GramStatTexNet: Efficient, Interpretable, and Neuro-Inspired Texture Analysis-by-Synthesis

Vasha DuTell[1,2] **Zeyu Yun**[2] **Anne Harrington**[1] **Michael Y-S. Fang**[4]

Mark Hamilton[1] **Christian Kovesdi**[1] **Edward H. Adelson**[1] **William T. Freeman**[1]

Ruth Rosenholtz[1,3]

[1]MIT CSAIL [2]UC Berkeley [3]NVIDIA [4]Spectacular Labs
{vasha}@berkeley.edu

Abstract

Deep networks achieve high realism in texture synthesis but often rely on opaque, over-parameterized representations that have limited direct connection to biological vision. Neuroscience-informed pyramid models are interpretable, compact, and biologically inspired, but rely on small, hand-curated statistics sets that are difficult to generalize or extend. We introduce GramStatTexNet, a hybrid analysis-by-synthesis framework that combines the multi-scale Gabor filter structure of classical models with the flexibility of Gramian-based correlations, organized into structured filter families. This structured representation enables analyses that are challenging for deep-feature methods: each statistic carries a named filter-family identity, allowing us to decompose synthesis fidelity into per-family contributions and to learn an interpretable, low-dimensional embedding that preserves perceptual texture content. We compare our framework to classical and deep texture-synthesis baselines across diverse texture categories, and on multiple perceptual metrics, highlighting the compactness and interpretability advantages of this hybrid representation, alongside competitive synthesis quality. This same framework extends naturally to peripheral textures via spatial pooling, and to dynamic textures using a spatiotemporal filter bank. GramStatTexNet provides a unified, interpretable framework for analyzing and modeling visual information with natural extensions across space, time, and eccentricity.

1 Introduction

The human visual system demonstrates impressive performance and efficiency at processing and interpreting the dynamic visual world. In particular, human vision has the ability to recognize complex textures that are essential for identifying objects, surfaces, and materials. Humans achieve this both for static scenes and in the context of motion, enabling us to navigate and interact seamlessly with our environment. Understanding how to model these capabilities is crucial for advancing computer vision and machine learning applications. These goals are also closely related to interpretability, where biological-vision can serve as a natural source of inspiration for learning effective and more transparent visual representations.

Neuroscience-informed approaches to texture synthesis (and closely related peripheral vision) often model visual areas using non-parametric multi-scale pyramids, decomposing images into filter responses, and aggregating features into statistics sets [39]. These statistics sets are predictive of

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

neural responses to natural scenes [56], produce perceptually plausible textures, and serve as models of peripheral vision when applied over spatial pooling regions [42, 15]. Despite the success of such models for texture images, they rely on hand-curated statistics sets, which are challenging to design and to extend. One example of this is the spatiotemporal domain, where spatial statistics often lack clear analogues in the temporal domain.

Deep-learning based approaches to texture and peripheral synthesis offer strong results in terms of quality and speed, and eliminate the need for hand-curated features, but their reliance on over-parameterized latent spaces comes at the cost of interpretability and biological feasibility. Gram-matrix-based approaches synthesize textures by matching correlations between filter responses, using either deep multi-layer networks [18] or single-layer random filters [51]. This approach has also shown success for peripheral syntheses [52, 12]. However, these models rely on pretrained layers from networks like VGG [48], and thus offer no representational compression compared to the highly efficient texture models from visual neuroscience.

In this work, we introduce GramStatTexNet (GSTN), an analysis-by-synthesis approach that bridges interpretability, biological plausibility, and synthesis quality in texture modeling. Our model generates texture syntheses comparable in quality to deep network-based models while remaining efficient and interpretable. We organize statistical representations into structured families and use contrastive learning to identify the most critical statistics for texture differentiation, reducing representation size while minimizing synthesis quality reduction and exposing which spatial-frequency, orientation, and color correlations carry texture identity. We further demonstrate that the same framework applies beyond static foveal images: it is compatible with classical peripheral-vision synthesis pipelines via spatial pooling, and extends naturally to dynamic textures. Together, GramStatTexNet provides a unified, interpretable framework for modeling visual information that operates across space, time, and eccentricity.

2 Previous Work

Early approaches to texture synthesis were deeply rooted in efforts to model human vision both in neuroscience and computer vision. The field began with Julesz' [24] work, which utilized N-th order statistics in pixel space. Texture representation in Fourier space [30], and later in pyramid space [5, 17] improved texture models significantly [21]. Along with considerable improvement in synthesis quality, came a strong link to the neuroscience of human visual processing, with the oriented multi-scale filters of the steerable pyramid [47] mirroring both the receptive fields of neurons found in area V1 [50, 29], as well as their response properties [35]. The Portilla & Simoncelli model [39] further improved synthesis quality by including pixel, autocorrelation, and magnitude statistics on pyramid responses. For videos, models like [44] construct video texture by shuffling frames, but cannot model spatiotemporal texture.

These summary statistics texture models were later extended as peripheral vision models, with the incorporation of spatial pooling [42, 15]. This method involves synthesizing texture locally in many small overlapping regions, which when optimized jointly, lead to full-field image syntheses. By progressively increasing the size of these regions with increased distance from central vision, these models reflect loss of information in peripheral vision beyond the simple blurring of photoreceptor density loss. As analysis-by-synthesis models, they enable the synthesis of 'metamers' or 'mongrels'. These image syntheses appear lossy when viewed foveally (with central vision), but appear normal when viewed peripherally (metamers), and have been shown to capture the information available to human peripheral vision, therefore predicting human performance on a wide range of visual tasks (mongrels). These models have been since extended to incorporate color [52], GPU acceleration [4], and expanded field of view [3]. These models have also been combined with deep learning to explore higher-level similarities and divergences in human and machine vision [20]. Other lines of work have explored spatiotemporal texture using oriented spacetime energies [11, 10] and have extended deep correlation-based synthesis to video [49]. However, the compact pyramid-plus-correlation-statistics tradition exemplified by [39] has not to our knowledge been extended into the temporal domain, despite extensive characterization of human spatiotemporal sensitivity [25, 27].

A line of work in deep learning offers a way to generalize these statistics, enabling a path to biologically plausible spatiotemporal texture and peripheral syntheses (see [41] for a recent review). Style transfer [19] is the extension of a spatial texture method [18] which synthesizes textures by

matching summary statistics defined as correlations between layers of pre-trained image networks such as VGG-Net [48]. Like the classical models, this texture synthesis method has also been extended to peripheral vision to create successful syntheses [12, 52]. On one hand, these methods demonstrate strong results in terms of synthesis quality while eliminating the need for hand-curated statistics sets. However, compared to pyramid-based models they have three major disadvantages: 1) They are highly-overparametrized, with orders of magnitude more statistics. 2) They replace human visual neuroscience-informed filters with less biologically-feasible convolutional layers. 3) Their summary statistics of deep network hidden layer correlations are difficult to interpret.

In addition to neural-inspired approaches, pure computer vision and deep learning approaches have had success in realistic synthesis of both static and dynamic textures. In the spatial domain, state of the art texture synthesis approaches often use diffusion models and generate photorealistic results [6, 54]. For dynamic texture synthesis, early work [13] laid the groundwork for representing the spatiotemporal dependencies needed in this task. Many modern spatiotemporal models build on the style transfer approach [19], with two-stream architectures extending it to video [49]. This same model has been shown to also succeed for single-layer networks [51]. The addition of long-range spatial and temporal dependencies have been shown to produce good spatiotemporal syntheses [55], and neural cellular automaton models have also been used to generate ultra-compact spatial [32] and real-time controllable spatiotemporal [37] texture synthesis.

3 GramStatTexNet

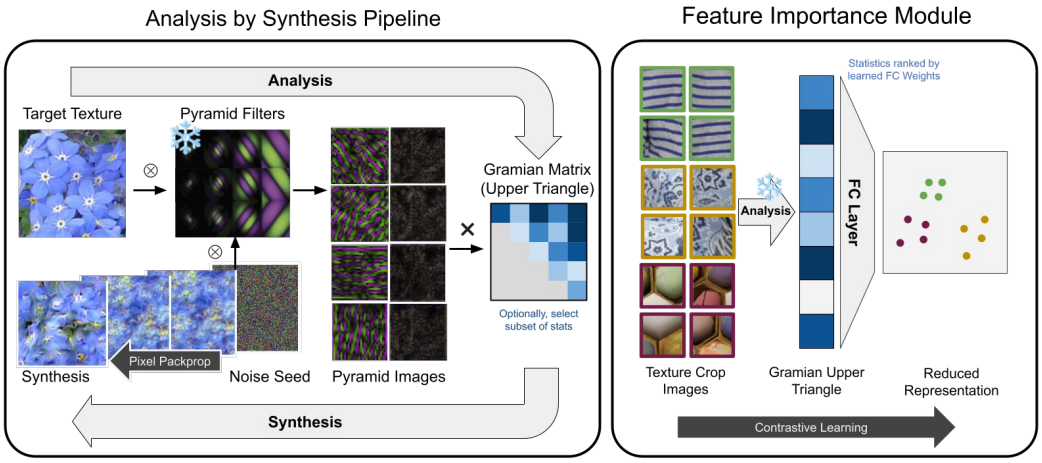


Figure 1: Left: GramStatTexNet, our analysis-by-synthesis texture model decomposes images into stacks (pyramids) of varying scales, orientations, and color channels, using a multi-scale pyramid filter bank. Pair-wise correlations between all pyramid images are represented in the Gram matrix; the upper triangle of this Gram matrix represents the full statistics set. Synthesis proceeds by calculating loss on statistics between a target and noise seed image, backpropagating this loss through the network to adjust pixel values until a synthesized image is produced. Right: To learn a reduced statistics set, a single fully connected layer compresses to the Gram upper triangle to a reduced representation. These weights are trained using a contrastive loss, self-supervised by the objective of crops from the same texture image lying nearby in the reduced feature space.

We present a simple but powerful model (Figure 1, left) that combines the interpretable and biologically-inspired pyramid filters of [39] with the powerful Gramian matrix representation of [18]. Our model leverages the GPU-accelerated multi-scale pyramids from [4], convolving input images with pyramid filters at individual combinations of orientation, phase, scale, and color, the output we call 'pyramid images'. We treat each pyramid image as a channel, and calculate the pair-wise correlations between each pyramid image pair, collapsing over space as a Gram matrix. We use the upper triangle and diagonal of this matrix as the full statistics set for downstream analysis, where each statistic corresponds to a pairwise correlation between filters. This can be used not only to analyze the properties of a given texture, but also to synthesize new texture images.

3.1 Texture Synthesis

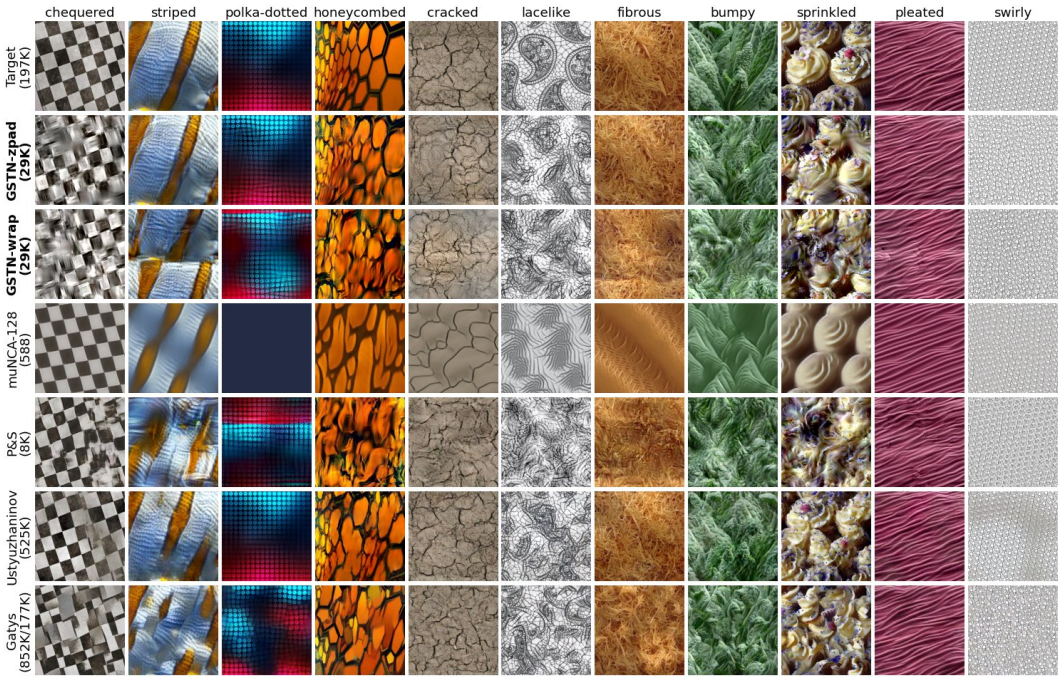


Figure 2: Texture syntheses across DTD categories spanning a range of structural regularity. Top row: target textures. GSTN matches the Gramian of multi-scale Gabor filters, with GSTN-zpad using zero-padding retains more spatial image content by avoiding wrapping artifacts, and GSTN-wrap using a wrapped Fourier transform, which runs $\sim$2x faster. Mμnca is a small footprint cellular automaton model. P&S uses a similar multi-scale Gabor strategy, but matches 8,000 hand-picked statistics rather than a full Gramian. Gatys matches the Gramian of deep network VGG-19. Ustyuzhaninov, like Gatys, matches a Gramian but calculated on random convolutional filters.

We first apply our model in an analysis-by-synthesis framework to generate novel texture images, matching the statistics set of an input image, and synthesizing new textures by optimizing noise images to match the statistics set. Comparing synthesized images to their targets, we demonstrate good synthesis quality over a variety of texture types and regularities, sampling textures from a variety of categories from the Describable Texture Dataset (DTD) [8] (Figure 2, top 2 rows).

Comparing GSTN to other related texture synthesis models (Figure 2), we see quality improvement over previous multi-scale-pyramid synthesis methods [39] (P&S), with competitive quality to the much larger, deep-network and convolution-based synthesis methods [18, 51], despite over an order of magnitude fewer statistics. Evaluating these results quantitatively, we find that the zero-padded GSTN outperforms P&S on the majority of categories across all nine evaluation metrics tested (30–46 of 47 categories per metric; Appendix Figure 10). Against Gatys, a metric-target effect emerges with Gatys winning on deep-feature metrics (DISTS, LPIPS) built on the same VGG features that its synthesis loss optimizes against, while GSTN wins on classical perceptual and structural metrics (FSIM, SSIM, VIF, GMSD, HaarPSI, PSD-L2) in 32–46 of 47 categories, despite using a 30x smaller representation (Appendix Figure 11).

4 Interpretable Representation of Texture

4.1 Correlation Families

An advantage of our method is that the Gramian representation for a given input texture is comprised of correlations between known pyramid filters, and is thus transparent and interpretable such that the properties of any given texture can be analyzed and understood from first-principles. In addition, through ablation, the quality of syntheses can be attributed to different statistical families. To make sense of these Gramian correlation statistics, we organize them into families falling into two main groups: one of statistics correlating one or more non-sub-band (non-pyramid level) filter images,

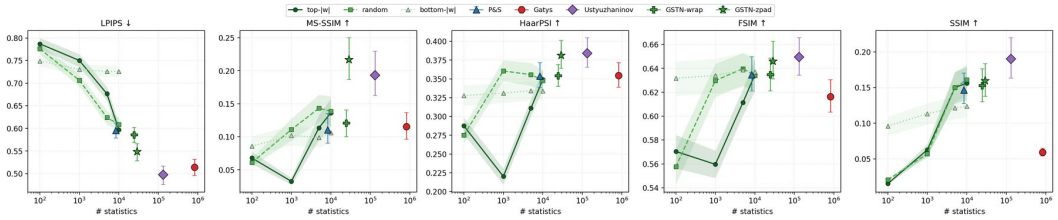


Figure 3: Pareto comparison: representation size versus synthesis quality across five perceptual metrics on 141 DTD validation textures. GSTN-zpad sits on the Pareto frontier between P&S and the order-of-magnitude-larger feature-space methods (Ustyuzhaninov, Gatys), achieving competitive quality with 30x fewer statistics than Gatys on four of five metrics. On LPIPS, Gatys wins, consistent with the metric evaluating similarity in the same VGG features that Gatys' synthesis loss optimizes against. Depletion variants (top-|w|, random, bottom-|w|) extend the GSTN frontier toward smaller representation sizes. Full metric breakdown in Appendix Figures 10 and 11.

Table 1: Summary of statistical families for the spatial model. Non-sub-band statistics contain at least one marginal, highpass, or lowpass statistic. Each sub-band statistic correlates two channels of the pyramid representation, where a channel is specified by phase, scale (level), orientation, and color. A family is defined by which of these attributes are shared and which differ between the two correlated channels. Families marked † are structured: only a single attribute varies, making them approximately analogous to the hand-curated statistics of classical pyramid models [39]. Unmarked families are unstructured, with multiple attributes varying at once. S = scale, C = color, O = orientation, P = phase (re = real, im = imaginary, mag = magnitude).

Name	Code	Shared	Varies	Correlations	#
Pixel and residual					
Marginal†	`marginal`	–	–	Per-color pixel mean, variance, std (appended)	9
Highpass†	`highpass`	–	–	Highpass residuals across three color channels	6
Lowpass†	`lowpass`	–	–	Two lowpass channels differing in S, C, both, or neither	231
Non-oriented × band	`pass_multi`	–	–	Highpass or lowpass statistic with any sub-band channel	5,247
Sub-band					
Cross-orientation†	`sub_Xori`	S, C, P	O	Differ only in orientation	324
Cross-color†	`sub_Xcolor`	S, O, P	C	Differ only in color (incl. non-differing band energies)	432
Cross-scale†	`sub_Xlevel`	C, O, P	S	Differ only in scale	540
Cross-phase, re × im†	`sub_Xri_eq`	S, C, O	P (re, im)	Real and imaginary parts of the same filter output	72
Cross-phase, re × mag†	`sub_Xrm_eq`	S, C, O	P (re, mag)	Real part and magnitude of the same filter output	72
Cross-phase, im × mag†	`sub_Xim_eq`	S, C, O	P (im, mag)	Imaginary part and magnitude of the same filter output	72
Real, multi-attr.	`sub_real_Xmulti`	P (re)	S, C, O	Two real outputs differing in $\geq$2 of S, C, O	2,196
Imaginary, multi-attr.	`sub_imag_Xmulti`	P (im)	S, C, O	Two imaginary outputs differing in $\geq$2 of S, C, O	2,196
Magnitude, multi-attr.	`sub_magn_Xmulti`	P (mag)	S, C, O	Two magnitude outputs differing in $\geq$2 of S, C, O	2,196
Mixed-phase, multi-attr.	`sub_Xmulti`	–	P, S, C, O	Different phase types, differing in $\geq$1 of S, C, O	15,336
Total					**28,929**

which we call 'Non-sub-band', and includes marginal statistics. The second group we call 'Sub-band' and computes correlations exclusively between sub-band images (Appendix Table 1). We further organize the Sub-band group into sub-groups, based on which properties are shared and differ between the pyramid images correlated. The hand-curated statistics sets from [39] rely heavily on correlation statistics analogous to the first 3 non-sub-band plus first 6 sub-band families, which we call 'structured' statistics. These share all properties except for one (i.e. sub_Xlevel at neighboring levels).

To analyze how statistical families differ across texture types, we group images by DTD category [8] and evaluate the per-category mean and within-category standard deviation of each family's z-scored correlations (Figure 4). Unsupervised hierarchical clustering on the mean fingerprints recovers an interpretable global organization, with regular/structured textures (chequered, striped, polka-dotted, grid) on the left, balanced soft/irregular textures (frilly, veined, lacelike, bubbly) on the right, and stochastic mid-frequency textures (flecked, marbled, cracked, fibrous) in the middle.

Several categories show distinctive single-family signatures that map onto recognizable image properties: chequered has the strongest marginal deviation, reflecting that its pixels are mostly either fully dark or fully light; lined and banded show elevated sub_magn_Xmulti, capturing the repetition of a single dominant orientation across spatial scales; frilly and veined show strong lowpass elevation,

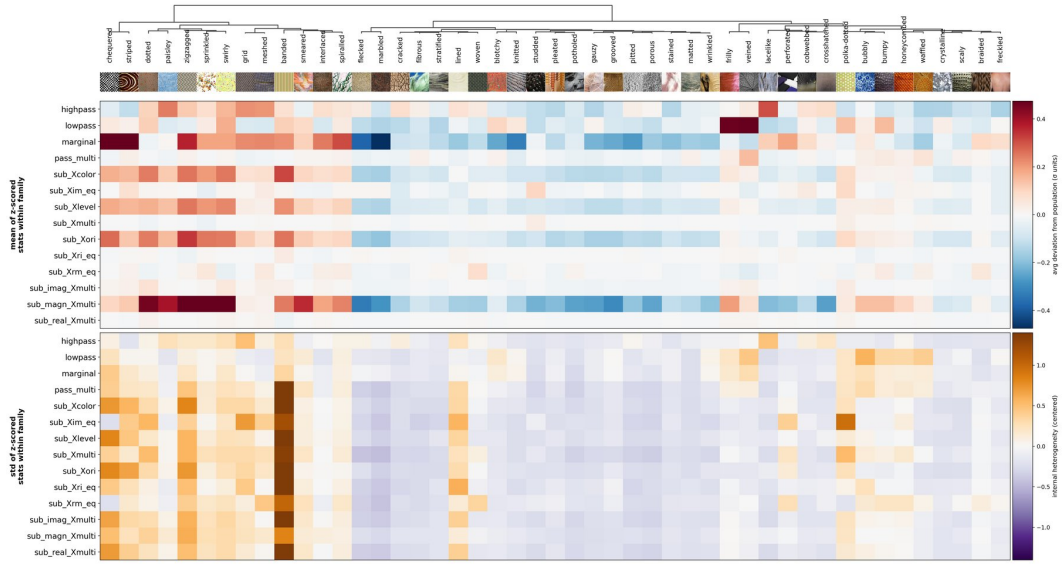


Figure 4: Family-level texture fingerprints across DTD. Top: Hierarchical clustering on per-texture category mean signatures recovers a smoothness vs structure ordering without supervision; one representative exemplar per category at each leaf. Top heatmap: per-category mean of within-family z-scored stat values. Bottom heatmap: per-category within-family std of z-scored stats (centered; orange = above-average within-family heterogeneity, purple = below).

reflecting their smooth, gradual variation rather than sharp edges; and crosshatched shows elevated highpass, reflecting the fine-scale edges produced by its overlapping line patterns. Three families (sub_Xri_eq, sub_Xrm_eq, and sub_imag_Xmulti) are nearly flat, indicating that they may carry little information; we test their causal relevance via family ablation.

The heterogeneity heatmap (Figure 4, bottom) reveals a complementary axis: banded, lined, and zigzagged are statistically diverse across exemplars (a broad range of stripe widths, orientations, and contrasts), while soft/irregular categories on the right (frilly, cobwebbed, freckled) are statistically tight despite visual diversity, suggesting the family-level summary captures a stable statistical regime even when individual exemplars look quite different.

4.2 Family-Based Ablation

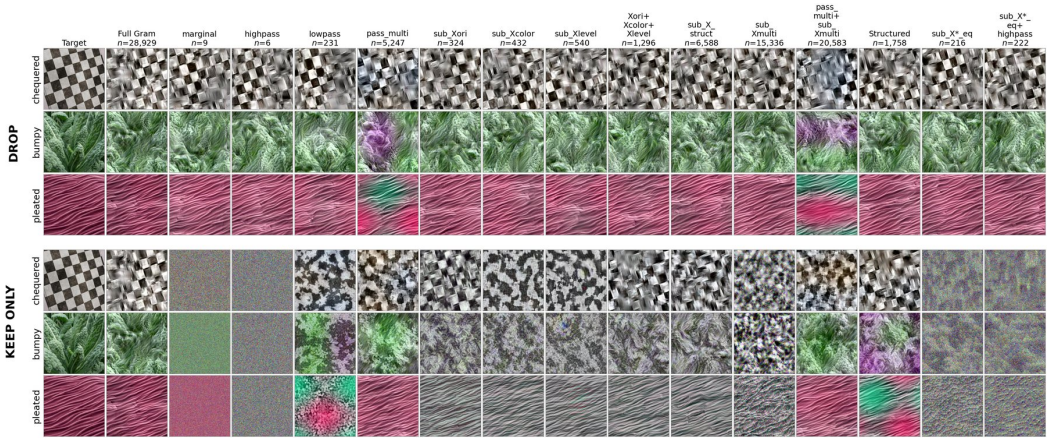


Figure 5: Ablated syntheses optimized using statistical subsets defined by categorical families. Families with 'structured' statistics pairs where only one parameter differs generate poor synthesis compared to larger families of unstructured pairs.

To understand how synthesis quality is related to different family groups, we perform paired ablation experiments: synthesizing textures with each family removed (DROP) and with each family alone (KEEP ONLY) (Figure 5). DROP reveals which families are necessary, and KEEP ONLY reveals which are sufficient. We find that no single family is sufficient on its own. Even the largest individual family (sub_multi, n=15,336) produces visibly degraded textures, and the union of *structured* families analogous to classical pyramid statistics (n=1,758) preserves only coarse texture identity. The bulk of synthesis-relevant information is carried by the large *unstructured* families whose correlations span multiple filter attributes simultaneously (pass_multi, sub_multi). Interestingly, synths from the low-signal *dead_eq* families (three real/imag/magnitude equivalence families identified as near-zero in Figure 4) show some of the worst quality, consistent with these carrying little information. Conversely, dropping individual families produces only minor visual degradation in most cases, with even the sub_multi drop of 15,336 statistics causing little visible loss, and reflecting substantial redundancy in the statistical representation. One notable exception is that dropping pass_multi visibly disrupts synthesis. Together, these results suggest that the classical structured-statistics approach captures only a fraction of synthesis-relevant content; the bulk of texture identity is encoded in cross-attribute correlations (pass_multi, sub_multi) that are absent from classical pyramid models, and may explain their characteristic synthesis limitations. The structured families approximate P&S statistics, which predict V2 responses and human textured discrimination [16, 34], whereas the cross-attribute families are untested, making their importance a testable hypothesis about higher visual areas.

5 Feature Importance Ordering

To further investigate the contribution of individual texture statistics and families to synthesis quality, we use contrastive learning to reduce the correlation statistics to a compressed feature vector that can group textures (Figure 1, right). We train a single fully connected layer (28,929x100) on the output of GSTN using InfoNCE [36] to group texture crops from the same parent texture in the compressed space. We find that the trained single layer is sufficient to group same texture crops in embedding space (Appendix Figure 15). Here, we solve the contrastive learning problem as is done typically, using gradient-based optimization. However, in this case with a single-layer encoder, it can also be solved analytically; we derive this solution in Appendix A.18.

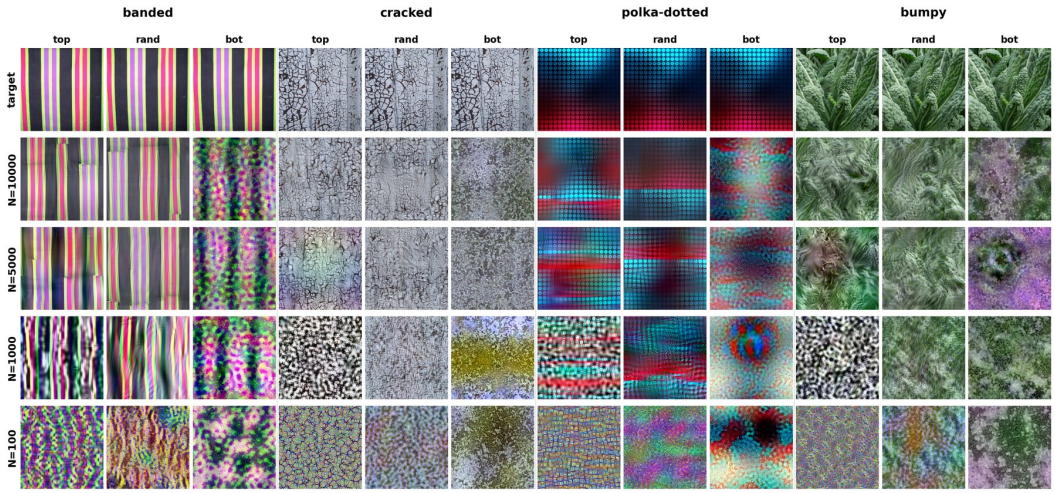


Figure 6: Stat-depletion comparison. Syntheses produced when only N stats are kept out of the full 28,929-dimensional GSTN representation. Columns within each category compare three selection rules: top-N by trained-compressor importance, rand-N randomly sampled, and bot-N least-important. Synths use GSTN-wrap for computational efficiency.

To evaluate the importance of different statistics to the contrastive learning task, we order statistics from most to least important based on both the absolute value of their weightings, as well as by their Shapley values [43] (not shown, similar results) (Appendix Figure 16). We find correlations between sub-band filter outputs with few or no shared attributes (sub_Xmulti) are most important to the contrastive learning task. By contrast, correlations between lowpass filter outputs are least

important for grouping textures. We visualize the relative contributions of individual families to the total statistics set when ordered by importance (Appendix Figure 17), and find that sub_Xmulti, highpass, sub-magnitude_Xmulti, and pass_multi families make an out-sized contribution to the most important statistics.

To evaluate the transfer of importance in the contrastive task to synthesis quality, we perform a set of additional synthesis ablation experiments, matching only statistics of the highest weighting in the contrastive learning task (Figure 6). We find that importance in contrastive learning broadly transfers to synthesis quality: by selecting the top-ranked statistics, we achieve acceptable syntheses with only 5,000 of the total 28,929 statistics, an 83% reduction. Statistics from bottom of the ranking, by contrast, produce extremely poor syntheses, confirming that the ranking carries real information about which statistics are needed for texture reproduction.

When we compare top-ranked statistics to randomly-selected control sets of the same size, we find that randomly chosen statistics in fact out-perform the top-ranked ones for synthesis, even achieving acceptable synthesis with only 1000 statistics in some cases (Figure 6). This trend is reflective quantitatively in Figure 3, and reflects a distinction between two objectives: The contrastive task selects statistics that discriminate between texture categories, concentrating in a few specific families (sub_Xmulti, highpass) maximally separating categories in a compressed embedding space. Synthesis, on the other hand, requires statistics that broadly constrain perceptual appearance across many aspects of a texture. A random subset samples broadly across the full feature space, including individually low-discriminability statistics that nonetheless contribute to constraining the synthesis to produce a coherent texture. The top-N subset may be more compact and more interpretable, but provides a narrower basis for reconstruction. This finding aligns with the classical distinction between discriminative and generative objectives in machine learning [33]: representations optimized to discriminate between categories are not necessarily optimal for modeling the joint statistical structure within them. This also echoes [51] where correlations on random convolutional filters prove sufficient for texture synthesis.

6 Beyond Static Foveal Synthesis

Having established an interpretable statistical representation for static textures, we now demonstrate that the same framework extends naturally beyond the static uniform, foveal texture image setting. We present two such extensions: applying the model over spatial pooling regions to produce foveated syntheses compatible with peripheral-vision pipelines, and replacing the spatial filter bank with a spatiotemporal one to synthesize dynamic textures. Neither extension requires changes to the underlying analysis-by-synthesis pipeline, only to the spatial aggregation of the model, or a temporal dimension to the filter bank; these illustrate the generality and interoperability of our pyramid + correlation-statistics approach.

Peripheral Synthesis

To demonstrate compatibility with classical peripheral-vision synthesis pipelines, we combine our statistical synthesis method with spatial pooling following [15, 42], motivated by evidence that pooled summary statistics predict crowding [1], generating foveated synthesis on full-size images which have non-uniform texture (Figure 7). The resulting syntheses preserve high fidelity at gaze center and progressively introduce spatial distortions toward the periphery, matching the smooth, spatially-varying degradation pattern produced by prior peripheral synthesis pipelines [15, 4]. Utilizing the reduced representation learned from contrastive learning, we are able to reduce the size of our peripheral syntheses significantly, while retaining high quality. While we do not run psychophysical validation here, this demonstration establishes that the compressed statistical representation remains compatible with the spatially-pooled synthesis pipeline of [15, 4].

Dynamic Textures

Extending human vision inspired models such as the Portilla–Simoncelli image model to the spatiotemporal domain is non-trivial, as its hand-selected cross-scale and cross-orientation statistics lack clear analogues in the time domain. Instead of hand picking important statistics, our method, leveraging all combinations of filter correlation statistics with the Gram matrix, can be naturally extended to video textures. We extend the spatial pyramid into the time domain by designing a bank of 3D Gabor filters arranged in stacked concentric rings in spatiotemporal frequency space,

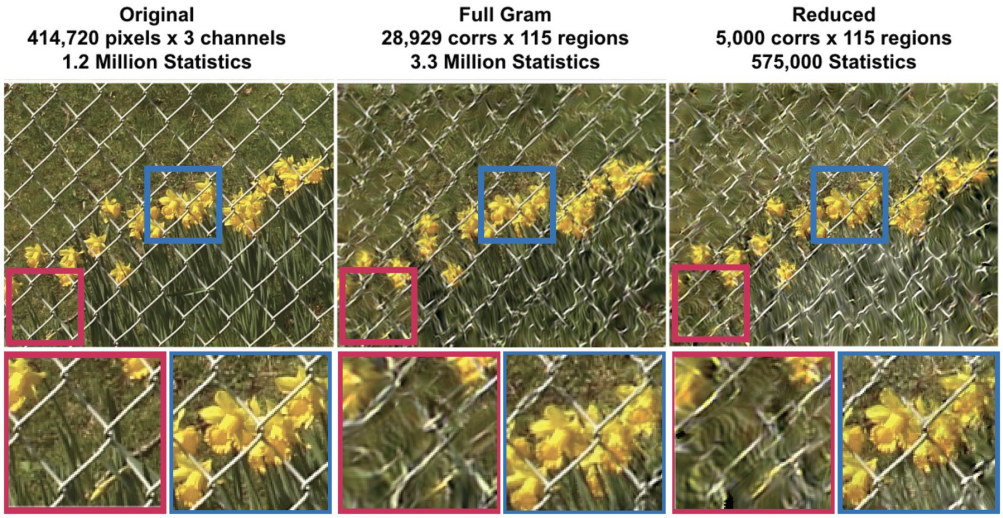


Figure 7: Synthesis incorporating spatial pooling. Spatial pooling produces foveated syntheses analogous to those used in peripheral-vision studies, where the center of the visual field is rendered at high fidelity, and the edges of vision are lossy. Original image (left) represents 1.2 Million values per image. Syntheses using the full model (middle) with 28,929 statistics per pooling region, and 115 pooling regions increases the representation size to 3.3 million, and creates quality syntheses. Our reduced model (right) decreases the statistical representation to 5,000 statistics per pooling region, for 575,000 total values, while still achieving good results. This results in a greater than 50% compression compared to the input image.

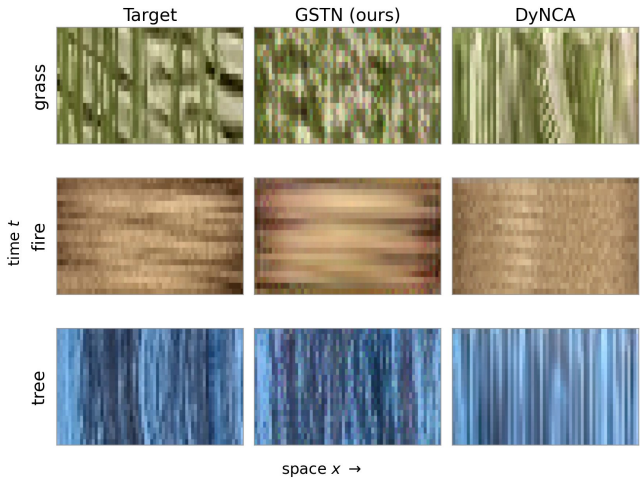
with each filter paired with its quadrature complement. The filter bank is parameterized to span the spatiotemporal frequency range of human visual sensitivity [25]; details and visualizations are given in Appendix and Appendix Figure 18.

We apply the same Gram matrix model to calculate correlations between our spatiotemporal filters, synthesizing texture videos by matching the set of filter correlation statistics. While contrastive learning in the spatiotemporal case is not computationally feasible given resource constraints, we are able to reduce our model size somewhat based on our findings from the spatial-only case, removing lowpass filters. Our model synthesizes videos that visually match well to the target textures (Figure 8), and results span a set of texture videos chosen to maximize variety in structure and motion properties, while maintaining computational tractability (Table 6). Our syntheses retain recognizable temporal structure consistent with the input video. We do not claim improvement over prior dynamic-texture synthesis methods (e.g., [49, 37]); rather, we show that the same pyramid + correlation framework that succeeds on static textures extends naturally to video.

7 Discussion

GramStatTexNet merges the interpretability of biologically inspired multi-scale pyramid filters with the flexibility of Gramian-based correlations. By organizing statistics into structured filter families, we identify a compressed subset of correlations that retains synthesis quality while substantially reducing representation size, and expose which spatial-frequency, orientation, and color correlations carry texture identity. The same framework extends naturally to peripheral and dynamic textures via spatial pooling and a spatiotemporal filter bank, providing a unified, interpretable framework for modeling visual information across space, time, and eccentricity.

Limitations Our depletion analysis uses GSTN-wrap for computational efficiency; replication on GSTN-zpad is left to future work, though we expect qualitatively similar findings, with improvement gains mirroring the full statistics set. We provide no new neural or psychophysical validation of this representation, for static, peripheral, or dynamic textures; validation against human observers remains future work. Finally, contrastive compression is currently tractable only for the spatial model.



Metric	GSTN	DyNCA
Deep-feature appearance		
LPIPS-Alex ↓	0.323	**0.159**
LPIPS-VGG ↓	0.515	**0.323**
DISTS ↓	0.324	**0.199**
FID ↓	162	**83**
CMMD ↓	2.441	**1.339**
Pixel-level		
SSIM ↑	**0.114**	0.107
MSE ↓	**0.044**	**0.044**
PSNR ↑	**14.2**	14.1
Temporal		
Spectrum dist. ↓	**0.176**	0.518
$\lvert\Delta$Curvature$\rvert$ ↓	**12.5**	17.8
$\lvert\Delta$Diversity$\rvert$ ↓	0.065	**0.037**

Figure 8: Video texture synthesis GSTN vs. DyNCA [37]. **Left:** space–time (x–t) slices: each panel shows one row of pixels (horizontal) across all frames (time runs down), so vertical streaks are static structure, slanted streaks are motion, and horizontal bands are frame-wide temporal change. GSTN slices vary over time much as the targets do, including fire's flicker bands, though with less fine spatial detail and some chromatic noise; DyNCA's grass and tree slices are dominated by vertical streaks (little change over time), and its fire slice shows only weak flicker bands. **Right:** mean over all five DynTex videos (per-video results in Appendix Table 6). DyNCA leads on deep-feature appearance metrics [23] and diversity match; GSTN leads on temporal power-spectrum match and perceptual straightness [22]; pixel-level metrics are close, with GSTN marginally ahead on SSIM and PSNR and MSE tied. Bold = better (both when tied); $\lvert\Delta \cdot\rvert$ = distance from the target's own value.

Future work Psychophysical evaluation of peripheral and dynamic-texture syntheses would test whether the compressed representation preserves perceptually relevant information. Scaling contrastive compression to spatiotemporal statistics would reveal which spatiotemporal correlations carry the most information for dynamic texture identity. Finally, the compressed representation is amenable to gradient-guided integration with pretrained generative models such as diffusion networks, which we leave to future work.

Broader Impacts This is foundational research on interpretable, biologically-inspired visual representations, with no direct path to negative societal applications. Our syntheses are visibly lower-fidelity than current diffusion-based methods, so this work does not meaningfully advance capabilities relevant to generative misuse such as deepfakes. Potential positive impacts include: (i) interpretable, family-decomposed representations that link learned features to well-characterized properties of biological vision: our pyramid filters approximate V1 response properties, and the family taxonomy offers a hypothesis space, not validated here, for how V2 might combine V1 outputs, complementing macaque single-unit [56] and macaque/human [16] evidence of V2 selectivity for the structured-family statistics; (ii) compressed representations that reduce storage and compute for downstream pipelines; and (iii) compatibility with foveated rendering, with potential applications in low-bandwidth AR/VR and accessibility tools.

8 Acknowledgments and Disclosures

This work was funded by the CSAIL MEnTorEd Opportunities in Research (METEOR) Fellowship, US National Science Foundation under grant number 1955219, as well as National Science Foundation Grant BCS-1826757 to PI Rosenholtz. The authors acknowledge the MIT SuperCloud Reuther et al. (2018) and Lincoln Laboratory Supercomputing Center for providing HPC resources that have contributed to the research results reported within this paper.

References

[1] Benjamin Balas, Lisa Nakano, and Ruth Rosenholtz. A summary-statistic representation in peripheral vision explains visual crowding. *Journal of vision*, 9(12):13–13, 2009.

[2] Benjamin J Balas. Texture synthesis and perception: Using computational models to study texture representations in the human visual system. *Vision research*, 46(3):299–309, 2006.

[3] William F Broderick, Gizem Rufo, Jonathan Winawer, and Eero P Simoncelli. Foveated metamers of the early visual system. *bioRxiv*, pages 2023–05, 2023.

[4] Rachel Brown, Vasha DuTell, Bruce Walter, Ruth Rosenholtz, Peter Shirley, Morgan McGuire, and David Luebke. Efficient dataflow modeling of peripheral encoding in the human visual system. *arXiv preprint arXiv:2107.11505*, 2021.

[5] Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier, 1987.

[6] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18558–18568, 2023.

[7] Yubei Chen, Zeyu Yun, Yi Ma, Bruno Olshausen, and Yann LeCun. Minimalistic unsupervised learning with the sparse manifold transform. *arXiv preprint arXiv:2209.15261*, 2022.

[8] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[9] Aaron Defazio and Samy Jelassi. Adaptivity without compromise: A momentumized, adaptive, dual averaged gradient method for stochastic optimization. *arXiv*, 2021.

[10] Konstantinos GP Derpanis and Richard Wildes. Spacetime texture representation and recognition based on a spatiotemporal orientation analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1193–1205, 2012.

[11] Konstantinos G Derpanis and Richard P Wildes. Dynamic texture recognition based on distributions of spacetime oriented structure. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 191–198. IEEE, 2010.

[12] Arturo Deza, Aditya Jonnalagadda, and Miguel Eckstein. Towards metamerism via foveated style transfer. *arXiv preprint arXiv:1705.10041*, 2017.

[13] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International journal of computer vision*, 51:91–109, 2003.

[14] Vasha DuTell, William Freeman, and Ruth Rosenholtz. Evaluating pyramid-based image statistics using contrastive learning. *Journal of Vision*, 23(9):5744–5744, 2023.

[15] Jeremy Freeman and Eero P Simoncelli. Metamers of the ventral stream. *Nature neuroscience*, 14(9): 1195–1201, 2011.

[16] Jeremy Freeman, Corey M Ziemba, David J Heeger, Eero P Simoncelli, and J Anthony Movshon. A functional and perceptual signature of the second visual area in primates. *Nature neuroscience*, 16(7): 974–981, 2013.

[17] William T Freeman, Edward H Adelson, et al. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

[18] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. *Advances in neural information processing systems*, 28, 2015.

[19] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[20] Anne Harrington, Vasha DuTell, Mark Hamilton, Ayush Tewari, Simon Stent, William T Freeman, and Ruth Rosenholtz. Coco-periph: Bridging the gap between human and machine perception in the periphery. In *The Twelfth International Conference on Learning Representations*, 2024.

[21] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238, 1995.

[22] Olivier J Hénaff, Robbe LT Goris, and Eero P Simoncelli. Perceptual straightening of natural videos. *Nature neuroscience*, 22(6):984–991, 2019.

[23] Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking fid: Towards a better evaluation metric for image generation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9307–9315. IEEE, 2024.

[24] Bela Julesz. Visual pattern discrimination. *IRE transactions on Information Theory*, 8(2):84–92, 1962.

[25] Donald H Kelly. Motion and vision. ii. stabilized spatio-temporal threshold surface. *Josa*, 69(10): 1340–1349, 1979.

[26] Christian Koevesdi, Vasha DuTell, Anne Harrington, Mark Hamilton, William T Freeman, and Ruth Rosenholtz. Stattexnet: Evaluating the importance of statistical parameters for pyramid-based texture and peripheral vision models. In *NeuRIPS 2023 Workshop on Gaze Meets ML*, 2023.

[27] Brooke Krajancich, Petr Kellnhofer, and Gordon Wetzstein. A perceptual model for eccentricity-dependent spatio-temporal flicker fusion and its applications to foveated graphics. *ACM Transactions on Graphics (TOG)*, 40(4):1–11, 2021.

[28] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[29] Jitendra Malik and Pietro Perona. Preattentive texture discrimination with early vision mechanisms. *JOSA A*, 7(5):923–932, 1990.

[30] Takashi Matsuyama, Shu-Ichi Miura, and Makoto Nagao. Structural analysis of natural textures by fourier transformation. *Computer vision, graphics, and image processing*, 24(3):347–362, 1983.

[31] Konstantin Mishchenko and Aaron Defazio. Prodigy: An expeditiously adaptive parameter-free learner. *arXiv preprint arXiv:2306.06101*, 2023.

[32] Alexander Mordvintsev and Eyvind Niklasson. munca: Texture generation with ultra-compact neural cellular automata. *arXiv preprint arXiv:2111.13545*, 2021.

[33] Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14, 2001.

[34] Gouki Okazawa, Satohiro Tajima, and Hidehiko Komatsu. Image statistics underlying natural texture selectivity of neurons in macaque v4. *Proceedings of the National Academy of Sciences*, 112(4):E351–E360, 2015.

[35] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[36] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[37] Ehsan Pajouheshgar, Yitao Xu, Tong Zhang, and Sabine Süsstrunk. Dynca: Real-time dynamic texture synthesis using neural cellular automata. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20742–20751, 2023.

[38] Renaud Péteri, Sándor Fazekas, and Mark J. Huiskes. DynTex : a Comprehensive Database of Dynamic Textures. *Pattern Recognition Letters*, doi: 10.1016/j.patrec.2010.05.009, 2010. http://projects.cwi.nl/dyntex/.

[39] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1):49–70, 2000.

[40] Rafael Reisenhofer, Sebastian Bosse, Gitta Kutyniok, and Thomas Wiegand. A haar wavelet-based perceptual similarity index for image quality assessment. *Signal Processing: Image Communication*, 61: 33–43, 2018.

[41] Carlos Rodriguez-Pardo, Dan Casas, Elena Garces, and Jorge Lopez-Moreno. Textile: A differentiable metric for texture tileability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4439–4449, 2024.

[42] Ruth Rosenholtz, Jie Huang, Alvin Raj, Benjamin J Balas, and Livia Ilie. A summary statistic representation in peripheral vision explains visual search. *Journal of vision*, 12(4):14–14, 2012.

[43] Alvin E Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[44] Arno Schödl, Richard Szeliski, David Salesin, and Irfan A. Essa. Video textures. In *SIGGRAPH*, pages 489–498, 2000.

[45] Hamid R Sheikh, Muhammad F Sabir, Alan C Bovik, et al. A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Trans. Image Process.*, 15(11):3440–3451, 2006.

[46] Eero Peter Simoncelli. *Distributed representation and analysis of visual motion*. PhD thesis, Massachusetts Institute of Technology, 1993.

[47] Eero P Simoncelli and William T Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *Proceedings., International Conference on Image Processing*, pages 444–447. IEEE, 1995.

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[49] Matthew Tesfaldet, Marcus A Brubaker, and Konstantinos G Derpanis. Two-stream convolutional networks for dynamic texture synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6703–6712, 2018.

[50] Mark R Turner. Texture discrimination by gabor functions. *Biological cybernetics*, 55(2-3):71–82, 1986.

[51] Ivan Ustyuzhaninov, Wieland Brendel, Leon Gatys, and Matthias Bethge. What does it take to generate natural textures? In *International conference on learning representations*, 2017.

[52] Thomas Wallis, Christina Funke, Alexander Ecker, Leon Gatys, Felix Wichmann, and Matthias Bethge. Towards matching peripheral appearance for arbitrary natural images using deep features. *Journal of Vision*, 17(10):786–786, 2017.

[53] Wufeng Xue, Lei Zhang, Xuanqin Mou, and Alan C Bovik. Gradient magnitude similarity deviation: A highly efficient perceptual image quality index. *IEEE transactions on image processing*, 23(2):684–695, 2013.

[54] Kim Youwang, Tae-Hyun Oh, and Gerard Pons-Moll. Paint-it: Text-to-texture synthesis via deep convolutional texture map optimization and physically-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4347–4356, 2024.

[55] Kaitai Zhang, Bin Wang, Hong-Shuo Chen, Xuejing Lei, Ye Wang, and C-C Jay Kuo. Dynamic texture synthesis by incorporating long-range spatial and temporal correlations. In *2021 International Symposium on Signals, Circuits and Systems (ISSCS)*, pages 1–4. IEEE, 2021.

[56] Corey M Ziemba, Jeremy Freeman, J Anthony Movshon, and Eero P Simoncelli. Selectivity and tolerance for visual texture in macaque v2. *Proceedings of the National Academy of Sciences*, 113(22):E3140–E3149, 2016.

A Appendix

A.1 Image and Video Synthesis

We synthesized visualized textures following the synthesis procedure in Koevesdi et al. [26], using the filter portion of the codebase from [4], with the pyramid settings at 4 orientations, 6 edge levels, edge start at level 1, with marginal statistics included. For image synthesis batched over the DTD dataset, we used Prodigy optimizer [31]. For image synthesis on individual textures, foveated synthesis, and spatiotemporal generation, we used the LBFG-S solver [28] as demonstrated for textures in [18]. See Appendix section A.3 for details on optimizers.

Syntheses for random filters [51] were created by a custom Pytorch port from the official Theano repo (https://github.com/ivust/random-texture-synthesis) with 20k iterations and the Prodigy solver [9]. Synths for Gatys et al [18] were created from an unofficial Pytorch repo for [18] (https://github.com/trsvchn/deep-textures). Synths for Portilla & Simoncelli were created using the official color extension package in Matlab (https://www.cns.nyu.edu/pub/eero/colorTextureSynth.tgz).

Spatial pooling of our model for peripheral syntheses was incorporated by tiling the image with overlapping pooling regions of 16x16 pixels tiled across the image, then synthesizing our texture model at each region in parallel. We utilize the pooling method from [4], warping the image, synthesizing over uniformly-tiled regions in the warped image space, and un-warping after synthesis, with parameters matched as closely as possible to [15].

To synthesize novel videos matching the statistics of input videos, we follow the same procedure as for images, calculating the spatiotemporal statistics vector from a target video. Then, starting with a white-noise video, we use LBFG-S to adjust pixel values in the synthetic video, using backpropagation of the mean squared error in statistics space between the synthetic and target video. This results in a synthetic video that matches the statistics of the original target video, but with scrambling of absolute spatial and temporal localization. Given this loss of spatial and temporal information, we use texture videos from the DynTex Database [38], which we spatially center-crop and downsample to minimize artifacts and ensure uniformity.

A.2 Spatiotemporal Filters

We chose the range of the spatiotemporal space up to 16 CPD and 20 Hz, following the human spatiotemporal contrast sensitivity calculations for foveal vision [25], and spatiotemporal flicker fusion for peripheral vision [27]. Within this range, we designed filters as 3D Gaussians arranged in concentric stacked rings (shaped like a mochi-donut), similar to [46]. A single ring represents 8 filters of varying orientations (motion directions) at the same spatial and temporal frequency. Rings are arranged concentrically to select for increasing spatial frequencies, and stacked along the temporal axis to select for increasing temporal frequencies. Each filter located in the positive half of the temporal frequency space has a quadrature pair filter mirrored in the negative half of the temporal frequency space, in the opposite spatial quadrant. To tile the space, we chose a set of 8 orientations, 4 spatial scales, and 3 temporal scales, with a real and imaginary pair each. We also include one spatial and one temporal low-pass filter, each with a real and imaginary pair, for a total of 200 filters. These spatiotemporal filter pairs correspond to a set of Gabor filters.

A.3 Optimizers: L-BFGS, Madgrad, Prodigy Comparison

L-BFGS [28] is often the optimizer of choice for statistical texture synthesis given its speed and efficiency at jointly optimizing the pixel space to find the target image, due to it being a second order method. However, the calculation of the Hessian makes batching image syntheses problematic, and non-batched synthesis on dataset scale intractably slow. To address this, we explored the use of MADGRAD [9] and Prodigy [31] as alternative optimizers that are batch-compatible, and produce synthesis comparable to L-BFGS, the de-facto gold standard for pixel-backpropagation based synthesis (Figure 9). L-BFGS uses 100 outer iterations of Strong-Wolfe-line-searched LBFGS (lr=0.5, max_iter=100, history=20, tolerance_grad=1e-7); MADGRAD uses 20k iterations at lr=0.07, momentum=0.99; Prodigy uses 20k iterations with auto-tuned step size (lr=0.04, d0=1e-2, betas=(0.99, 0.999), safeguard_warmup=True). All optimizers minimize MSE between target and synthesis Gram-statistic vectors with wrap or zero-padded mode, and the best-loss iterate is reported.

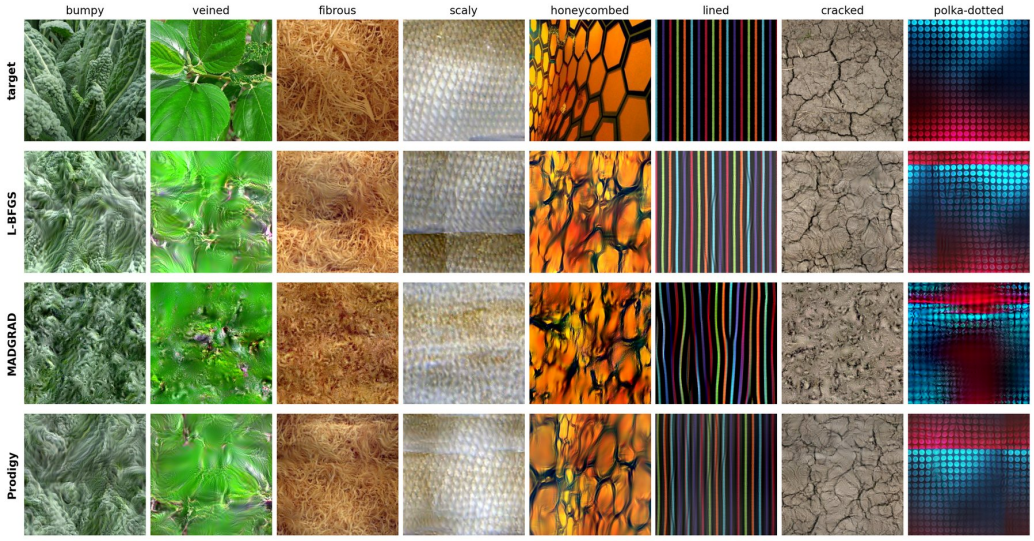


Figure 9: We find that Prodigy and Madgrad produce good quality synthesis like L-BFGS, and are batch compatible, saving significant time when synthesizing at scale, with Prodigy overall achieving slightly higher quality than Madgrad.

The choice of optimizer was determined by individual experiment requirements. For single-image qualitative comparisons, foveated peripheral synthesis, and video synthesis, we use L-BFGS, which converges efficiently on individual targets due to its second-order Hessian approximation. However, as discussed above, the Hessian computation does not separate cleanly across batched images, making L-BFGS intractable for large-scale evaluation. For experiments requiring batched synthesis across the validation set including texture variety, depletion experiments, and all quantitative results, we use Prodigy.

A.4 Computational Resources

For the image synthesis experiments, synthesizing a single condition for the the DTD-141 (141 subset) used two NVIDIA GeForce RTX 4090 GPUs, taking approximately 3-4 hours for the GSTN-wrap, and 6-8 hours for the GSTN-zpad condition with the Prodigy solver. Total estimated runtime for the image experiments is 1-2 weeks GPU time. For the video syntheses, synthesis ran on a single NVIDIA A-100 and took 1-2 hours for a single video. Foveated syntheses ran on a single NVIDIA A-100 and took 2-4 hours. Storage was <100GB for all datasets and experiments. Many additional preliminary and failed experiments, testing different optimizers, hyperparamters, resolutions, and pyramid settings were conducted than were reported in the paper, and while none individually used significant resources, altogether the use was not negligible.

A.5 Datasets

For the spatial texture model, we utilized the Describable Textures Database (DTD) [8], keeping the largest centered square and resizing the image to 256x256 for synthesis, and 5-crops for the contrastive learning. For training the contrastive learning module, we used the entire training subset of split 1, which samples all 47 texture types. For the evaluations of our syntheses and depletions, for computational tractability, we sampled 3 randomly chosen images from each category of the validation subset of validation split 1, for a total of 141 images (DTD-141). For computational reasons, we also evaluate [39], [18], and [51] on this DTD-141 subset.

For the video texture model, we utilized the Dynamic Texture Database (DynTex) [38]. For computational tractability, we synthesized a small subset of videos selected to vary in frequency, shape, and motion type, but do not evaluate on the entire dataset.

A.6 Full Statistics Image Synthesis

We compare quantitative performance of GSTN per texture category on a variety of metrics. A set of bias-neutral metrics (VIF, GMSD, HaarPSI, PSD-L2) was added to address the concern that local-moment metrics (SSIM, FSIM) are structurally aligned with GSTN's Gram-of-pyramid-bands representation while deep-feature metrics (DISTS, LPIPS, DreamSim) are aligned with Gatys' deep-VGG training objective. [45] is grounded in natural-scene statistics and Gaussian scale-mixture wavelet models; GMSD [53] compares the spatial deviation of local gradient-magnitude similarity; HaarPSI [40] uses Haar-wavelet feature similarity; PSD-L2 is the L2 distance between log power-spectral densities. None of these share architecture or training objective with GSTN or Gatys, but all four consistently favor GSTN-zpad in head-to-head per-category comparisons (Figures 10, 11), supporting the conclusion that GSTN-zpad's overall quality is not a metric-bias artifact.

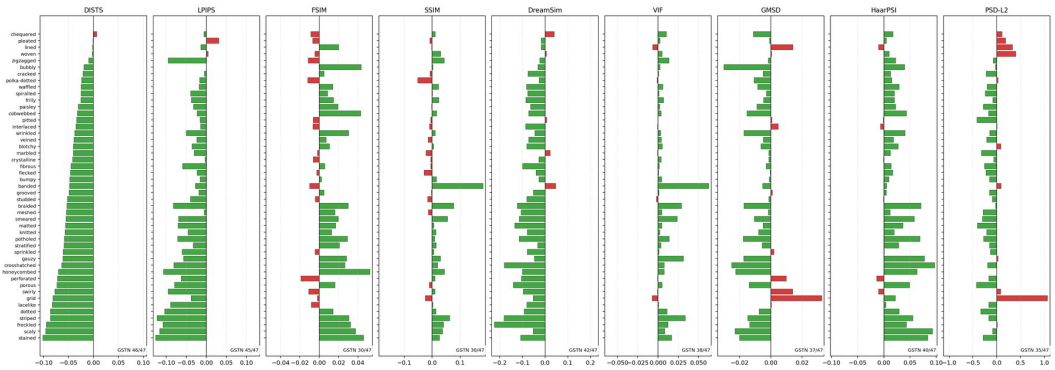


Figure 10: Per-category quantitative comparison of GSTN-zpad vs P&S across nine perceptual and structural metrics. Each bar shows the mean per-category metric difference (GSTN − P&S); green = GSTN wins, red = P&S wins. Categories sorted by DISTS-difference. Metrics span deep-network features (DISTS, LPIPS, DreamSim), local moments (FSIM, SSIM), and four bias-neutral metrics independent of either family (VIF, GMSD, HaarPSI, PSD-L2). DTD-141, 47 DTD categories with n=3 per category. GSTN-zpad wins the majority of categories on every metric (per-panel counts: DISTS 46/47, LPIPS 45/47, DreamSim 42/47, FSIM 30/47, SSIM 30/47, VIF 38/47, GMSD 37/47, HaarPSI 40/47, PSD-L2 35/47).

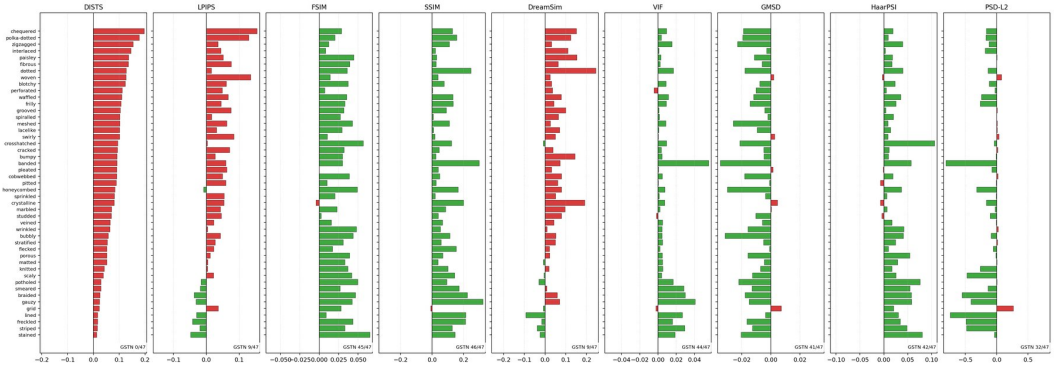


Figure 11: Per-category quantitative comparison of GSTN-zpad vs Gatys, same layout as Figure 10. DTD-141, 47 DTD categories with n=3 per category. Gatys wins on deep-network feature metrics (GSTN wins only DISTS 0/47, LPIPS 9/47, DreamSim 9/47) the metric family closest to Gatys' own VGG-feature training objective, and deep-learning based DreamSim, while GSTN-zpad wins on local-moment metrics (FSIM 45/47, SSIM 46/47) and on all four bias-neutral metrics (VIF 44/47, GMSD 41/47, HaarPSI 42/47, PSD-L2 32/47), despite using 30× fewer parameters (29K vs 825K)

Table 2: **Sensitivity to random initialization.** The 11 textures of Figure 2, each synthesized from 5 independent random initializations with GSTN-zpad,. Entries are mean ± s.d. across seeds.

Texture	DISTS ↓	LPIPS ↓	DrmSim ↓	FSIM ↑	SSIM ↑	VIF ↑	GMSD ↓	HaarPSI ↑	PSD-L2 ↓
pleated_0136	0.231 ± 0.005	0.332 ± 0.005	0.086 ± 0.018	0.639 ± 0.004	0.078 ± 0.019	0.032 ± 0.003	0.263 ± 0.001	0.434 ± 0.003	1.481 ± 0.009
cracked_0056	0.243 ± 0.008	0.587 ± 0.010	0.317 ± 0.024	0.615 ± 0.005	0.098 ± 0.005	0.007 ± 0.001	0.251 ± 0.004	0.374 ± 0.008	1.740 ± 0.006
striped_0050	0.246 ± 0.020	0.633 ± 0.099	0.215 ± 0.028	0.663 ± 0.032	0.171 ± 0.031	0.021 ± 0.005	0.240 ± 0.012	0.314 ± 0.033	1.386 ± 0.025
fibrous_0112	0.269 ± 0.008	0.524 ± 0.005	0.167 ± 0.002	0.698 ± 0.002	0.102 ± 0.008	0.009 ± 0.001	0.208 ± 0.001	0.439 ± 0.002	1.677 ± 0.003
bumpy_0077	0.283 ± 0.008	0.568 ± 0.012	0.282 ± 0.024	0.669 ± 0.010	0.061 ± 0.011	0.009 ± 0.002	0.231 ± 0.006	0.377 ± 0.013	1.709 ± 0.005
lacelike_0004	0.307 ± 0.005	0.513 ± 0.008	0.252 ± 0.009	0.620 ± 0.007	0.015 ± 0.008	0.005 ± 0.001	0.254 ± 0.004	0.382 ± 0.006	1.766 ± 0.002
sprinkled_0042	0.310 ± 0.012	0.667 ± 0.031	0.441 ± 0.022	0.601 ± 0.015	0.085 ± 0.027	0.017 ± 0.008	0.266 ± 0.008	0.288 ± 0.020	1.675 ± 0.011
honeycombed_0149	0.329 ± 0.008	0.647 ± 0.023	0.287 ± 0.007	0.554 ± 0.012	0.171 ± 0.013	0.026 ± 0.006	0.304 ± 0.006	0.244 ± 0.017	1.664 ± 0.009
swirly_0065	0.344 ± 0.004	0.475 ± 0.003	0.111 ± 0.006	0.605 ± 0.002	0.010 ± 0.006	0.003 ± 0.000	0.218 ± 0.001	0.436 ± 0.002	3.347 ± 0.007
chequered_0177	0.403 ± 0.011	0.682 ± 0.006	0.409 ± 0.008	0.567 ± 0.004	0.106 ± 0.016	0.014 ± 0.005	0.297 ± 0.002	0.252 ± 0.005	1.748 ± 0.019
polka-dotted_0140	0.454 ± 0.053	0.609 ± 0.119	0.144 ± 0.026	0.546 ± 0.041	0.169 ± 0.059	0.017 ± 0.013	0.238 ± 0.023	0.317 ± 0.066	2.456 ± 0.018

Table 3: **Efficiency Comparisons** All timings and memory are measured on a single idle NVIDIA RTX 4090 GPU (P&S on Apple Macbook Air M2 with MATLAB as the model/codebase is not GPU compatible). Reduced statistics sets for GSTN and Gatys represent ablations reported that exhibit little loss of quality compared to the full models. Iteration counts differ by method, so per-image time reflects optimization budget as well as per-step cost. All inference and per-image synthesis times reported are non-batched. With Prodigy, GSTN syntheses can be batched (36 images on a single 4090), which reduces effective per-image cost substantially. Batching is limited by the optimizer rather than the model; MADGRAD or Prodigy could in principle replace L-BFGS in Gatys and Ustyuzhaninov, but here we report the standard configurations.

Method	Statistic size (reduced) ↓	Inference ms/eval ↓	Synthesis / image (non-parallel) ↓	Peak GPU memory ↓	Parallelizable?
GSTN-zpad (Prodigy)	28,929 (10k)	13.9	4.6 min	0.37 GB	**yes (GPU batch)**
GSTN-wrap (Prodigy)	28,929 (10k)	13.3	4.4 min	**0.30** GB	**yes (GPU batch)**
GSTN-zpad (L-BFGS)	28,929 (10k)	17.9	3.7 min	0.53 GB	no (L-BFGS)
GSTN-wrap (L-BFGS)	28,929 (10k)	16.5	2.2 min	0.46 GB	no (L-BFGS)
P&S	8k	700	34.8 s	n/a (CPU)	**yes (MATLAB parpool)**
Ustyuzhaninov	525k	114.4	8.3 min	1.42 GB	no (L-BFGS)
Gatys (VGG-19)	852k (177k)	**6.9**	**6.9 s**	0.42 GB	no (L-BFGS)
muNCA (256×256)	**588**	686	91 min	7.73 GB	*yes* (TPU only)
muNCA (128×128, native)	**588**	526	70 min	1.98 GB	*yes* (TPU only)

A.7 Seed Sensitivity

We tested the sensitivity of our method to multiple seeds to evaluate the stability of results presented. (Table 2, Figure 12). We find that texture identity accounts for far more of the variation than seed, and this trend is consistent for all metrics

A.8 Efficiency Comparisons

We performed an efficiency evaluation across models, size, inference, and synthesis time 3. Overall, GSTN shows balanced efficiency compared to other methods, with the lowest GPU memory load, and with Prodigy optimizer can be highly parallelized on a single GPU to decrease effective per-image synthesis time. While Gatys dominates in speed, its representation size (number of statistics) is very large in comparison. And while muNCA is by far the smallest model in terms of number of statistics, as a cellular automaton it unrolls over many steps at full resolution, leading to a huge effective memory footprint, and its synthesis time is by far the worst (though the synthesis time may perform better and be parallelizable natively on a TPU).

A.9 Resolution Scaling

The full non-depleted GSTN model uses a fixed budget of 28,929 statistics to represent images, regardless of resolution (Figure 13), resulting in a varying amount of compression. Thus, the relative size efficiency of GSTN improves as image resolution increases, but at the tradeoff of synthesis quality. This effect is visible in the tiled syntheses of Figure 7 comparing the center (blue, smaller region) and edge (red, larger region) cutouts. We tested the effect of differing image size on synthesis quality and wall clock given this fixed representational budget (Figure 14, Table 4). We find that as expected, synthesis quality decreases with increased image size and increased ratio to representation size. This varying relationship mirrors the increase in pooling region size in peripheral syntheses and

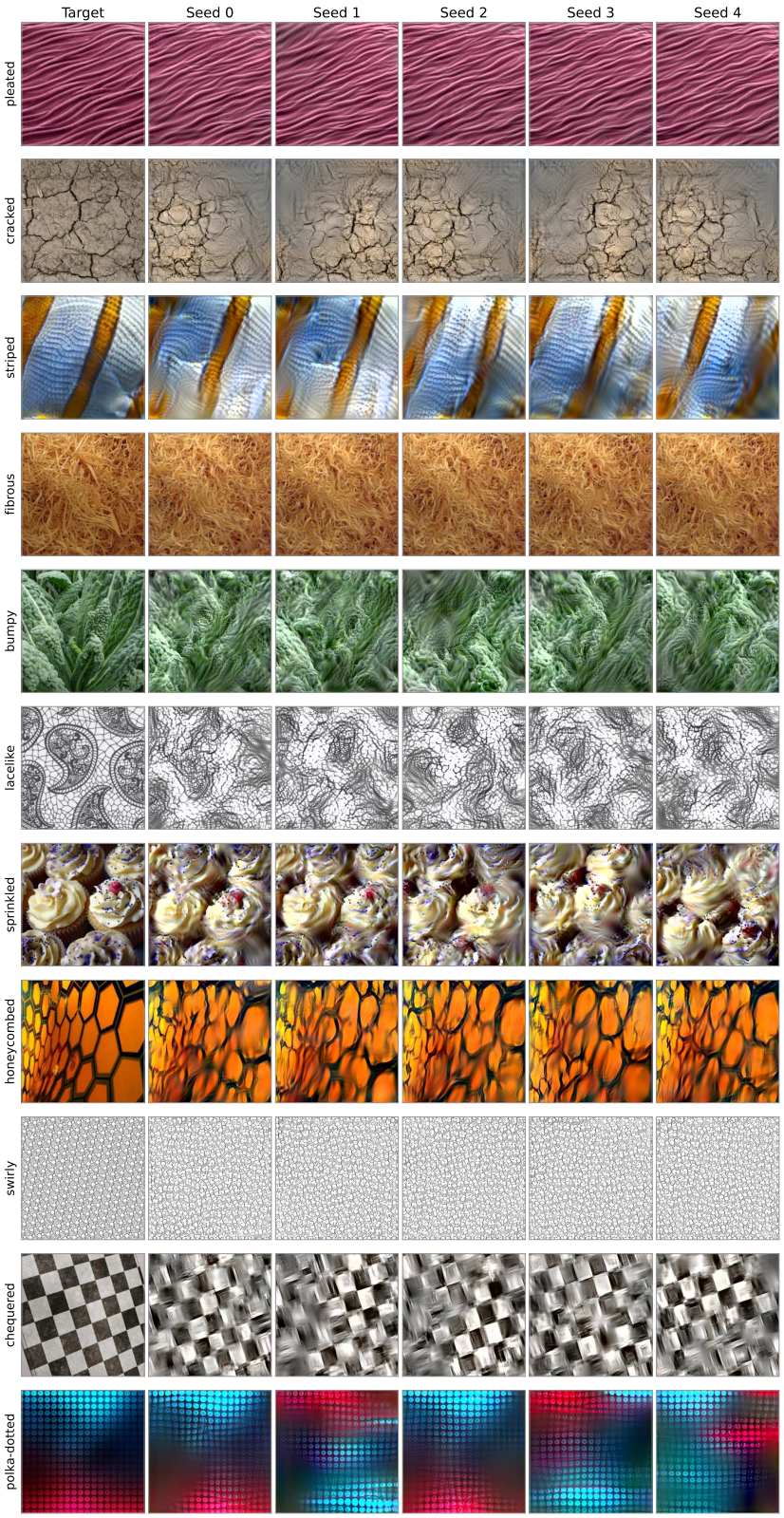


Figure 12: **Synthesis is stable across random initializations.** The 11 textures of Figure 2, each synthesized by GSTN-zpad from five independent random initializations, shown against the target.

serves as a tractable but controlled and principled qualitative proxy for the eccentricity-dependent degradation shown in Figure 7. Likewise, synthesis time increases with target image size, although this relationship is sub-linear with respect to number of pixels, with a 512x512 pixel color image taking less than 7x the synthesis time of a 16x16 pixel color image.

Table 4: **Scaling of compression, quality and wall-clock with image resolution.** Six DTD textures (`bubbly_0064`, `dotted_0078`, `grooved_0054`, `polka-dotted_0140`, `scaly_0168`, `swirly_0065`) × 3 seeds (18 syntheses per row), 20,000 iterations, GSTN-zpad. Metrics are computed with synthesis and target both at the synthesis resolution. Representation size is fixed at 28,929 statistics at every resolution, so compression ("values per statistic") grows with the square of the side length and crosses unity at $\sim 98 \times 98$ px. Wall-clock scales strongly sublinearly: $1024\times$ the pixels costs $7\times$ the time. Resolutions above 512×512 are untestable because DTD contains no 1024×1024 images. *Below 64 px the VGG-based metrics (DISTS, LPIPS) operate outside their validated regime, and VIF is undefined.

Resolution	Image size ($3N^2$)	Values per stat.	DISTS ↓	LPIPS ↓	FSIM ↑	GMSD ↓	HaarPSI ↑	VIF ↑	Wall-clock / synthesis
$16\times16^*$	768	$0.027\times$	0.101	0.093	0.851	0.101	0.747	n/a	96 s
$32\times32^*$	3,072	$0.106\times$	0.165	0.213	0.770	0.136	0.627	n/a	103 s
64×64	12,288	$0.425\times$	0.212	0.322	0.719	0.199	0.526	0.049	154 s
128×128	49,152	$1.699\times$	0.305	0.483	0.629	0.232	0.401	0.021	248 s
256×256	196,608	$6.796\times$	0.356	0.578	0.581	0.253	0.328	0.015	313 s
512×512	786,432	$27.185\times$	0.321	0.603	0.579	0.282	0.270	0.016	670 s

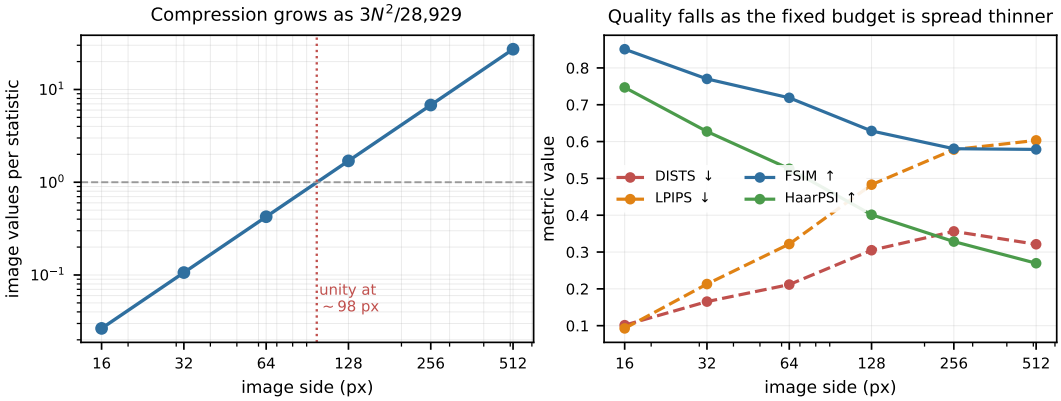


Figure 13: **Compression and quality as a function of synthesis resolution.** *Left:* representation size is constant at 28,929 statistics, so compression grows as $3N^2/28{,}929$ and GSTN becomes a lossy code above $\sim$98 px per side (dashed line) and an expansion below it. *Right:* the same sweep scored against the target at the synthesis resolution. FSIM, HaarPSI and LPIPS degrade monotonically as the budget is spread over larger images; DISTS peaks at 256×256 and improves slightly at 512×512. Below 64 px DISTS and LPIPS are outside their validated regime and are plotted for completeness.

A.10 Contrastive Learning

For contrastive learning, we trained a single fully connected linear layer of size (28,929 x 100), reducing the size of our representation space by nearly 300 times. We used DTD train split 1, taking 5-crops (4 corners plus center) of the image of size 128x128 with random vertical and horizontal flips, and feeding these through the analysis pipeline, extracting the upper triangle of the Gramian matrix for 28,929 statistics, and training the network in batches of 50 textures (200 crops total). We trained the network for 100 epochs with the Normalized Temperature-Scaled Cross-Entropy (NT-Xent) loss from pytorch metric learning, defined as:

$$\mathcal{L}_i = -\log \frac{\exp(\mathrm{sim}(z_i, z_{j(i)})/\tau)}{\sum_{k \neq i} \exp(\mathrm{sim}(z_i, z_k)/\tau)} \tag{1}$$

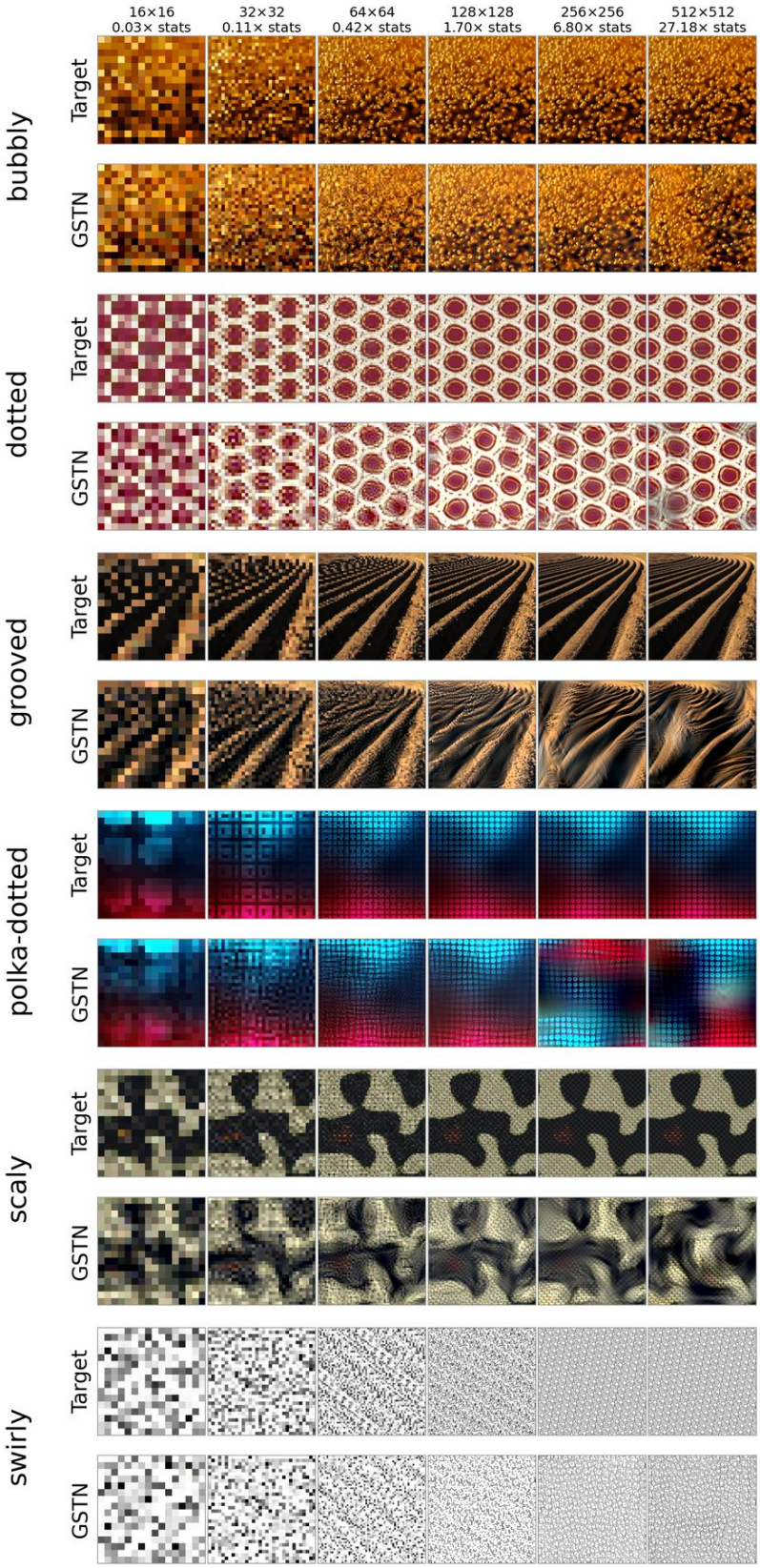


Figure 14: **Synthesis quality with variable target resolution:** Because images of different target sizes are encoded with the same fixed representational budget, quality of match to target degrades while region size grows with eccentricity. Like synthesis quality generally, the magnitude of this degradation is very texture dependent.

Conceptually, this loss encourages the learned representation (100 dimensions) to represent textures from the same parent texture similarly, and textures from different parent textures differently. Figure 15 shows the learned embedding for training and validation textures. Crops drawn from the same parent texture cluster tightly in the 100-dimensional compressed space, while crops from different parent textures are well separated, both on training data and on the held-out validation split, indicating that the learned compression preserves texture identity rather than memorizing parent-image features.

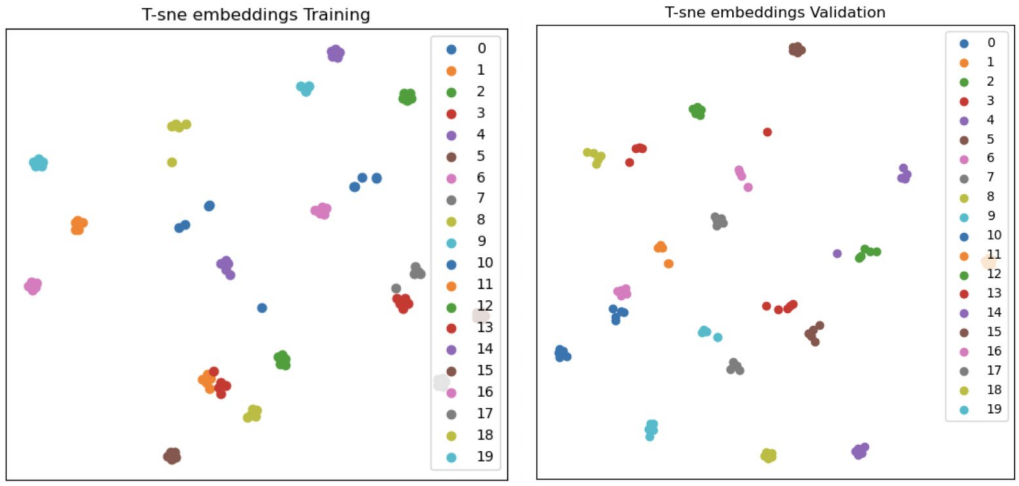


Figure 15: Contrastive learning successfully groups texture images cropped from the same parent image (same color), and separates images from different parent images. Training set (left). Validation set (right)

A.11 Contribution of Statistics by Contrastive Learning

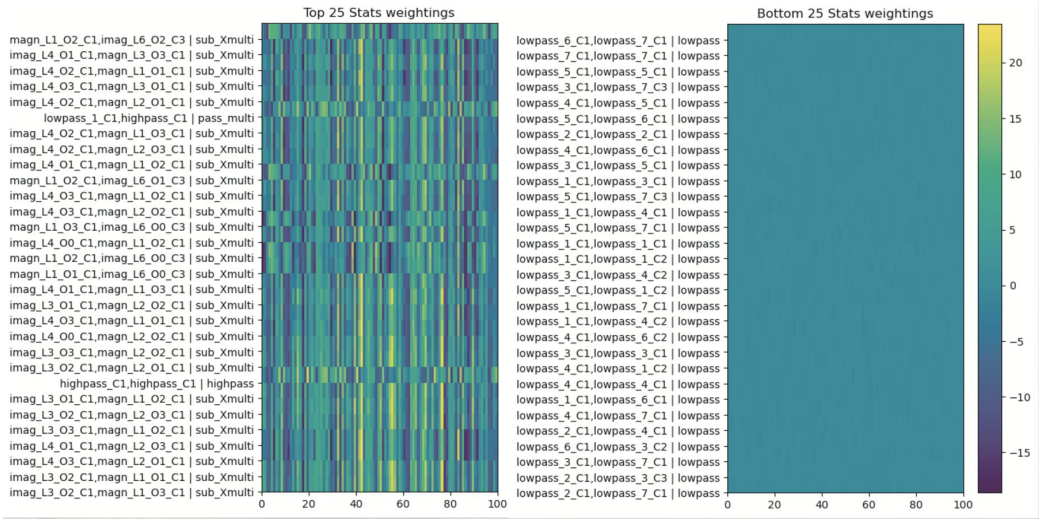


Figure 16: We use the weight matrix of the compression layer from the contrastive learning module to order correlation statistics by their contribution to the compressed representation. Learned weights of the single fully connected layer show high weightings for unstructured statistics, and lowest weightings for lowpass filter statistics.

Once the contrastive compression layer is trained, the weight matrix can be interpreted as an importance map over individual statistics: each column corresponds to one Gram-matrix statistic, and the magnitude of that column reflects how strongly the statistic contributes to the learned 100-dimensional

representation. We compute a per-statistic importance score as the L_1 norm of the corresponding column, and rank all 28,929 statistics in descending order by this score. Figure 16 visualizes this ranking grouped by statistical family, showing that unstructured cross-correlation families (`sub Xmulti`, `pass multi`) receive the highest weights on average, while lowpass-filter statistics receive the lowest. Figure 17 further breaks this down by reporting the percent contribution of each family to the top-N most important statistics for varying N, showing that the top-ranked statistics are dominated by a small number of large unstructured families.

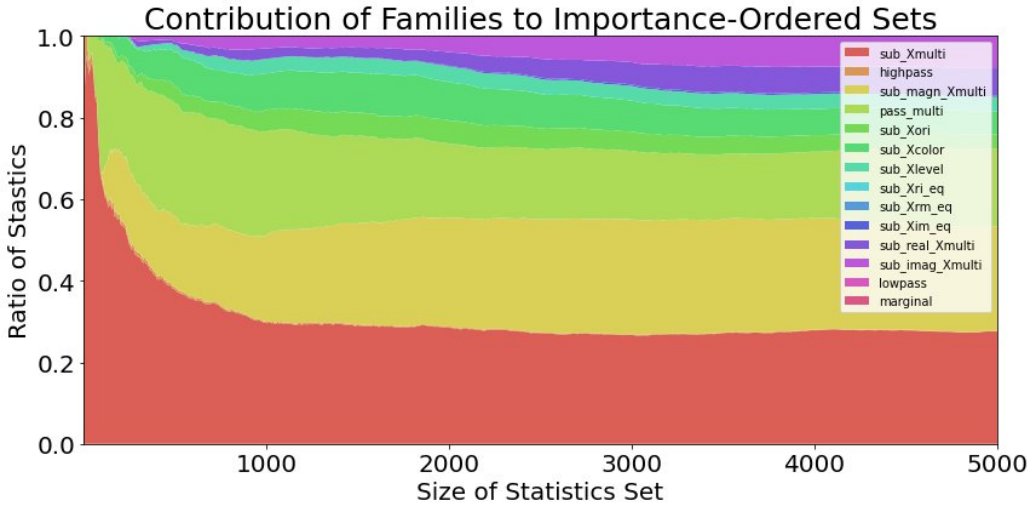


Figure 17: Using the contrastively learned importance ordering, we visualize the percent contribution of individual families to the total set, for varying statistics set size. Families sub_Xmulti, highpass, and pass_multi contribute most when selecting for the most important statistics (left).

A.12 Depleted Syntheses Control

To verify that the contrastively-learned importance ranking carries information beyond what is available from set size alone, we ran a random-depletion control. For each set size (100,1000,2500,5000,10000), we generated synthesis using both (a) the top-N statistics ranked by contrastive importance and (b) a randomly-sampled subset of N statistics.

A.13 Spatial Pyramid Statistics Families

A benefit of our approach is that as a Gram-matrix element, each statistic represents a correlation between a pair of known filter responses, making the full set of 28,929 statistics directly interpretable. To organize this set, we group individual statistics into families based on the relationship between the two filter images being correlated. Filter images differ along several attributes: the pyramid type (subband edge image vs. highpass vs. lowpass), the pyramid level (spatial scale), the orientation, the color channel, and the real/imaginary/magnitude phase. A given pair-wise correlation can hold any subset of these attributes fixed and vary the rest. Table 1 groups all 28,929 spatial statistics by which attributes are shared and which differ between the two filter images in each correlation. We distinguish two top-level groupings. Non-subband pyramid statistics involve at least one filter that is not a subband edge image — these include marginal pixel statistics (mean, variance, standard deviation), highpass-only correlations, lowpass-only correlations, and mixed pass-band correlations between highpass, lowpass, and subband filters. Subband pyramid statistics are correlations between two subband edge images, further partitioned by which subband attribute differs: orientation only (`sub Xori`), color only (`sub Xcolor`), level only (`sub Xlevel`), or various combinations involving real, imaginary, and magnitude components. The largest family by count is `sub Xmulti` (15,336 statistics), which contains all subband-to-subband correlations in which the two filter images differ along multiple attributes simultaneously (pyramid type, level, color, and orientation). Smaller families such as `sub Xori` (324 statistics) and `sub Xri eq` (72 statistics) capture more constrained relationships in which only a single attribute varies between the two filters being correlated. This

taxonomy supports the family-based ablation experiments and provides the grouping used to interpret the contrastively-learned importance weights.

A.14 Spatiotemporal Filter Equations

In the Fourier domain, our filter banks take the form of anisometric Gaussian functions centered on (μ_x, μ_y, μ_t) with variances $(\sigma_x^2, \sigma_y^2, \sigma_t^2)$

$$\tilde{f}(\mathbf{k}) \propto \exp\left[-\frac{1}{2}\left(\left(\frac{k_x - \mu_x}{\sigma_x}\right)^2 + \left(\frac{k_y - \mu_y}{\sigma_y}\right)^2 + \left(\frac{k_t - \mu_t}{\sigma_t}\right)^2 \right)\right]. \tag{2}$$

The filter positions are determined by the spatial scale (ρ), temporal scale (ζ), and orientation (ϕ). These are defined in cylindrical coordinates as:

$$\mu_x = \rho \cdot \cos(\phi) \tag{3}$$
$$\mu_y = \rho \cdot \sin(\phi) \tag{4}$$
$$\mu_t = \zeta \tag{5}$$

Finally, the standard deviations σ_i scale with the centers via spatial and temporal scaling factors $\alpha_S = 0.4$ and $\alpha_T = 0.7$ such that:

$$\sigma_x = \alpha_S \cdot \mu_x \tag{6}$$
$$\sigma_y = \alpha_S \cdot \mu_y \tag{7}$$
$$\sigma_t = \alpha_T \cdot \mu_t \tag{8}$$

For our filter bank, we sample the spatial scales, temporal scales, and orientations over the ranges:

$$\rho \in \{1, 2, 4, 8, 16\} \tag{9}$$
$$\zeta \in \{1, 2, 4, 8, 16\} \tag{10}$$
$$\phi \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}\} \tag{11}$$

We synthesize videos by matching on the Gramian of these filters. Unlike the spatial case, we find that for video textures, histogram matching over the whole image is needed in addition to statistics matching to achieve good syntheses.

A.15 Spatiotemporal Pyramid Filters

The first step for expansion of our model into the temporal domain is the design of a human-vision informed spatiotemporal filter bank. We extended the classical multi-scale spatial pyramid [21] into the time domain with a set of spatiotemporal filters centered around the spatiotemporal frequency space of human visual sensitivity (see Appendix). A subset of these filters in the Fourier domain with boundaries at half maximum amplitude is shown in Figure 18 (top). See A.14 for the mathematical parameterization for these spatiotemporal filters. When the inverse Fourier transform of a quadrature pair of frequency defined filters is taken, both a real and imaginary component is generated. These spatiotemporal filter pairs can be visualized in the pixel-frame space, enabling intuition for the type of motion a filter pair selects for. A sample of these filter pairs is shown in Figure 18, (bottom).

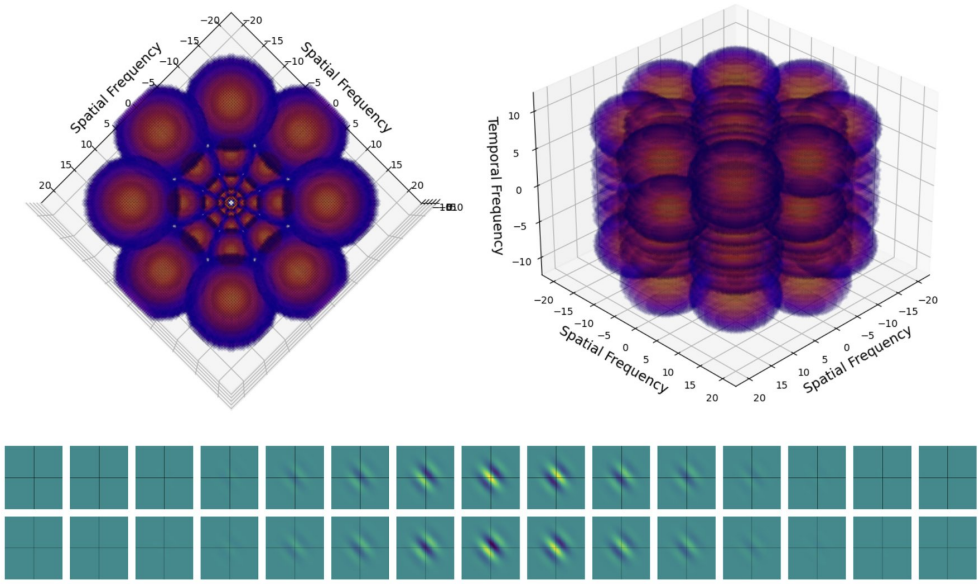


Figure 18: We design a filter bank of spatiotemporal Gabor filters that tile the spatiotemporal frequency space. Viewed in Fourier space (top), filters are arranged in concentric rings that are stacked vertically, selecting for 8 orientations at increasing spatial and temporal frequencies. Each filter in the positive temporal frequency space has a quadrature pair mirrored across a line through the origin. Left: top view, Right: side view. Bottom: A real+imaginary pair of spatiotemporal filters, generated by inverse Fourier transform of the spatiotemporal frequency defined filters above. Left to right denotes the temporal axis, with frames sub-sampled for visualization. This pair of filters is sensitive to low spatial and low temporal frequency motion to the upper-right.

A.16 Spatiotemporal Statistics Families

The benefit of interpretability of our method as pairwise correlation between filters is true for both the spatial-only filters as well as the spatiotemporal filters. For the spatiotemporal filters, given the much larger number of filters, we use a slightly different organizational approach, separating them by each parameter that varies. Table 5 organizes these statistics into families based on the relationship between the two filter images being correlated. We take specific note of the groups denoted by a single letter (C,O,S,T,P), as these correspond to correlations between filters that vary in only one attribute. These statistics are analogous to the hand-selected spatial statistics in previous pyramid-based texture and peripheral vision models including [39, 15, 2]. In addition, spatial texture investigations have shown that low-pass filter correlation statistics (LPS) are less important to successful synthesis, and correlation statistics across many filter attributes (such as COSTP) are most important [14].

A.17 Video Syntheses

In addition, we evaluate the quantiative performance on a variety of metrics, comparing GSTN video model to DyNCA, a recent model for video synthesis (Table 6). We find that DyNCA outperforms the GSTN video model on some metrics, with the exception of the difference of the temporal power spectrum from the original target video. We emphasize here that DyNCA is a fundamentally different type of model than GSTN with differing tradeoffs, optimized for synthesis quality and model size, and with a per-image/texture trained parameterization, that lacks interpretability and biological/perceptual connection.

Table 5: Spatiotemporal correlation statistics grouped into families based on which filter attributes are cross-correlated (*Cross*) versus held fixed (*Same*). The five attributes are color channel, motion direction (orientation), spatial frequency, temporal frequency, and phase (real/imaginary). Special families (top): AUTO contains filter autocorrelations on the Gram diagonal; LPS/LPT contain correlations involving spatial / temporal low-pass filters. Remaining families are partitioned by the number of attributes that vary, from one (e.g., C, O) to all five (COSTP).

Group	**Cross (varies)**	**Same (fixed)**	**#Stats**
Special families			
AUTO	Auto-correlations (Gram diagonal)		1,212
LPS	Spatial low-pass correlations		7,233
LPT	Temporal low-pass correlations		7,233
One attribute varies			
C	color	orientation, spatial, temporal, phase	1,200
O	orientation	color, spatial, temporal, phase	4,200
S	spatial	color, orientation, temporal, phase	2,400
T	temporal	color, orientation, spatial, phase	2,400
P	phase	color, orientation, spatial, temporal	600
Two attributes vary			
CO	color, orientation	spatial, temporal, phase	8,400
CS	color, spatial	orientation, temporal, phase	4,800
CT	color, temporal	orientation, spatial, phase	4,800
CP	color, phase	orientation, spatial, temporal	1,200
OS	orientation, spatial	color, temporal, phase	16,800
OT	orientation, temporal	color, spatial, phase	16,800
OP	orientation, phase	color, spatial, temporal	4,200
ST	spatial, temporal	color, orientation, phase	9,600
SP	spatial, phase	color, orientation, temporal	2,400
TP	temporal, phase	color, orientation, spatial	2,400
Three attributes vary			
COS	color, orientation, spatial	temporal, phase	33,600
COT	color, orientation, temporal	spatial, phase	33,600
COP	color, orientation, phase	spatial, temporal	8,400
CST	color, spatial, temporal	orientation, phase	19,200
CSP	color, spatial, phase	orientation, temporal	4,800
CTP	color, temporal, phase	orientation, spatial	4,800
OST	orientation, spatial, temporal	color, phase	67,200
OSP	orientation, spatial, phase	color, temporal	16,800
OTP	orientation, temporal, phase	color, spatial	16,800
STP	spatial, temporal, phase	color, orientation	9,600
Four attributes vary			
COST	color, orientation, spatial, temporal	phase	134,400
COSP	color, orientation, spatial, phase	temporal	33,600
COTP	color, orientation, temporal, phase	spatial	33,600
CSTP	color, spatial, temporal, phase	orientation	19,200
OSTP	orientation, spatial, temporal, phase	color	67,200
All five attributes vary			
COSTP	color, orientation, spatial, temporal, phase	—	134,400
Total			**735,078**

Table 6: GSTN vs. DyNCA [37] per video on five DynTex dynamic textures. Metrics are computed with identical code for both methods. Bold = better of the two (both when tied). LP-A/LP-V = LPIPS (AlexNet/VGG); Spec = L1 distance between normalized temporal power spectra; $|\Delta\mathrm{Div}|$ and $|\Delta\mathrm{Curv}|$ = absolute distance from the target's own frame diversity and perceptual-trajectory curvature [22]. SSIM, MSE and PSNR are alignment-based and weak for stochastic syntheses; per-video FID is noisy at small n (read the mean). GSTN uses histogram projection every 50 evaluations, 3,000 evaluations.

Video	Method	LP-A↓	LP-V↓	DISTS↓	SSIM↑	FID↓	CMMD↓	MSE↓	PSNR↑	Spec↓	$\lvert\Delta\mathrm{Div}\rvert$↓	$\lvert\Delta\mathrm{Curv}\rvert$↓
grass	GSTN	0.287	0.463	0.318	0.016	160	2.030	**0.045**	**13.4**	**0.196**	0.057	**8.7**
	DyNCA	**0.124**	**0.281**	**0.205**	**0.054**	**75**	**0.664**	0.046	**13.4**	0.786	**0.013**	20.2
fire	GSTN	0.557	0.602	0.347	**0.398**	120	2.156	0.014	18.7	**0.152**	0.117	8.2
	DyNCA	**0.140**	**0.334**	**0.182**	0.356	**31**	**1.696**	**0.013**	**18.8**	0.902	**0.090**	**3.9**
fence	GSTN	0.265	0.548	0.360	**0.047**	378	3.737	0.054	12.7	**0.253**	0.043	**18.1**
	DyNCA	**0.152**	**0.309**	**0.187**	0.044	**174**	**1.609**	**0.052**	**12.8**	0.453	**0.034**	21.0
blooms	GSTN	0.302	0.493	0.301	**0.049**	89	2.781	0.073	11.4	**0.188**	0.084	22.6
	DyNCA	**0.206**	**0.367**	**0.188**	0.024	**79**	**1.954**	**0.061**	**12.1**	0.193	**0.035**	**14.0**
tree	GSTN	0.202	0.467	0.295	**0.061**	64	1.501	**0.032**	**15.0**	**0.094**	0.025	**4.7**
	DyNCA	**0.174**	**0.321**	**0.234**	0.057	**54**	**0.770**	0.046	13.4	0.254	**0.012**	29.8
mean	GSTN	0.323	0.515	0.324	**0.114**	162	2.441	**0.044**	**14.2**	**0.176**	0.065	**12.5**
	DyNCA	**0.159**	**0.323**	**0.199**	0.107	**83**	**1.339**	**0.044**	14.1	0.518	**0.037**	17.8

A.18 Analytical Solution for Contrastive Learning

Let i denote the ith parent texture images, and $n(i)$ denotes the set of texture images cropped from ith parent texture images. Let $a(x)$ denote the texture statistic of image x, and let P be the weight of the fully connected layer of the contrastive learning. The contrastive learning problem can be formulated as:

$$\min_{P} \sum_{i} \sum_{j,k \in n(i)} ||Pa(x_j) - Pa(x_k)||_F^2 \tag{12}$$

such that

$$P\mathbb{E}_x[a(x)a(x)^T]P^T = I \tag{13}$$

This optimization problem can be written in the following matrix form:

$$\min_{P} ||PAD||_F^2 \quad s.t. \quad PVP^T = I \tag{14}$$

Where $V = \frac{1}{N}$, N is the total number of used images crops. V is the covariance matrix of statistics. A contains the statistic of all possible crops. Each row of A is the $a(x_i)$ for some i. D is analogous to the first order derivative operator defined on a graph. Specifically, $D = [d_1, d_2, \cdots, d_M]$, where M denotes the number of pair of crops in the training set. For ith pair of crops x_k and x_j, we have $d_{ik} = 1$ and $d_{ij} = -1$. All other entries of D are zeros.

The solution to this generalized eigen-decomposition problem is given by $P = UV^{-\frac{1}{2}}$, where U is a matrix of L trailing eigenvectors (i.e. eigenvectors with the smallest eigenvalues) of the matrix $Q = V^{-\frac{1}{2}}ADD^TA^TV^{-\frac{1}{2}}$ [7].

This means solving the contrastive learning problem requires calculating Q, which requires calculating the matrix ADD^TA^T and $V^{-\frac{1}{2}}$. The latter can be easily computed by diagonalizing A. To compute ADD^TA^T on the other hand is generally computational expensive. Luckily, since D is highly sparse, we can efficiently calculate matrix ADD^TA^T as the following:

$$Q = 2(\sum_{i} |n(i)| \sum_{k \in n(i)} a(x_k)a(x_k)^T - \bar{a_i}\bar{a_i}^T) \tag{15}$$

where $\bar{a_i} = \sum_{j \in n(i)} a(x_j)$. Let a_i denote $a(x_i)$ for convenience. The derivation is the following:

$$
\begin{aligned}
Q &= ADD^TA^T \\
&= \sum_i \sum_{j,k\in n(i)} (a_k - a_j)(a_k - a_j)^T \\
&= \sum_i \sum_{j\in n(i)} \sum_{k\in n(i)} a_k a_k^T + a_j a_j^T - a_k a_j^T - a_j a_k^T \\
&= \sum_i \sum_{j\in n(i)} (\sum_{k\in n(i)} a_k a_k^T + |n(i)| a_j a_j^T - \\
&\quad (\sum_{k\in n(i)} a_k) a_j^T - a_j (\sum_{k\in n(i)} a_k)^T) \\
&= \sum_i (|n(i)| \sum_{k\in n(i)} a_k a_k^T + |n(i)| \sum_{j\in n(i)} a_j a_j^T - \\
&\quad (\sum_{k\in n(i)} a_k)(\sum_{j\in n(i)} a_j)^T - \\
&\quad (\sum_{j\in n(i)} a_j)(\sum_{k\in n(i)} a_k)^T) \\
&= \sum_i 2|n(i)| \sum_{k\in n(i)} a_k a_k^T - 2(\sum_{k\in n(i)} a_k)(\sum_{j\in n(i)} a_j)^T \\
&= 2(\sum_i |n(i)| \sum_{k\in n(i)} a_k a_k^T - \bar{a_i}\bar{a_i}^T)
\end{aligned}
$$

A.19 LLM Use

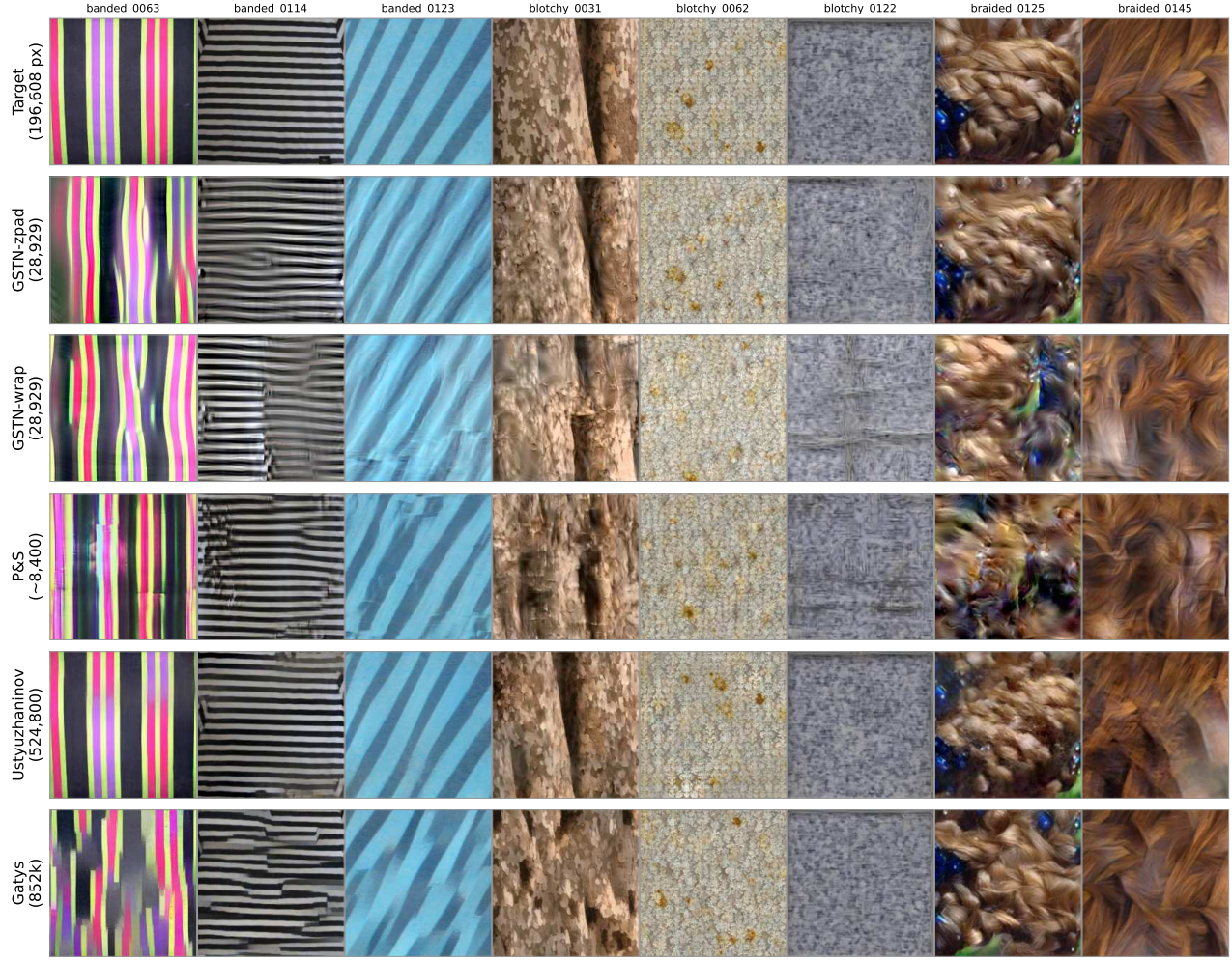
The GSTN model was created without the use of LLMs. Generation of the code for foveated and video syntheses were done without the use of LLMs. Code to run, manage, and queue parameter sweeps and depletion runs on the GPU was generated with an LLM. Code to aggregate synthesis images, calculate quantitative metrics, and plot summary figures was generated with an LLM.

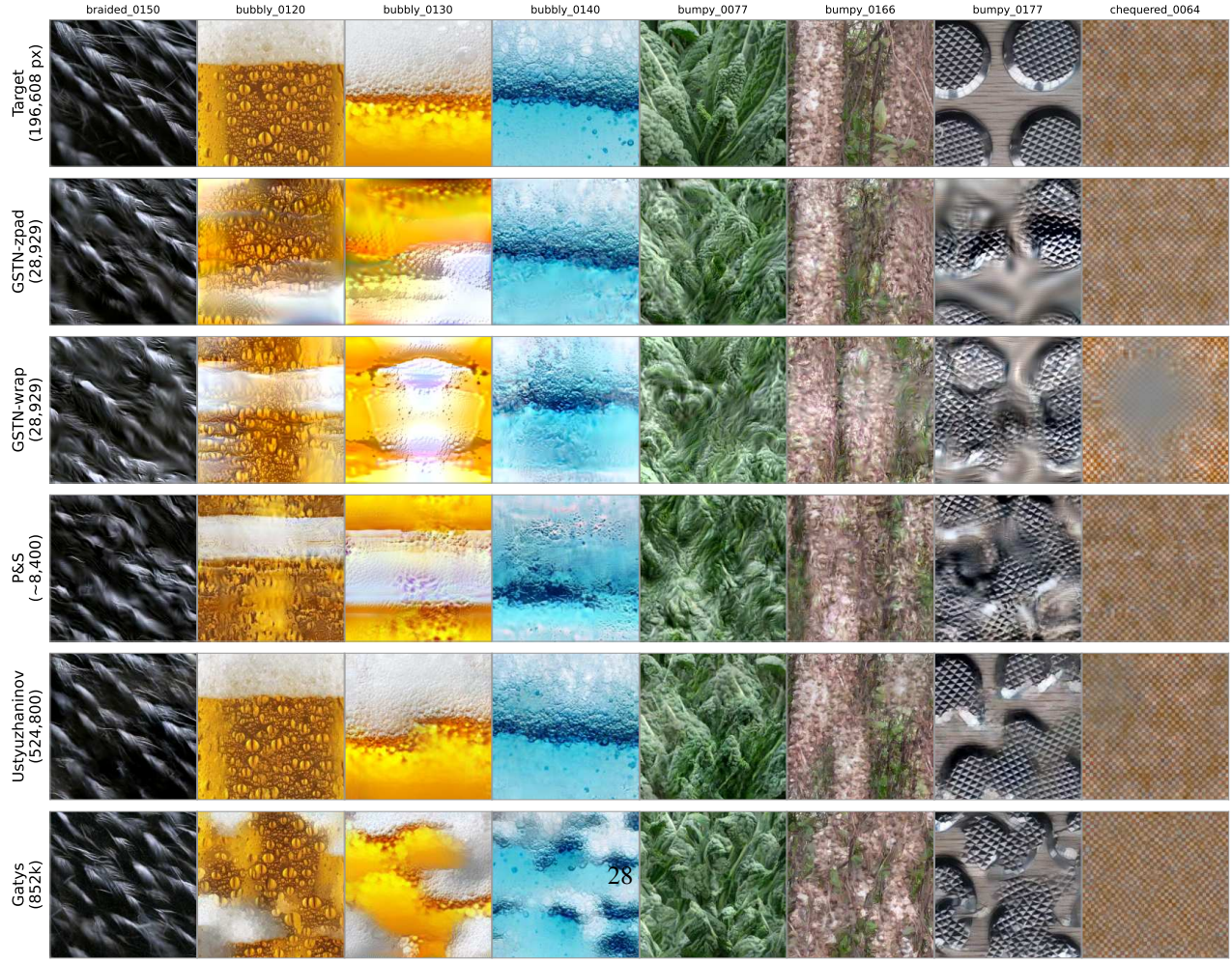
B Full qualitative comparison on DTD-141

For completeness we include every image in the evaluation set below: all 141 images (47 categories, $n = 3$ each). This is the complete subset of the DTD dataset used in quantitative comparisons throughout the manuscript except where otherwise specified.

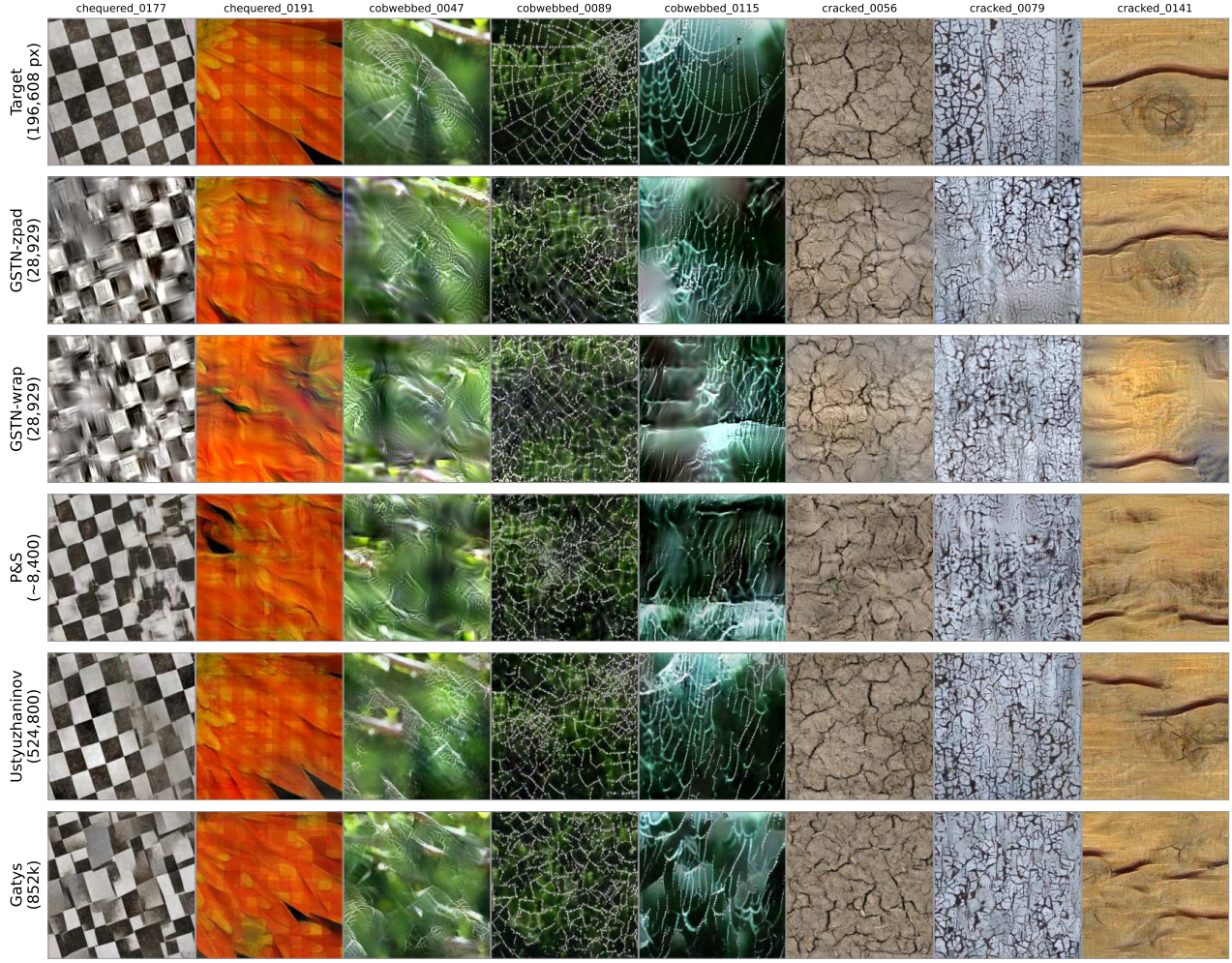
Figure 19: **Full qualitative comparison over the DTD-141 evaluation set** (next 9 pages).

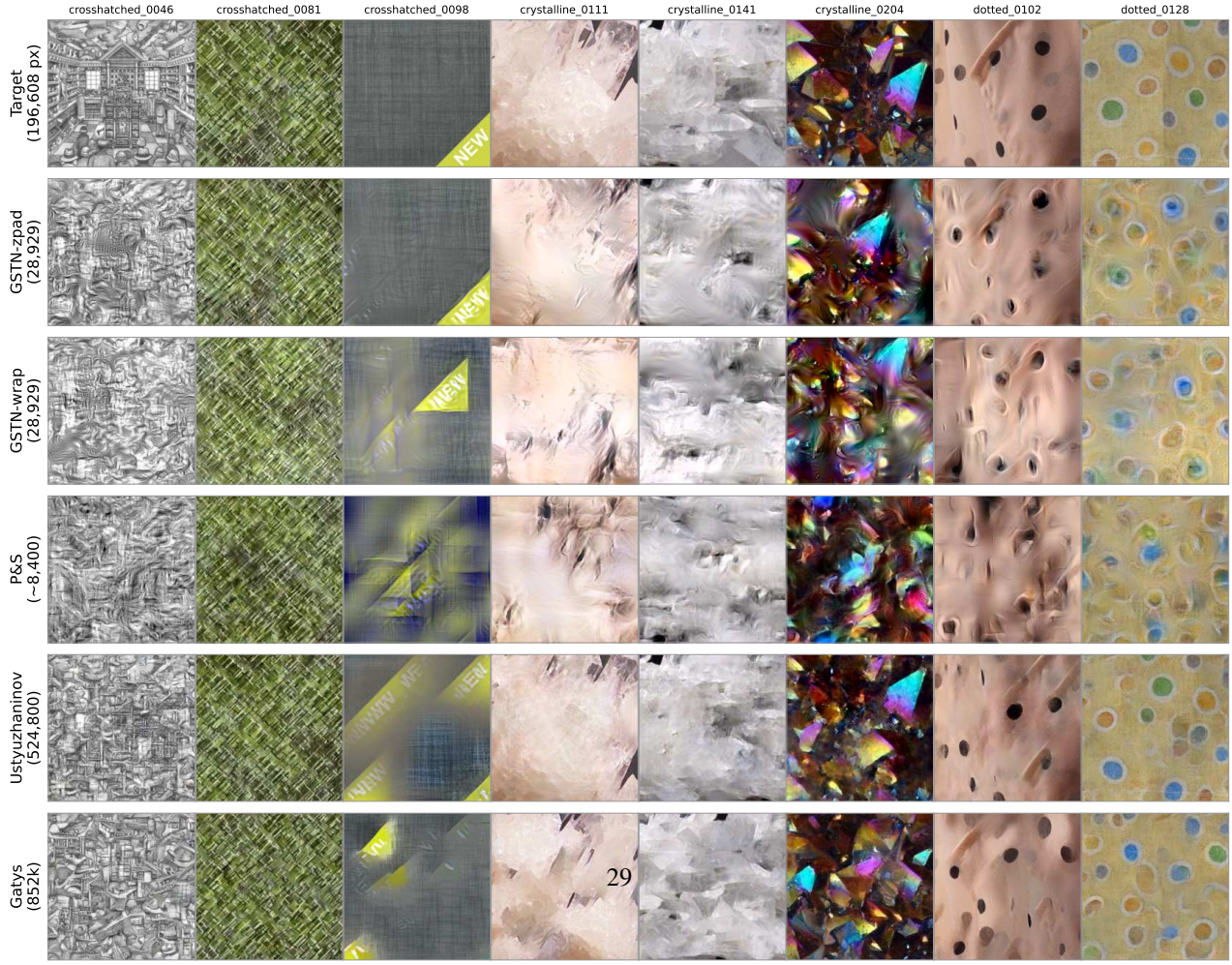
DTD-141 full qualitative comparison — page 1 of 9



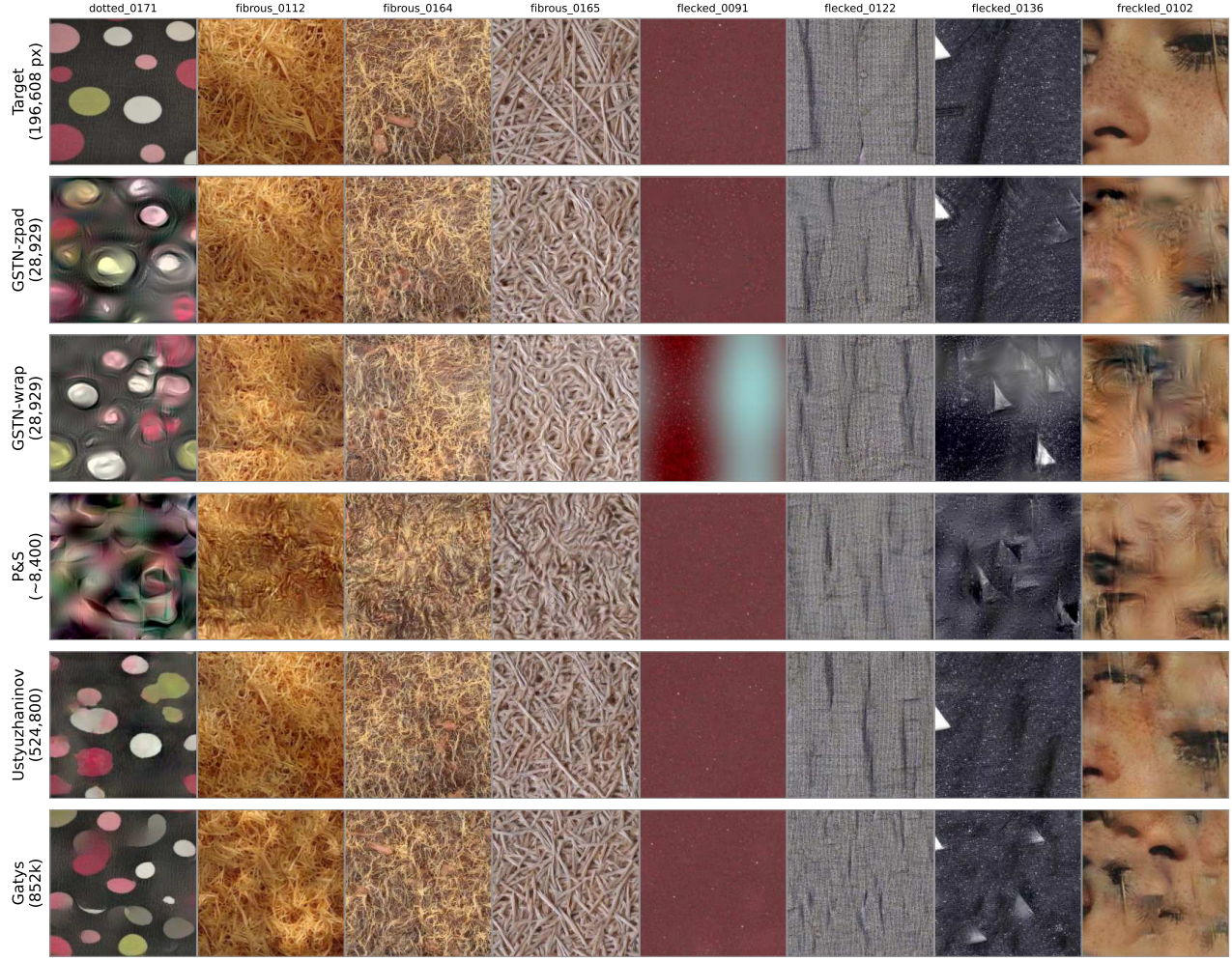


DTD-141 full qualitative comparison — page 2 of 9

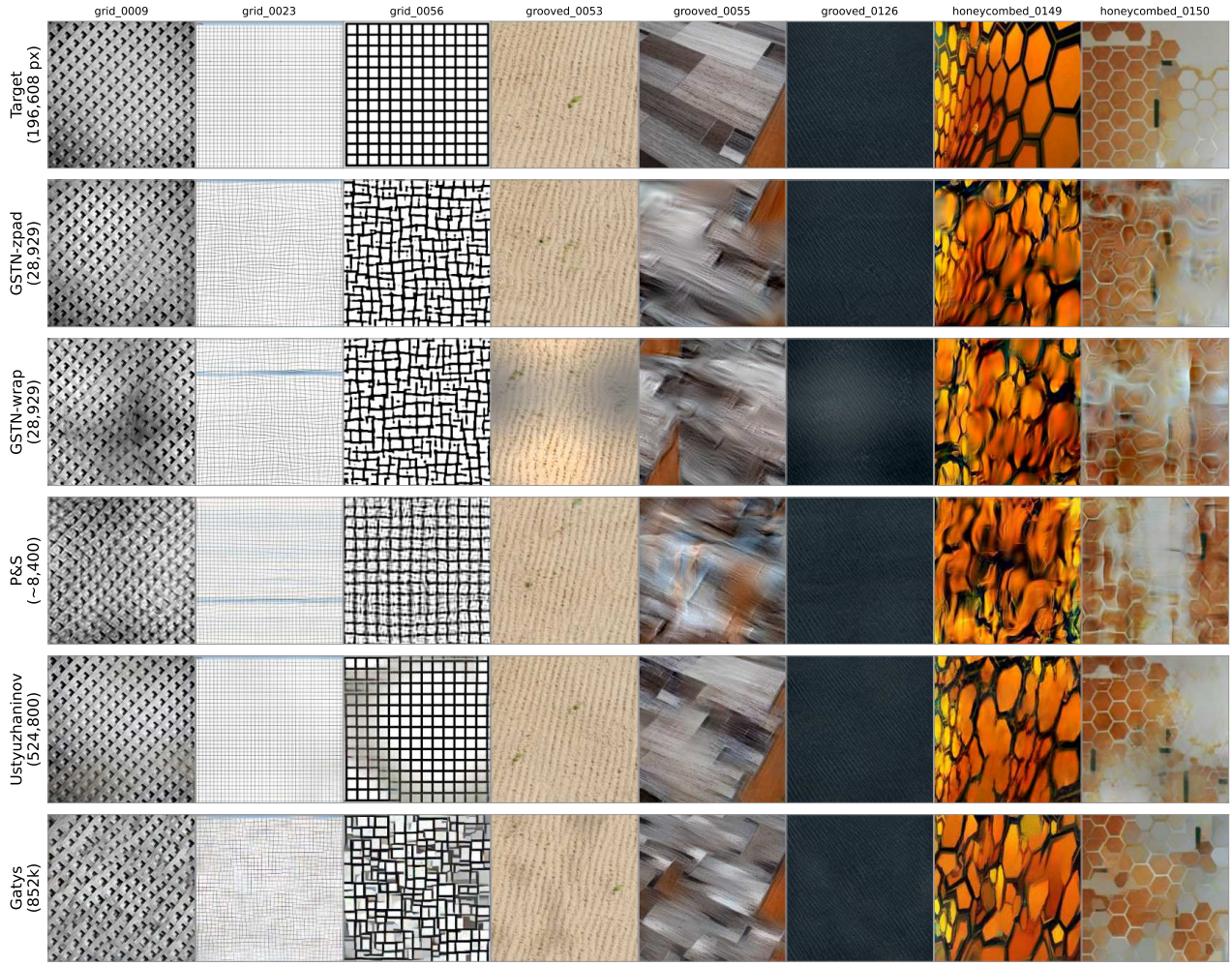


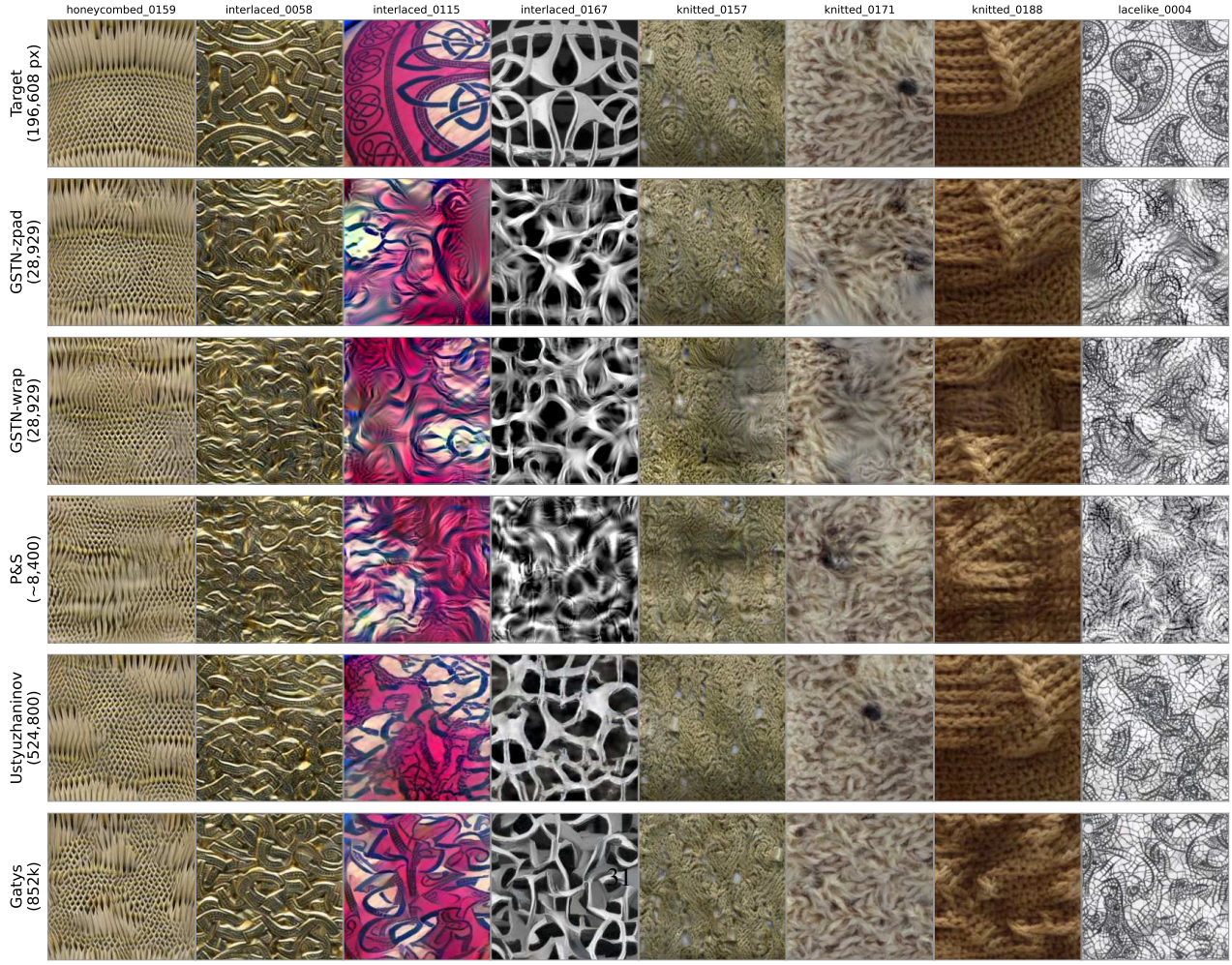


DTD-141 full qualitative comparison — page 3 of 9

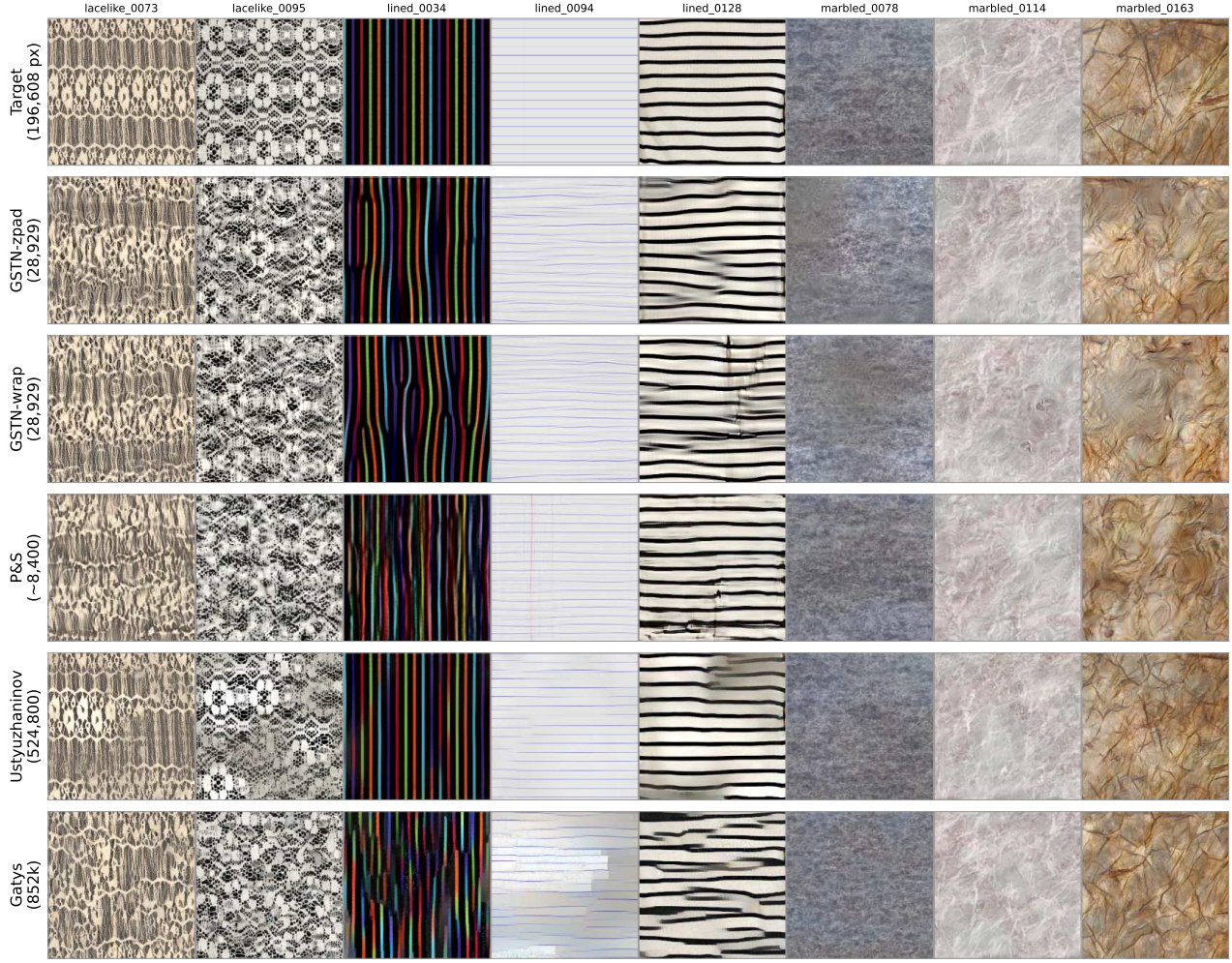


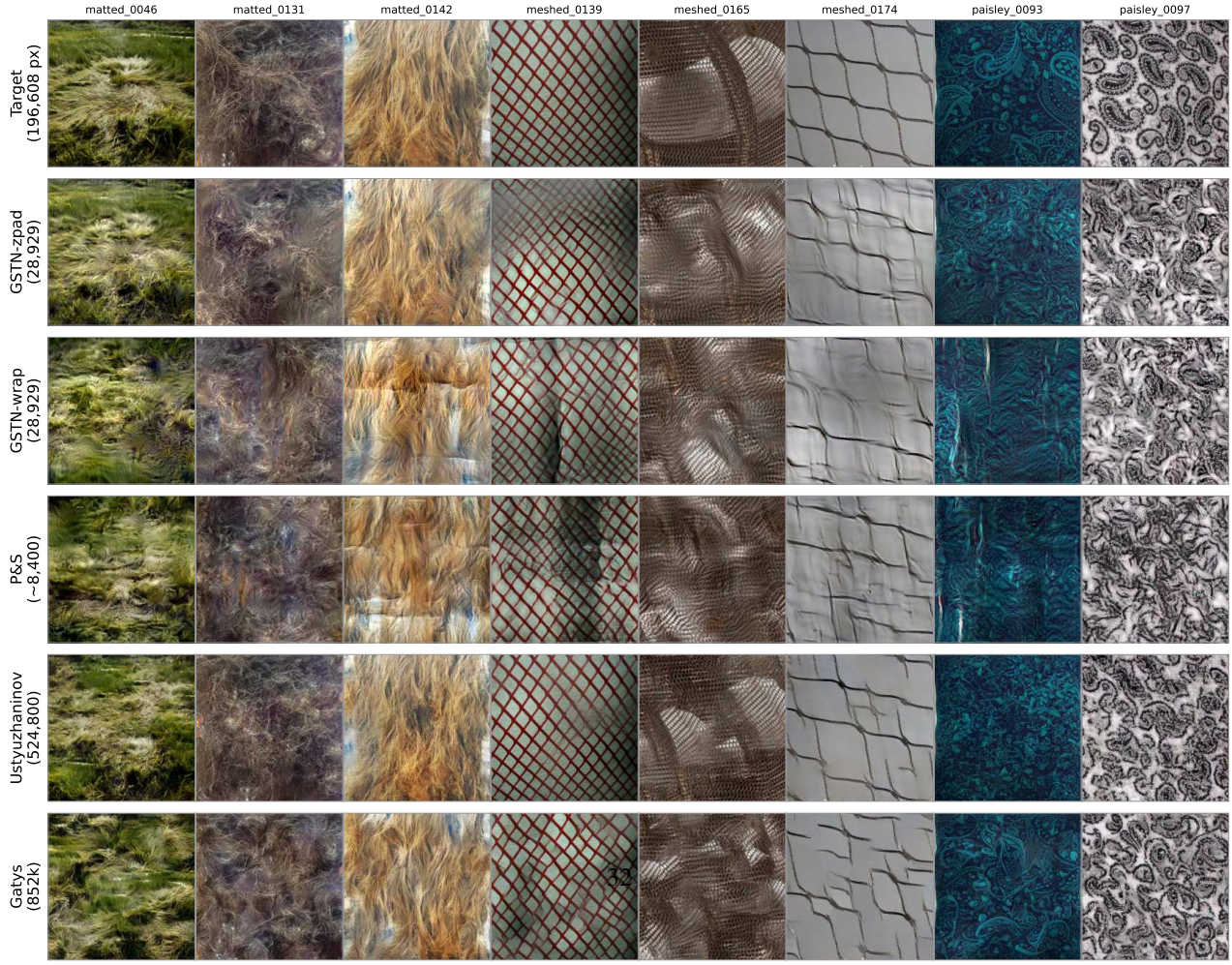



grid_0009
grid_0023
grid_0056
grooved_0053
grooved_0055
grooved_0126
honeycombed_0149
honeycombed_0150
Target (196,608 px)
GSTN-zpad (28,929)
GSTN-wrap (28,929)
P&S (~8,400)
Ustyuzhaninov (524,800)
Gatys (852k)

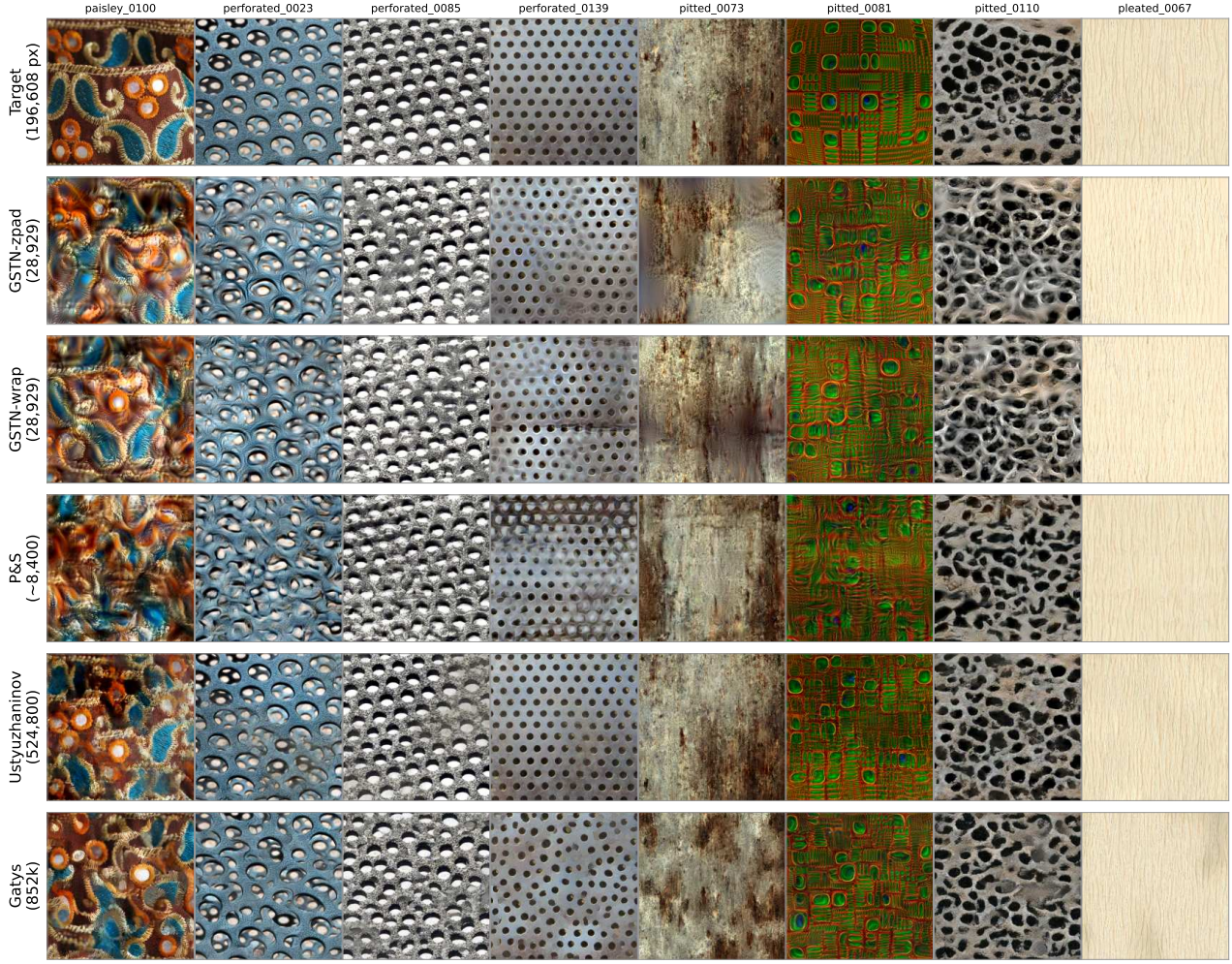

honeycombed_0159
interlaced_0058
interlaced_0115
interlaced_0167
knitted_0157
knitted_0171
knitted_0188
lacelike_0004
Target (196,608 px)
GSTN-zpad (28,929)
GSTN-wrap (28,929)
P&S (~8,400)
Ustyuzhaninov (524,800)
Gatys (852k)

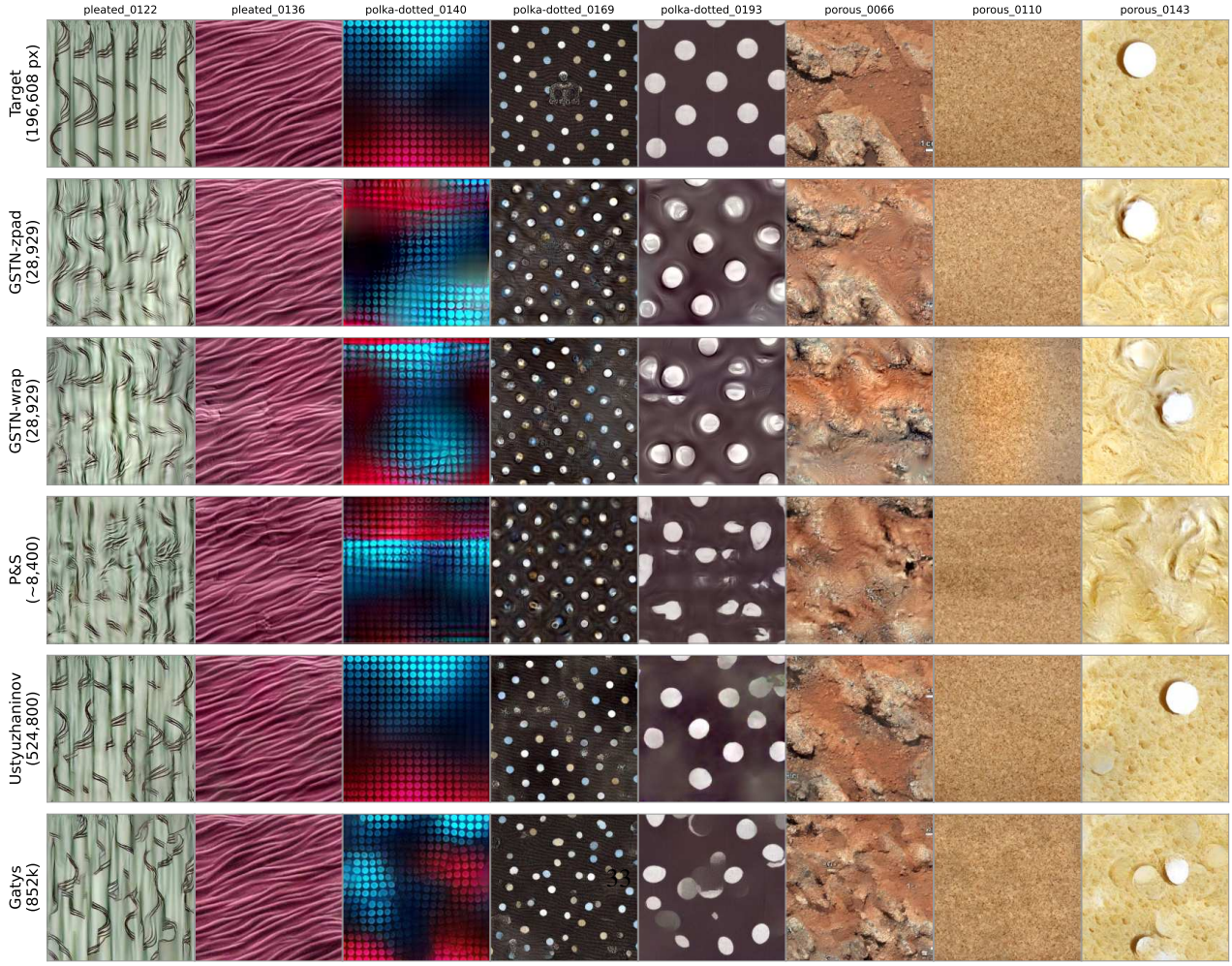
DTD-141 full qualitative comparison — page 5 of 9



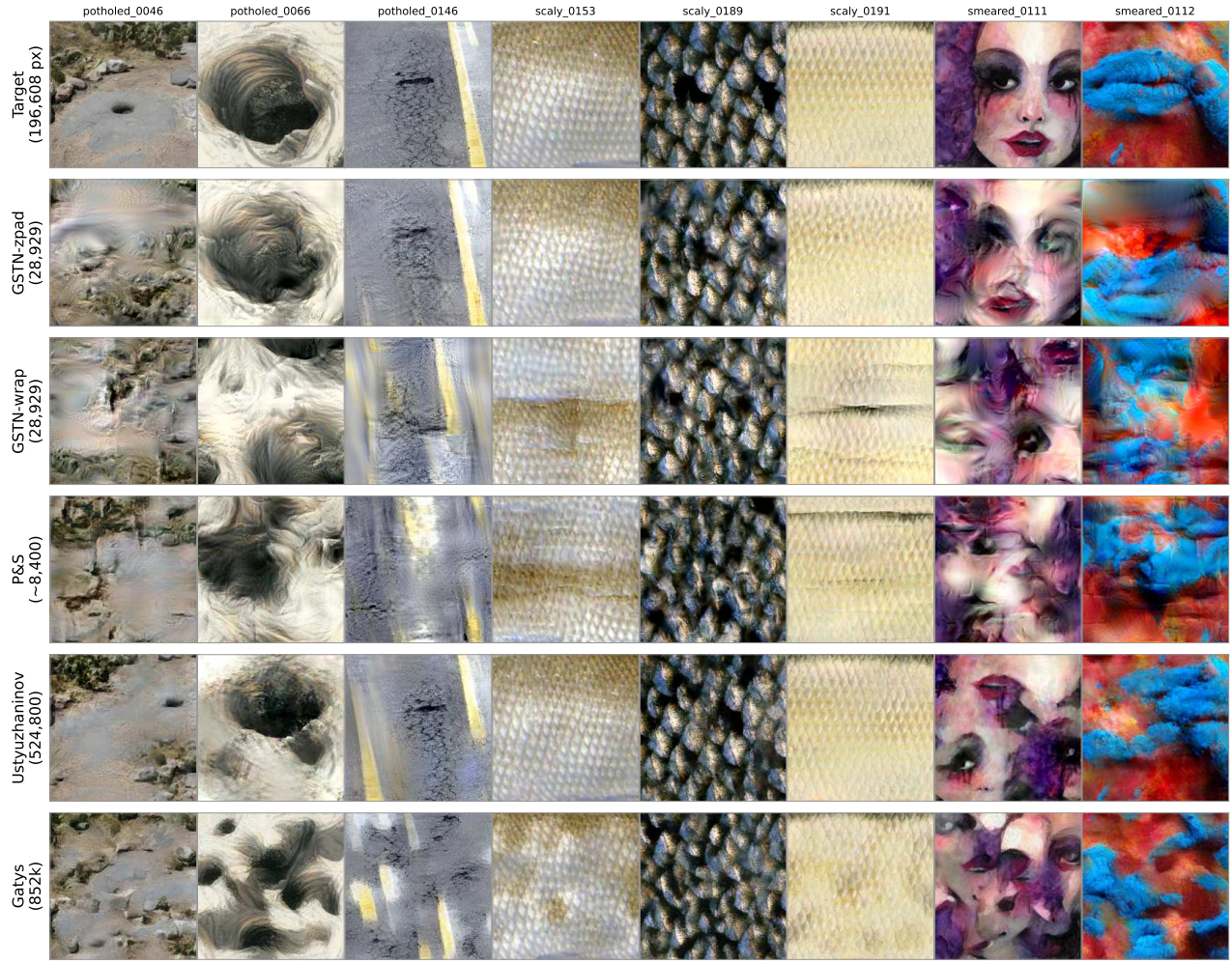


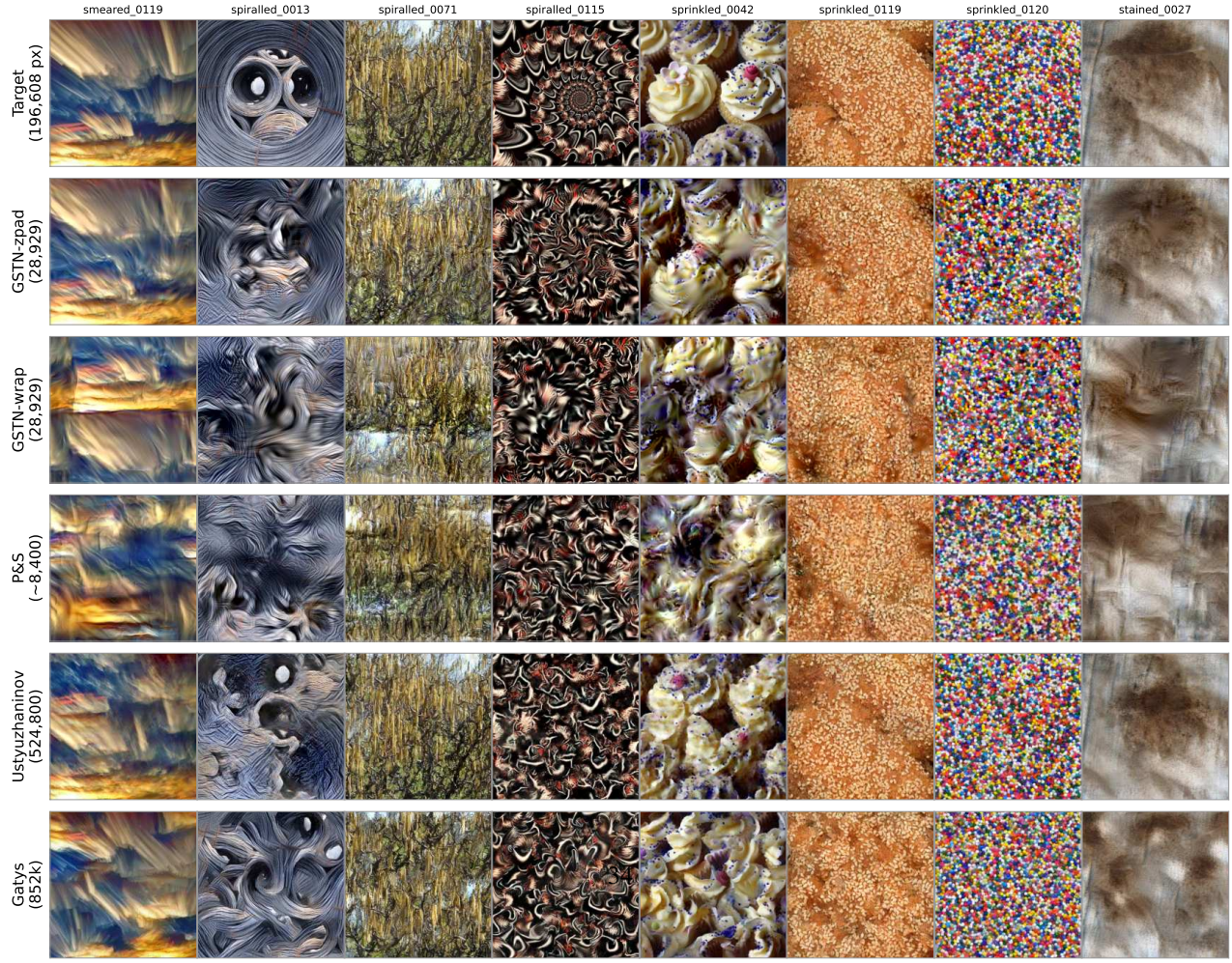
DTD-141 full qualitative comparison — page 6 of 9

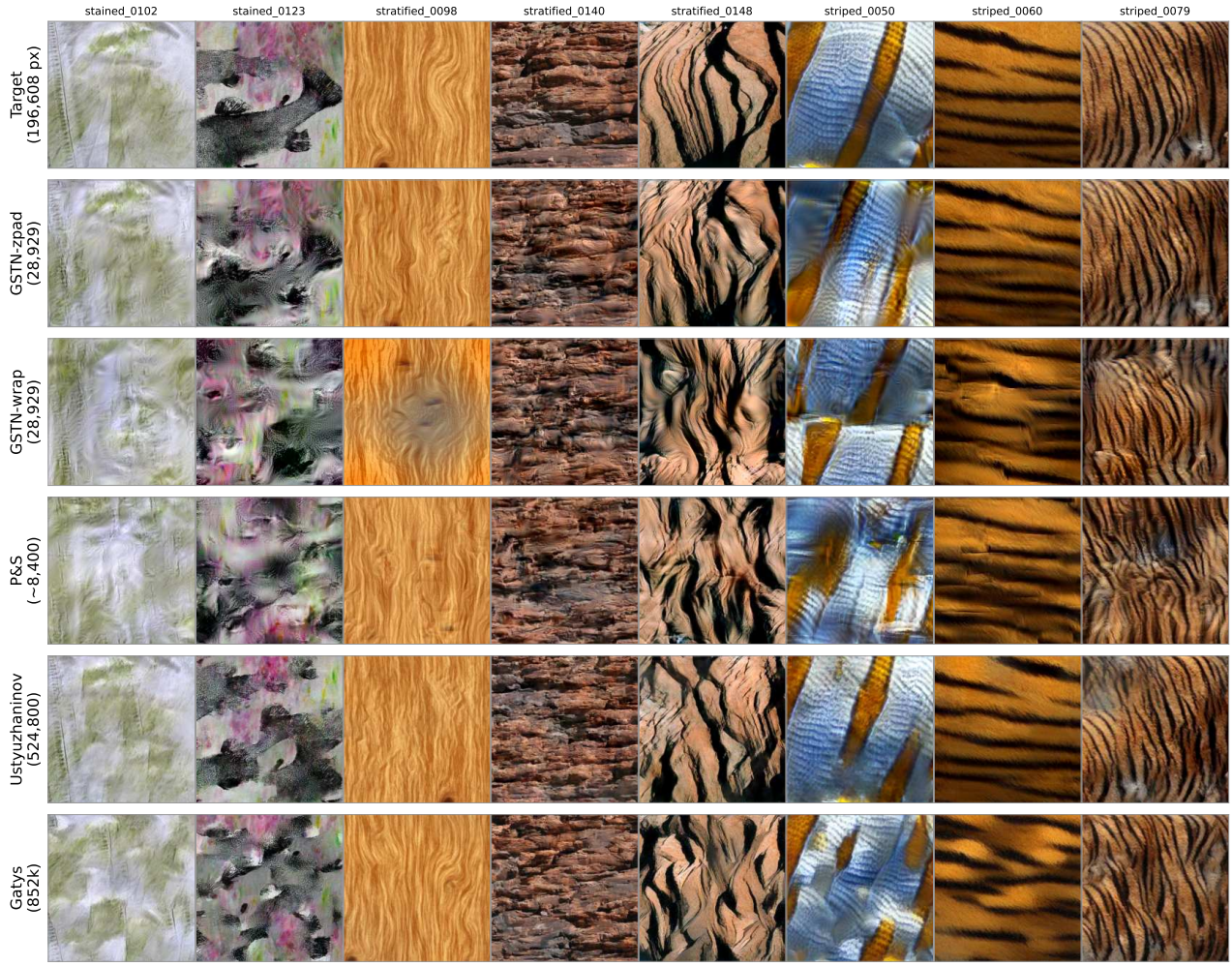


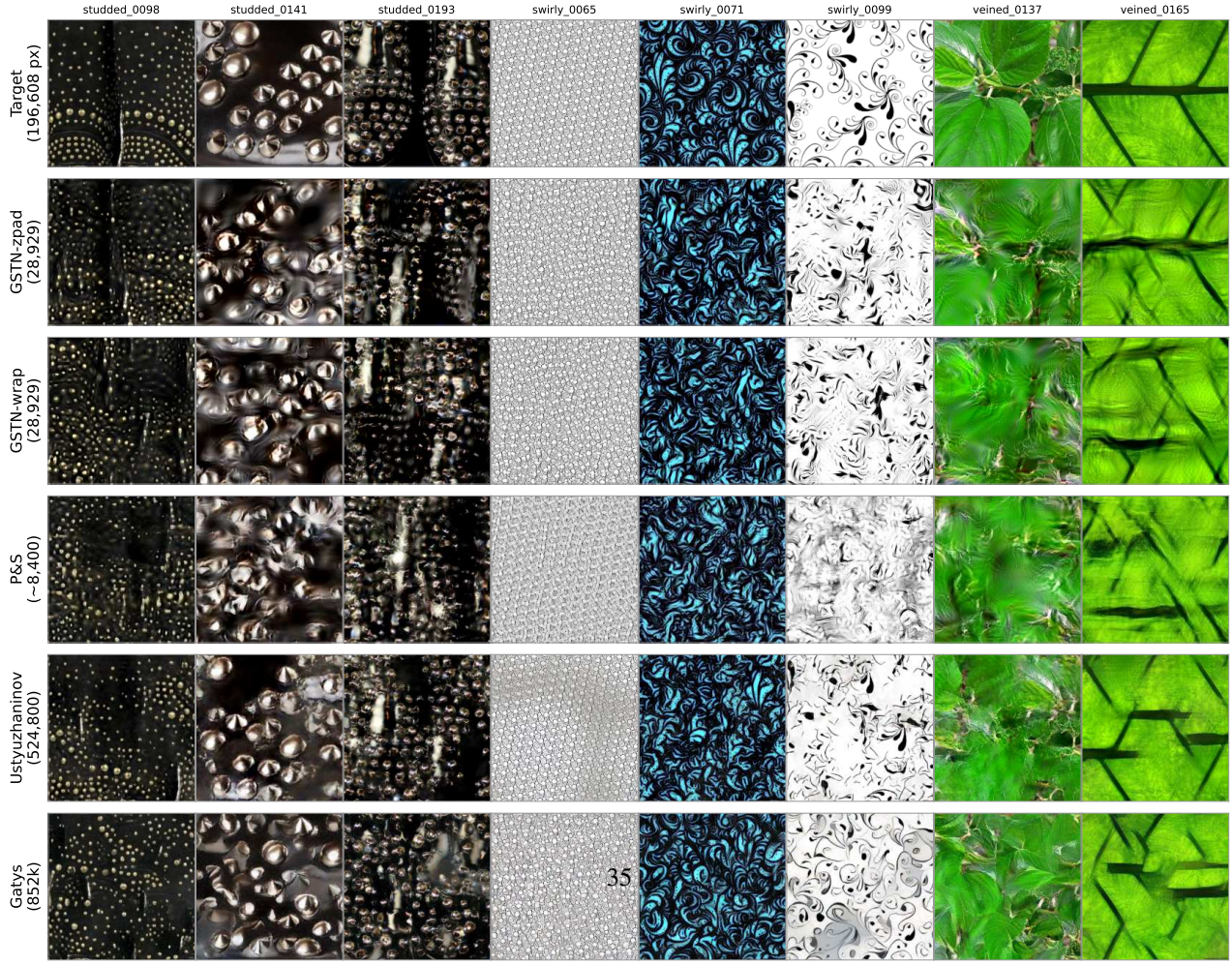


DTD-141 full qualitative comparison — page 7 of 9

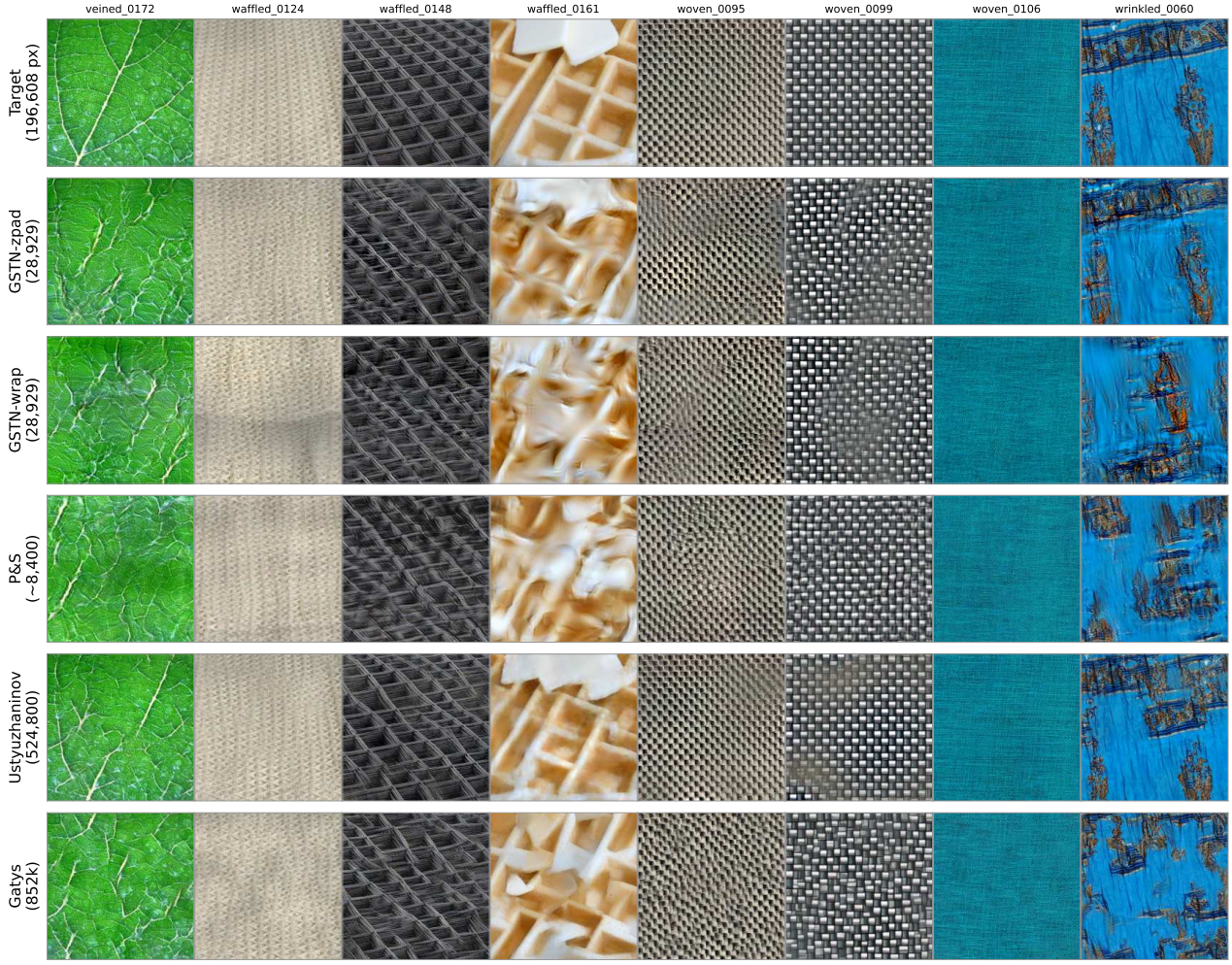


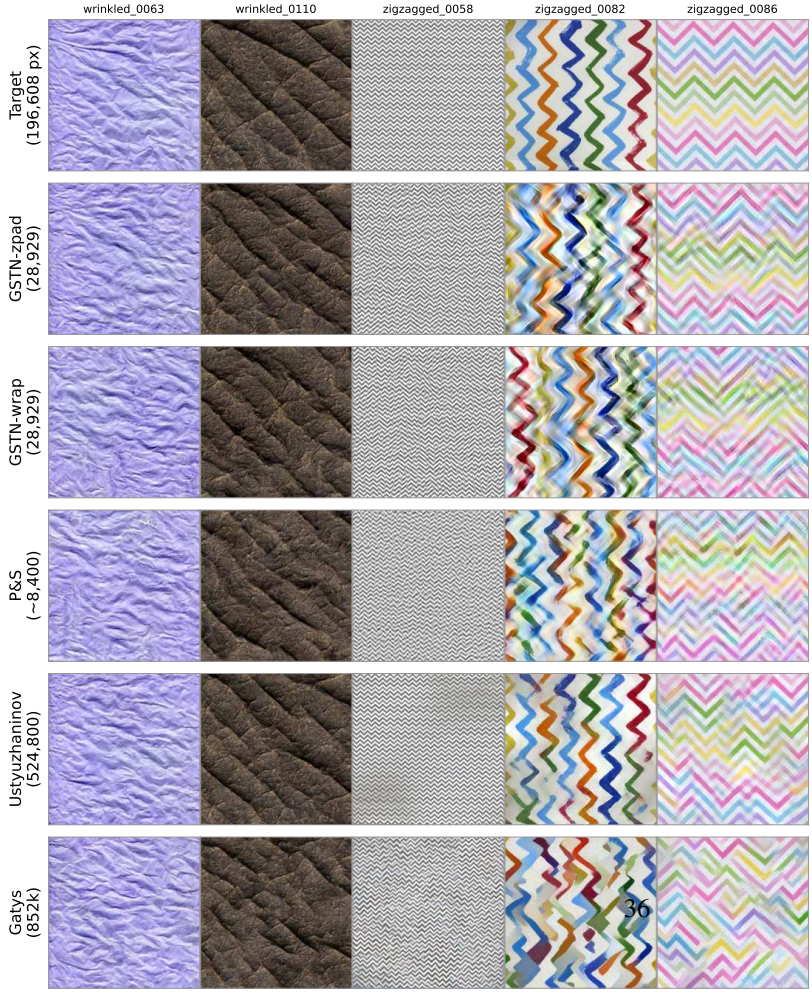



stained_0102
stained_0123
stratified_0098
stratified_0140
stratified_0148
striped_0050
striped_0060
striped_0079
Target (196,608 px)
GSTN-zpad (28,929)
GSTN-wrap (28,929)
P&S (~8,400)
Ustyuzhaninov (524,800)
Gatys (852k)


studded_0098
studded_0141
studded_0193
swirly_0065
swirly_0071
swirly_0099
veined_0137
veined_0165
Target (196,608 px)
GSTN-zpad (28,929)
GSTN-wrap (28,929)
P&S (~8,400)
Ustyuzhaninov (524,800)
Gatys (852k)

DTD-141 full qualitative comparison — page 9 of 9





NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: There are no claims made in abstract or introduction that do not appear in body.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: There are multiple limitations section included in discussion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: No theoretical results beyond the analytical solution for Contrastive Learning in Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All of the settings for the pyramid architecture and parameters, Gramian structure, and optimization details are included in the appendix. Code will be released upon acceptance, and datasets used are open-source

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code will be released upon acceptance with camera ready to protect anonymity, datasets are already open source and cited, but currently the experiments are described in sufficient detail for reproduction.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The training/test set is defined already in the datasets used, the hyperparameters are reported for all optimizations, and a section is dedicated to optimizer choice. Paper reports these were chosen by sweeps to identify optimal hyperparams.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Qualitative syntheses analyses do not allow for error bars to be reported, but quantitative results contains error bars.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Computational resources section is in appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the Code of Ethics and believe the research conforms to this code.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: A broaded impacts section is included the conclusion.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not have high risk for misuse as discussed in broader impacts section.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Authors are original creators and owners of this code, codebases used to build this model and benchmark it are referenced, datasets used are open source and referenced.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: Paper does not release new assets at this time, but code will be released with an open-source license with camera ready.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Paper does not involve any crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper uses no research participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Section on LLM use is included in supplemental

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.