

U-Attention to Textures: Hierarchical Hourglass Vision Transformer for Universal Texture Synthesis

Shouchang Guo^{1*}, Valentin Deschaintre², Douglas Noll¹, Arthur Roullier²

¹University of Michigan, Ann Arbor, ²Adobe Research

Abstract

We present a novel U-Attention vision Transformer for universal texture synthesis. We exploit the natural long-range dependencies enabled by the attention mechanism to allow our approach to synthesize diverse textures while preserving their structures in a single inference. We propose a hierarchical hourglass backbone that attends to the global structure and performs patch mapping at varying scales in a coarse-to-fine-to-coarse stream. Completed by skip connection and convolution designs that propagate and fuse information at different scales, our hierarchical U-Attention architecture unifies attention to features from macro structures to micro details, and progressively refines synthesis results at successive stages. Our method achieves stronger 2× synthesis than previous work on both stochastic and structured textures while generalizing to unseen textures without fine-tuning. Ablation studies demonstrate the effectiveness of each component of our architecture.

1. Introduction

Texture synthesis aims at expanding a given texture image while preserving its structure and texture content. This finds application in multiple domains such as image inpainting or large-scale content generation. In this work we leverage the attention mechanism [1] in Transformers [37] to synthesize a 2× larger version of an input texture.

Recent learning-based approaches for texture synthesis showed impressive results using CNNs but require the training of a network for each texture [46, 16]. These methods cannot generalize to unseen textures due to the locality of operations in CNNs [29]. To account for the global structural information required for the texture synthesis task, a different trend in recent work incorporates new designs for CNNs [29, 28] leveraging similarity maps or synthesizing textures in the Fourier domain.

In this work, we focus on the emerging attention mech-

*This work was mostly done when Shouchang Guo was an intern at Adobe Research.

anism in Transformers that naturally exploits long-range structural dependencies required by texture synthesis and network generalization while reducing its computational overhead. Using our novel Transformer based architecture, our trained network is able to blend and match patches with various scales. This versatility lets our texture synthesis approach generalize to a broad range of texture patterns while remaining more computationally efficient than recent CNN-based networks [29, 28].

We build a novel texture synthesis model that: (1) performs fast synthesis with a single forward inference, and (2) generalizes to different classes of textures with various amount of structure, with a single trained network.

We propose a novel U-Attention net with multi-stage hierarchical hourglass Transformers as the generative model for texture synthesis. Our model presents two novel designs that distinguish it from other Transformer networks for image-to-image mapping, enabling fast texture synthesis for various types of textures. (1) Our hierarchical hourglass network exhibits an hourglass-like scale change of the input patches for different Transformer stages and enables progressive patch mapping with a coarse-fine-coarse scheme. (2) Each Transformer block is followed by a convolution and a down/up sampling layer, providing increased network capacity, with skip connection propagating pre-bottleneck information to later stages.

To summarize, our main contributions are:

- A novel hierarchical hourglass backbone for coarse-to-fine and fine-back-to-coarse processing, allowing to apply self-attention at different scales and to exploit macro to micro structures.
- Skip connections and convolutional layers between Transformer blocks, propagating and fusing high-frequency and low-frequency features from different Transformer stages.
- A 2× texture synthesis method with a single trained network generalizing to various texture complexity in a single forward inference.

2. Related Works

2.1. Algorithmic texture synthesis

Texture synthesis aims at generating a larger extent of a given texture. This research area has been active for the past decades, as described in the survey of Raad et al. [32]. More specifically, traditional methods follow two main approaches: statistics-based, using statistical descriptors of the original texture to generate the new image [18, 13, 14, 17, 15, 19], and patch-based layout shuffling, leveraging available patches in the original texture to synthesize the new one [12, 11, 24]. The statistics-based methods can reproduce micro-structures well, but fail to preserve larger-scale patterns. Our method shares conceptual similarities with patch-based approaches but leverages multi-scale self-attention to evaluate patch relevance at a given position rather than a direct optimization.

2.2. Deep-learning based texture synthesis

More recently, deep learning based approaches have been proposed to synthesize textures. These methods can be divided into two families: iterative optimization approaches that use pretrained CNNs as feature extractors to match deep feature statistics of inputs and outputs [16, 21, 27, 20, 33], and specialized networks which are trained per texture [36, 46, 34, 30, 23, 3]. The latter produce impressive results but do not generalize to unseen textures without new training. Optimization methods using pretrained CNNs on the other hand often miss important texture elements, in particular in their structure [27, 20, 16] as they are based on convolutions and struggle to capture long-range dependencies in the textures.

To alleviate these issues, Mardani et al. [29] proposed to synthesize textures in the Fourier domain in which global changes in image space are locally represented. Closer to our approach, Liu et al. [28] leverage a self-similarity map computed by comparing every possible combination of the shifted input texture at different positions. As opposed to previous work, we design a Transformer based architecture, leveraging the attention mechanism to naturally "attend" to different – arbitrarily distant – parts of textures at multiple scales. This new architecture allows our method to generalize to a large variety of textures and better handle structural elements.

2.3. Transformers for images

The Attention mechanism has been introduced in the context of natural language processing [1] as a way to solve the problem of long-range dependency on input sequences and was later adapted to the Transformer architecture [37]. Motivated by the numerous successes in the field, it was soon adapted to computer vision tasks [6, 31].

However, the attention mechanism is not straightforward

to adapt to images, specifically due to its computational (quadratic) complexity. Consequently, recent research focused on improving the attention mechanism efficiency. Different approaches approximate the formula itself: the Performer [7] linearizes the softmax using the kernel trick, the Linformer [39] lowers the rank of the attention matrix, the Perceiver [22] uses a recurrent update of a low dimensional latent representation, and the Synthesizer [35] approximates the attention matrix itself. Another line of work relaxes the (spatial) range over which information is retrieved: Lambda Networks [2] restrict the attention to a local neighborhood, Axial DeepLab [38] restricts attention to specific directions, the Vision Transformer [10] splits the image into crops instead of working at the pixel level.

Different visual tasks are tackled by recent works on image Transformers, including classification [10], image processing [5], super-resolution [42], and video inpainting [43]. Similar to these works we leverage the attention mechanism to image patches. For our universal texture synthesis goal, we propose a hierarchical hourglass architecture that progressively refines patch mapping with successive Transformer stages and unifies attention to multi-scales with an hourglass-like scale change between stages.

3. U-Attention Model Architecture

3.1. Overview

Problem statement Texture synthesis takes a small texture exemplar as input and generates a larger scale image of the same texture. Many texture synthesis approaches leverage patch matching and mapping at a given scale to enlarge texture content based on patches of the given exemplar texture [12, 11, 24]. This leads to two challenges: (1) small scale patch mapping often fails to capture global structures and long-range dependencies in an input texture, and (2) choosing a scale for patch mapping requires manual tuning for different levels of structure depending on the input texture images.

To take global structure into account, we propose to use attention mechanisms attending to the entire input image and able to process long-range dependencies. To automatically synthesize texture without manual scale parameter tuning, we propose a novel hierarchical attention network that unifies attention at different scales of texture details.

Architecture Fig. 1 presents our proposed U-Attention Network architecture. The input image to be outpainted is: (1) encoded by a set of convolutional layers into a feature domain before (2) being processed by our main hierarchical Transformer network and (3) being decoded back to the image domain by a set of convolutions.

Our main network consists of an odd number (e.g., 5) of Transformer blocks arranged in an hourglass shape for multi-scale and hierarchical patch mapping. For each

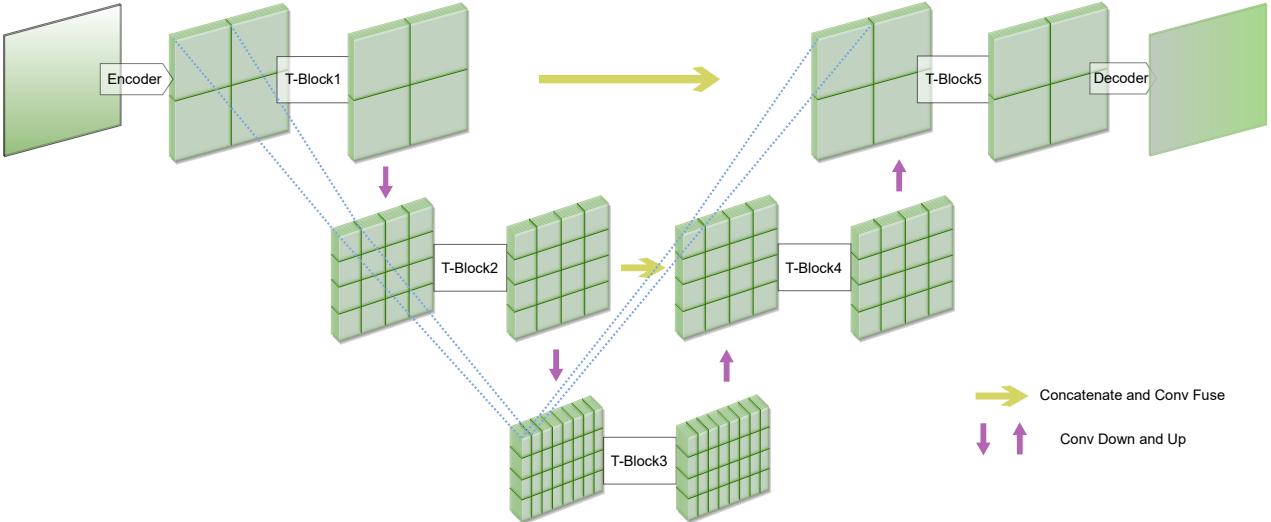


Figure 1. Proposed U-Attention framework with hierarchical hourglass Transformers. We introduce a multi-scale partition of the feature map between hierarchical Transformer blocks to form input patches of different scales for different Transformers. The input texture image is first projected into feature space by an encoder. We then leverage a succession of Transformer blocks, with up and down convolutions in between (purple arrows), processing the feature maps at different resolutions. Each Transformer block takes the whole feature maps as input, and we partition the feature maps to be sequences of patches of progressively smaller or larger sizes at consecutive stages of the network. Therefore, the input patch size of all the stages forms an hourglass-like scale change (dotted blue line), enabling attention to finer/coarser details at different attention steps. Finally, we add skip connections that propagate and concatenate outputs from different previous stages as part of the inputs for later Transformer stages (yellow arrows).

Transformer block i , we partition the current latent feature map of dimension $H_i \times W_i \times C$ into a sequence of P_i^2 patches of dimension $\frac{H_i}{P_i} \times \frac{W_i}{P_i} \times C$ as the input. The Transformer block processes the patch sequence and exploits long-range dependencies with a self-attention mechanism that attends to all the patches. After processing, the transformed sequence of patches forms the transformed feature map, which is passed to the next stages of the model.

The first 2 Transformer blocks are followed by strided convolutions that down-sample their inputs and enlarge the channel dimension, while outputs from the following 2 blocks are up-sampled and shrunk in the channel dimension. We use skip connections, respectively between the outputs of Transformer blocks 1 and 2, and the inputs of Transformer blocks 4 and 5 (i.e., at the same resolution), allowing to propagate high frequency information extracted by the Transformer blocks before the bottleneck to be part of the input for later Transformer blocks.

We train our network using a structural l_1 loss, VGG-based perceptual and style losses [28], and a patch GAN loss as proposed in [43, 4].

3.2. Hierarchical hourglass vision Transformers

Transformer block Our Transformer block consists of 2 stacked Transformer layers, where each Transformer layer [37] contains attention, feed forward operation, residual connections, and normalization as illustrated in Fig. 2.

Each of our Transformer block i operates on patches of

latent space feature maps. The sequence of P_i^2 patches with dimensions $\frac{H_i}{P_i} \times \frac{W_i}{P_i} \times C$ is turned into a tensor $X \in \mathbb{R}^{P_i^2 \times \frac{H_i}{P_i} \times \frac{W_i}{P_i} \times C}$ and is processed via self-attention:

$$\text{self-attention}(X) = \sigma \left(\frac{W_Q(X) W_K(X)^\top}{\sqrt{d}} \right) W_V(X) \quad (1)$$

where σ is the softmax function and \cdot^\top is the matrix transpose. $W_Q(\cdot)$, $W_K(\cdot)$, and $W_V(\cdot)$ denote operations that perform 1×1 convolutions along the channel dimension as in [44, 43] to embed input patches to [queries, keys, and values], and further reshape the [queries, keys, and values] to matrices of dimension $P_i^2 \times d$, where $d = \frac{H_i}{P_i} \frac{W_i}{P_i} C$. The self-attention operation maps a sequence of patches to another sequence of patches with a learned attention map. Specifically, the keys $W_K(X)$ can be thought of as *content descriptors* broadcast by each input patch (e.g. “I contain some brick texture”), the queries $W_Q(X)$ can be thought of as what each patch is *interested in* (e.g. “I contain half a brick texture and would need some brick texture”), and the values $W_V(X)$ are the actual quantities that are merged together to compute the output, based on the matching of keys and queries.

Essentially, the Transformer block performs a learned patch mapping with global information, and the output sequences of patches are reshaped back to form new feature maps as inputs for the next stages.

Hierarchical network To unify attention at varying

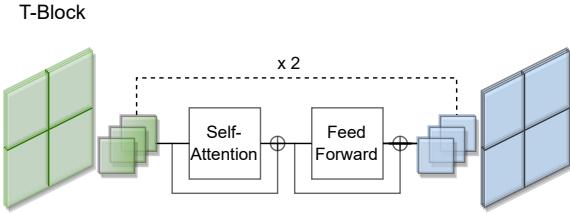


Figure 2. Details of a single Transformer block (T-block). An input latent feature map is partitioned into a sequence of patches, and the input patch sequence is mapped to another patch sequence via a stack of 2 Transformer layers. The output sequence is then reshaped back into a whole feature map.

scales, we propose a multi-stage hierarchical design that progressively reduces/enlarges both the size of the feature maps in between consecutive Transformer blocks and the size of the patches attended to by the attention mechanisms. Indeed, the resolution of the attention mapping is determined by the spatial extent of the considered patches. If the patches are too small, the correlation calculation for attention can be noisy and computationally expensive. While if the patch is too large, the mapping would be too coarse.

With these considerations in mind, we design our network with an hourglass-shape scale change. Following each Transformer block – except for the last one – we add a $2\times$ down or up convolution as shown in Fig. 1. These convolutions reduce (resp. enlarge) the input spatial dimension and increase (resp. decrease) the channel dimension. Because the operations for attention are implemented with 1×1 convolutions along the channel dimensions, the progressively increasing (resp. decreasing) channel dimensions allow the attention to leverage deeper representations of the original image at different Transformer blocks of our model.

Importantly, as we reduce or augment the feature map resolution with convolutions, we further increase or decrease the spatial partitioning in each dimension by $2\times$ to form longer or shorter patch sequences as inputs for different Transformer stages. Therefore, the size of the patches processed by the attention mechanism is divided or multiplied by $4\times$ between consecutive Transformer blocks as shown in Fig. 3. This ensures that the Transformers can attend to different scales of features at different levels of the hourglass hierarchy. Specifically, as each spatial dimension is divided or multiplied by 2, while each patch dimension is divided or multiplied by 4, the network effectively attends to finer or coarser details with the footprint of attended patches being divided or multiplied by 2 at each stage.

We add skip connections to propagate high-frequency information from earlier stages to their mirrored later stage Transformer blocks as shown in yellow in Fig. 1. Convolutional layers then fuse the skip connection information (yellow arrow) with the up-sampled output of the previous stage (purple arrow), forming the input of the next stage.

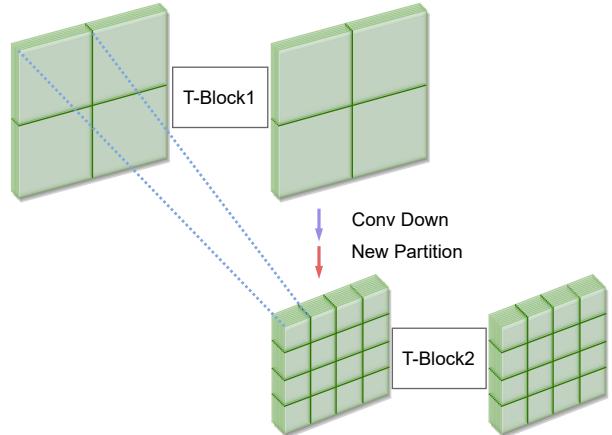


Figure 3. Details of a transition between two Transformer stages that formulate the change of input patch scales for different Transformers. A feature map formed after a Transformer block (T-Block1) is downsampled by $2\times$ with $4\times$ enlarged channel dimension using a strided convolution, and then partitioned into $4\times$ more patches as the input for the next Transformer block. Because the spatial extent is halved by the strided convolution, and the number of patches is doubled along each spatial dimension with the new partition, the input patches for the next Transformer block (T-Block2) are $4\times$ smaller in each spatial dimension compared to the input patches for T-Block1. Each patch effectively represents $2\times$ smaller regions than the previous step.

This U-Attention network unifies attention at different scales from different stages and builds progressive patch mapping for both global pattern outline and local refinements of detailed structures.

Coarse to fine to coarse We start from coarse scale attention to allow the first Transformer block to leverage the input texture and the overall structure. As the attention mechanism requires some information in the patches of the patch sequence to build a query and assess self-similarity, starting from small patches would require recursive processing that slowly builds the texture outward with little information on the global structure. In our approach, at the coarsest level, the union of the patches covers the whole – signal free – area to synthesize and each patch contains part of the input texture. This enables our approach to synthesize a first rough scale texture with preserved original structure and relevant generation to be further refined in the following stages, as illustrated in Fig. 7.

Encoder-Decoder Before processing the inputs with our Transformer blocks, we encode them in feature space using a small 2 layer CNN, preserving the spatial extent but increasing the channel dimension. This lets the network manipulate a richer representation of images. Likewise, the last Transformer block outputs features that are then decoded by another 2 layer CNN similar to the encoder, returning features to the image space.

	SSIM	LPIPS	c-LPIPS	Inference speed (s)
Naive Tiling	0.104	0.529	0.500	-
WCT	0.179	0.401	0.350	0.7
pix2pixHD	0.132	0.526	0.497	0.011
Self Tuning	0.104	0.540	0.514	150
Texture CNN*	0.199	0.272	0.256	100
Ours	0.333	0.272	0.152	0.022

Table 1. Quantitative comparison to previous work. Our approach performs better on structural and perceptual metrics.

*TextureCNN aims at the generation of a new texture variation from the entire ground truth image.

3.3. Training and implementation details

The loss function \mathcal{L} we use to train our model is a combination of structural l_1 loss in pixel space \mathcal{L}_{l_1} , perceptual loss \mathcal{L}_p as the l_1 distance in the feature space of a pretrained VGG model, and a style loss \mathcal{L}_s defined by the l_1 distance between the Gram matrices of the features from the same VGG model [16]. The perceptual loss and the style loss match the overall statistics of the texture without constraining a pixel perfect match. We also used a temporal-patch adversarial loss (T-PatchGAN) \mathcal{L}_{GAN} [4, 43] that helps crafting realistic details. We set the batch dimension as the temporal dimension for the T-PatchGAN to distinguish spatial features of a batch as real or fake. Specifically, our complete loss is

$$\mathcal{L} = \lambda_{l_1} \mathcal{L}_{l_1} + \lambda_p \mathcal{L}_p + \lambda_s \mathcal{L}_s + \lambda_{GAN} \mathcal{L}_{GAN} \quad (2)$$

Similar to previous work [29, 28], we set $\lambda_p = 0.01$, $\lambda_s = 200$, and $\lambda_{GAN} = 0.1$. We add a component \mathcal{L}_{l_1} with $\lambda_{l_1} = 1$ to emphasize the structural synthesis and stabilize the network training.

We develop our model using Pytorch and train it using a single NVIDIA Tesla V100 GPU for 7 days. We use a batch size of 8 and train for 100 epochs with Adam [25] optimizer at a constant learning rate of 0.001. At inference time, our proposed network runs in 22 ms.

We use publicly available texture datasets with a wide variety of structures and patterns from 3 different sources [8, 9, 26]. There are about 27000 texture images in total, and we randomly split them into 26400 images for training, and about 600 images for validation and testing. The images of different scales and resolutions are resized to $128 \times 128 \times 3$ as target images, and their $64 \times 64 \times 3$ central crops are zero-padded to $128 \times 128 \times 3$ as inputs.

4. Experiments

4.1. Evaluation metrics

For evaluation, we randomly select 400 images from the testing set and evaluate the synthesis performance based on

	SSIM	LPIPS	c-LPIPS
Baseline	0.256	0.388	0.294
Pyramid	0.325	0.276	0.154
Hourglass	0.334	0.284	0.156
No GAN loss	0.381	0.350	0.204
Ours	0.333	0.272	0.152

Table 2. Qualitative evaluation of the ablation study. We see that our full approach better matches the perceptual appearance of the target (LPIPS/c-LPIPS). While the version without adversarial loss performs a little better when evaluated with SSIM, its perceptual appearance is significantly worse, as particularly visible in the qualitative experiment in Fig. 5. With respect to structure, the pyramid cascaded Transformer seem to provide the strongest benefit.

qualitative results and quantitative metrics including structural similarity index measure (SSIM) [41] and learned perceptual image patch similarity (LPIPS) [45]. SSIM and LPIPS measure structural and perceptual similarity between image pairs and we use c-LPIPS which operates on crops of images, for texture synthesis evaluation. We compute c-LPIPS on 8 random crops of the target and output images.

4.2. Comparisons and results

We compare our method to the baseline and state-of-the-art, publicly available, approaches: (1) naive tiling, which tiles the input to the relative locations; (2) WCT [27], a style transfer method using the small texture input as the style image and a $2 \times$ larger random noise image as the content image; (3) pix2pixHD [40], the state-of-the-art image-to-image translation GAN network, which we trained with the same input and output pairs as for our work; (4) self tuning [24], the state-of-the-art optimization method; (5) texture CNN* [16] which uses the ground-truth image as input style and a noise image of the same size as the content image.

As presented in Fig. 4, our model outperforms other approaches with generalization to unseen textures, and better preserves global structures including global color shift as shown in the third row and brick wall structures in the last row. We present the numerical evaluation in Tab. 1 and demonstrate that our method outperforms previous work on structural (SSIM, higher is better) and perceptual (LPIPS and c-LPIPS, lower is better) metrics.

4.3. Attention visualizations

Our novel use of attention mechanisms enables new ways for patch mapping at varying scales and allows for interesting visualizations. We are able to visualize the attention to each considered patch at each stage of the proposed model. We can also visualize how the network progressively improves results with consecutive Transformer stages operating on varying scales. Fig. 7 presents the hierarchical patch mapping for one output patch. The figure illus-

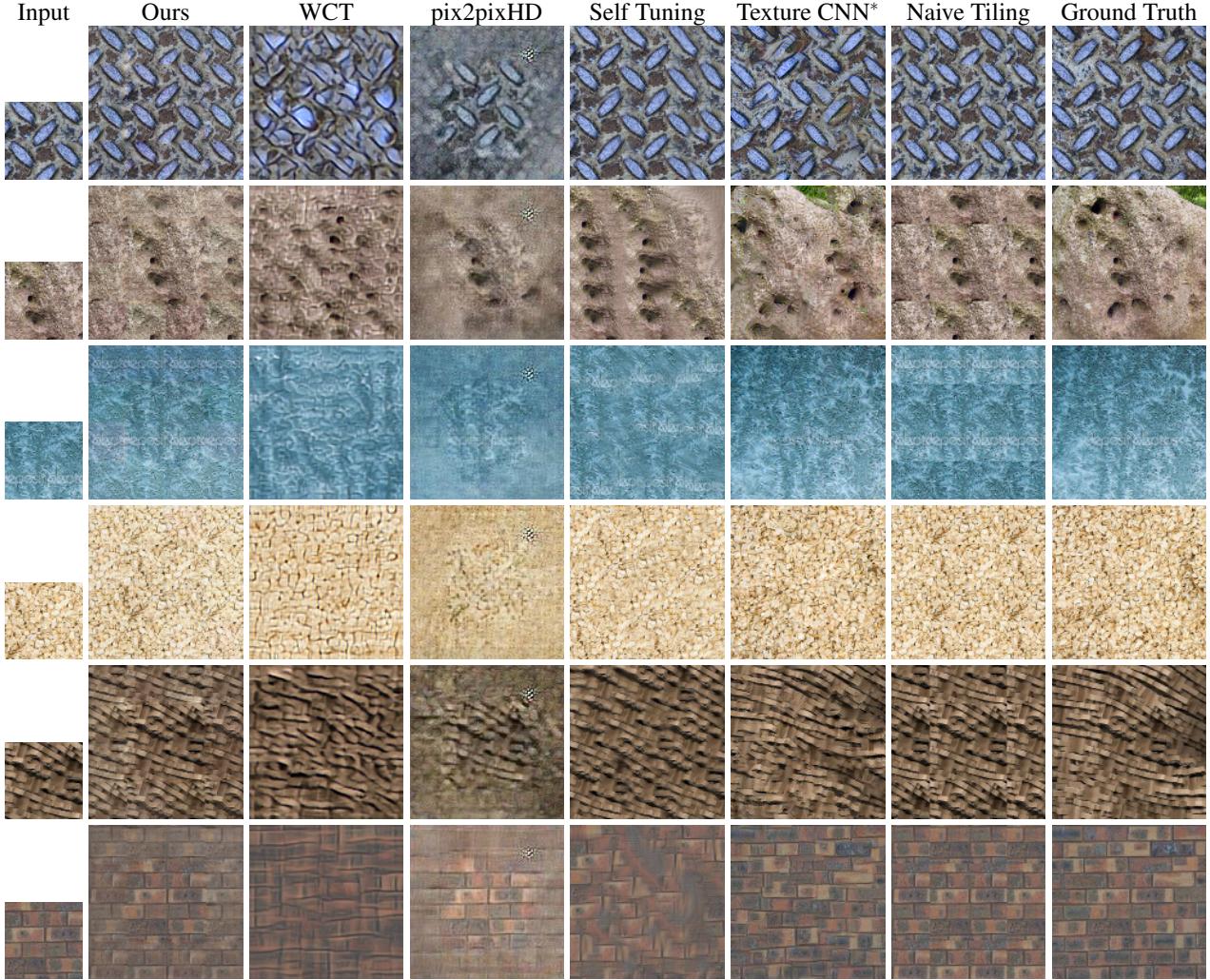


Figure 4. Comparison to previous works. We can see that our method does not suffer from structure randomness as WCT [27] or Texture CNN [16]. Our approach is also more robust in avoiding repetition artifacts as seen in Self Tuning [24] or blurry boundaries like pix2pixHD [40]. We also show that our method suffers less from the strong discontinuities introduced by naive tiling.

trates how the proposed network progressively (top to bottom) refines the patch mapping process at different scales with coarse-to-fine and fine-back-to-coarse schemes.

4.4. Ablation study

We perform an ablation study to demonstrate the effectiveness of each component of our network. Our model is compared to: (1) a baseline architecture with three cascaded Transformer blocks; (2) the baseline (1) and our proposed progressive patches size reduction for later cascaded Transformers (also called pyramid architecture hereafter); (3) the hourglass backbone with two mirrored cascaded Transformer (2) and skip connections, without convolution downs and ups between Transformer blocks; (4) our proposed U-Attention net without the adversarial loss.

We present the qualitative result of the ablation study in

Fig. 5. The baseline cannot handle multiple scales and patch sizes. The pyramid architecture (2) allows to attend to different patch sizes, but cannot unify the multi-scale structures properly. The simplified hourglass misses the convolutions between Transformer blocks and leads to inconsistent color shifts and worst structure alignment as seen in lines 2 and 4 of Fig. 5. Our use of GAN loss is particularly important to enhance the realism of the results of our network. Without the GAN loss, the borders of the image are well structured but suffer from a color and contrast shift.

We provide the quantitative evaluation with L-PIPS and c-L-PIPS in Tab. 2 confirming the visual qualitative differences seen in Fig. 5. More details of the ablated designs and additional synthesis results on a wide range of textures are provided in the supplemental materials.

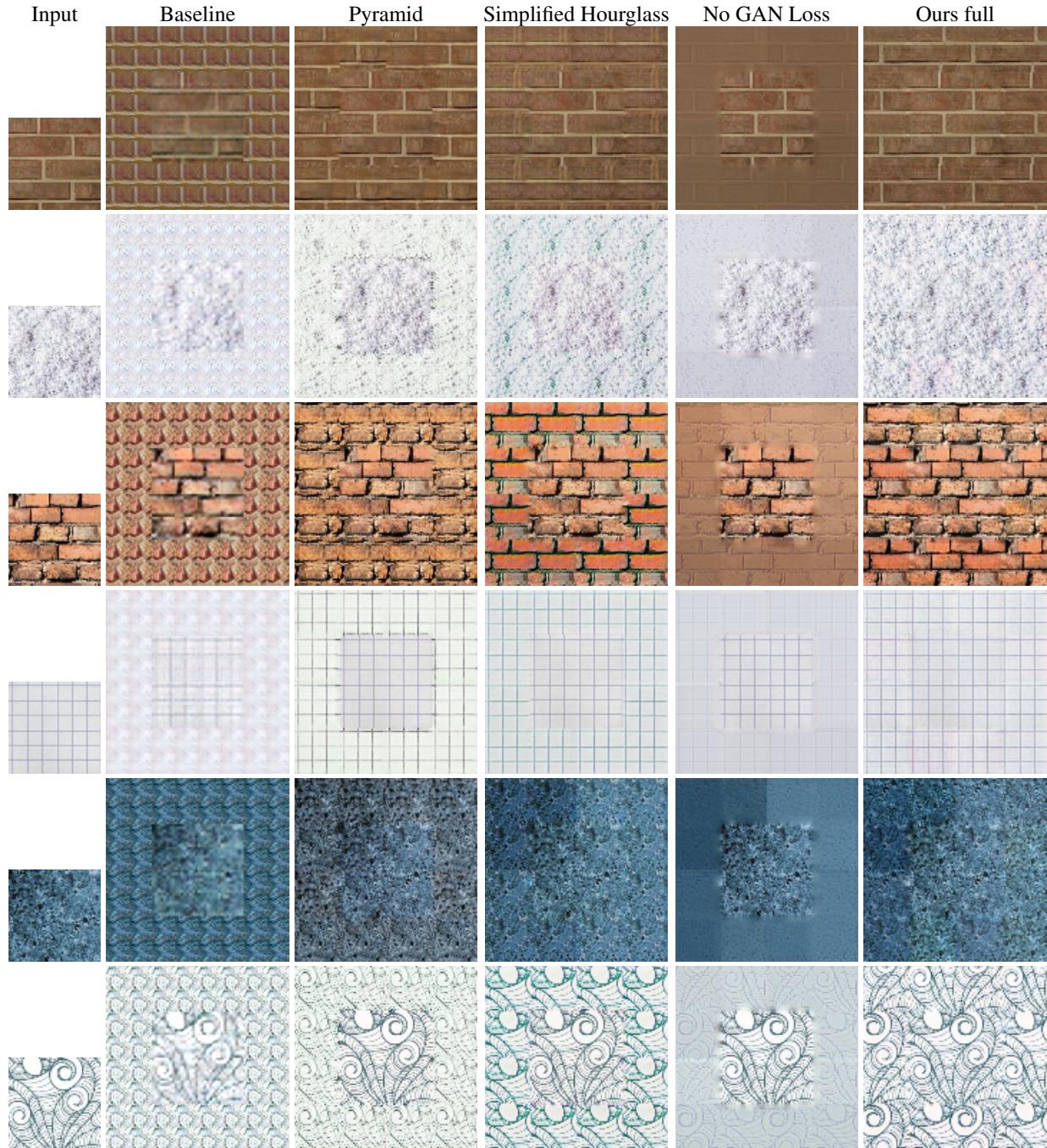


Figure 5. Qualitative results for our ablation study. We can see that the baseline cannot attend to multiple attention scales. The Pyramid approach allows to attend to multiple scales, but cannot properly fuse the multi-scale attention information. The simplified hourglass does not have convolutions between Transformer blocks, failing to handle the overall structure and color coherence. Finally we ablate our GAN loss and show that it significantly improves the perceptual quality of the results.

4.5. Limitations and future work

Fig. 6 illustrates the limitations of our approach. As our method relies on patch feature matching, we cannot handle perspective as shown in the first row nor non-(semi)-

periodic textures such as the unseen shapes of the bark in row 2 and the leather in row 3. Additionally, as we rely on an attention grid, our network may fail to align structures with limited training samples, creating small discontinuities

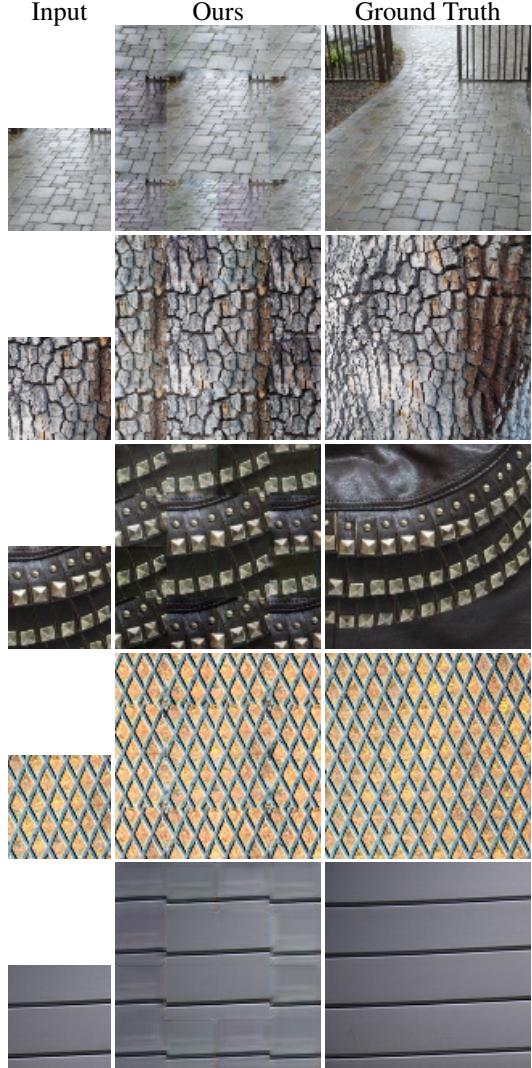


Figure 6. Example for limitations of our method. As we rely on patch mapping, our method does not handle well perspective or non periodic inputs (rows 1-3). With the limitation of patch operations, in some cases the alignment cannot be well corrected by our current approach (rows 4-5).

visible in row 4. These discontinuities are however mitigated by our multi-scale approach and processing in feature space.

In some particularly adverse cases where the texture is not self-repeating enough and the coarse grid does not allow for relevant patches retrieval, some alignment artifacts can appear such as in the last row. In future work, it would be interesting to augment the pool of patches to attend to or increase the number of scales/stages of the hierarchical hourglass network for the attention mechanism to select the best alignment.

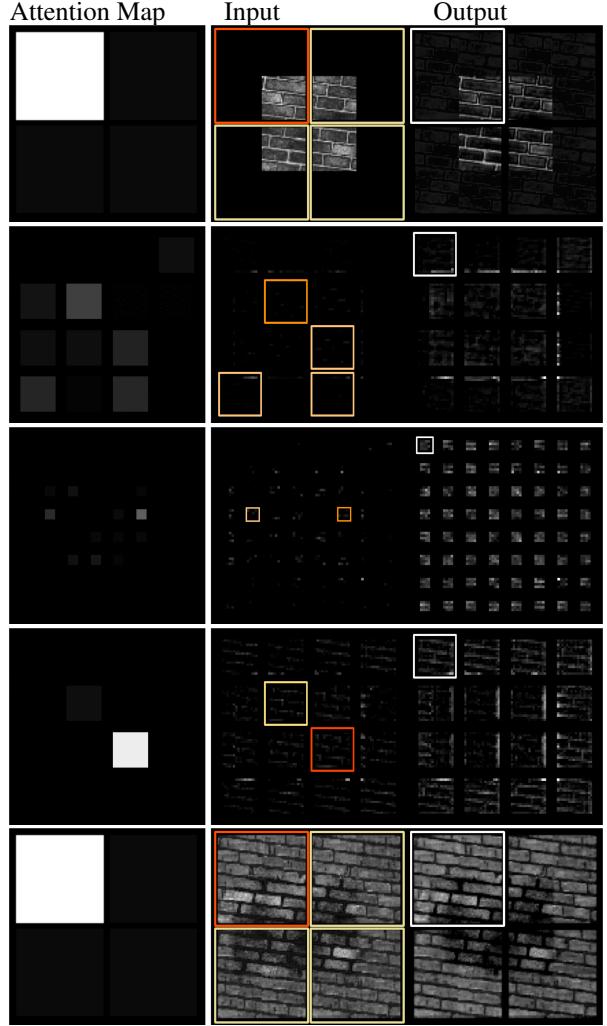


Figure 7. Input/output visualization for all 5 Transformer blocks of the model (the images represent averaged feature maps along channel dimension, and black partitioning lines are added to distinguish the patches). At each stage, the attention map denotes the weights for computing the top-leftmost patch of the output (white square highlight) based on the input patches. More red highlighted squares on the inputs represent a higher impact on the output.

5. Conclusions

We propose a novel U-Attention network with multi-stage hierarchical hourglass vision Transformers enabling $2\times$ texture synthesis on a broad range of texture patterns with a single trained network. Our approach uses features at multiple scales to synthesize patches with varying details while preserving the input texture appearance and runs in a few milliseconds. Our proposed network is a first step in leveraging the attention mechanism for texture synthesis, and we believe that the different components we demonstrate will be useful for broader research on attention-based image-to-image mapping.

Acknowledgment

We thank Jeff Fessler, Tammy Boubekeur, Paul Parneix, and Jerome Derel for insightful discussions.

References

- [1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. [1](#) [2](#)
- [2] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. *arXiv preprint arXiv:2102.08602*, 2021. [2](#)
- [3] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial GAN. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 469–477. PMLR, 06–11 Aug 2017. [2](#)
- [4] Ya-Liang Chang, Zhe Yu Liu, Kuan-Ying Lee, and Winston Hsu. Free-form video inpainting with 3d gated convolution and temporal patchgan. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9066–9075, 2019. [3](#) [5](#) [11](#) [12](#)
- [5] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12299–12310, 2021. [2](#)
- [6] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pre-training from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020. [2](#)
- [7] Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020. [2](#)
- [8] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. [5](#)
- [9] D. Dai, H. Riemenschneider, and L. Van Gool. The synthesizability of texture examples. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. [5](#)
- [10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. [2](#)
- [11] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001. [2](#)
- [12] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2:1033–1038 vol.2, 1999. [2](#)
- [13] B. Galerne, Y. Gousseau, and J. Morel. Random phase textures: Theory and synthesis. *IEEE Transactions on Image Processing*, 20(1):257–267, 2011. [2](#)
- [14] Bruno Galerne, Ares Lagae, Sylvain Lefebvre, and George Drettakis. Gabor noise by example. *ACM Trans. Graph.*, 31(4), July 2012. [2](#)
- [15] Bruno Galerne, Arthur Leclaire, and Lionel Moisan. Texton noise. *Computer Graphics Forum*, 36, 2017. [2](#)
- [16] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. *Advances in neural information processing systems*, 28:262–270, 2015. [1](#) [2](#) [5](#) [6](#)
- [17] Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler, and Djamchid Ghazanfarpour. Local random-phase noise for procedural texturing. *ACM Transactions on Graphics (TOG)*, 33:1 – 11, 2014. [2](#)
- [18] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995. [2](#)
- [19] Eric Heitz and Fabrice Neyret. High-performance by-example noise using a histogram-preserving blending operator. *Proc. ACM Comput. Graph. Interact. Tech.*, 1:31:1–31:25, 2018. [2](#)
- [20] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Learning a neural 3d texture space from 2d exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8356 – 8364, June 2020. [2](#)
- [21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017. [2](#)
- [22] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021. [2](#)
- [23] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016. [2](#)
- [24] Alexandre Kaspar, Boris Neubert, Dani Lischinski, Mark Pauly, and Johannes Kopf. Self tuning texture optimization. *Computer Graphics Forum*, 34, 2015. [2](#) [5](#) [6](#)
- [25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. [5](#)
- [26] Roland Kwitt and Peter Meerwald. Salzburg texture image database (stex), Feb 2008. [5](#)
- [27] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems*, 2017. [2](#) [5](#) [6](#)
- [28] Guilin Liu, Rohan Taori, Ting-Chun Wang, Zhiding Yu, Shiqiu Liu, Fitzsum A Reda, Karan Sapra, Andrew Tao, and Bryan Catanzaro. Transposer: Universal texture synthesis using feature maps as transposed convolution filter. *arXiv preprint arXiv:2007.07243*, 2020. [1](#) [2](#) [3](#) [5](#)

- [29] Morteza Mardani, Guilin Liu, Aysegul Dundar, Shiqiu Liu, Andrew Tao, and Bryan Catanzaro. Neural ffts for universal texture image synthesis. *Advances in Neural Information Processing Systems*, 33, 2020. [1](#), [2](#), [5](#)
- [30] Eyvind Niklasson, Alexander Mordvintsev, Ettore Randazzo, and Michael Levin. Self-organising textures. *Distill*, 6(2):e00027–003, 2021. [2](#)
- [31] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018. [2](#)
- [32] Lara Raad, Axel Davy, Agnès Desolneux, and Jean-Michel Morel. A survey of exemplar-based texture synthesis. *CoRR*, abs/1707.07184, 2017. [2](#)
- [33] Carlos Rodriguez-Pardo, Sergio Suja, David Pascual, Jorge Lopez-Moreno, and Elena Garces. Automatic extraction and synthesis of regular repeatable patterns. *Computers & Graphics*, 83:33–41, 2019. [2](#)
- [34] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4569–4579, 2019. [2](#)
- [35] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *International Conference on Machine Learning*, pages 10183–10192. PMLR, 2021. [2](#)
- [36] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016. [2](#)
- [37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. [1](#), [2](#), [3](#)
- [38] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Loddon Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*, 2020. [2](#)
- [39] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. [2](#)
- [40] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. [5](#), [6](#)
- [41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. [5](#)
- [42] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baineng Guo. Learning texture transformer network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5791–5800, 2020. [2](#)
- [43] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. Learning joint spatial-temporal transformations for video inpainting. In *European Conference on Computer Vision*, pages 528–543. Springer, 2020. [2](#), [3](#), [5](#), [11](#), [12](#)
- [44] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019. [3](#)
- [45] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. [5](#)
- [46] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Non-stationary texture synthesis by adversarial expansion. *ACM Transactions on Graphics (TOG)*, 37:1 – 13, 2018. [1](#), [2](#)

Supplemental Material

U-Attention to Textures: Hierarchical Hourglass Vision Transformer for Universal Texture Synthesis

This supplemental material presents:

- Details of our proposed network architecture (Sec. A)
- Additional synthesis results on a wide range of textures (Sec. B)
- Details of ablated designs and additional results for the ablation study (Sec. C)
- Texture synthesis of $128 \rightarrow 256$ (Sec. D)

A. Network Architecture Details

Table 1 presents details of our proposed U-Attention network. We zero-pad the small texture exemplar (to the output size) as the input and solve the texture inpainting problem. Note that although Tab. 1 uses an overall input/output dimension of 128×128 as an example, our network is not constrained to a particular size of input/output. Details of the temporal-patch adversarial loss [4, 43] we adopted are in Tab. 2.

B. Additional Synthesis Results

This section presents additional synthesis results of the proposed model compared to other methods and input-output pairs of a wide range of textures using the proposed model.

B.1. Comparisons to other methods

Figure 1 provides additional comparison results of the proposed approach to previous works.

B.2. Results of our model for different textures

Figure 2 and Figure 3 present more test results from our single trained network.

C. Ablation Study

This section presents details about the “pyramid” and “simplified hourglass” network architectures introduced in the ablation study, along with additional synthesis results.

C.1. Pyramid and simplified hourglass architectures

We demonstrate the effective components of the proposed network for ablation study. The “pyramid” network shown in Fig. 4 represents a multi-stage hierarchical Transformer network with an encoder, 3 Transformer blocks, and a decoder. The Transformer blocks take sequences of patches of feature maps as inputs. We introduce a multi-scale partition of the feature map between Transformer blocks to form input patches of different scales for different Transformers. Specifically, the input patch size presents a pyramid-like shrinking pattern with $2 \times$ smaller spatial extent between Transformer stages.

This “pyramid” network design establishes a coarse-to-fine patch mapping scheme that combines attention between patches of varying scales for global pattern outline and local refinements of detailed structures. In the extreme form of the pyramid, the last stage of the Transformer would take sequences of patches of size 1×1 as input, and the results from the pyramid network might be improved at the expense of computation.

The “simplified hourglass” in Fig. 5 concatenates the pyramid network with a mirrored pyramid network and is complemented with skip connections to propagate and fuse information. The mirrored pyramid network at the second half of the simplified hourglass takes progressively larger patches ($2 \times$ larger in each spatial dimension between consecutive Transformer stages) with shorter sequence lengths as inputs for later Transformer stages. We also add skip connections that propagate output from an earlier stage to a later stage of the network, concatenate and fuse the outputs from the two stages with two layers of 1×1 convolutions to form the input of the next Transformer.

	Operation	Filter Size	# Channels	Stride/Up Factor	Nonlinearity	Input	Output
Encoder	2D Conv	3×3	$3 \rightarrow 16$	1	LeakyReLU (0.2)	128×128	128×128
	2D Conv	1×1	$16 \rightarrow 16$	1	LeakyReLU (0.2)	128×128	128×128
T-Block1	Partition_1	-	-	-	-	128×128	$4 \times 64 \times 64$
	TransformerLayers_1	$1 \times 1, 3 \times 3$	16	1	LeakyReLU (0.2)	$4 \times 64 \times 64$	$4 \times 64 \times 64$
	ArrangeBack_1	-	-	-	-	$4 \times 64 \times 64$	128×128
ConvDown1	2D Conv	4×4	$16 \rightarrow 64$	2	LeakyReLU (0.2)	128×128	64×64
	2D Conv	1×1	$64 \rightarrow 64$	1	LeakyReLU (0.2)	64×64	64×64
T-Block2	Partition_2	-	-	-	-	64×64	$16 \times 16 \times 16$
	TransformerLayers_2	$1 \times 1, 3 \times 3$	64	1	LeakyReLU (0.2)	$16 \times 16 \times 16$	$16 \times 16 \times 16$
	ArrangeBack_2	-	-	-	-	$16 \times 16 \times 16$	64×64
ConvDown2	2D Conv	4×4	$64 \rightarrow 256$	2	LeakyReLU (0.2)	64×64	32×32
	2D Conv	1×1	$256 \rightarrow 256$	1	LeakyReLU (0.2)	32×32	32×32
T-Block3	Partition_3	-	-	-	-	32×32	$64 \times 4 \times 4$
	TransformerLayers_3	$1 \times 1, 1 \times 1$	256	1	LeakyReLU (0.2)	$64 \times 4 \times 4$	$64 \times 4 \times 4$
	ArrangeBack_3	-	-	-	-	$64 \times 4 \times 4$	32×32
ConvUp1	BilinearUpSample	-	256	2	-	32×32	64×64
	2D Conv	1×1	$256 \rightarrow 64$	1	LeakyReLU (0.2)	64×64	64×64
	2D Conv	1×1	$64 \rightarrow 64$	1	LeakyReLU (0.2)	64×64	64×64
ConvFuse1	2D Conv	1×1	$128 \rightarrow 64$	1	LeakyReLU (0.2)	64×64	64×64
	2D Conv	1×1	$64 \rightarrow 64$	1	LeakyReLU (0.2)	64×64	64×64
T-Block4	Partition_2	-	-	-	-	64×64	$16 \times 16 \times 16$
	TransformerLayers_4	$1 \times 1, 3 \times 3$	64	1	LeakyReLU (0.2)	$16 \times 16 \times 16$	$16 \times 16 \times 16$
	ArrangeBack_2	-	-	-	-	$16 \times 16 \times 16$	64×64
ConvUp2	BilinearUpSample	-	64	2	-	64×64	128×128
	2D Conv	1×1	$64 \rightarrow 16$	1	LeakyReLU (0.2)	128×128	128×128
	2D Conv	1×1	$16 \rightarrow 16$	1	LeakyReLU (0.2)	128×128	128×128
ConvFuse2	2D Conv	1×1	$32 \rightarrow 16$	1	LeakyReLU (0.2)	128×128	128×128
	2D Conv	1×1	$16 \rightarrow 16$	1	LeakyReLU (0.2)	128×128	128×128
T-Block5	Partition_1	-	-	-	-	128×128	$4 \times 64 \times 64$
	TransformerLayers_5	$1 \times 1, 3 \times 3$	16	1	LeakyReLU (0.2)	$4 \times 64 \times 64$	$4 \times 64 \times 64$
	ArrangeBack_1	-	-	-	-	$4 \times 64 \times 64$	128×128
Decoder	2D Conv	3×3	$16 \rightarrow 3$	1	LeakyReLU (0.2)	128×128	128×128
	2D Conv	1×1	$3 \rightarrow 3$	1	Tanh	128×128	128×128

Table 1. Details of the proposed U-Attention network architecture and parameters. The “Input” and “Output” denotes the spatial dimensions of the input and output at different stages of the network in forms of $H_i \times W_i$ for whole images or $P_i^2 \times \frac{H_i}{P_i} \times \frac{W_i}{P_i}$ for a sequence of P_i^2 patches. “ConvFuse1” fuses outputs of “T-Block2” and “ConvUp1” to be the input of “T-Block4”. Similarly, “ConvFuse2” fuses outputs of “T-Block1” and “ConvUp2” to be the input of “T-Block5”. “Partition_1” represents the multi-scale partition for the proposed multi-scale attention in hierarchical hourglass Transformers. “ArrangeBack_1” denotes the operation that rearranges a sequence of patches back to whole feature maps. “TransformerLayers_1” are composed of two stacked Transformers.

Operation	Filter Size	# Channels	Stride	Nonlinearity
3D Conv	$3 \times 3 \times 3$	$3 \rightarrow 32$	(1, 1, 1)	LeakyReLU (0.2)
3D Conv	$3 \times 3 \times 3$	$32 \rightarrow 64$	(1, 1, 1)	LeakyReLU (0.2)
3D Conv	$3 \times 3 \times 3$	$64 \rightarrow 128$	(1, 1, 1)	LeakyReLU (0.2)
3D Conv	$3 \times 3 \times 3$	$128 \rightarrow 128$	(1, 1, 1)	LeakyReLU (0.2)
3D Conv	$3 \times 3 \times 3$	$128 \rightarrow 128$	(1, 1, 1)	LeakyReLU (0.2)
3D Conv	$3 \times 3 \times 3$	$128 \rightarrow 128$	(1, 1, 1)	-

Table 2. Details of the temporal-patch adversarial loss (T-PatchGAN) [4, 43]. The 3D convolution layers for T-PatchGAN use spectral normalization to stabilize training [4, 43].

This hourglass network design further regularizes the smallest output patches from the bottleneck of the hourglass and helps resolve blocking artifacts caused by patch-based operations of the pyramid network. The hourglass-like scale change of the patches enables coarse-to-fine and fine-back-to-coarse mapping.

Note that for both pyramid and simplified hourglass networks, the input feature map (before partitioning and arranging the patches as a sequence) for each Transformer stage are of the same spatial extent and same channel dimension, while in our U-Attention network the spatial extent and channel dimension of the input feature maps do vary with depth, as shown in Tab. 1.

C.2. More results of ablation study

In Figure 6 we show that our proposed network outperforms the multiple Transformer blocks baseline network. We further demonstrate the contribution of each component of our design through more examples to support our ablation study.

D. 128 to 256 Synthesis

We fine-tune the network that is pre-trained for 64 to 128 synthesis with input and target images of larger size (256 instead of 128) for 40 epochs (about 4 days) and present 128 to 256 synthesis results. The 128 to 256 results in Fig. 7 are very similar to the quality of 64 to 128 with a higher resolution.

We believe that for larger texture synthesis, we could extend our proposed network to be composed of more stages of multi-scale attention networks and train the extended network for further improvement of performance.

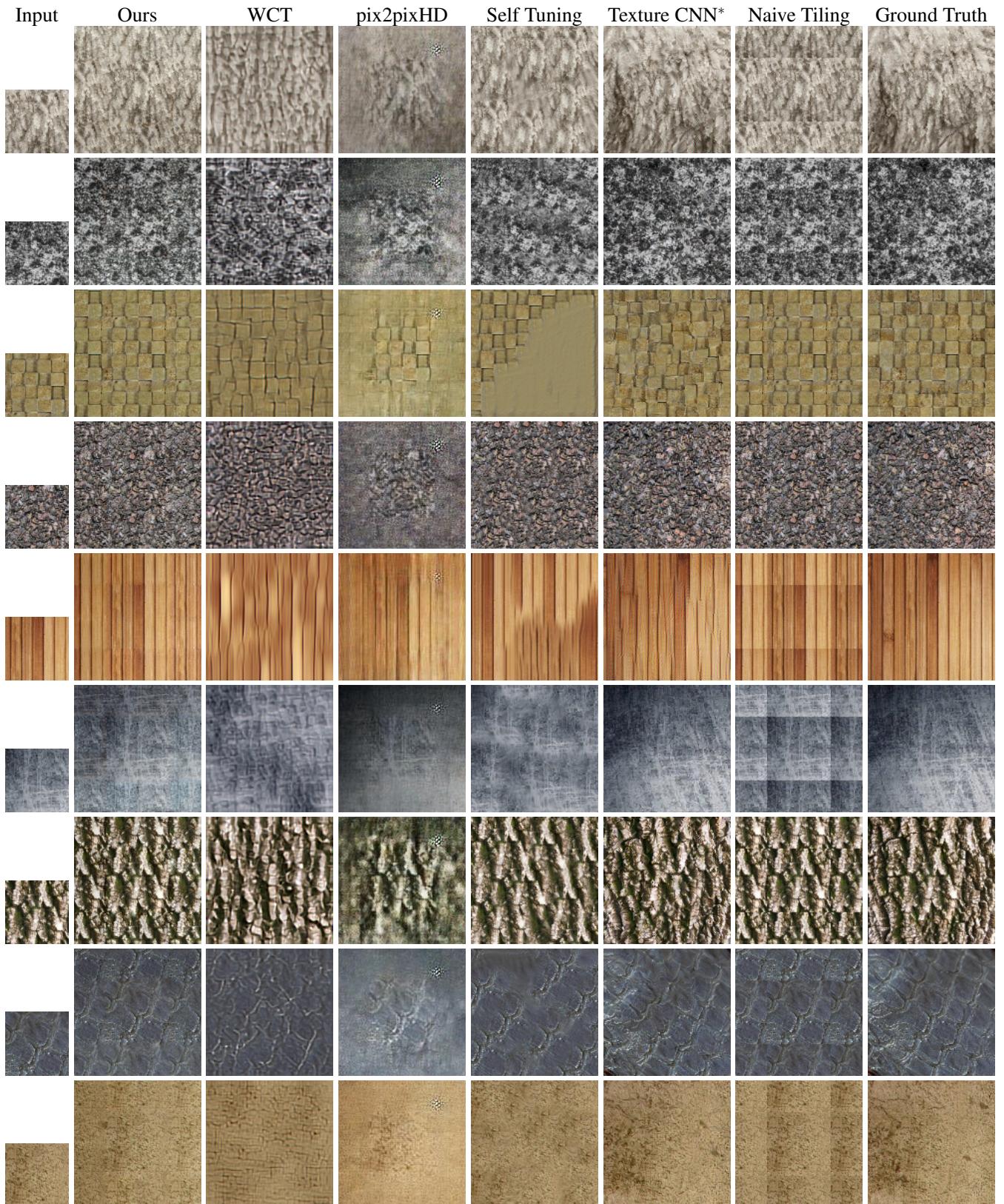


Figure 1. Comparison to previous works demonstrates that our approach can handle textures of varying patterns and structures, and preserve the coherent color shift and structural details.



Figure 2. Synthesis results of our U-Attention network demonstrate that our approach generalizes to a broad range of textures with varieties of randomness and structure using one trained network.

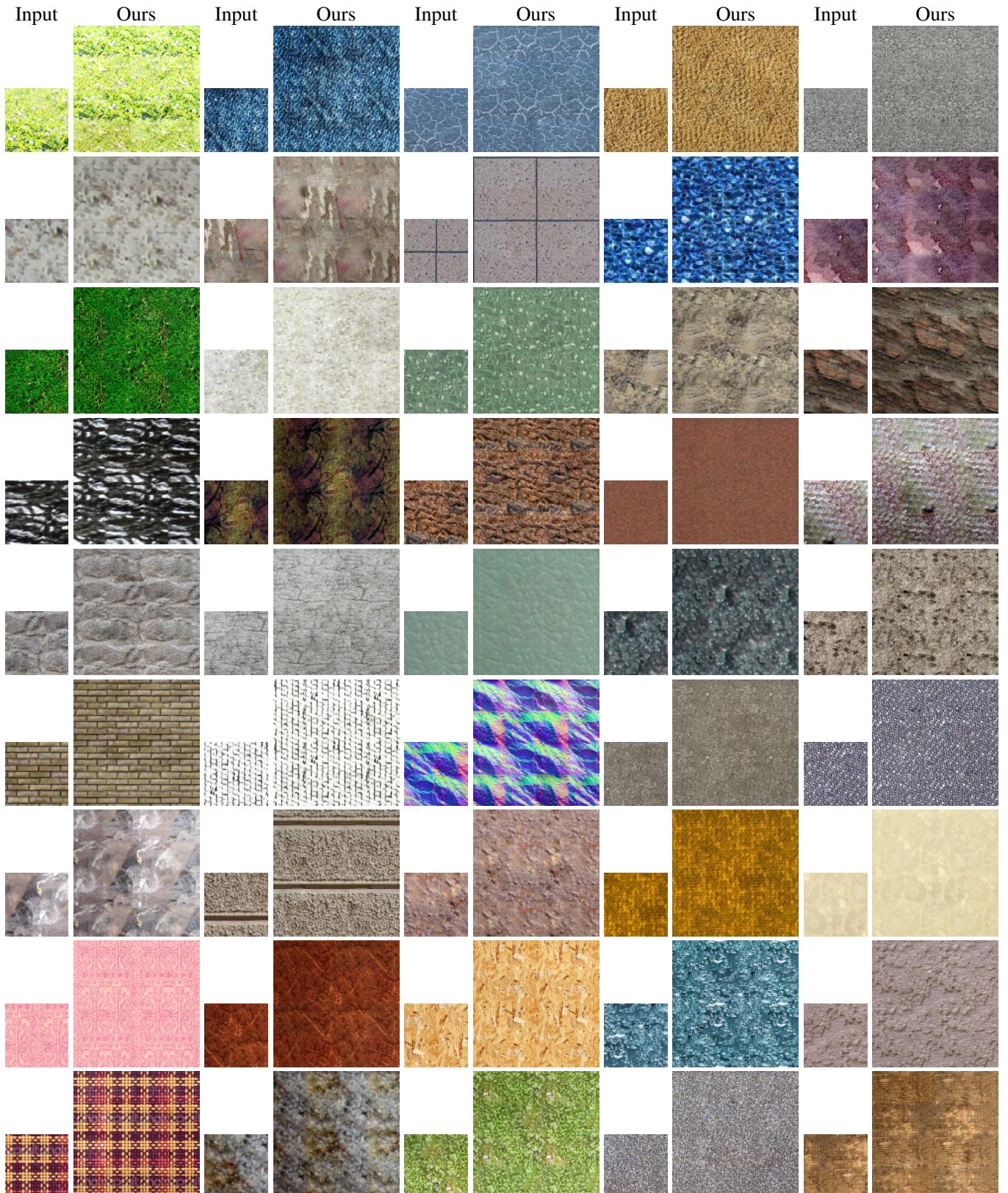


Figure 3. More synthesis results of our U-Attention network demonstrate that our approach generalizes to a broad range of textures with varieties of randomness and structure using one trained network.

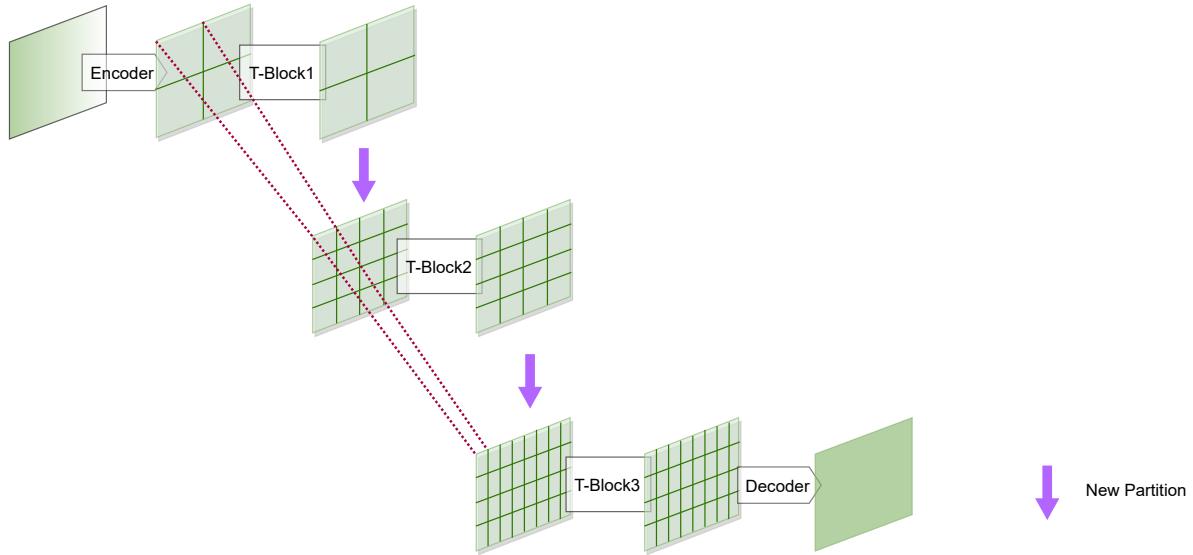


Figure 4. The “pyramid” network. Each Transformer block takes the whole feature maps as inputs, and we partition the feature maps to be sequences of patches of shrinking ($2 \times$ smaller in each dimension) spatial extents for consecutive stages of the network.

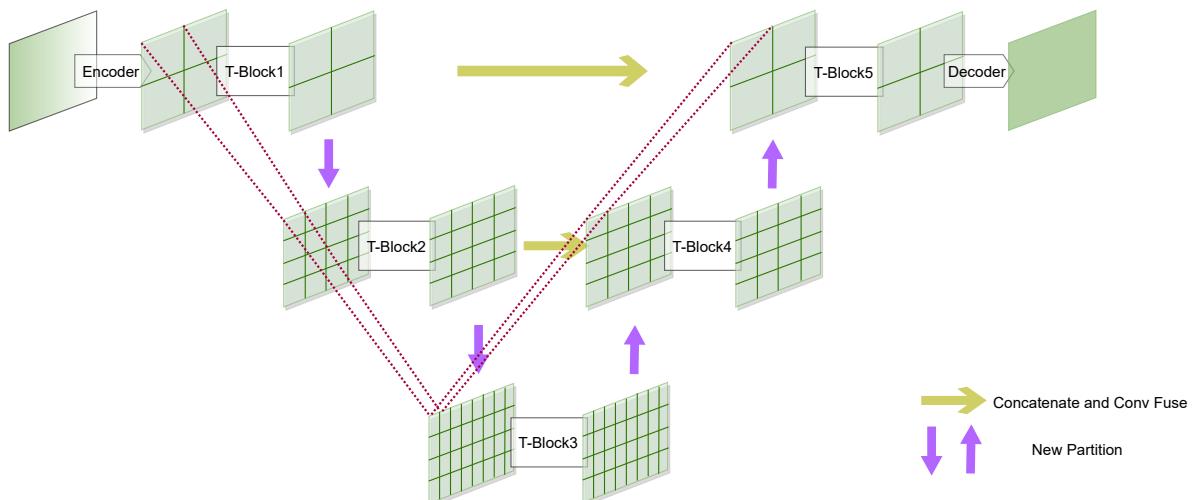


Figure 5. The “simplified hourglass” network. Each Transformer block takes the whole feature maps as inputs, and we partition the feature maps to be sequences of patches of shrinking or enlarging spatial extents ($2 \times$ smaller or larger in each dimension) for different stages of the hourglass network.



Figure 6. More qualitative synthesis results for our ablation study.

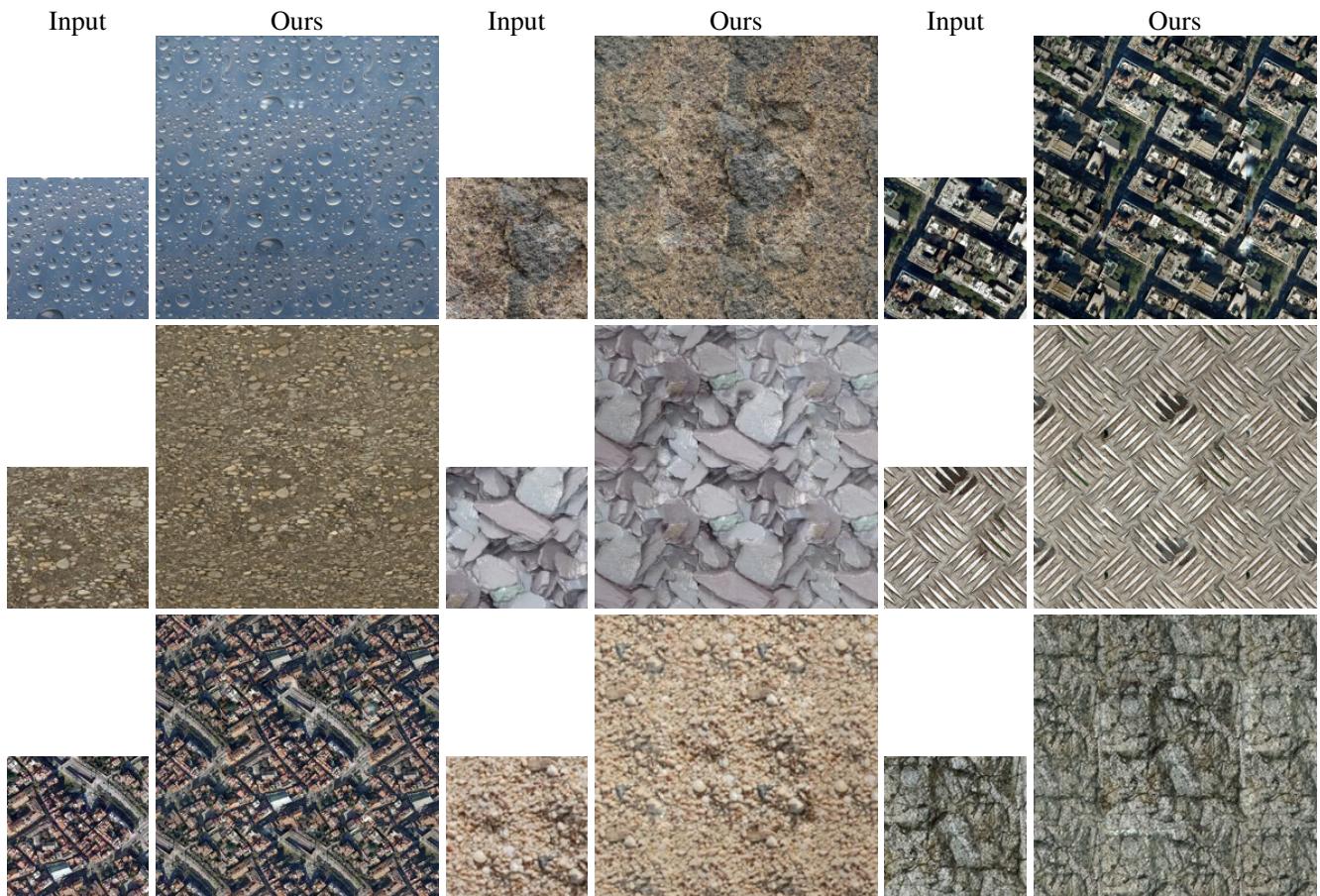


Figure 7. Higher-resolution synthesis results for image of size 128 to 256.