Hindawi
Security and Communication Networks
Volume 2022, Article ID 1626747, 11 pages
https://doi.org/10.1155/2022/1626747

Research Article

Texture Synthesizability Assessment via Deep Siamese-Type Network

Chuanyan Hao,[1] **Zhi-Xin Yang**,[2] **Liping He**,[1] **and Weimin Wu**[1]

[1]*School of Education Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
[2]*State Key Laboratory of Internet of Things for Smart City, Department of Electromechanical Engineering, University of Macau, Macao, China*

Correspondence should be addressed to Zhi-Xin Yang; zxyang@um.edu.mo

Received 20 December 2021; Revised 19 January 2022; Accepted 27 January 2022; Published 27 February 2022

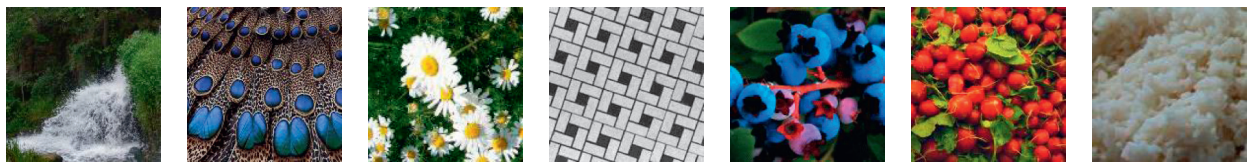
Academic Editor: Zhili Zhou

Example-based texture synthesis plays a significant role in many fields, including computer graphics, computer vision, multimedia, and image and video editing and processing. However, it is not easy for all textures to synthesize high-quality outputs of any size from a small input example. Hence, the assessment of the synthesizability of the example textures deserves more attention. Inspired by the broad studies in image quality assessment, we propose a texture synthesizability assessment approach based on a deep Siamese-type network. To our best knowledge, this is the first attempt to evaluate the synthesizability of sample textures through end-to-end training. We first train a Siamese-type network to compare the example texture and the synthesized texture in terms of their similarity and then transfer the experience knowledge obtained in the Siamese-type network to a traditional CNN by fine-tuning, so that to give an absolute score to a single example texture, representing its synthesizability. Not relying on laborious human selection and annotation, these synthesized textures can be generated automatically by example-based synthesis algorithms. We demonstrate that our approach is completely data-driven without hand-crafted features and/or prior knowledge in the field of expertise. Experiments show that our approach improves the accuracy of texture synthesizability assessment qualitatively and quantitatively and outperforms the manual feature-based method.

1. Introduction

Although textures can be obtained by various methods such as manual drawing, photography, modeling, and simulation, texture synthesis, especially, example-based texture synthesis, remains one of the most powerful approaches due to its advantages on universality, efficiency, and quality. However, not all textures can be equally well reproduced in this way, and thus, it is necessary to investigate how well an example texture can be synthesized. As far as we know, some work has been done to explore this issue though not much. One kind focuses on the quality assessment for the synthesized textures, that is, the output texture after synthesis [1, 2]. Another one attempts to evaluate whether an example texture/input texture is suitable for example-based synthesis methods before synthesis [3, 4], called texture synthesizability prediction in [3]. Our work falls into the latter category.

To achieve this goal, Dai et al. [3] design a group of hand-crafted features for qualitative texture analysis and use Random Forest to train a regression model to predict the synthesizability scores. Figure 1 shows an example of texture synthesizability assessment. As seen, a specific texture could be given a synthesizability score after computation, and a higher score means a better fit for the example-based synthesis algorithms. Nevertheless, the manual feature-based method limits its application, making it lack generality. Recently, deep convolutional neural networks (deep CNNs) have been successful in various fields [5–8]. It also motivates us to investigate their potential application to the texture synthesizability assessment problem. But shallow CNNs cannot achieve ideal performance while deep CNNs require large enough datasets. As a matter of fact, Dai et al. [3] have offered a fairly large texture dataset of 21,302 texture samples along with synthesizability annotations (the ETH



(a) (b) (c) (d) (e) (f) (g)

Figure 1: An example of texture synthesizability assessment. The number under each example texture is its synthesizability score, varying in [0, 1], and a higher value means that it is easier to synthesize by the example-based synthesis algorithms. (a) Waterfall: 0.18. (b) Peacock hair: 0.35. (c) Chrysanthemum: 0.53. (d) Floor jigsaw: 0.64. (e) Blueberry: 0.78. (f) Cherry tomato: 0.88. (g) Rice: 0.93.

synthesizability dataset), but for deeper and wider CNNs, larger datasets are necessary.

Inspired by the intensive studies on image quality assessment [9–18], we propose a deep neural network-based approach to address these issues. The main idea is that although it is difficult to obtain human-annotated texture data, synthesized textures can be generated easily in a fully automated manner. That is, we can produce synthesized texture sets in which, though there is neither an absolute score for the synthesizability of the sample texture nor a quality score for the synthesized texture, for any pair of textures (samples and synthesized ones), we could set the measures between them in terms of their similarity. So far, it is easy to understand that the Siamese-type network is a good option to make our regression model to learn to compare textures according to the similarity. Then, the knowledge learned from the Siamese-type network is transferred to a traditional CNN, in our case, a single branch of the Siamese-type network, which is fine-tuned on the texture database offered by Dai et al. [3] to enhance the accuracy of texture synthesizability assessment. Experiments show that our regression model significantly improves results qualitatively and quantitatively.

In summary, our main contributions are as follows: (1) we propose a deep Siamese-type network-based approach for texture synthesizability assessment, which allows for end-to-end optimization for feature extraction, fusion, and regression; (2) we offer a recommendation mechanism for suggesting the best-matched synthesis algorithm; (3) such a methodology could also be considered for other texture analysis, such as the synthesizability of nonstationary textures.

2. Related Work

2.1. Example-Based Texture Synthesis. Over the recent two decades, a great deal of work is presented on example-based texture synthesis. A detailed survey could be referred to in [19]. The dominant approaches are patch-based algorithms [20–22] due to their more superior quality and performance and thus still the most popular methods for texture synthesis util now. One of the most representative works is the Quilting algorithm [20]. It generates the new textures by copying the whole patches from the input one at a time. Then, Kwatra et al. [21] generalize the uniform patch sizes to arbitrarily shaped patches which are determined entirely by performing a minimum cost graphcut. By using a fast randomized patch search strategy, Barnes et al. propose the PatchMatch algorithm [23], which successfully accelerates this family of work and has been extended to various graphics and vision tasks, such as image melding [24] and large-scale texture synthesis [22]. However, these classical example-based approaches are unable to reproduce good outputs for textures with large-scale structures and/or inhomogeneity, which violate the assumption of Markov Random Field (MRF). Therefore, some guidance control methods [25, 26] are shown to be effective in middle scale, large-scale, and inhomogeneous textures.

With the fast development of deep learning techniques, approaches based on deep neural networks have also been raised in the field of texture and image synthesis. The pioneering work was conducted by Gatys et al. [27]. Moreover, they extend their work to the artistic style transfer task [28] by adding a content loss to the Gram-based style loss. Recently, Generative Adversarial Networks (GANs) [29] have been employed directly to perform texture synthesis and style transfer [30–34], but many of them are trained for a specific texture. For instance, the work in [32] is mainly limited to periodic textures; the method presented in [33] shows promising results for nonstationary textures; Hertz et al. [34] use deep GANs for geometric texture synthesis. Meanwhile, this technique has been also transferred to other related tasks, including textureGAN [35], texture mixer [36], tileGAN [37], and loss study for texture synthesis [38].

2.2. Image Quality Assessment. Studies very relevant to our work also include achievements in image quality assessment (IQA). In general, IQA approaches can be divided into three categories based on the availability of reference images. While full reference image is available, its IQA method is called full reference IQA (FR-IQA); correspondingly, no information available means no reference IQA (NR-IQA); lying between them is named as reduced reference IQA (RR-IQA), whose reference information is usually represented by a set of features extracted from original reference images. Among them, early work is mainly based on traditional machine learning algorithms, but they only show good performance on small data sets and have a large overfitting risk existing. Comparatively, deep learning-based methods owe a wide range of applications, especially when the data set is large. Since the latter one is most related to our work, we make a brief review of it.

For IQA problems, deep learning-based approaches [9–18] can learn the mapping relationship between image and image quality end-to-end, resulting in superior performance over traditional ones. The work of [9] introduces CNN to the design of IQA models. Bosse et al. [10] train a deep Siamese network for both FR-IQA and NR-IQA. Their work is purely data-driven and achieves end-to-end optimization for feature extraction and regression. Similarly, Ma et al. [14] propose an IQA model supporting multitask end-to-end optimization. Kim et al. [11] present a deep image quality assessor, called DIQA. Its training includes two stages: the first stage trains a CNN to learn the objective errors; the second stage fine-tunes the CNN model using human subjective scores to improve the prediction accuracy. Pan et al.'s deep CNN comprises two parts: generating network and pooling network [16]. Zhang et al. [15] demonstrate a deep bilinear model suitable for both synthetic and true distortions. Another very interesting job is RankIQA [13], which implements the no-reference image quality assessment by learning from rankings. In the last two years, metalearning [17] and Transformer architecture [18] have also been introduced into the IQA job. Inspired by these excellent algorithms, in this article, we take the advantage of Siamese network and fine-tuning mechanism to explore the methodology of texture synthesizability assessment.

3. Our Approach

In this section, we describe our method to utilize synthetically generated similarity comparisons for texture synthesizability assessment. We start with a general framework for our approach, then to move on to the Siamese-type network architecture to learn from similarity comparison, and finally to the fine-tuning stage.

3.1. Overview. In this article, we attempt to explore a more general and accurate strategy to evaluate texture synthesizability. Enlightened by the method of RankIQA [13], we take advantage of large, unlabeled databases where synthesized textures could be easily generated by utilizing example-based synthesis algorithms. Corresponding to the architecture of the Siamese network, we call the sample texture/input texture as the example texture while the synthesized one as the reference texture. In order to be more robust, we collect three synthesized textures taken from three different example-based methods and generate the dataset of the reference texture patches by random sampling. The thinking behind it is that if an example texture has good synthesizability, it would match any three example-based synthesis algorithms. In other words, the synthesized texture has a high similarity with the example texture, and the similarity measurement will output a lower value. And vice versa, bad synthesizability means that at least one synthesis algorithm, sometimes two, or even three, does not produce a good result. In this case, the similarity measurement between the example texture and the reference one will output a higher value.

After learning the similarity between the example texture and its reference one, we can use fine-tuning on small texture synthesizability datasets [3] to give the absolute score. Since now much larger datasets with synthesized textures could be available, we are able to obtain more accurate synthesizability scores by learning a distance embedding with deeper and wider networks. An overall pipeline of our approach is shown in Figure 2, including three steps: synthesize reference textures, train a Siamese-type network, and fine-tune on the ETH [3] dataset.

3.2. Synthesize Reference Textures. The definition of texture synthesizability in this paper is primarily for the texture under the MRF hypothesis, and the selection of the synthesis approaches falls into the category of example-based methods. While deep neural network-based methods have emerged, they are still less universal than classical methods and generally require more hardware and time costs. As a result, we produced similar textures using three algorithms that are most popular and easiest to implement and run: Quilting [20], Graphcut [21], and Random Search [22, 23]. Then, it is conceivable that a texture from near-regular textures, nonstationary textures, or inhomogeneous textures possibly has a low score of its synthesizability. Figure 3 gives an example. With the synthesizability score decreasing from top to bottom, the three synthesis methods yield different visual results. For the first texture with a high score (top), all of its three outputs are visually near to the input texture, almost without artifacts. For the second one, one method, Graphcut [21], fails to produce a good result. Alike, two methods do not work well for the third texture, Quilting [20] and Random Search [22], and all the three methods do not apply to the fourth texture. It can be seen that sampling reference texture patches from the outputs of these three methods would be beneficial to the generalization and robustness of the dataset.

3.3. Train RGB-2Channel Siamese-Type Network for Similarity Learning. To investigate our method, it is necessary to learn more about the Siamese network. Different from the traditional networks, the Siamese network [10, 13] combines two branch networks with shared weights and has an output layer network. During training, the two branches extract and learn the features of the images, and the shared output layer calculates the difference between the two different feature vectors. Generally, single column neural networks compute the absolute error against the estimated label with the labeled input, while the Siamese network computes the relative error between the two inputs. In essence, the implementation of the Siamese architecture has only one branch network, into which the two inputs are successively fed to extract features. Hence, pseudo-Siamese networks whose two branches have different weights emerged. Although it is more flexible than the traditional Siamese network, it is also more difficult to train.

Therefore, Sergey Zagoruyko and Komodakis [39] proposed another improvement scheme. In this method, two image patches are combined together by using a two-

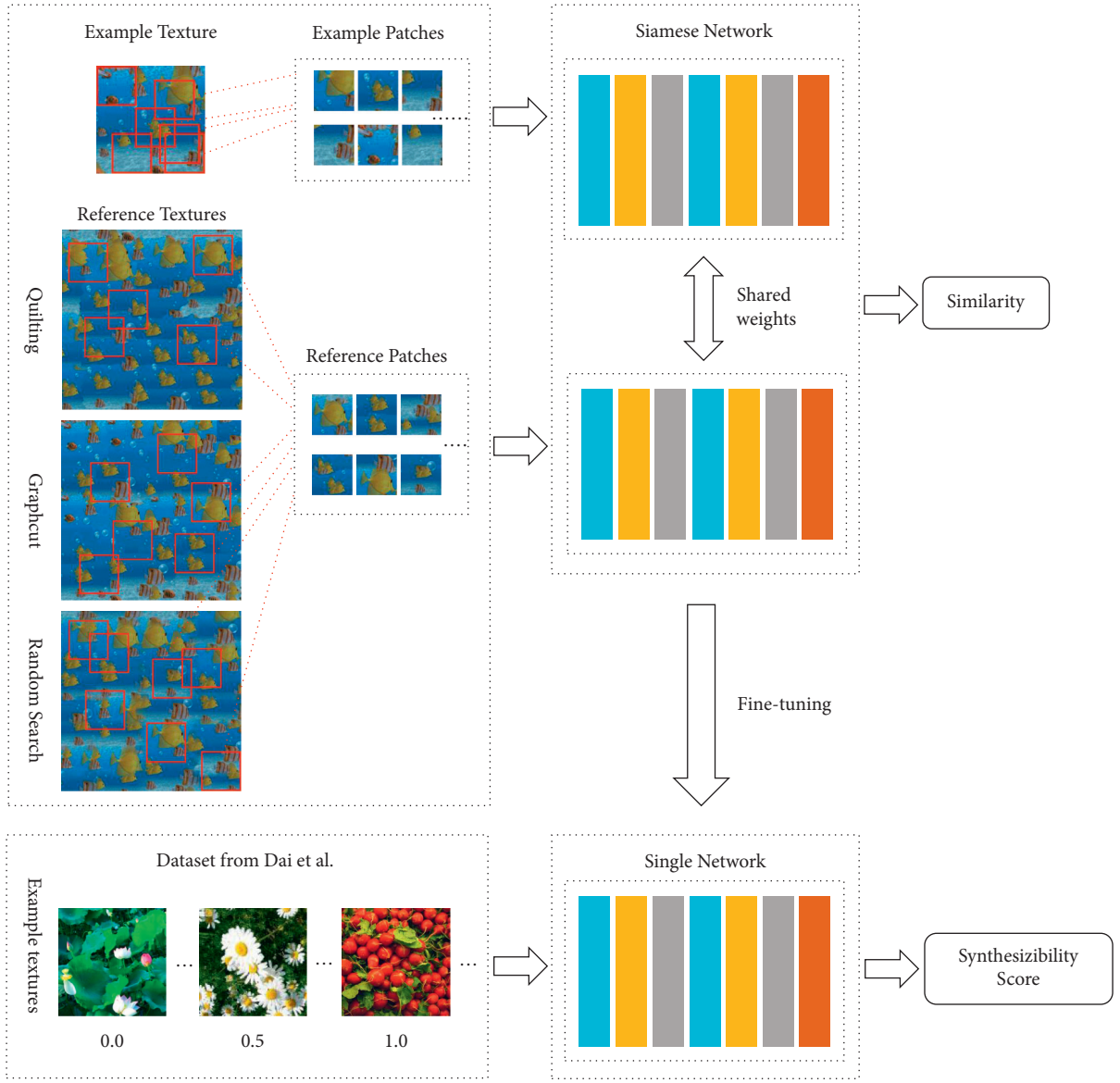


Figure 2: An overall pipeline of our method. It trains a deep Siamese-type network from a dataset with reference textures. These reference textures can be easily generated by using example-based synthesis algorithms. The network parameters are then transferred to the single branch regression network for fine-tuning.

channel network structure, and the similarity computation is conducted only through one-time training. In fact, it is the process of doing image channel compression. That is, two grayscale image patches of an input pair are compressed into a two-channel image, which is then sent into the neural network, and the information of the image patches is fused together right from the first layer of the network. The similarity metric is obtained from the activation of the last layer. Such a network architecture further accelerates the training and shows better performance. These results bring to our attention the significance of combining information from image pairs right from the front layers of the network. For texture synthesizability assessment, we hence extend a two-channel Siamese-type network to an RGB-two-channel Siamese-type network for joint feature extraction. This has the advantage of avoiding manual feature fusion, for example, using subtraction and concatenation operations as in [10], and specifying some statistical functions as in [40]. Finally, the fused features are input to the regression layer. The structure of the proposed network is shown in Figure 4 and will be elaborated in the following.

Due to its superior performance and achievements in a variety of computer vision tasks, VGGNet [41] is selected as the backbone structure for our proposed network. Generally, the input images to the VGG network have a fixed size of 224×224 pixels. But for classical example-based texture synthesis algorithms, the usually used patch sizes vary among 16×16, 32×32, 64×64, $1\ 28 \times 128$, and sometimes 256×256 pixels. In order to make the network adaptable to smaller and multiscale input sizes, we add three layers (conv3-32, conv3-32, and max-pool) in front of the original VGG-16, similar to the operations in [10], and insert the spatial

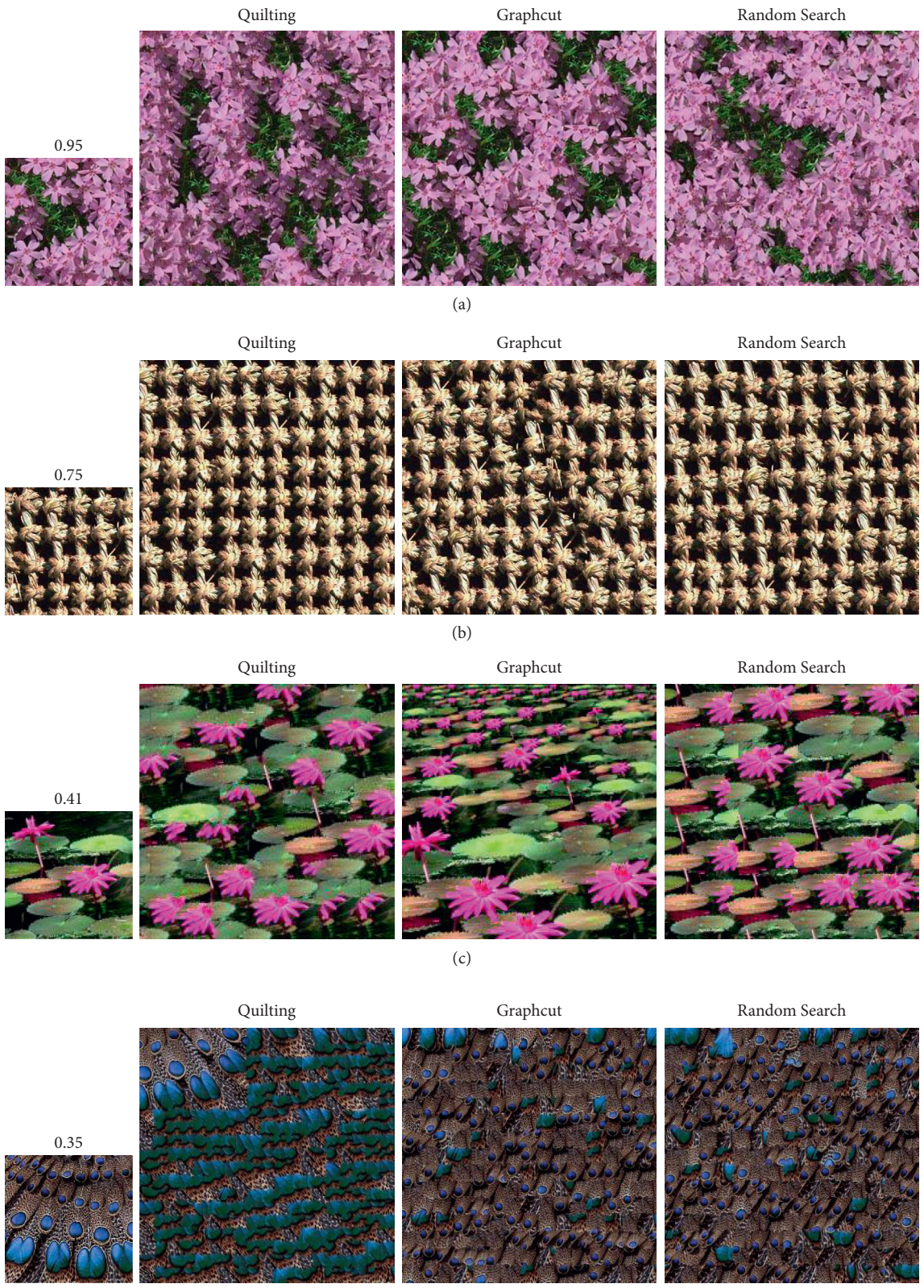


Figure 3: An example of similar textures generated from three synthesis algorithms, Quilting [20], Graphcut [21], and Random Search [22], from which different synthesizability scores lead to different visual appearances. (a) Flowers. (b) Knots. (c) Waterlily. (d) Peacock hair.

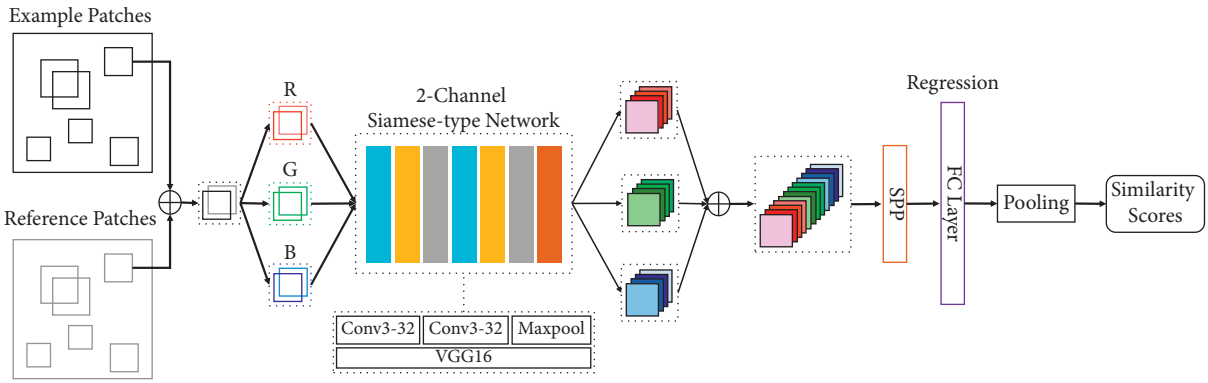


Figure 4: The architecture of our deep RGB-2Channel Siamese-type network.

pyramid pool layer (SPP) [42] between the convolutional layers and the fully connected layers. As a result, this VGGNet-based RGB-two-channel Siamese-type network consists of 17 weight layers, including conv3-32, conv3-32, 13 weight layers in the original VGG-16, and the last two FC layers, FC-512 and FC-1 for regression. Following the settings of VGG-16, the convolution kernel has a uniform size of 3×3 pixels for all convolutional layers and the activation is done through the rectified linear unit function (ReLU); all the max-pool kernel are 2×2 pixels; zero padding is applied so that the output feature maps have the same size with the input patches; the dropout regularization ratio is set to 0.5 to prevent overfitting; and three sizes, 32×32, 64×64, and 128×128 pixels, are chosen as the sizes of input patches.

To obtain the global texture-wise similarity estimation, one of the simplest solutions is to employ the average pooling on all the local sampled patches. That is, it is assumed that every texture patch has an identical relative influence on the entire texture. Although it is straightforward, it is reasonable since texture patterns under the MRF hypothesis are nonsensitive to outliers or saliency effects. Specifically, given each texture patch i and its locally estimated similarity metric y_i, the global texture-wise similarity estimate Y_s is computed as follows:

$$Y_s = \frac{1}{N} \sum_{i=1}^{N} y_i, \tag{1}$$

where N means the number of patches randomly sampled from textures. As our regression problem corresponds to the Euclidean distance (L2 norm) based neighborhood searching process, we choose the commonly used MSE as the minimization criterion. The loss function is then to be

$$L = \frac{1}{N} \sum_{i=1}^{N} y_i - Y_s^2, \tag{2}$$

where Y is the label of a texture image, and the back-propagation error is the average loss over the texture images (denoted as M) in a minibatch, as shown as follows:

$$E = \frac{1}{M} \sum_{j=1}^{M} \left(\frac{1}{N} \sum_{i=1}^{N} y_i - Y_{s_j} \right)^2. \tag{3}$$

In practice, the number of patches N can be set to be arbitrary, including both nonoverlapping patches and overlapping patches. These two kinds of patches need to keep spatial consistent. 32×32, 64×64, and 128×128 sized patches are independently fed into the neural network for training to update the weights. The initial labels are from the ETH dataset [3], where $(0.0, 0.5, 1.0)$ are the annotations, representing "bad," "acceptable," and "good" synthesizability separately. We hence simply define the similarity labels following the synthesizability annotation. In other words, it could be assumed that good synthesizability means that the synthesized output would have a good similarity to the input texture. Then, texture patches are assigned the same similarity labels with their corresponding texture images.

3.4. Fine-Tune on Synthesizability Data. After training the RBG-two-channel Siamese-type network to learn the similarity between the reference textures and the similar textures, we then apply the single branch network, which has the same backbone structure as our Siamese-type network, only to the example textures for fine-tuning. At this stage, the dataset is adjusted to texture images with real synthesizability labels (the ETH dataset [3]), and the loss function is replaced by the MAE so as to further reduce the influence of outliers. At the same time, the two-branch RGB-two-channel Siamese-type network can recommend the best synthesis algorithms from those involved in generating the reference textures. For instance, in our training, the best algorithm is picked up from three approaches, Quilting [20], Graphcut [21], and Random Search [22], depending on the similarity scores; in other training, the best one could be generated from methods like Quilting [20], Graphcut [21], and GAN-based texture synthesis [33].

4. Experiments

In this section, we report results on two stages of experiments designed to evaluate the synthesizability of example textures. Meanwhile, the performance of our approach is analyzed in terms of qualitative and quantitative results and provides comparisons with the state of the art.

4.1. Datasets. In the first stage of training the Siamese-type network, the example textures are from the ETH Synthesisability dataset [3], which have 21,302 sample textures with expert annotations of three labels, 1.0 for "good," 0.5 for "acceptable," and 0.0 for "bad"; the reference textures are from the expanded dataset based on the ETH Synthesisability dataset [3], that is, the reference texture dataset generated by the three synthesis algorithms, Quilting [20], Graphcut [21], and Random Search [22], which includes 63,906 synthesized textures. The size of the example textures is 300×300 pixels while the synthesized size is 600×600 pixels. The patch size is the same for both example texture images and reference texture images, 32×32, 64×64, and 128×128 pixels. The second stage of fine-tuning uses the original ETH Synthesisability dataset [3].

4.2. Training. During training the Siamese-type network, 17,000 example textures and 51,000 reference textures, both 80% of the total datasets, are used for similarity training. Specifically, we sample texture patches as inputs in epochs and employ minibatches for optimization. Since one example texture versus three reference textures, each minibatch includes two example textures and six reference textures, all of which are represented by 16 randomly sampled texture patches, totally leading to 128 patches in a minibatch. The average loss of the textures in one minibatch is counted as the backpropagated error. The entire training is performed in three turns, corresponding to three sampling sizes of 32×32, 64×64, and 128×128 pixels. To optimize the loss function, ADAM algorithm [43] is selected, also with its recommended parameters as $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and the initial learning rate $\alpha = 10^{-4}$. The learning rate decreasing factor is set to 0.1. Its variation in loss is shown in Figure 5, the top one. We can see that the training converges quickly and reaches a relatively stable low value.

In the stage of fine-tuning, 19,000 example textures, 90% of the total datasets, were used for scoring training. Parameter settings keep consistent with those in the training stage of the Siamese-type network, except the initial learning rate as $\alpha = 10^{-6}$. Besides, textures used for validation are 20% of the total datasets for the Siamese-type network and 10% for the fine-tuning. We finally chose the model with the best validation loss for evaluation. As shown in Figure 5, the bottom image is the loss changes during the fine-tuning phase. Although, in the beginning, the loss function has slight fluctuation, with the increase of epochs, it also gradually converges to a low value. This is probably due to the fact that we turn down the batch size because the input images for the fine-tuning have a larger size $(300 \times 300 \times 3)$.

4.3. Results. In this section, we evaluate the performance of our approach in terms of qualitative and quantitative results.

Qualitative Results. Figure 6 provides the predicted synthesizability scores on several example textures by our approach, with their human quality judgments, the annotations in the ETH dataset [3], and the best synthesis algorithm. Each group of examples consists of two images: the example texture and its synthesized one, under which is its predicted synthesizability score, its human annotation, and the recommended synthesis algorithm in the parentheses. It can be seen that the predicted synthesizability score can keep consistent with the quality of the synthesized texture. That is, an example texture with a higher synthesizability score would generate better quality synthesized texture using example-based synthesis methods; as shown in Figure 6, the synthesized quality decreases with the decrease of the synthesizability score. Meanwhile, the predicted synthesizability score can be consistent with the human annotation as well. In the parentheses in Figure 6, the numbers of 1.0, 0.5, 0.0 are the human labels. Although the human labels only give three levels of "good," "acceptable," and "bad," our predicted scores fall into these three ranges.

In addition, experiments also show that our method is superior to the manual feature-based approach. As seen in Figure 7, two typical failures mentioned in [3], false negatives (a low synthesizability score assigned to a synthesizable texture) like cloud and rust examples and false positives (high synthesizability scores for textures hard to synthesize) like hair and tree ring examples, have been corrected. This is because our method is not limited by the hand-crafted features which cannot completely describe texture patterns with irregular structures, heterogeneous information, and so on. Figure 8 shows that our approach also works for textures at different scales. Generally, the synthesizability scores fall down with the increase of the scales of textures. Besides, some interesting results deserve deep investigation. As shown in Figure 9, example textures are assigned the human annotations as 0.5, that is, they are "acceptable" for example-based synthesis algorithms, but actually their synthesized results can achieve "good" visual effects. Finally, affected by the human labels, these textures cannot obtain higher synthesizability scores (for example, "apple" is 0.7, and "feather" is 0.68.). Probably, refining the manual labels and using the unsupervised methodology are solutions that can be explored.

Quantitative Results. In this section, we make an analysis of the performance of the proposed method in terms of its ability to predict the synthesizability of example textures. To quantify the evaluation, a number of network architectures are tested, and several commonly used performance metrics are applied.

Although there is no completely consistent previous work, we can investigate some typical network architectures which vary from shallow to deep, single column to Siamese, one image to multipatches. After combining, they are referred to as single column shallow (one patch), single column VGG-16 (one patch), Siamese VGG-16 (one patch), Siamese VGG-16 (multipatches), and ours (RGB-2Channel Siamese-type network (multipatches) + fine-tuning). We use four convolutional layers and one fully connected layer for

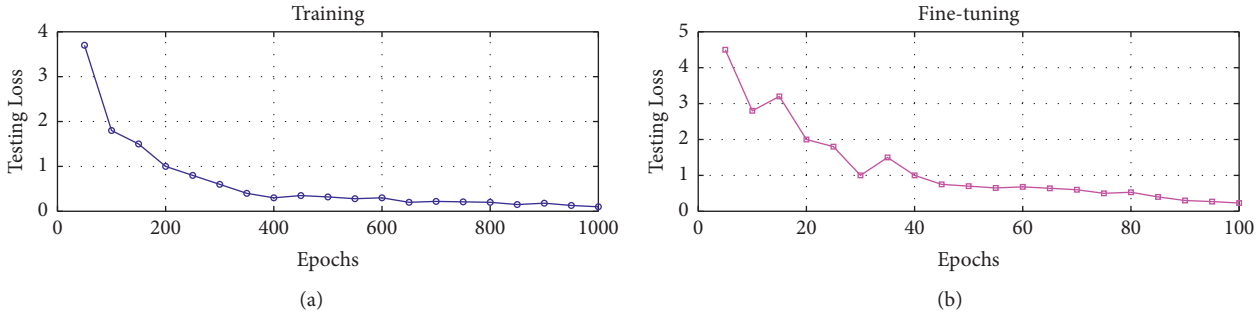


Figure 5: Convergence of loss for our approach in the training stage and fine-tuning stage.

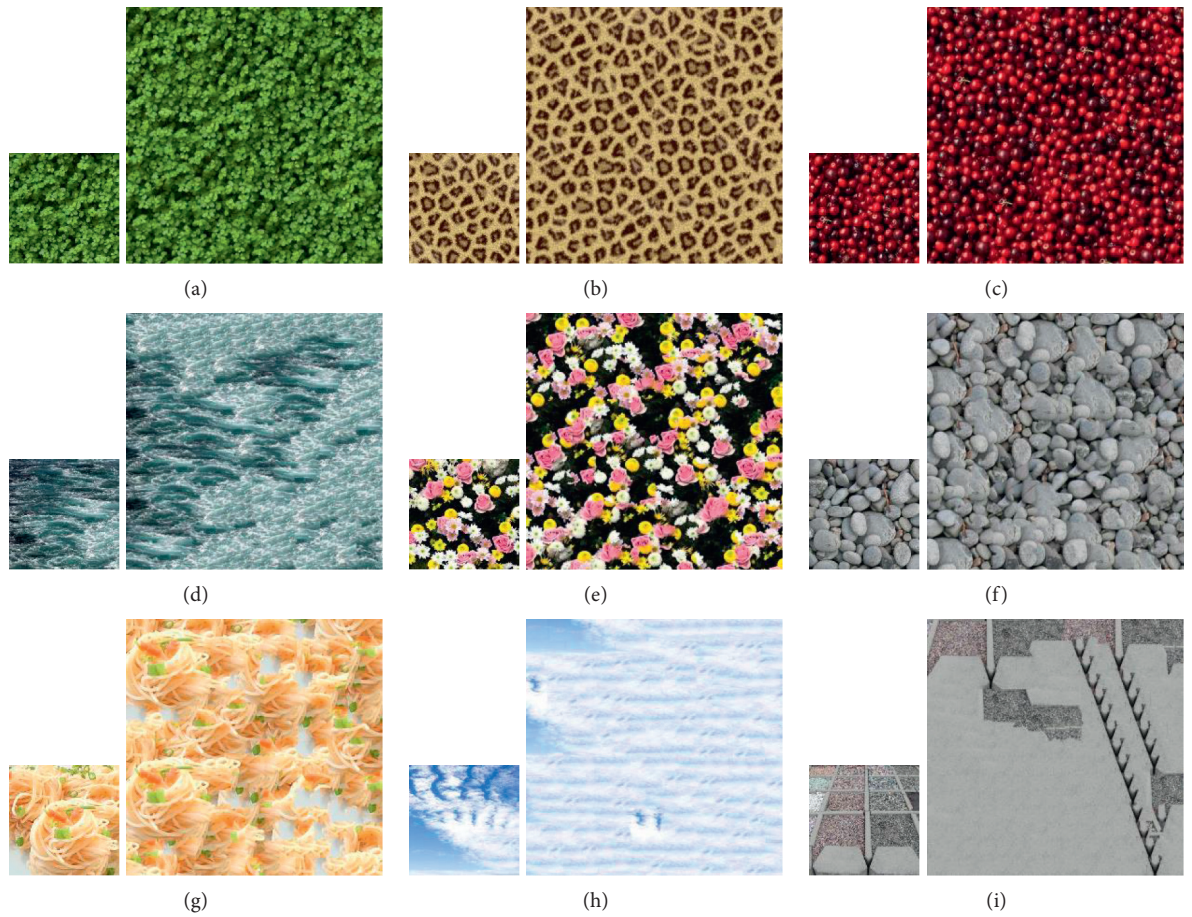
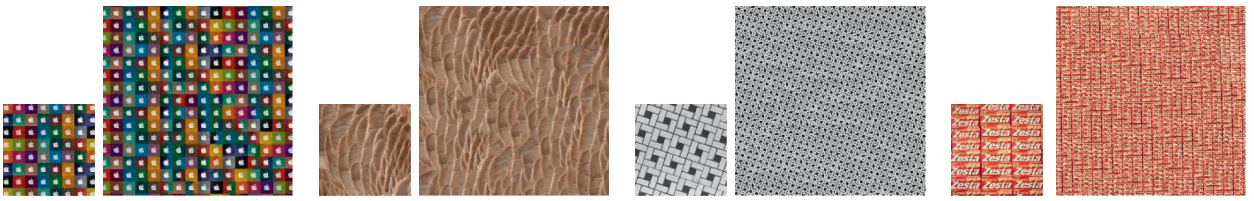


Figure 6: Examples of textures with their synthesizability scores (under the texture image), human annotations (in the parentheses), and the best synthesis algorithm (also in the parentheses) recommended from Quiliting [20], Graphcut [21], and Random Search [24] by our approach, in which "Null" means none of the three methods work for this texture, and "RS" is for Random Search. (a) Four-leaf clover: 0.89 (1.0, RS). (b) Leopard print: 0.78 (1.0, Quilting). (c) Cherry: 0.68 (1.0, RS). (d) Foam : 0.59 (1.0, Graphcut). (e) Monthly roses: 0.52 (0.5, Quilting). (f) Stones: 0.47 (0.5, Quilting). (g) Pasta: 0.37 (0.0, Graphcut). (h) Cloud: .20 (0.0, Null). (i) Grid road: 0.15 (0.0, Null).

Figure 7: Comparison with method in [3]. Our method gives more reasonable scores to textures with irregular structures, heterogeneous information, subtle features, and so on. (a) Cloud: 0.58 (0.35 [3]). (b) Rust: 0.40 (0.25 [3]). (c) Hair: 0.33 (0.59 [3]). (d) Tree ring: 0.13 (0.65 [3]).

(a) (b) (c) (d)

Figure 8: Examples of synthesizability scores predicted for textures at different scales. (a) 0.65 (0.5). (b) 0.78 (1.0). (c) 0.58 (0.5). (d) 0.87 (1.0).



(a) (b) (c) (d)

Figure 9: Some cases deserve further discussion. (a) 0.70 (0.5). (b) 0.68 (0.5). (c) 0.64 (0.5). (d) 0.57 (0.5).

Table 1: Performance comparison between our network and other architectures. The dataset is the ETH Synthesisability dataset [3].

Approach	One patch	Multipatches	PLCC	SROCC	RMSE
Single column shallow	Y		0.112	0.087	0.921
Single column VGG-16	Y		0.413	0.305	0.507
Siamese VGG-16	Y		0.532	0.401	0.421
Siamese VGG-16		Y	0.614	0.522	0.335
Ours		Y	**0.785**	**0.723**	**0.250**

Bold entries are the best performers.

the shallow network, while the change of VGG-16 is only to set the number of outputs as one.

On the other side, since, in the dataset, the qualities of textures are measured by experts' analysis and scoring, which is a very subjective task, we choose PLCC (the Pearson linear correlation coefficient) and SROCC (the Spearman Rank-order Correlation Coefficient) as metrics to measure the prediction accuracy and the prediction monotonicity between MOS/DMOS and the objective scores. We also employ RMSE (the root mean squared error) to assess the prediction consistency between MOS/DMOS and the objective scores. In order to compare with the hand-designed features [3], we measure the precision at different levels of recall as well.

Table 1 provides the comparison between our network and others. It can be seen that our network achieves the best evaluation results. At the same time, the deep network performs better than the shallow network in general, and the multipatches model can describe the features of texture images more effectively with respect to the one-patch model. We also make comparisons with the hand-crafted feature-based method [3], as shown in Figure 10, where our method achieves higher average precision at the corresponding recall levels (the magenta lines), for both textures belonging to "acceptable" and "good." When the recall is one, Dai et al. [3] report that the average precision score (under all features) for "acceptable" textures is 94.5%, and that for "good" textures is 75.5%, while our approach obtains that of 95.7% and 83.2%, respectively. In particular, for "good" textures, the improvement is more noticeable.

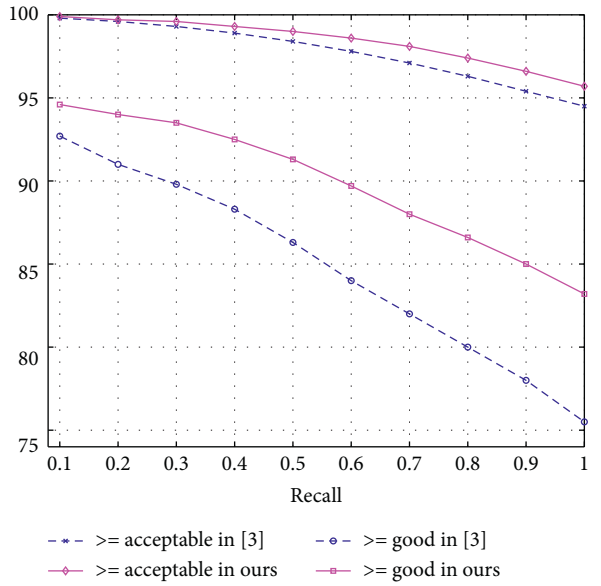


Figure 10: Comparison of average precision of synthesizability prediction at different recall levels.

5. Conclusions

This study proposes a deep Siamese-type network, RGB-two-channel Siamese-type network, for the task of texture synthesizability assessment. To improve versatility and accuracy, we employed the deep learning-based method, which is, as far as we know, the first attempt to introduce end-to-end training for feature extraction, fusion, and regression in this job. To solve the problem of the data insufficiency of the training dataset, this algorithm constructs the reference textures dataset for similarity learning and adapts the multipatches scheme when training. At last, a single branch of fine-tuning on available synthesizability data generates more accurate predictions for synthesizability and also provides a recommended synthesis algorithm which is best matched the example texture. The proposed approach obtains good results qualitatively and quantitatively, when evaluated among several network architectures, and compared with the state of the art. However, there are still some issues that need further exploration; for example, the no-reference method is an interesting direction. Additionally, we do not notice the efficiency of the proposed approach, which means that its speed cannot meet application requirements in actual application scenes. Therefore, balancing performance and efficiency is also a future study.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grant nos. 61907025, 61807020) and is also funded in part by the Science and Technology Development Fund, Macau SAR (Grant no. 0018/2019/AKP and SKL-IOTSC(UM)-2021-2023), in part by the Guangdong Science and Technology Department (Grant no. 2018B030324002), and in part by the Zhuhai Science and Technology Innovation Bureau Zhuhai-Hong Kong-Macau Special Cooperation Project (Grant no. ZH22017002200001PWC).

References

[1] S. A. Golestaneh and L. Karam, "Synthesized texture quality assessment via multi-scale spatial and statistical texture attributes of image and gradient magnitude coefficients," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision & Pattern Recognition Workshops (CVPRW)*, pp. 851–857, Salt Lake City, USA, June 2018.

[2] X. Dong and H. Zhou, "Texture synthesis quality assessment using perceptual texture similarity," *Knowledge-Based Systems*, vol. 194, Article ID 105591, 2020.

[3] D. Dai, H. Riemenschneider, and L. Van Gool, "The synthesizability of texture examples," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3027–3034, Columbus, OH, USA, June 2014.

[4] F. Yang, G.-S. Xia, D. Dai, and L. Zhang, "Learning the synthesizability of dynamic texture samples," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2502–2517, 2019.

[5] L. L. Tang, K. Chen, C. Wu, Y. Hong, K. Jia, and Z.-X. Yang, "Improving semantic analysis on point clouds via auxiliary supervision of local geometric priors," *IEEE Transactions on Cybernetics*, vol. 12, no. 1, pp. 1–11, 2021.

[6] P.-B. Zhang and Z.-X. Yang, "A new learning paradigm for random vector functional-link network: rvfl+," *Neural Networks*, vol. 122, pp. 94–105, 2020.

[7] Z.-X. Yang, L. Tang, K. Zhang, and P. K. Wong, "Multi-view CNN feature aggregation with ELM auto-encoder for 3D shape recognition," *Cognitive Computation*, vol. 10, no. 6, pp. 908–921, 2018.

[8] C. Hao, Y. Chen, Z.-X. Yang, and E. Wu, "Higher-order potentials for video object segmentation in bilateral space," *Neurocomputing*, vol. 401, pp. 28–35, 2020.

[9] V. V. Lukin, N. N. Ponomarenko, O. I. O. Ieremeiev, K. O. Egiazarian, and J. Astola, "Combining full-reference image visual quality metrics by neural network," in *Proceedings of the SPIE 9394*, p. 93940K, San Francisco, CA, USA, March 2015.

[10] S. Bosse, D. Maniry, K.-R. Muller, T. Wiegand, and W. Samek, "Deep neural networks for no-reference and full-reference image quality assessment," *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 206–219, 2018.

[11] J. Kim, A.-D. Nguyen, and S. Lee, "Deep cnn-based blind image quality predictor," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 1, pp. 11–24, 2019.

[12] F. Gao, Y. Wang, P. Li, M. Tan, J. Yu, and Y. Zhu, "DeepSim: deep similarity for image quality assessment," *Neurocomputing*, vol. 0, pp. 1–11, 2017.

[13] X. Liu, J. Van De Weijer, and A. D. Bagdanov, "Rankiqa: learning from rankings for no reference image quality assessment," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1040–1049, Venice, Italy, October 2017.

[14] K. Ma, W. Liu, K. Zhang, Z. Duanmu, Z. Wang, and W. Zuo, "End-to-end blind image quality assessment using deep neural networks," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1202–1213, 2018.

[15] W. Zhang, K. Ma, J. Yan, D. Deng, and Z. Wang, "Blind image quality assessment using a deep bilinear convolutional neural network," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 1, pp. 36–47, 2020.

[16] D. Pan, P. Shi, M. Hou, Z. Ying, S. Fu, and Y. Zhang, "Blind predicting similar quality map for image quality assessment," in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6373–6382, Salt Lake City, USA, June 2018.

[17] H. Zhu, L. Li, J. Wu, W. Dong, and G. Shi, "Metaiqa: deep meta-learning for no reference image quality assessment," in *Proceedings of the 2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14131–14140, Seattle, WA, USA, August 2020.

[18] J. You and J. Korhonen, "Transformer for image quality assessment," in *Proceedings of the 2021 IEEE International Conference on Image Processing (ICIP)*, pp. 1389–1393, Anchorage, AK, USA, September 2021.

[19] L. Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the art in example-based texture synthesis," in *Eurographics 2009*

State of the Art Reports, Eurographics Association, Munich, Germany, 2009.

[20] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proceedings of the 28th Annual Conference on Computer graphics and Interactive Techniques, SIGGRAPH '01*, pp. 341–346, New York, NY, USA, August 2001.

[21] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 277–286, 2003.

[22] C. Hao, Y. Chen, W. Wu, and E. Wu, "An iterated randomized search algorithm for large-scale texture synthesis and manipulations," *The Visual Computer*, vol. 31, no. 11, pp. 1447–1458, 2015.

[23] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: a randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 24:1–24:11, 2009.

[24] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, "Image melding: combining inconsistent images using patch-based synthesis," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 82:1–82:10, 2012.

[25] A. Kaspar, B. Neubert, D. Lischinski, M. Pauly, and J. Kopf, "Self tuning texture optimization," *Computer Graphics Forum*, vol. 34, no. 2, pp. 349–359, 2015.

[26] Y. Zhou, H. Shi, D. Lischinski, M. Gong, J. Kopf, and H. Huang, "Analysis and controlled synthesis of inhomogeneous textures," *Computer Graphics Forum*, vol. 36, no. 2, pp. 199–212, 2017.

[27] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems, NIPS'15*, pp. 262–270, Montreal Canada, December 2015.

[28] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2414–2423, Las Vegas, NV, USA, June 2016.

[29] I. J. Goodfellow, J. Pouget-Abadie, and M. Mirza, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS'14*, pp. 2672–2680, Montreal Canada, December 2014.

[30] C. Li and M. Wand, "Precomputed real-time texture synthesis with Markovian generative adversarial networks," in *Proceedings of the 2016 European Conference on Computer Vision (ECCV)*, pp. 702–716, Amsterdam, The Netherlands, October 2016.

[31] N. Jetchev, U. Bergmann, and R. Vollgraf, "Texture synthesis with spatial generative adversarial networks," 2017, https://arxiv.org/abs/1611.08207.

[32] U. Bergmann, N. Jetchev, and R. Vollgraf, "Learning texture manifolds with the periodic spatial gan," in *Proceedings of the 34th International Conference on Machine Learning, ICML'17*, pp. 469–477, Sydney, Australia, August 2017.

[33] Y. Zhou, Z. Zhu, X. Bai, D. Lischinski, D. Cohen-Or, and H. Huang, "Non-stationary texture synthesis by adversarial expansion," *ACM Transactions on Graphics*, vol. 37, no. 4, p. 13, Article ID 49, 2018.

[34] A. Hertz, R. Hanocka, R. Giryes, and D. Cohen-Or, "Deep geometric texture synthesis," *ACM Transactions on Graphics*, vol. 39, no. 4, 2020.

[35] W. Xian, P. Sangkloy, and V. Agrawal, "Texturegan: controlling deep image synthesis with texture patches," in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8456–8465, Salt Lake City, USA, June 2018.

[36] N. Yu, C. Barnes, E. Shechtman, S. Amirghodsi, and M. Lukàc, "Texture mixer: a network for controllable synthesis and interpolation of texture," in *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12164–12173, Long Beach, CA, USA, June 2019.

[37] A. Frühstück, I. Alhashim, and P. Wonka, "Tilegan: synthesis of large-scale non-homogeneous textures," *ACM Transactions on Graphics*, vol. 38, no. 4, 2019.

[38] E. Heitz, K. Vanhoey, T. Chambon, and L. Belcour, "A sliced wasserstein loss for neural texture synthesis," in *Proceedings of the 2021 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9412–9420, Nashville, TN, USA, June 2021.

[39] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4353–4361, Boston, MA, USA, June 2015.

[40] X. Lu, Z. Lin, X. Shen, R. Mech, and J. Z. Wang, "Deep multi-patch aggregation network for image style, aesthetics, and quality estimation," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 990–998, Santiago, CL, USA, December 2015.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015, https://arxiv.org/abs/1409.1556.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[43] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2017, https://arxiv.org/abs/1412.6980v5.