Perception-Aware Texture Similarity Prediction

Weibo Wang, and Xinghui Dong, *Member, IEEE*

***Abstract*— Texture similarity plays important roles in texture analysis and material recognition. However, perceptually-consistent fine-grained texture similarity prediction is still challenging. The discrepancy between the texture similarity data obtained using algorithms and human visual perception has been demonstrated. This dilemma is normally attributed to the *texture representation* and *similarity metric* utilised by the algorithms, which are inconsistent with human perception. To address this challenge, we introduce a Perception-Aware Texture Similarity Prediction Network (PATSP-Net). This network comprises a Bilinear Lateral Attention Transformer network (BiLAViT) and a novel loss function, namely, RSLoss. The BiLAViT contains a Siamese Feature Extraction Subnetwork (SFEN) and a Metric Learning Subnetwork (MLN), designed on top of the mechanisms of human perception. On the other hand, the RSLoss measures both the ranking and the scaling differences. To our knowledge, either the BiLAViT or the RSLoss has not been explored for texture similarity tasks. The PATSP-Net performs better than, or at least comparably to, its counterparts on three data sets for different fine-grained texture similarity prediction tasks. We believe that this promising result should be due to the joint utilization of the BiLAViT and RSLoss, which is able to learn the perception-aware texture representation and similarity metric.**

***Index Terms*—Texture similarity, texture retrieval, texture synthesis quality assessment, bilinear attention, ranking loss.**

I. Introduction

As one of the inherent properties of the material surface, texture enables humans to accurately perceive and identify objects. Texture similarity aims at judging whether or not two textures are similar, or the extent that they are similar. However, this task normally encounters two difficulties: *texture representation* and *similarity metric*. As a result, prediction of perceptually-consistent fine-grained texture similarity using algorithms is struggling. In Fig. 1, the four textures shown at the right side share the same **SSIM** [1] value compared with the query texture. In other words, it is decided using the **SSIM** that they show the same level of similarity to the query. Nevertheless, this is not the case for human perception.

Clarke et al. [2] used four algorithms to predict the fine-grained texture similarity data collected using 30 human subjects. It was showed that only a weak correlation (Spearman's correlation coefficient ρ = 0.21) between the algorithmic decisions and human perception was obtained. Recently, Dong et al. [3] showed that the similarity data obtained using 51 texture descriptors were considerably inconsistent with that perceived by humans. They attributed this inconsistency to the finding that these descriptors do not exploit long-range interactions (dependencies) as well as humans.

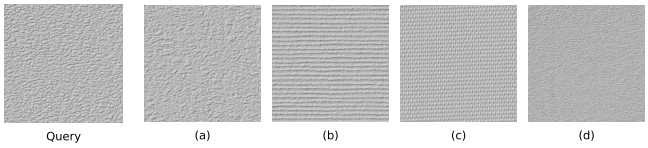

Fig. 1. A query texture and four retrieval textures contained in the *Pertex* [2] data set. The SSIM value computed between each retrieval texture and the query texture is 0.0264. However, the corresponding similarity values contained in the *Isomap* perceptual similarity matrix [18] manifest great variations, which are 0.85286, 0.37449, 0.36677 and 0.544 in turn.

Traditionally, texture descriptors were normally designed based on the response of filters [4], [5], the statistics of gray level values [6], the structural characteristics [1], [7], [8] and a texture model [9]. With the development of deep learning, in particular, Convolutional Neural Networks (CNNs) have been widely applied to computer vision tasks and produced the better results than traditional methods [10]–[14]. They can directly learn features from a large set of training samples. It has been demonstrated that the features extracted from the convolutional layers of a pre-trained CNN model can be used for texture representation [15]. Motivated by this study, Gao et al. [16] proposed a texture similarity prediction framework on top of the pre-trained VGG-VD-19 model [17]. In this framework, the cosine similarity was computed in the feature space in order to measure the similarity between a pair of textures. However, there is no evidence whether or not the cosine similarity is consistent with human perception.

More studies also built a texture similarity prediction network on top of a pre-trained CNN along with the Euclidean or Cosine distance [14], [19]–[21]. Usually, the Mean Square Error (MSE) was employed as the loss function while Pearson's Correlation Coefficient (PCC) was used as the performance metric. In this context, a low rank correlation coefficient (e.g., Spearman's ρ) was often produced together with a high PCC value. However, it has been addressed that the Euclidean distance, the Cosine distance and PCC are equivalent under the z-score normalized data distribution [22]. Therefore, the embarrassment should be attributed to the fact that these metrics do not measure the ranking discrepancy. Nevertheless, the ranking information is particularly important to human perception even if it is normally ignored by existing studies which only pay attention to the proximity of the data.

To address the challenge that texture similarity tasks encounter, we propose a novel Perception-Aware Texture Similarity Prediction Network (PATSP-Net). This network con-

This study was in part supported by the National Natural Science Foundation of China (NSFC) (No. 42176196) and was in part supported by the Young Taishan Scholars Program (No. tsqn201909060) (Corresponding author: Xinghui Dong).

W. Wang and X. Dong are with the School of Computer Science and Technology, Ocean University of China, Qingdao, 266100. (e-mail: wwb@stu.ouc.edu.cn, xinghui.dong@ouc.edu.cn)

sists of a Bilinear Lateral Attention Transformer network (BiLAViT) and a loss function, namely, RSLoss. The BiLAViT is motivated by the lateral interactions exploited by the Human Visual System (HVS). On the other hand, existing evaluation metrics and loss functions are usually linearly equivalent in the numerical scale [22] while they ignore the difference in the ranking. However, the ranking information is important to human perceptual texture similarity estimation. To incorporate the ranking discrepancy into the similarity metric as perceived by humans, we are inspired to design the RSLoss.

The BiLAViT comprises a Siamese Feature Extraction Subnetwork (SFEN) and a Metric Learning Subnetwork (MLN). The SFEN is built on top of a siamese backbone network and a Chanel-wise Interaction and Bilinear Fusion Module (CI-BFM). The MLN contains a series of Bilinear Lateral Attention (BiLA) stages and three linear layers. The BiLA stages learn the lateral interactions between different feature channels and compress the features. The RSLoss measures not only the ranking difference but also the scaling discrepancy. Both the BiLAViT and RSLoss are applied to fine-grained texture similarity prediction tasks. In particular, we also popularize this task to Texture Synthesis Quality Assessment (TSQA). To this end, we collect a Perceptual Synthesized Texture Similarity data set, referred to as *PerTexSynQS*. The proposed BiLAViT can be trained using the RSLoss on a texture data set, for example, *Pertex* [2], *PTD* [23] and *PerTexSynQS*, which contains the fine-grained human perceptual similarity data.

In contrast to conference paper [24], this paper shows four differences. (1) We replace the pre-trained Conformer [25] with the SwinTransformer [26] network. (2) The metric learning network is rebuilt using the CI-BFM and BiLA that we design. (3) The new ranking loss function is applied rather than the MSE function. (4) An additional TSQA task is performed using the new data set that we collect. The contributions of this study can be summarized as fourfold. (1) We introduce a Chanel-wise Interaction and Bilinear Fusion Module (CI-BFM), which fuses the HOS calculated from the features which are extracted from a pair of textures. (2) We develop a new Bilinear Lateral Attention (BiLA) block, which encodes the lateral interactions between different feature channels and produces the more compact features. (3) To help the network learn the ranking difference, we design a new loss function by weighting scale differences using rank differences. (4) We collect a new perceptual similarity data set, i.e., *PerTexSynQS*, for conducting the TSQA task. To our knowledge, the four aspects have not been explored before.

A. Contributions

The contributions of this paper is organized as three aspects.

(1) We present BiLAViT, a novel BiLateral Attention Transformer network with SFEN and MLN components. It utilizes lateral interactions inspired by the human visual system, encoding HOS information through lateral attention mechanisms, simulating short-range, long-range, and lateral interactions.

(2) Introducing RSLoss, a novel loss function based on ranking differences, to measure diversity in rankings and scale differences. RSLoss quantifies both ranking and scale differences, providing better guidance for the network to learn texture similarity perceived by humans.

(3) Expanding the task scope to a new data set for texture synthesis quality assessment, PerTexSynQS. To address fine-grained texture similarity prediction and TSQA tasks, we collect PerTexSynQS, a new perceptual synthetic texture similarity data set, utilized to train BiLAViT.

The rest of this paper is organized as follows. We review the related literature in Section II. In Sections III and IV, the BiLAViT and RSLoss are introduced respectively. The *PerTexSynQS* data set is described in Section V. We report the experimental setup and results in Sections VI and VII respectively. Finally, we draw our conclusion in Section VIII.

II. Related Work

A. CNNs and Transformers

Convolutional Neural Networks (CNNs) have been widely applied to computer vision [27]–[29]. Since Vaswani [30] built the first Transformer network and applied it to Natural Language Processing (NLP), more studies have been performed, such as BERT [31] and GPT-3 [32]. Dosovitskiy et al. [33] further popularized this technique to computer vision. In contrast to CNNs which exploit local features, Transforms encode the global representation by computing the self-attention data. Thus, both CNNs and Transformers have become the dominant methods for vision tasks [34]–[37]. In [24], Wang and Dong were motivated to utilize the features extracted using the CNN and Transformer sub-networks of the Conformer [25] for texture representation. The SwinTransformer [26] is able to exploit both the local characteristics and long-range dependencies using the Shift Window Attention which is more efficient than the self-attention mechanism. Therefore, we use the pre-trained SwinTransformer as the backbone in this study.

B. Metric Learning

Metric learning [38] aims to learn representations in the embedding space. The primary purpose of it is to learn a new metric which can reduce the distance between similar samples and increase the distance between dissimilar samples [39]. Traditionally, the Euclidean and Mahalanobis distances were used to measure the similarity between two samples. However, these distances can only capture a few nonlinear information. Instead, deep metric learning techniques have been used to learn the nonlinear information. In general, these techniques were designed based on the Siamese network [40] or the Triplet network [41]. Loss functions are important to differentiating and distinguishing between the positive and negative samples. The contrastive loss [42], triplet loss [41] and mixed loss [43] normally focus on modeling the relationship between samples and the distance between positive and negative samples, while the clustering loss [44] and Magent loss [45] pay attention to preventing the overlapping between different categories. However, these functions normally do not consider the ranking difference. In contrast, we design a metric learning network together with a new ranking and scaling based loss function.

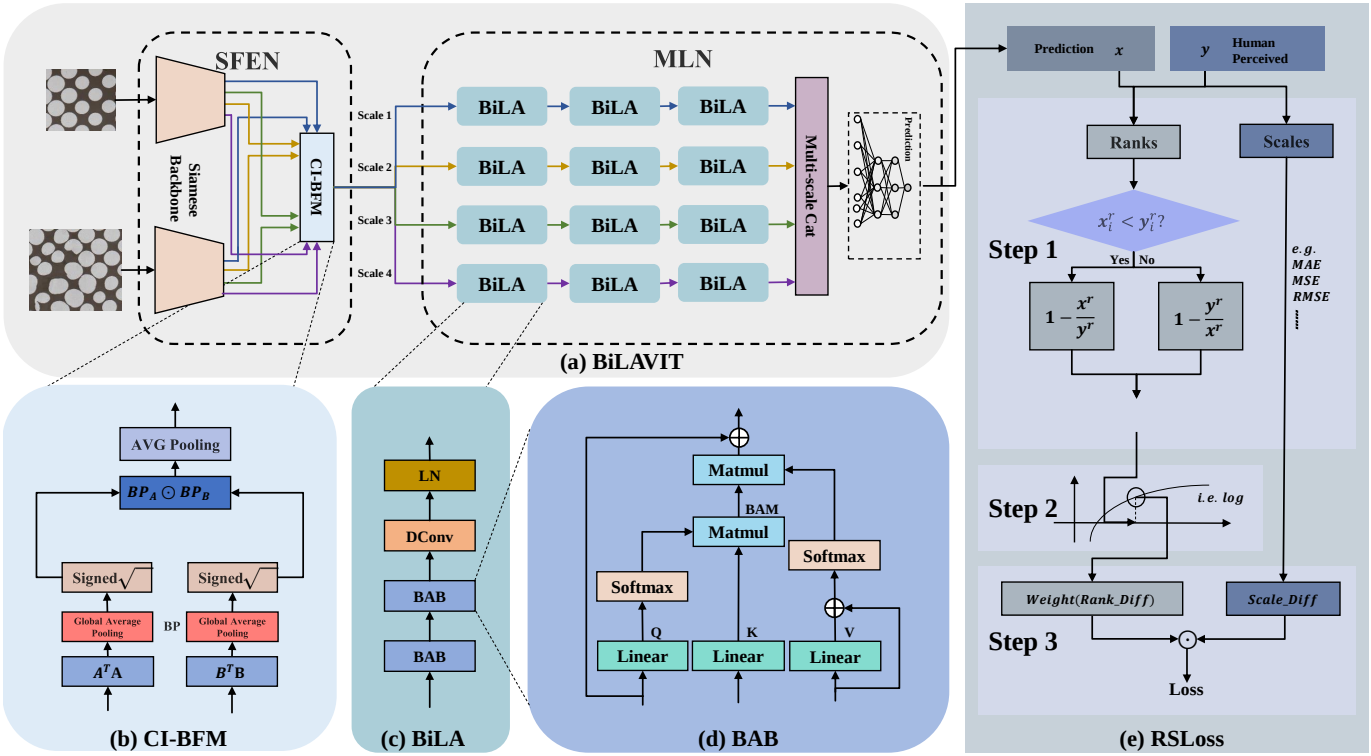

Fig. 2. The architecture of the proposed PATSP-Net, which contains the BiLAViT (a) and the RSLoss (e). Also, the internal structures of CI-BFM, BiLA and BAB are shown in (b), (c) and (d) respectively.

C. Perceptual Similarity Data Collection

In [2], Clarke et al. derived a 334×334 perceptual similarity matrix using a free-grouping experiment. Clarke et al. [18] further used the Isomap dimensionality reduction method [46] to obtain a more compact Isomap similarity matrix. A similar study was conducted by Liu et al. [23] on procedural textures. Rosenfeld et al. [20] collected 6,016 pairs of similar images that human observers consistently decided. Prashnani et al. [47] argued that it is challenging for humans to assign a specific score to a single image. We were motivated to design a new texture synthesis quality marking software. Specifically, four synthesized textures were displayed around the sample texture. The observer was required to mark each synthesized texture with a quality score according to the similarity between them. In this case, the observer was able to compare the four synthesized textures and mark them more accurately.

D. Perceptual Texture Similarity Prediction

Texture similarity is key to human perception and machine vision for recognition of objects and scenes. However, it is challenging to derive the perceptually-consistent fine-grained texture similarity using traditional image features [3]. Due to the powerful representation ability of Deep Neural Networks (DNNs), a pre-trained deep model was usually used to extract features. Gao et al. [16] used the features extracted using the pre-trained VGG [28] model together with cosine similarity to predict the perceptual texture similarity. In [14], Zhang et al. utilized the L_2 distance computed between the feature maps extracted from two textures using the same model as the similarity score. Ding et al. [48] computed the multi-scale representation using the VGG [28] model. The correlations of the spatial averages of different feature maps were combined with the correlations of these maps. In these studies, the similarity was only measured using a simple metric. In contrast, we introduce a Bilinear Lateral Attention Transformer network (BiLAViT), which contains a siamese feature extraction network and a metric learning network. The latter can be used to learn the similarity of two textures on top of the features extracted using the former.

III. Bilinear Lateral Attention Transformer

Inspired by the lateral interactions that the Human Visual System (HVS) utilizes, we propose a Bilinear Lateral Attention Transformer network (BiLAViT). As shown in Fig. 2 (a), the BiLAViT consists of a Siamese Feature Extraction Network (SFEN) and a Metric Learning Network (MLN).

A. Siamese Feature Extraction Network

When designing the feature extraction module, our goal was to ensure that this module possesses a robust capability to extract both local and global information effectively. The Siamese Feature Extraction Network (SFEN) is built on top of a siamese backbone network and a Channel-wise Interaction and Bilinear Fusion Module (CI-BFM). The two streams share the same set of weights. A pair of textures are fed into the backbone network. In terms of each stream, features are extracted at different stages, to derive a multi-scale representation. As a result, two sets of features are extracted at the same

stage of both the streams respectively. The features extracted at a stage are then sent to the CI-BFM separately. This module first computes Higher Order Statics (HOS) from each set and then fuses these into a single set.

1) Siamese Backbone Network: In this study, the pre-trained SwinTransformer [26] was used in the siamese backbone network. The SwinTransformer incorporated some concepts from the CNN architecture design, such as downsampling and local dependency, to redesign the Transformer architecture. The sliding-window design that it used is able to introduce the locality of convolutional operations. But as a member of Transformers, the SwinTransformer can still maintain the sufficient globality. In addition, the window partitioning operation makes it efficient. Therefore, we use the SwinTransformer [26] as the backbone of the SFEN rather than the Conformer [25] that we used in [24] for the purpose of efficiency. The siamese backbone network takes a pair of textures as the input. In terms of each stream, a set of features are extracted using the last block of one of four stages. In other words, two sets of features are extracted at the same stage with regard to both the streams. As a result, a multi-scale representation, which encodes the features at four different scales, is derived. Given a single stage, the features extracted are fed into the CI-BFM separately.

2) Channel-wise Interaction and Bilinear Fusion Module: To construct rich detailed correspondences between a pair of textures based on fine-grained information, we designed the Channel-wise Interaction and Bilinear Fusion Module (CI-BFM). The CI-BFM receives the two sets of features, extracted using the same stage of the two streams of the siamese backbone network respectively. As shown in Fig. 2 (b), the CI-BFM contains a dual-stream Bilinear Pooling (BP) unit and an Interactive Fusion unit. Within this module, each set of features are sent to an individual BP unit, which computes Higher Order Statics (HOS) in different spatial extents. The two sets of features processed by both the BP units are further fused into a single set using the Interactive Fusion unit. In the BP unit, the features of each texture undergo bilinear pooling operations with themselves to maximally represent fine-grained information. Although fine-grained information contains a significant amount of detailed redundancy, it provides sufficient detailed information. And the Interactive Fusion unit constructs a "give-and-take" relationship mapping between a pair of textures through the Hadamard product, capturing a rich set of detailed correspondences between textures.

Dual-Stream Bilinear Pooling Unit. Two parallel bilinear pooling [49] operations are comprised of the dual-stream bilinear pooling unit. The bilinear pooling operation computes a set of pair-wise statistics between feature channels and captures the channel-to-channel relationship. The computation of this operation can be expressed as:

$$\mathbf{BP} = \mathbf{sign}(\cdot) \odot |\frac{1}{m \cdot n} \cdot \sum\nolimits_{l \in \mathcal{L}} F_l^T \times F_l|^{\alpha}, \tag{1}$$

where F_l represents the vector of the feature map of the channel $l \in \mathcal{L}$, α is the Power Law coefficient, $\mathbf{sign}(\cdot)$ denotes the sign function of the sum and $\odot$ is the element-wise multiplication (or the Hadamard product) operator. In essence, the bilinear pooling is used to encode the pair-wise relationship between homologous feature channels. Since the pooling operation is performed across the entire feature map, the resultant features are suitable for comparing the similarity of two textures which have different sizes.

Interactive Fusion Unit. The element-wise multiplication is able to achieve the multimodal joint representation [50]. We are motivated to design the interactive fusion unit on top of an element-wise multiplication operation and an average pooling operation. Given the two sets of features outputted by the dual-stream BP unit, they are fused using the element-wise multiplication operation, in which the features in one set are used as the weight of those in the other set. The features fused are then processed by the average pooling operation in order to derive the more compact features.

B. Metric Learning Network

To reduce redundant information in fine-grained texture correspondences, promoting the transformation of detailed information (local) towards abstract information (global) aligned with human perception, we designed The Metric Learning Network (MLN). The MLN consists of a four-stream BiLA subnetwork and a three-layer linear subnetwork. This network aims to learn a similarity score which measures the fine-grained similarity of two textures. Four sets of features, extracted using the SFEN at four scales respectively, are passed through the four streams separately. Each stream contains three consecutive Bilinear Lateral Attention (BiLA) stages. The features produced by the four streams are flattened and concatenated into an individual feature vector, which encodes both the long-range and lateral interactions, to measure the similarity of the two textures. This feature set is further fed into the three-layer linear subnetwork. Finally, a similarity score is predicted using the MLN.

1) Bilinear Lateral Attention: The Bilinear Lateral Attention (BiLA) stage (see Fig. 2(c)) contains two Bilinear Attention Blocks (BABs), a dilated convolutional layer and a Layer Normalization (LN) operation. This stage learns the lateral interactions between feature channels and compresses the features. Indeed, it is the process of constructing abstract relationships from texture correspondences, establishing long-range dependencies.

Bilinear Attention Block. Inspired by the previous work [51], we design a Bilinear Attention Block (BAB) (see Fig. 2(d)) based on a position-independent bilinear attention mechanism, which aggregates the features across all channels into a set of bilinear weights and redistributes them. The BAB can be computed in two steps: channel aggregation and channel redistribution. In the channel aggregation stage, we aggregate the channel information into a Bilinear Attention Matrix (BAM). Given the input feature maps X, this process can be expressed as follows:

$$BAM = K@(softmax(Q))^T, \tag{2}$$

where @ is the matrix multiplication operator, and K and Q are the results produced by applying two different linear layers to the input X, respectively. Q is passed through a $softmax$

function along the last dimension. As a result, an attention map is obtained. Then K and the transpose of the attention map are multiplied. The result is a BAM, which contains the weights of each feature in all channels. The BAM is similar to the global descriptors which contain the HOS at different positions [51]. During the channel redistrib ution stage, we redistribute the aggregated data contained in the BAM to the input data on the channels. This stage can be formularized as follows:

$$X^{'} = BAM@softmax(V + X), \tag{3}$$

where V is the output of a third linear layer in terms of the input feature maps X. The redistribution operation facilitates the exploitation of the long-range interactions. Finally, a second residual connection is applied to $X^{'}$, which accelerates the training of the network.

Dilated Convolutional Layer. To handle the high dimensionality of the features due to the bilinearity, we add a dilated convolutional layer behind the second BAB in the BiLA stage. Compared with the normal convolutional kernel of the same size, this layer can preserve more information in the downsampling process by using an increased dilation rate to enlarge the receptive field. The output is then processed using a layer normalization operation.

2) Linear Layers: Regarding the four streams, the features produced are flattened and aggregated into a single feature vector, which captures both the long-range and the lateral interactions. This vector is further sent to a three-layer linear subnetwork, which consists of three fully-connected layers ($N \rightarrow 1024, 1024 \rightarrow 1024, 1024 \rightarrow 1$), two $ReLU$ activation functions and a $Sigmoid$ activation function. As a result, a similarity score of two textures is predicted.

IV. The Ranking and Scaling Based Loss

Regarding perceptual similarity prediction, both the *Ranking* and *Scaling* information are crucial to measuring the discrepancy between the algorithm performance and the ground-truth data. Existing DNN methods normally used the Mean Square Error (MSE) as the loss function, which ignores the ranking difference (see Fig. 3 (a)). The performance was usually measured using correlation coefficients, including Pearson's Correlation Coefficient (PCC), Spearman's Rank Correlation Coefficient (SRCC) and Kendall's Rank Correlation Coefficient (KRCC). Since the ranking data was not used in the training process of the network, the SRCC or KRCC result was often worse than the PCC result. For the sake of measuring both the ranking and scaling discrepancies (see Fig. 3 (b)), we design a Ranking and Scaling Based loss function, i.e., RSLoss. To our knowledge, this is the first attempt to use the ranking data in the loss function for texture similarity tasks.

A. Limitations of Loss Functions and Performance Measures

Given $x_i \in X^n$ and $y_i \in Y^n$, where n is the sample size and X^n and Y^n are the predicted and ground-truth data sets respectively, the MSE can be computed as:

$$MSE(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2, \tag{4}$$

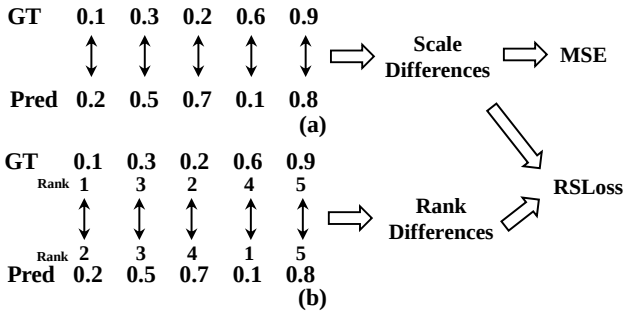

Fig. 3. Comparison of the computation processes of the MSE loss (a) and the proposed RSLoss (b) when the ground-truth and predicted data sets are used for examples. As can be seen, the MSE only measures the scaling difference while the RSLoss measures both the scaling and ranking differences.

while the PCC can be calculated as:

$$PCC(X,Y) = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X\sigma_Y}). \tag{5}$$

In the case that $\mu_X = \mu_Y = 0$ and $\sigma_X = \sigma_Y = 1$, we have

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 &= \sum_{i=1}^{n}x_i^2 - 2\times\frac{1}{n}\sum_{i=1}^{n}x_iy_i + \sum_{i=1}^{n}y_i^2\\ &= \sigma_X - 2\times\frac{1}{n}\sum_{i=1}^{n}x_iy_i + \sigma_Y\\ &= 2\times(1 - \frac{1}{n}\sum_{i=1}^{n}x_iy_i)\\ &= 2\times(1 - \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X\sigma_Y}),\end{aligned} \tag{6}$$

i.e., $MSE(X,Y) = 2\times(1 - PCC(X,Y))$. In other words, the MSE and PCC are linearly equivalent in the aspect of scaling even though the PCC can serve as a supplement to the MSE by providing the trend guidance. However, they do not consider the discrepancy between the rankings of X^n and Y^n. We can also establish a similar relationship between the cosine similarity and the MSE, which can be expressed as

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 &= \sum_{i=1}^{n}x_i^2 - 2\times\frac{1}{n}\sum_{i=1}^{n}x_iy_i + \sum_{i=1}^{n}y_i^2\\ &= \sigma(X) - 2\times\frac{1}{n}\sum_{i=1}^{n}x_iy_i + \sigma(Y)\\ &= 2\times(1 - \frac{1}{n}\sum_{i=1}^{n}x_iy_i)\\ &= 2\times(1 - \frac{XY}{||X||\cdot||Y||}),\end{aligned} \tag{7}$$

i.e., $MSE(X,Y) = 2\times(1 - Cos(X,Y))$.

Existing similarity prediction methods often used the PCC, cosine similarity and Euclidean distance ($\sqrt{n\cdot MSE(X,Y)}$) as performance measures and adopted the MSE as the loss function. Since minimizing the MSE was equivalent to maximizing the PCC or the cosine similarity, they mainly learned the difference in the numerical values of the ground-truth data rather than the important ranking difference.

On the other hand, the SRCC can be defined as

$$SRCC = \frac{Cov(R(X)R(Y))}{\sigma_{R(X)}\sigma_{R(Y)}}, \tag{8}$$

where $R(X)$ and $R(Y)$ are the ranks of X^n and Y^n respectively. The KRCC can be expressed as

$$KRCC = \frac{2}{n(n-1)} \sum_{i<j} \text{sgn}(x_i - x_j)\text{sgn}(y_i - y_j), \tag{9}$$

where $\text{sgn}(x_i - x_j)\text{sgn}(y_i - y_j)$ measures whether or not the ordering of the i_{th} data and the j_{th} data in X^n and that of the i_{th} data and the j_{th} data in Y^n are consistent. Therefore, both the SRCC and the KRCC consider the difference in the rankings of the ground-truth data and the predicted data. However, they do not exploit the scaling difference directly.

B. Our Solution to the Limitations: RSLoss

To ensure that the proposed network measures both the scaling and ranking differences, we design a new loss function (see Fig. 2 (e)), namely, **RSLoss**. This function leverages a differentiable ranking method to obtain the ranking data and weight the scaling data using these data. Regarding the predicted and ground-truth similarity values: X^n and Y^n, the RSLoss can be computed in three steps.

Step 1: Computing the rank and scale differences.

In terms of both the rank and scale, two sets of differences, i.e., $\mathcal{R}(x_i^r, y_i^r)$ and $\mathcal{S}(x_i, y_i)$, between X^n and Y^n are computed as follows:

$$\mathcal{R}(x_i^r, y_i^r) = \begin{cases} 1 - \frac{x_i^r}{y_i^r}, & \text{if } x_i^r < y_i^r, \\ 1 - \frac{y_i^r}{x_i^r}, & \text{otherwise,} \end{cases} \tag{10}$$

$$\mathcal{S}(x_i, y_i) = (x_i - y_i)^2, \tag{11}$$

where x_i and y_i stand for the i-th values in X^n and Y^n respectively, while x_i^r and y_i^r denote the ranks of the two values respectively.

Step 2: Discretizing the rank difference by an activator.

It has been demonstrated that the $Sigmoid$ function may lead to the outcome that the activation value reaches to the "plateau" (or saturation) status [52], [53]. In contrast, the log function can alleviate the impact of the exponential function in the $Sigmoid$ function [54]. Therefore, we use the log function as the default activator for the rank difference. As a result, the rank difference is transformed by the activator into a weight $W_{\mathcal{R}}$. This weight will be applied to the corresponding scale difference. The transform can be expressed as:

$$W_{\mathcal{R}}(x_i^r, y_i^r) = log_\gamma(\mathcal{R}(x_i^r, y_i^r) + \beta), \tag{12}$$

where γ determines the smoothness of the activation value and β controls the range of the log function. Given a value of β, the larger value of γ results in the smoother log function. Both γ and β are set to e by default. It should be noted that the settings of γ and β can be influenced by the learning rate because they affect the gradient of the backward-propagation.

Step 3: Weighting the scale difference by the rank data.

Given a set of scale differences $\mathcal{S}(x_i, y_i)$ and a set of weights $W_{\mathcal{R}}(x_i^r, y_i^r)$ computed in a mini-batch, we weight the scale differences using the weights based on the element-wise multiplication operation. In this case, the scaling data is strengthened by the ranking information. The RSLoss is defined as the average computed across the weighted scale differences. This loss function can be formulated as:

$$RSLoss = \frac{1}{n} \sum_{i=1}^{n} W_{\mathcal{R}}(x_i^r, y_i^r) \odot \mathcal{S}(x_i, y_i), \tag{13}$$

where $\odot$ denotes the element-wise multiplication operation. In essence, the ranking difference is used to facilitate the reduction of the discrepancy in the scaling. On the other hand, the reduction of the scaling discrepancy will promote that the values predicted become consistent with the ground-truth in the ranking. Since the RSLoss is more consistent with human perception, the performance of the model trained is boosted.

The challenge to the back-propagation process is, however, due to the embarrassment that the sorting operation is not differentiable. Alternatively, we use the *Fast Differentiable Sorting and Ranking*[1] [55] method to obtain a differentiable ranking. This method has the space complexity of *O(n)* and the time complexity of *O(nlogn)*. We use the isotonic regression solver of the method, which was re-implemented using PyTorch C++ and CUDA in *torchsort*[2]. In this situation, the ranks of the x_i and y_i are computed as:

$$x_i^r = Soft\_Rank(x_i, r, rs), \tag{14}$$

$$y_i^r = Soft\_Rank(y_i, r, rs), \tag{15}$$

respectively, where $Soft\_Rank(\cdot)$ is the Fast Differentiable Sorting and Ranking method, r denotes the regularization which is set to L_2 (default) or *KL* and rs represents the strength of the regularization. In our experiment, the default value of rs is set to 0.001.

V. Perceptual Texture Synthesis Quality Scores

Texture Synthesis Quality Assessment (TSQA) plays an important role in evaluation of texture synthesis algorithms. It is natural to perform this task by measuring the similarity between a sample and the related synthesized texture. Recently, Dong and Zhou [56] utilized the features extracted using the pre-trained VGG [17] model together with a Random Forest (RF) [57] regressor for the TSQA task. However, the ground-truth data that they used was originally collected for evaluating the synthesizability of textures. Since these data only contained three values: 0, 0.5 and 1.0, the granularity impaired the training of the regressor. For the sake of improving the performance of the TSQA task, a large set of finer-grained perceptual texture synthesis quality score data is demanded.

To bridge this gap, we developed a perceptual texture synthesis quality score marking software. As shown in Fig. 4, a sample texture was displayed at the center, while four corresponding synthesized textures produced by four different algorithms were randomly shown at the top, bottom, left and right side of the sample respectively. As a result, the position bias was alleviated. For the purpose of avoiding distorting

[1]https://github.com/google-research/fast-soft-sort

[2]https://github.com/teddykoker/torchsort

Fig. 4. The GUI of the perceptual texture synthesis quality score marking software. The sample texture is shown at the center while the four associated synthesized textures generated using four different algorithms are randomly displayed at the top, bottom, left and right side of the sample respectively.

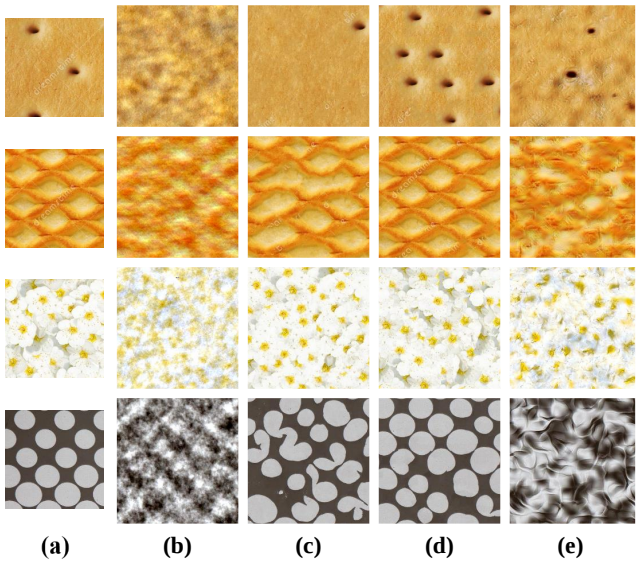

Fig. 5. Four sets of textures [58] are displayed. In each set, a sample texture and four synthesized textures produced by four methods, including FFT [59], MRF [60], Quilting [61] and Wavelet [62], respectively, are shown in turn.

these images, they were scaled in terms of the original aspect ratios in order that those images could be shown within the area of the screen. In practice, the resolution of the sample texture was set to 300×300 pixels and the synthesized textures were shown in the resolution of 400×400 pixels. The observer was required to mark each synthesized texture with a score in the range of [0, 1] with the interval of 0.1. This strategy forced him/her to carefully observe the textures in order to mark a proper score. The experiment was carried out on the same computer in order to keep the display condition constant.

We used the 21,302 sets of sample and synthesized textures that Dai et al. [58] collected. This data set contained both natural and artificial textures which exist in daily life, including biscuit, canvas, flour, hair, leaves, pasta, tile, wool, wall, woven, etc. Four sets of textures are shown in Fig. 5. The experiment was performed for 21,302 trials. Four scores were acquired for a sample texture with regard to the four synthesized textures produced by different algorithms, including FFT [59], MRF [60], Quilting [61] and Wavelet [62], in each trial. In total, 85,208 perceptual quality scores were derived. These scores were comprised of a Perceptual Texture Synthesis Quality Score data set, referred to as *PerTexSynQS*.

The score contained in the *PerTexSynQS* suggests the quality of the associated synthesized texture perceived by the observer via referring to the sample texture. Each score lies in the range of [0, 1], where 0 indicates the completely bad synthesis quality and 1 implies the completely good quality according to human perception. It should be noted that a pair of textures with the score of 0 do not necessarily lack locally similar elements, which are the so-called "hard samples" for traditional algorithms. The *PerTexSynQS* encodes the perception of humans on texture similarity and can be used to train a texture similarity prediction network.

VI. Experimental Setup

In this section, we will describe the setup used in our experiment, including the data sets, performance measures and implementation details.

A. Data Sets

The *Pertex* [2], *PTD* [23] and *PerTexSynQS* data sets that we used contain three types of textures, including natural, procedural and synthesized textures. The *Pertex* data set [2] contains 334 textures, captured from the decorative materials, such as wallpaper, canvas, carpet and drapery. A free-grouping experiment was conducted using this data set and a 334×334 similarity matrix was derived. The Isomap dimensionality reduction [46] method was applied to it. The result was a more compact Isomap similarity matrix. The human perceptual similarity scores contained in this matrix were linearly normalized to the range of [0, 1]. In contrast, the *PTD* data set [23] comprises 450 textures, generated using 23 procedural texture models. The similar experiment was performed and an Isomap similarity matrix was also achieved from this data set. The *PerTexSynQS* data set has been introduced in Section V.

Following the setup that Gao et al. [16] used, we randomly divided the *Pertex* textures into two subsets, which contained 300 training images and 34 test images, respectively. To keep consistent with the work presented in [16], we fixed the random seed. As a result, we derived a total of 45,150 pairs of textures for training and a set of 595 pairs of textures for testing. Likewise, the 450 *PTD* textures were randomly split into two groups, including 400 training images and 50 test

TABLE I
THE RESULTS DERIVED USING THE PROPOSED METHOD ALONG WITH 14 BASELINES ON THE *Pertex* [63], *PTD* [23] AND *PerTexSynQS* DATA SETS FOR THE FINE-GRAINED TEXTURE SIMILARITY PREDICTION TASK. THE VALUES OF FOUR DIFFERENT PERFORMANCE MEASURES ARE REPORTED.

	Pertex [63]				PTD [23]				PerTexSynQS			
Method	MSE↓	PCC↑	SRCC↑	KRCC↑	MSE↓	PCC↑	SRCC↑	KRCC↑	MSE↓	PCC↑	SRCC↑	KRCC↑
SSIM [1]	0.2030	0.6169	0.4330	0.3003	0.1952	0.6306	0.5020	0.3540	0.4248	0.1738	0.1972	0.1412
MS-SSIM [8]	0.2100	0.6085	0.3933	0.2735	0.1185	0.5620	0.9328	0.7986	N/A	0.1229	0.0959	0.0689
CW-SSIM [64]	0.0578	0.6425	0.4038	0.2816	0.0668	0.6285	0.4472	0.3117	650	-0.0033	0.0435	0.0314
VIF [65]	0.2811	0.5935	0.3994	0.2751	0.2086	0.5750	0.2287	0.1579	N/A	0.0119	-0.0041	-0.0029
FSIM [66]	0.0484	0.5230	0.1928	0.1307	0.0407	0.6097	0.4052	0.2802	N/A	0.0969	0.0853	0.0636
MAD [67]	N/A	0.5898	0.3914	0.2733	N/A	+	+	+	N/A	0.0069	0.0312	0.0233
GMSD [68]	0.0715	0.5577	0.1295	0.0859	0.1048	0.6073	0.3225	0.2249	0.0596	0.0942	0.0918	0.0683
VSI [69]	0.1050	0.5126	0.2104	0.1455	0.1090	0.5797	0.4460	0.3096	N/A	0.1151	0.1167	0.0853
NLPD [70]	0.0974	0.6069	0.3509	0.2391	0.3300	0.5493	0.2614	0.1759	0.8603	0.1613	0.1486	0.1066
LPIPS* [14]	0.0155	0.7498	0.6327	0.4580	0.0201	0.7802	0.6780	0.4937	0.0992	0.4088	0.3952	0.2890
PDLF-PTSNet [16]	0.0151	0.7949	0.7188	0.5412	*0.0032*	0.9546	0.9388	0.7980	-	-	-	-
DISTS* [48]	0.0122	0.7546	0.6412	0.4732	0.0131	0.8006	0.7016	0.5200	0.0389	0.6379	0.6065	0.4568
DISTS [48]	0.0100	0.8473	0.7949	0.6048	0.0049	0.9292	0.9005	0.7327	0.0202	0.8005	0.7308	0.5723
A-DISTS* [21]	0.0249	0.6963	0.5427	0.3807	0.0412	0.6584	0.3906	0.2762	0.2453	0.2397	0.2251	0.1621
PMTSPN [24]	*0.0056*	*0.9171*	*0.8783*	*0.7076*	**0.0017**	*0.9763*	*0.9628*	*0.8376*	*0.0158*	*0.8518*	*0.7936*	*0.6356*
PATSP-Net (Ours)	**0.0033**	**0.9504**	**0.9280**	**0.7783**	**0.0017**	**0.9767**	**0.9644**	**0.8429**	**0.0156**	**0.8544**	**0.7946**	**0.6375**

N/A: The MSE value is not meaningful because the result of the MAD [67] has an uncertain range ($[0, +\infty)$)
-: The PDLF-PTSNet [16] did not work because a contour map could not be extracted from some images
+: The MAD [67] method is not applicable to the *PTD* [23] data set, as all the results computed are zeros
*: These methods employ pre-trained weights that are not finetuned during the model training process

images, respectively. In this case, a set of 80,200 pairs of textures were used for training and 1,275 pairs of textures were utilized for testing. The sample textures in the *PerTexSynQS* data set were randomly split into a training set and a test set, which consisted of 18,000 and 3,302 images respectively. In terms of each sample texture, the four associated synthesized textures were put into the set that this texture belonged to.

B. Baselines

We used nine traditional algorithms, including SSIM [1], MS-SSIM [8], CW-SSIM [64], VIF [65], FSIM [66], MAD [67], GMSD [68], VSI [69] and NLPD [70], and five deep learning methods, including LPIPS [14], PDLF-PTSNet [16], DISTS [48], A-DISTS [21] and PMSTPN [24], as baselines, for the purpose of comparison.

C. Performance Measures

Given a texture data set with the perceptual similarity data, let $\mathbf{x}$ and $\mathbf{y}$ denote a pair of textures and $\mathbf{s}$ stands for the perceptual similarity of them. For the *Pertex* [2] and *PTD* [23] data sets, $\mathbf{s}$ is a decimal number between 0 and 1, where 0 suggests that the two textures are dissimilar at all while 1 means that they are completely similar or even are the same. We use $\bar{\mathbf{s}}$ to denote the similarity predicted using an algorithm. For measuring the difference between a set of $\mathbf{s}$ values and a set of $\bar{\mathbf{s}}$ values, two groups of metrics were used. First, the Mean Square Error (MSE) was calculated, to measure the average of the differences between each pair of $\mathbf{s}$ and $\bar{\mathbf{s}}$. However, this measure does not consider the ranking difference. Thus, a second group of measures, including Spearman's Rank Correlation Coefficient (SRCC) and Kendall's Rank Correlation Coefficient (KRCC), were used. Besides, we utilised Pearson's Correlation Coefficient (PCC) that Gao et al. [16] used for comparison.

D. Implementation Details

We used the pre-trained SwinTransformer [26] model as the backbone of the siamese network, whose gradients were updated during the training stage. The Stochastic Gradient Descent (SGD) optimiser was used. The learning rate and momentum were set to $2e^{-3}$ and 0.9 respectively. We set the size of mini-batches to 16. The network was trained for 30 epochs. We performed the experiment on a single Nvidia Geforce RTX 3080ti graphics card. Typically, the training stage took around 13.5 hours on the *Pertex* data set. Specifically, we first normalized each image using the mean and standard deviation of the colours that the pre-trained model used. Then each image was resized to the resolution of 224×224 pixels. Two images $\mathbf{x}$ and $\mathbf{y}$ along with the associated similarity score $\mathbf{s}$ were fed into the siamese network for training. The goal is to minimise the difference between the values of $\bar{\mathbf{s}}$ and $\mathbf{s}$ using an optimiser. In the test stage, each pair of test images were sent to the model that we trained and the output was the similarity score between them. All scores were comprised of a similarity matrix in terms of the test subset.

VII. EXPERIMENTAL RESULTS

We conducted the fine-grained texture similarity prediction experiment using the setup described in Section VI along with the *Pertex* [63], *PTD* [23] and *PerTexSynQS* data sets. We also performed a series of ablation studies. The results will be reported in this section.

A. Results Obtained Using Pertex

1) Texture Similarity Prediction: We first applied the proposed PATSP-Net to the *Pertex* [63] data set for the texture similarity prediction task. The results obtained using our method and 14 baselines are compared in Table I. As can be seen, the proposed method achieved the better performance

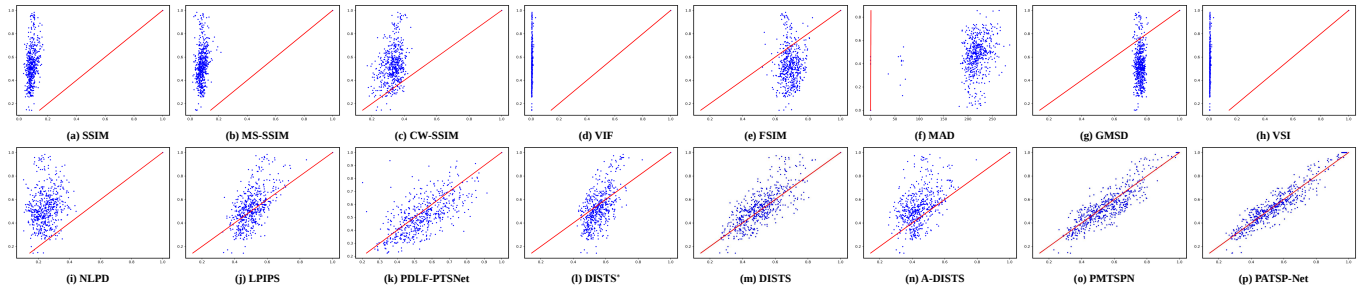

Fig. 6. The scatter plots drawn using the similarity data predicted using 14 baselines and the proposed method against the human perceptual similarity data, when the *Pertex* [63] data set is used.

than all the baselines. In particular, our method outperformed the PMTSPN that we developed in conference paper [24].

2) Scatter Plots: To compare the similarity data obtained using humans against that predicted using the proposed method and the baselines, we draw a scatter plot for each method in Fig. 6. It can be observed that the traditional methods normally deviated from human perception. On the other hand, the five deep learning baselines [14], [16], [21], [24], [48] exhibited the better consistency with human perception. However, the scatter plots produced by these methods still lack the compactness. In contrast, the scatter plot generated using our method presents the higher consistence between the predicted and human perceived similarity data. It is indicated that the proposed method is superior to its counterparts on mimicking human visual perception.

3) Texture Retrieval: We further performed a texture retrieval experiment using the *Pertex* [63] data set. For simplicity, only the best model that we trained was used. Each of 34 test textures was used as a query texture. In terms of a query texture, 34 test textures were retrieved in the descending order of the similarity to this texture. In Table III, we report the G Measure [71] values which measure the consistency between the rankings predicted using one of six deep learning methods and those sorted by humans. It can be seen that our method produced the best consistency with human perception. Figure 8 presents two sets of retrieval textures along with a query texture. As can be observed, the retrieval textures present the high similarity to the query texture.

4) Texture Recovery From Random Noise: We employed 8 methods to recover images from random noise, and the images are displayed in Fig 7. We recruited 10 subjects to rate the recovered images. The scoring criterion was as follows: a score of 1 if the participant deemed the recovery perfect, a score of 0 if the recovery was poor, and ratings in increments of 0.1. The scoring results are presented in Table II.

B. Results Obtained Using PTD

1) Texture Similarity Prediction: When the proposed PATSP-Net was employed on the *PTD* [23] data set for texture similarity prediction, the same baselines were used as those used on the *Pertex* [63] data set. The results are reported in Table I. As can be observed, the proposed method normally outperformed the baselines with large margins, no matter which metric was utilised.

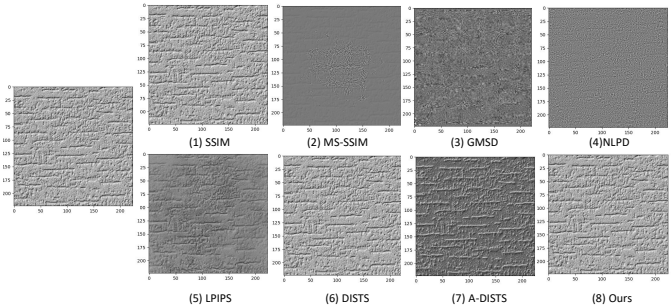

Fig. 7. We employed 8 methods to recover images from random noise.

TABLE II
12 SUBJECTS RATE 8 METHODS FOR TEXTURE RECOVERY FROM RANDOM NOISE.

SUBJECIS	SSIM	MS-SSIM	GMSD	NLPD	LPIPS	DISTS	A-DISTS	Ours
1	1	0	0.2	0	0.6	1	0.6	1
2	1	0.2	0.1	0.1	0.6	0.9	0.7	0.8
3	1	0.1	0.3	0.2	0.6	0.9	0.7	0.8
4	0.8	0	0.2	0.1	0.5	0.8	0.7	0.9
5	0.9	0	0.2	0.1	0.5	0.8	0.7	1.0
6	1	0.1	0.1	0	0.5	1	0.6	0.9
7	1	0.1	0.2	0.1	0.6	1	0.6	1
8	0.9	0	0.4	0.1	0.6	1	0.7	1
9	0.9	0.1	0.2	0.1	0.4	0.9	0.6	0.8
10	1	0.2	0	0.1	0.5	1	0.6	1
AVG	0.95	0.08	0.19	0.09	0.54	0.93	0.65	0.92

2) Texture Retrieval: The texture retrieval experiment was also conducted on the *PTD* [23] data set. Table IV presents the G Measure [71] values calculated between the rankings predicted using one of six deep learning approaches and those sorted by humans. As can be seen, the proposed method outperformed its counterparts when all 50 test textures were retrieved. However, our method performed slightly worse than both PDLF-PTSNet [16] and PMTSPN [24] when only top 10 textures were retrieved. In Fig. 9, we display two sets of retrieval textures. These textures show the high similarity to the query texture.

C. Results Obtained Using PerTexSynQS

1) Texture Similarity Prediction: The 21,302 sets of sample and synthesized textures that Dai et al. [58] collected together with the *PerTexSynQS* data were used. The results are shown

TABLE III
THE G MEASURE [71] VALUE COMPUTED BETWEEN THE RANKINGS PREDICTED USING A DEEP LEARNING METHOD AND THOSE SORTED BY HUMANS ON THE *Pertex* [63] DATA SET.

	LPIPS	PDLF-PTSNet	DISTS	ADISTS	PMTSPN	Ours
All	0.8141	0.8296	0.8233	0.7974	*0.8896*	**0.9224**
Top 10	0.7262	0.7257	0.7380	*0.7401*	0.7348	**0.7829**

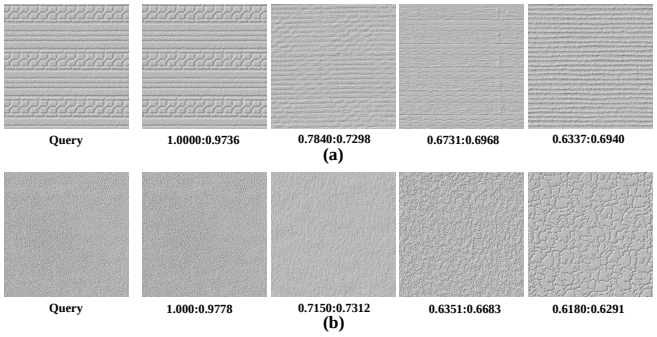

Fig. 8. Two sets of texture retrieval results derived using the *Pertex* [63] data set. Regarding a query texture, the similarity value annotated by humans and that predicted using our best model are shown for each retrieval texture.

TABLE IV
THE G MEASURE [71] VALUE COMPUTED BETWEEN THE RANKINGS PREDICTED USING A DEEP LEARNING METHOD AND THOSE SORTED BY HUMANS ON THE *PTD* [23] DATA SET.

	LPIPS	PDLF-PTSNet	DISTS	ADISTS	PMTSPN	Ours
All	0.8197	0.9136	0.8904	0.7387	*0.9271*	**0.9307**
Top 10	0.7367	**0.8044**	0.7644	0.7338	*0.7909*	0.7822

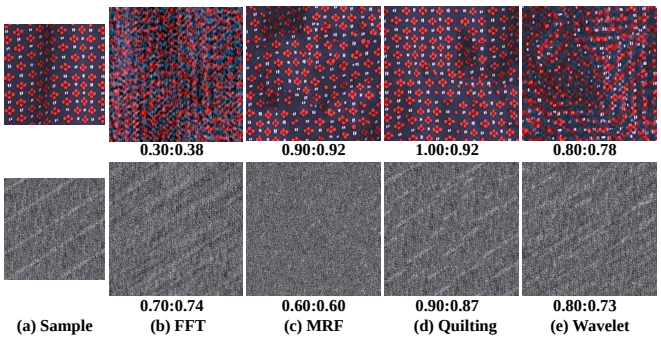

Fig. 10. Two sets of results derived in the TSQA task. Regarding a sample texture, the similarity value annotated by humans and that predicted using our model are shown for each synthesized texture.

in Table I. Again, our method outperformed the 14 baselines with large margins.

2) Texture Synthesis Quality Assessment: We then performed a texture synthesis quality assessment (TSQA) experiment. Given a sample texture, four similarity values were computed between the associated four synthesized textures and this texture using one of the 14 baselines or our method. The correlation coefficient (CC) value was calculated between these similarity values and those annotated by human observers. The average was computed across the CC values calculated with regard to the testing set which contained 3,302 sets of sample and synthesized textures. In Table. V, we report the average CC values calculated using PCC, SRCC and KRCC, in terms of different methods. As can be seen, the proposed method outperformed its counterparts with significantly large margins. It should be noted that the PMTSPN [24] was inferior to our method much in this task. Besides, two sets of results derived in the TSQA task are shown in Fig. 10. It can be observed that the similarity between the synthesized and sample textures predicted using our method was close to that annotated by humans.

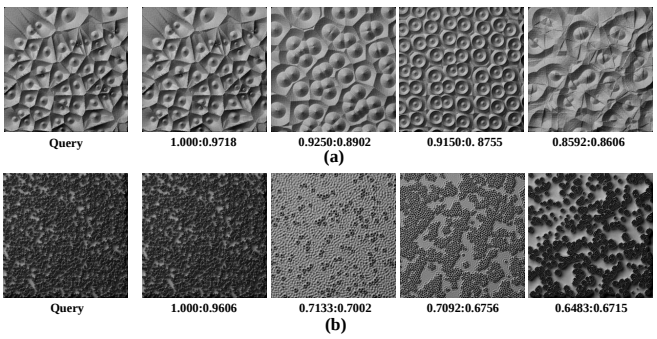

Fig. 9. Two sets of texture retrieval results obtained using the *PTD* [23] data set. Regarding a query texture, the similarity value annotated by humans and that predicted using our best model are shown for each retrieval texture.

D. The Best Synthesis Algorithm Prediction

The model predictions should be consistent with human perceptual results in both scalings and rankings. We ensure scaling consistency through BiLAViT and ranking consistency through RSLoss. To visually showcase the results, we conducted The best synthesis algorithm prediction experiment. The best synthesis algorithm prediction experiment was conducted by following an established setup [56]. Given a sample texture, the best synthesis algorithm can be decided using the formula [56]:

$$B(QS(i)) = argmax_{i\in\{1,\ldots,4\}} QS(i), \tag{16}$$

where $QS(i)$ are four synthesis quality scores predicted using an algorithm or annotated by humans. In terms of the 3,302 sample textures contained in the test set, two sets of best synthesis algorithm predictions were derived using an algorithm and human perception. However, Dong and Zhou [56] did not consider the cases in which the best synthesis algorithm prediction produced multiple best algorithms. In terms of each set of sample and synthesized textures, we therefore define the Prediction Score PS ($\in [0, 100]$) as:

$$PS(QS(i)) = \frac{len(B_C(QS(i)) \cap B_P(QS(i)))}{len(B_P(QS(i)))}, \tag{17}$$

where $B_C(QS(i))$ and $B_P(QS(i))$ denote the best synthesis algorithm prediction obtained using a computational method and human perception respectively. The accuracy ($\in [0, 100]$) of the best synthesis algorithm prediction experiment is computed as the average across all the 3,302 score values. The

TABLE V
THE AVERAGE COMPUTED ACROSS THE CORRELATION COEFFICIENT VALUES CALCULATED BETWEEN THE SIMILARITY DATA PREDICTED USING ONE OF 14 ALGORITHMS AND THAT ANNOTATED BY HUMAN OBSERVERS IN TERMS OF THE TEST SET OF THE TSQA [58] DATA SET. NOTE THAT WE DID NOT INCLUDE THE PDLF-PTSNET [16] BECAUSE IT DID NOT WORK WHEN A CONTOUR MAP COULD NOT BE EXTRACTED FROM SOME IMAGES.

CC	SSIM	MS-SSIM	CW-SSIM	VIF	FSIM	MAD	GMSD	VSI	NLPD	LPIPS	DISTS	A-DISTS	PMTSPN	Ours
PCC ↑	0.3072	0.2580	0.2147	0.2404	0.2806	0.1950	0.1964	0.3276	0.2448	0.3200	0.3687	0.2060	*0.7595*	**0.8345**
SRCC↑	0.2862	0.2405	0.2038	0.2252	0.2619	0.1809	0.1824	0.3078	0.2375	0.2994	0.3441	0.1917	*0.6974*	**0.7832**
KRCC↑	0.2613	0.2204	0.1918	0.2082	0.2402	0.1671	0.1685	0.2888	0.1984	0.2749	0.6655	0.1756	*0.6416*	**0.7242**

TABLE VI
COMPARISON BETWEEN THREE STATE-OF-THE-ART MODELS AND THE PROPOSED PATSP-NET WITH REGARD TO THE NUMBER OF PARAMETERS, COMPUTATIONAL COMPLEXITY (GMAC) AND INFERENCE COST (S).

Model	#Param (M)	GMac (G)	Inference Cost (S)
PDLF-PTSNet [16]	20.178	78.093	0.0071
DISTS [48]	14.715	30.720	0.0086
PMSTPN [24]	79.018	26.430	0.0900
PATSP-Net (Ours)	57.419	17.752	0.0664

accuracy values derived using the 14 methods are displayed in Fig. 11.

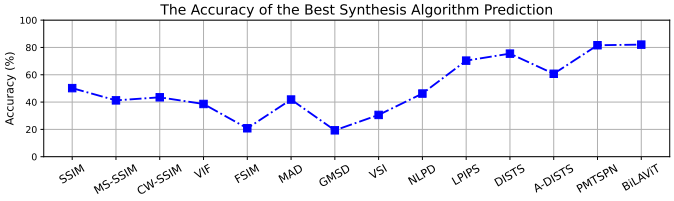

Fig. 11. The accuracy values of the best synthesis algorithm prediction obtained using 14 methods.

E. Performance Analysis

We compared the proposed PATSP-Net with three state-of-the-art models, including PDLF-PTSNet [16], DISTS [48] and PMSTPN [24], in terms of the number of parameters, computational complexity (GMac) and inference cost (S). Both the number of parameters and the computational complexity were computed using THOP[3]. The two values computed for the four models are (57.419M, 17.752GMac), (20.178M, 78.093GMac), (14.715M, 30.720GMac) and (79.018M, 26.430GMac) in turn. Given that the PDLF-PTSNet [16], DISTS [48] and PMSTPN [24] were trained using the setup proposed in this paper, it took approximately 0.0071, 0.0086 and 0.09 seconds, respectively, to run inference on a pair of 224×224 images. In contrast, it took around 0.0664 seconds for our model. The above results are reported in Table VI. Compared with the PMSTPN that we proposed in conference paper [24], the PATSP-Net achieved the better performance with the fewer parameters and the less computational complexity.

[3]https://github.com/Lyken17/pytorch-OpCounter

TABLE VII
EFFECT OF THE BACKBONE ON THE PERFORMANCE OF THE PATSP-NET.

Backbone	MSE ↓	PCC ↑	SRCC ↑	KRCC ↑
VGG-VD-16 [28]	0.0054	0.9262	*0.9037*	*0.7369*
ResNet-101 [29]	0.0114	0.8608	0.8508	0.6609
Conformer [25]	0.0073	0.8990	0.8625	0.6853
Vision Transformer [33]	0.0068	0.8982	0.8682	0.6914
BEiT [72]	0.0284	0.5041	0.4981	0.3426
CSWin Transformer [73]	*0.0047*	*0.9282*	0.9005	0.7337
SwinTransformer [26]	**0.0033**	**0.9504**	**0.9280**	**0.7783**

TABLE VIII
EFFECT OF THE COMPONENTS OF THE PATSP-NET ON ITS PERFORMANCE.

Variant	MSE ↓	PCC ↑	SRCC ↑	KRCC ↑
Hadamard → Concatenation	*0.0041*	*0.9392*	*0.9178*	*0.7619*
AVG Pooling → MAX Pooling	0.0046	0.9303	0.8906	0.7222
DConv → PatchMerging	0.0042	0.9360	0.9105	0.7472
All Scales → Scale 4	0.0047	0.9291	0.8908	0.7194
log → no log	0.0067	0.9166	0.9108	0.7486
PATSP-Net (Ours)	**0.0033**	**0.9504**	**0.9280**	**0.7783**

F. Ablation Studies

A series of ablation studies were performed for the texture similarity prediction task. For simplicity, only the *Pertex* [63] data set was used.

1) Effect of the Backbone: Regarding the backbone of the PATSP-Net, we examined four additional pre-trained models, including VGG-VD-16 [28], ResNet-101 [29], Conformer [25], Vision Transformer [33], BEiT [72] and CSWin Transformer [73] for the purpose of comparison. The results are shown in Table VII. As can be seen, these results are inferior to those produced when the pre-trained SwinTransformer [26] was utilized as the backbone.

2) Effect of the Components of the PATSP-Net: We also investigated the effect of the components within the PATSP-Net, including the Hadamard product (Hadamard), the average pooling operation (AVG Pooling), the dilated convolutional layer (DConv) and the multi-scale concatenation operation (All Scales), by replacing them using the concatenate operation (Concatenation), the max-pooling operation (MAX Pooling), the patch merging operation (PatchMerging) [26] and the last scale (Scale 4), respectively. In this context, we obtained four variants of the PATSP-Net. The results are reported in Table VIII. It can be seen that the PATSP-Net outperformed all the four variants no matter which metric was utilized.

TABLE IX
EFFECT OF DIFFERENT LOSS FUNCTIONS ON THE PERFORMANCE OF THE BILAVIT.

Loss Function	MSE ↓	PCC ↑	SRCC ↑	KRCC ↑
MAE	0.0042	0.9360	0.9056	0.7450
MSE	*0.0038*	*0.9418*	*0.9142*	*0.7546*
PLCCLoss [74]	0.1297	0.9115	0.8740	0.7019
RSLoss	**0.0033**	**0.9504**	**0.9280**	**0.7783**

TABLE X
EFFECT OF DIFFERENT LOSS FUNCTIONS ON THE PERFORMANCE OF THE BILAVIT.

FOLD	MSE ↓	PCC ↑	SRCC ↑	KRCC ↑
Fold 1	0.0033	0.9504	0.9280	0.7783
Fold 2	0.0037	0.9442	0.9179	0.7624
Fold 3	0.0036	0.9469	0.9336	0.7805
AVG	0.0035	0.9471	0.9265	0.7737

3) Effect of the Loss Function: To assess the impact of the loss function to the performance of the BiLAViT, we compared Mean Absolute Error (MAE), MSE and the proposed RSLoss. Additionally, we employed PLCCLoss [74] for comparison. The results are reported in Table IX. It can be seen that our RSLoss achieved the better result than both the MAE and MSE loss functions when they were used on the *Pertex* [63] data set together with the BiLAViT. We also tried to test the Spearman [55] loss function. However, the BiLAViT failed to converge along with this loss function.

4) K-Fold Cross-Validation: In order to assess the stability performance of the method, we re-divided the *Pertex* [63] data set into three different training and testing sets and reported the results three times along with the averages. The results are presented in Table X.

VIII. CONCLUSION

To address the challenge, which caused the discrepancy between the texture similarity data predicted using an algorithm and human visual perception, we proposed a Perception-Aware Texture Similarity Prediction Network (PATSP-Net). This network contained a Bilinear Lateral Attention Transformer network (BiLAViT) and the RSLoss loss function. The BiLAViT comprised a Siamese Feature Extraction Network (SFEN) and a Metric Learning Network (MLN), which were developed based on the mechanisms of human perception. On the other hand, the RSLoss measured both the ranking and scaling differences. To our knowledge, none of the BiLAViT and the RSLoss had been exploited for texture similarity tasks before. Given three data sets, the BiLAViT which was trained using the RSLoss, performed better than, or at least comparably to, its 14 counterparts for fine-grained texture similarity prediction tasks. This promising result should benefit from the combination of the BiLAViT and RSLoss, which learned the perception-aware texture representation and similarity metric.

REFERENCES

[1] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[2] A. D. Clarke, F. Halley, A. J. Newell, L. D. Griffin, and M. J. Chantler, "Perceptual similarity: A texture challenge." in *British Machine Vision Conference*. Citeseer, 2011, pp. 1–10.

[3] X. Dong, J. Dong, and M. J. Chantler, "Perceptual texture similarity estimation: An evaluation of computational features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 7, pp. 2429–2448, 2021.

[4] A. Bovik, M. Clark, and W. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 55–73, 1990.

[5] B. Manjunath and W. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837–842, 1996.

[6] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 6, pp. 610–621, 1973.

[7] D. G. Low, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[8] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2, 2003, pp. 1398–1402.

[9] J. Mao and A. K. Jain, "Texture classification and segmentation using multiresolution simultaneous autoregressive models," *Pattern recognition*, vol. 25, no. 2, pp. 173–188, 1992.

[10] R. M. Battleday, J. C. Peterson, and T. L. Griffiths, "Modeling human categorization of natural images using deep feature representations," *arXiv preprint arXiv:1711.04855*, 2017.

[11] K. M. Jozwik, N. Kriegeskorte, K. R. Storrs, and M. Mur, "Deep convolutional neural networks outperform feature-based but not categorical models in explaining object similarity judgments," *Frontiers in psychology*, vol. 8, p. 1726, 2017.

[12] J. C. Peterson, J. T. Abbott, and T. L. Griffiths, "Adapting deep network features to capture psychological representations: An abridged report," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017, pp. 4934–4938.

[13] R. T. Pramod and S. P. Arun, "Do computational models differ systematically from human object perception?" in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1601–1609.

[14] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 586–595.

[15] M. Cimpoi, S. Maji, I. Kokkinos, and A. Vedaldi, "Deep filter banks for texture recognition, description, and segmentation," *International Journal of Computer Vision*, vol. 118, no. 1, p. 65, 2016.

[16] Y. Gao, Y. Gan, L. Qi, H. Zhou, X. Dong, and J. Dong, "A perception-inspired deep learning framework for predicting perceptual texture similarity," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 10, pp. 3714–3726, 2020.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale visual recognition," 2015, proc. International Conference on Learning Representations.

[18] A. D. Clarke, X. Dong, and M. J. Chantler, "Does free-sorting provide a good estimate of visual similarity," *Predicting Perceptions*, pp. 17–20, 2012.

[19] L. Hudec and W. Bencsova, "Texture similarity evaluation via siamese convolutional neural network," in *2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2018, pp. 1–5.

[20] A. Rosenfeld, M. D. Solbach, and J. K. Tsotsos, "Totally looks like - how humans compare, compared to machines," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2042–20 423, 2018.

[21] K. Ding, R. Zhong, Z. Wang, Y. Yu, and Y. Fang, "Adaptive structure and texture similarity metric for image quality assessment and optimization," *IEEE Transactions on Multimedia*, pp. 1–13, 2023.

[22] M. R. Berthold and F. Höppner, "On clustering time series using euclidean distance and pearson correlation," *arXiv preprint arXiv:1601.02213*, 2016.

[23] J. Liu, J. Dong, X. Cai, L. Qi, and M. J. Chantler, "Visual perception of procedural textures: Identifying perceptual dimensions and predicting generation models," *PLoS ONE*, vol. 10, 2015.

[24] W. Wang and X. Dong, "Unifying the visual perception of humans and machines on fine-grained texture similarity," in *33rd British Machine Vision Conference 2022*. BMVA Press, 2022.

[25] Z. Peng, W. Huang, S. Gu, L. Xie, Y. Wang, J. Jiao, and Q. Ye, "Conformer: Local features coupling global representations for visual recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 367–376.

[26] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *2021 IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9992–10 002.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[32] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[33] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[35] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[36] Y. Wang, Z. Xu, X. Wang, C. Shen, B. Cheng, H. Shen, and H. Xia, "End-to-end video instance segmentation with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 8741–8750.

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[38] M. Kaya and H. Ş. Bilge, "Deep metric learning: A survey," *Symmetry*, vol. 11, no. 9, p. 1066, 2019.

[39] Y. Duan, J. Lu, J. Feng, and J. Zhou, "Deep localized metric learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2644–2656, 2017.

[40] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," *Advances in neural information processing systems*, vol. 6, 1993.

[41] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International workshop on similarity-based pattern recognition*. Springer, 2015, pp. 84–92.

[42] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 1735–1742, 2006.

[43] L. Chen and Y. He, "Dress fashionably: Learn fashion collocation with deep mixed-category metric learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[44] H. Oh Song, S. Jegelka, V. Rathod, and K. Murphy, "Deep metric learning via facility location," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5382–5390.

[45] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev, "Metric learning with adaptive density discrimination," *arXiv:1511.05939*, 2015.

[46] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[47] E. Prashnani, H. Cai, Y. Mostofi, and P. Sen, "Pieapp: Perceptual image-error assessment through pairwise preference," in *The IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[48] K. Ding, K. Ma, S. Wang, and E. P. Simoncelli, "Image quality assessment: Unifying structure and texture similarity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 5, pp. 2567–2581, 2022.

[49] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear convolutional neural networks for fine-grained visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1309–1322, 2018.

[50] J.-H. Kim, K.-W. On, W. Lim, J. Kim, J.-W. Ha, and B.-T. Zhang, "Hadamard product for low-rank bilinear pooling," in *International Conference on Learning Representations*, 2017.

[51] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, "A2-nets: Double attention networks," in *Neural Information Processing Systems*, 2018.

[52] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[54] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[55] M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga, "Fast differentiable sorting and ranking," in *International Conference on Machine Learning*. PMLR, 2020, pp. 950–959.

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

[57] L. Breiman, "Random forests." *Mach Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[58] D. Dai, H. Riemenschneider, and L. V. Gool, "The synthesizability of texture examples," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3027–3034.

[59] B. Galerne, Y. Gousseau, and J.-M. Morel, "Random phase textures: Theory and synthesis," *IEEE Transactions on Image Processing*, vol. 20, no. 1, pp. 257–267, 2011.

[60] A. Zalesny, V. Ferrari, G. Caenen, and L. Van Gool, "Composite texture synthesis," *International journal of computer vision*, vol. 62, pp. 161–176, 2005.

[61] 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*, 2001, pp. 341–346.

[62] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *International journal of computer vision*, vol. 40, pp. 49–70, 2000.

[63] Fraser, "Perceptually relevant browsing environments for large texture databases," *Heriot-Watt University*, 2012.

[64] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik, and M. K. Markey, "Complex wavelet structural similarity: A new image similarity index," *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2385–2401, 2009.

[65] H. Sheikh and A. Bovik, "Image information and visual quality," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, 2006.

[66] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "Fsim: A feature similarity index for image quality assessment," *IEEE Transactions on Image Processing*, vol. 20, no. 8, pp. 2378–2386, 2011.

[67] E. C. Larson and D. M. Chandler, "Most apparent distortion: full-reference image quality assessment and the role of strategy," *J. Electronic Imaging*, vol. 19, p. 011006, 2010.

[68] W. Xue, L. Zhang, X. Mou, and A. C. Bovik, "Gradient magnitude similarity deviation: A highly efficient perceptual image quality index," *IEEE Transactions on Image Processing*, vol. 23, no. 2, pp. 684–695, 2014.

[69] L. Zhang, Y. Shen, and H. Li, "Vsi: A visual saliency-induced index for perceptual image quality assessment," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4270–4281, 2014.

[70] V. Laparra, J. Ballé, A. Berardino, and E. P. Simoncelii, "Perceptual image quality assessment using a normalized laplacian pyramid," in *Human Vision and Electronic Imaging*, 2016, pp. 43–48.

[71] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing top k lists," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003, p. 28–36.

[72] W. Wang, H. Bao, L. Dong, J. Bjorck, Z. Peng, Q. Liu, K. Aggarwal, O. K. Mohammed, S. Singhal, S. Som, and F. Wei, "Image as a foreign language: BEiT pretraining for vision and vision-language tasks," in

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.

[73] X. Dong, J. Bao, D. Chen, W. Zhang, N. Yu, L. Yuan, D. Chen, and B. Guo, "Cswin transformer: A general vision transformer backbone with cross-shaped windows," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 12 114–12 124.

[74] W. Xian, M. Zhou, B. Fang, X. Liao, C. Ji, T. Xiang, and W. Jia, "Spatiotemporal feature hierarchy-based blind prediction of natural video quality via transfer learning," *IEEE Transactions on Broadcasting*, vol. 69, no. 1, pp. 130–143, 2023.

Weibo Wang received the B.S. degree in Network Engineering from Shandong Jianzhu University, Jinan, China, in 2020. He is currently pursuing the M.S. degree in Network Information and Security at Ocean University of China, Qingdao, China, under the supervision of Prof. Xinghui Dong. His research interests include computer vision, texture retrieval and visual perception.

Xinghui Dong received the PhD degree from Heriot-Watt University, U.K., in 2014. He worked with the Centre for Imaging Sciences, the University of Manchester, U.K., between 2015 and 2021. Then he jointed Ocean University of China in 2021. He is currently a professor at the Ocean University of China. His research interests include computer vision, defect detection, texture analysis, and visual perception.