Deep convolutional neural networks for regular texture recognition

Ni Liu [Corresp., 1] , **Mitchell Rogers** [2] , **Hua Cui** [1] , **Weiyu Liu** [3] , **Xizhi Li** [4] , **Patrice Delmas** [2]

[1] School of Information Engineering, Chang'an University, Xi'an, ShaanXi Province, China

[2] Department of Computer Science, The University of Auckland, Auckland, New Zealand

[3] School of Electronics and Control Engineering, Chang'an University, Xi'an, China

[4] Henan Highway Development Co. LTD. Anxin Branch, Xinxiang City, Henan Province, China

Corresponding Author: Ni Liu
Email address: niliu@chd.edu.cn

Regular textures are frequently found in man-made environments and some biological and physical images. There is a wide range of applications for recognizing and locating regular textures. In this work, we used deep CNNs as a general method for modelling and classifying regular and irregular textures. We created a new regular texture database and investigated two set of deep CNNs based methods for regular and irregular texture classification. First, the classic CNN models (e.g. Inception, Residual network, etc.) were used in a standard way. These two-class CNN classifiers were trained by fine-tuning networks using our new regular texture database. Next, we transformed the trained filter features of the last convolutional layer into a vector representation using Fisher Vector pooling (FV). Such representations can be efficiently used for a wide range of machine learning tasks such as classification or clustering, thus more transferable from one domain to another. Our experiments show that the standard CNNs attained sufficient accuracy for regular texture recognition tasks. The Fisher representations combined with Support Vector Machine (SVM) also showed high performance for regular and irregular texture classification.

Deep Convolutional Neural Networks for Regular Texture Recognition

Ni Liu[1], Mitchell Rogers[4], Hua Cui[1], Weiyu Liu[2], Xizhi Li[3], and Patrice Delmas[4]

[1]**School of Information Engineering, Chang'an University, Xi'an, ShaanXi Province, 710064, China**

[2]**School of Electronics and Control Engineering, Chang'an University, Xi'an, ShaanXi Province, 710064, China**

[3]**Henan Highway Development Co. LTD. Anxin Branch**

[4]**IVSlab, Department of Computer Science, The University of Auckland, Auckland 1010, New Zealand**

Corresponding author:

Ni Liu[1]

Email address: niliu@chd.edu.cn

ABSTRACT

Regular textures are frequently found in man-made environments and some biological and physical images. There is a wide range of applications for recognizing and locating regular textures. In this work, we used deep CNNs as a general method for modelling and classifying regular and irregular textures. We created a new regular texture database and investigated two set of deep CNNs based methods for regular and irregular texture classification. First, the classic CNN models (e.g. Inception, Residual network, etc.) were used in a standard way. These two-class CNN classifiers were trained by fine-tuning networks using our new regular texture database. Next, we transformed the trained filter features of the last convolutional layer into a vector representation using Fisher Vector pooling (FV). Such representations can be efficiently used for a wide range of machine learning tasks such as classification or clustering, thus more transferable from one domain to another. Our experiments show that the standard CNNs attained sufficient accuracy for regular texture recognition tasks. The Fisher representations combined with Support Vector Machine (SVM) also showed high performance for regular and irregular texture classification.

INTRODUCTION

Textures are composed of atomic units called textons. The layout of textons can be regular or irregular (Hettiarachchi et al., 2014). Regular textures exhibit strongly periodic or quasi-periodic behavior, often found on building surfaces, bricks, floor tiles, fences, vegetation and crop fields. A period describes the distance between repeating textons. For images, we consider the spatial distance. In contrast to regular textures, irregular textures have "noisy" or "stochastic" patterns with random intensity variations. In practice, strong periodicity rarely occurs in nature, these textures are usually quasi-periodic, which means patterns recur, but the periodicity has random components (Hettiarachchi et al., 2014; Liu et al., 2015).

The layout of regular textures are an important property that allows an object to be correctly identified. The recognition and segmentation of regular textures could be used across many domains such as computer vision, computer graphics, and medical imaging (Cai and Baciu, 2011; Sun et al., 2019). Applications of regular texture detection include: analysis and quantification of building patterns for urban planning (Yu et al., 2017), terrain segmentation (Aksoy et al., 2012), texture replacement in images (Liu et al., 2004; Hettiarachchi et al., 2014), texture synthesis (Lin et al., 2006), quantification of drosophila eye surface regularity (Hermano, 2017), and pattern segmentation in woven fabric (Cai and Baciu, 2011).

The analysis of repetitive patterns is a long standing problem in texture analysis (Yu et al., 2017). A pioneering work by Leung and Malik in the 90's used a spatial tracking approach to find repeating elements of images (Leung and Malik, 1996). First, they detected windows of possible candidates for textons.

Then, neighboring regions of each candidate were searched for similar structures. Regions with similar repeating elements were grouped together, and individual elements were marked. The main drawback of this method is that it did not consider complex textons, periods or texton regularity. In (Schaffalitzky and Zisserman, 1998), Schaffalitzky and Zisserman proposed a grouping algorithm based on local affine transformations. However, their method was sensitive to structural distortions. More recently, the texton grouping problem was reformulated as a lattice detection problem (Hays et al., 2006; Park et al., 2009). In (Hays et al., 2006) the solution is found by iteratively proposing textons and assigning neighbors to the textons. In (Park et al., 2009) a Markov Random Field (MRF) with a Mean-Shift Belief Propagation method was used. Other approaches to texton grouping include optimization of shape alignment (Cai and Baciu, 2011), structural regularity using symmetry groups (Liu et al., 2008), projection profiles (Aksoy et al., 2012) and frequency filtering (Hettiarachchi et al., 2014; Sun et al., 2021).

As opposed to developing a specialized method for specific scenes and applications, our objective is to develop a general method for recognizing texture regularity, that can be easily applied to regular texture related areas only with light fine-tuning needed. According to the level of regularity, we classify textures into two categories, regular or irregular. This configuration allows us to seek standard representations of regular textures by learning their common characteristics with an easy collection of a large number of samples. Then the developed method can be fine-tuned into specific areas with only a few labeled images needed. We envision that our work could be used to enhance image retrieval, object recognition and 3D structure identification tasks (Leung and Malik, 1996). For this challenging task, we therefore decided to employ the power of deep convolutional neural networks (CNNs).

CNNs have been shown to outperform traditional approaches for image classification tasks. This prompted the quick adoption of deep learning methods in many image classification fields such medical (Litjens et al., 2017; Wang et al., 2021; Singh et al., 2020), aerial (Mnih and Hinton, 2012; Petrovska et al., 2020), vehicle recognition (Adu-Gyamfi et al., 2017), traffic congestion recognition (Cui et al., 2020), gait recognition (Tan et al., 2014; Zou et al., 2020), fruit recognition (Murean and Oltean, 2018; Saedi and Khosravi, 2020) as well as for general images (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; He et al., 2016; Szegedy et al., 2013). Unlike the first perceptrons (Rosenblatt, 1958) which have a single layer of neurons, deep learning models consist of multiple layers of inter-connected neurons (Widrow and Lehr, 1990) with back-propagation (Rumelhart et al., 1988). CNNs can learn increasingly complex representations of objects by exploiting the compositional nature of images. High-level features like faces are composed of lower-level features such as edges and lines (Llamas et al., 2017). In CNNs, the lower layers correspond to corners, edges and color conjunctions, while deeper layers correspond to more complex high-level features such as faces, text, wheels and flowers (Hossain and Serikawa, 2013a). CNN models have achieved state-of-the-art performance for image recognition with architectures such as LeNet-5 (Lecun et al., 2001), AlexNet (Krizhevsky et al., 2012), VGGNet (Simonyan and Zisserman, 2014), and more recently Inception network (GoogLeNet) (Szegedy et al., 2015, 2016b), and Residual network (ResNet) (He et al., 2016). Most of these architectures feature deeper layers, such as Inception network which has 22 layers.

In this paper, we investigated two sets of CNN based methods for regular texture modelling and classification. First, the classic CNN models were used in a standard way. The output layer was set to two categories while other layers maintained their original architectures. These two-class classifiers were trained using our newly created regular texture. The following state-of-the-art CNN models were tested: Inception network, Residual network and Inception-ResNet-v2 (Szegedy et al., 2016a). Second, we applied Fisher Vector pooling (FV) to the learned features (filter responses) of the above trained CNNs (Perronnin and Dance, 2007). The generated vector representations, combined with classifiers such as SVM, can be efficiently used for a wide range of machine learning classification or segmentation tasks, and as such are more transferable from one domain to another. The motivation for this method stems from the state-of-the-art texture analysis method FV-CNN (Cimpoi et al., 2015), proposed by the Visual Geometry Group of Oxford University. However, they only used pre-trained VGG features on ImageNet ILSVRC data, which may not be optimal for modelling texture regularity.

The main contributions of our paper are: i) A newly created regular texture database, which is publicly available. This database complements well the state-of-the-art Describable Textures Dataset (DTD) from the Visual Geometry Group (Cimpoi et al., 2014). Their database is comprised of visual perception of textures in features such as dotted, bumpy, cracked, striped and line-like and so on. Their dataset mainly focused on textons (texture elements) and did not consider the global configuration of textons

¹⁰² and the degrees of regularity in textures. ii) A generalized regular texture modelling and recognition
¹⁰³ framework. The trained deep CNN models and further generated Fisher representations were robust to
¹⁰⁴ different texture layouts, pattern complexity, texton variability and viewing angles, and is transferable
¹⁰⁵ from one domain to another. Our experiments showed that both methods reached remarkable accuracy,
¹⁰⁶ with a best performance of 98% for general regular texture classification.

METHODS

¹⁰⁸ In this work, our main goal is to classify texture images by assigning them to a specific label. We used
¹⁰⁹ Convolutional Neural Networks to learn from a set of labelled training data. Then, the model was used to
¹¹⁰ predict class labels for a test set which has not been seen during training. Here, we consider two texture
¹¹¹ categories, namely, regular or irregular patterns. The images were manually labelled by volunteers using
¹¹² a consensus approach.

Regular texture database

¹¹⁴ A considerable number of images are needed in order to train CNNs. Several image databases have been
¹¹⁵ developed for generic texture analysis including the natural texture image database, texture of materials
¹¹⁶ database, and dynamic texture database (Hossain and Serikawa, 2013b). However, they are mainly texture
¹¹⁷ image banks and contain limited regular textures. Only a small number of databases, CMU Near-Regular
¹¹⁸ Texture (NRT) database (Liu et al., 2004) and PSU Near-Regular texture database (PSU-NRTDB, 2005)
¹¹⁹ provide near-regular textures. However, these databases are small in size and the data is mostly unlabelled.
¹²⁰ A related dataset is the Describable Texture Dataset (DTD) (Cimpoi et al., 2014), which categorizes
¹²¹ textures by adjectives e.g., dotted, cracked, or wrinkled. This database focuses more on textons rather
¹²² than the global layout of textons. From these categories, we identified a subset of classes "grid, banded,
¹²³ chequered, grooved, meshed" that can be used to describe regular textures.

Table 1. The number of regular and irregular textures selected from online resources.

Database 1	# Regular	# Irregular
PSUN	335	98
Texturelib	363	588
Brodatz	36	46
CUReT	2	0
DTD	216	459
Kylberg	7	19
Texture Library	16	20
Sketchup texture	14	0
Google images	40	0
Flickr	201	0
Total number	1230	1230

¹²⁴ We created our own texture database which focused on regular elements. Our database consists of
¹²⁵ two categories: regular textures and irregular textures. We manually curated this database by selecting
¹²⁶ relevant textures from existing online libraries:

¹²⁷ • PSU near-regular texture (PSU-NRTDB, 2005)

¹²⁸ • Texturelib (Chugai, 2019)

¹²⁹ • Brodatz dataset (Randen, 2012)

¹³⁰ • Columbia-Utrecht Reflectance and Texture (CUReT) (Dana et al., 1999)

¹³¹ • Describable Texture Dataset (DTD) (Cimpoi et al., 2014)

132   • Kylberg database (Sarafraz, 2011)

133   • Texture Library (Francv, 2010)

134   • Sketchuptexture (SketchupTextureClub, 2010)

135   To supplement our database, we also obtained images from Flickr (MidCenturyStyles, 2015) and
136 Google images (Bayless, 2019). The number of images based on source location and image class are
137 listed in Table 1. Our regular database has a total of 1230 regular textures and 1230 irregular textures,
138 including images of different sizes.
139   As color information is irrelevant to the layout of textons, in our experiments, all images are first
140 transformed to grey-scale. Images are then expanded into three channels by repeating the gray-scale data.
141 Figure 1 shows examples of the two image categories in our database.
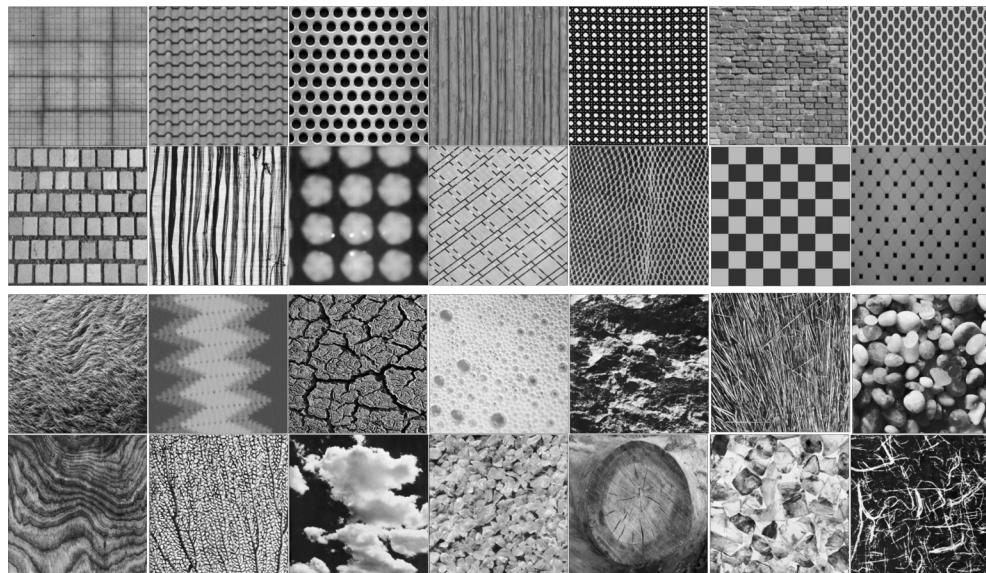
Figure 1. Examples of regular (top two rows) and irregular textures (bottom two rows) from our
database. Our database has a total of 1230 regular textures and 1230 irregular textures.

142 **Deep Convolutional Neural Networks**

CNNs are used commonly for image classification. The CNN architecture is formed by several intermedi-
ate layers, which are typically convolutional layers and pooling layers. Followed by a fully-connected
layer which performs the final classification using the output from the previous layers. Figure 2 shows an
example of a simple CNN architecture.
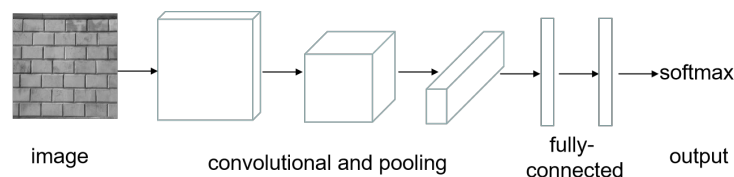
Figure 2. A typical convolutional neural network.

The layers of a CNN are described in detail below.

Convolution layer: The convolutional layer is the core building block of a CNN. A small filter (usually
3×3) moves over the image, generating a map of activation called a features map. This makes certain
characteristics become more dominant in the output image. For example, edges can be detected by filters

that highlight the gradient in a particular direction. The output of each layer can be formulated as:

$$\mathbf{a}^l = \sigma\left(\omega^l * \mathbf{a}^{l-1} + \mathbf{b}^l\right) \tag{1}$$

143   where l represents the lth layer, $*$ is a convolution operation (filter), ω^l is the weight matrix, $\mathbf{b}^l$ is the
144   vector (bias) and σ is the nonlinear activation function.

146   Activation layer: After the convolution, non-linear activation functions are applied to the features maps.
147   The most common used is the ReLU (Rectified Linear Unit): $f(x) = \max(0, x)$. This function propagates
148   the gradient efficiently and alleviates the problem of vanishing gradient when there are many layers.

150   Pooling: The pooling or down-sampling layer is responsible for reducing the size of the activation
151   maps. Grouping operations (max-pooling) find the maximum value of a sample window and pass this
152   value as a summary of that area. As a result, the size of the data is reduced by a factor equal to the size of
153   the sample window. In general, it reduces the computational power requirements progressively through
154   the network. It also reinforces invariance properties to small changes such as feature position and image
155   distortion.

157   Fully-connected layer (FCL): The objective of a fully-connected layer is to take the results of the
158   convolution/pooling process and use them to classify the image. Adding a fully-connected layer is
159   beneficial when learning combinations of non-linear features.

161   Training a neural network consists of minimizing a global error function. The error function calculates the
162   error between the output class and the true class, and averages this error over all input images. The training
163   steps are: i) calculate the output class using feedforward; ii) calculate the error; iii) back-propagate the
164   gradients and update the weights.
165   Stochastic gradient descent iteratively minimizes the errors by updating the network weights in the
166   opposite direction to the gradient of the cost function (Rumelhart et al., 1988). The frequency of the error
167   updates can be done per batch, per sample or anywhere in between. Typically mini-batches give the best
168   balance between computational time and accuracy.
169   For this method, there are many optimizations; learning rate, weight decay (a regularization term,
170   which penalizes changes in the weights and prevents them from being too large), and momentum (the
171   average of the previous gradients which reduces oscillations that cause local minima, thus accelerating
172   convergence).
173   The following equation is used to update the values of the network weights w_i:

$$v_{i+1} := r \times v_i - \lambda \times \varepsilon \times w_i - \varepsilon \times \left\langle \frac{\partial L}{\partial \omega}\Big|_{\omega_i} \right\rangle_{\mathbf{D}_i} \tag{2}$$

$$\omega_{i+1} := \omega_i + v_{i+1} \tag{3}$$

174   where i is the iteration index, ε is the learning rate, λ is the weight decay, v is the momentum
175   variable, r is the momentum weight and $\left\langle \frac{\partial L}{\partial \omega}\Big|_{\omega_i} \right\rangle_{\mathbf{D}_i}$ is the average error over the ith batch $\mathbf{D}$ of the
176   derivative of the objective function with respect to ω, evaluated at w_i. The cross-entropy loss function
177   was: $L(\omega) = -\sum_j \sum_c y_{jc} \log f_c(g_j)$, where y_{jc} denotes the label for an image with indexed j and class c;
178   $f_c(g_j)$ is the prediction probability of class c for image g.

Regular textures classification
180   We investigated two sets of Deep CNNs based methods for modelling regular textures and classifying
181   regular and irregular textures.
182   *Standard Deep CNNs* First, we used classic CNNs in a standard way. The output layer was set
183   to the number of texture categories while other layers maintained their original architectures. These
184   two-classes classifiers were trained using our newly created regular texture database, by fine-tuning
185   ImageNet pre-trained networks. Further details for the three state-of-the-art CNNs are introduced as
186   follows.

187 The Inception-v3 network (Szegedy et al., 2016b) uses Google's Inception architecture (Szegedy
188 et al., 2015) for image recognition. The "Inception" modules concatenate filters of different sizes and
189 dimensions into a single new filter. They use a combination of a small kernel (1×1, 3×3, and 5×5
190 convolutions) and a few convolutional filters with large kernel size (Figure 3). Convolutions of different
191 sizes were used to capture details at various scales. Another feature of the module is a bottleneck layer of
1×1 convolutions. This is designed to reduce the computational complexity.
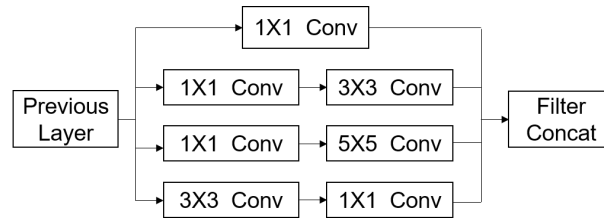
Figure 3. Inception module.

192
193 The residual network (He et al., 2016) introduces an identity function between layers. Conventional
194 networks learn underlying nonlinear mappings $y = \mathcal{H}(x)$ of stacked layers (Figure 4 (left)), with x
195 denoting the inputs to the first of these layers. In residual networks, a new nonlinear function $y = \mathcal{F}(x) + x$,
196 here $\mathcal{F}(x)$ is called the residual (Figure 4 (right)). The Residual mapping avoids the vanishing gradient
197 problem, so the depth of the neural network can be increased without increasing the number of parameters
to optimize.
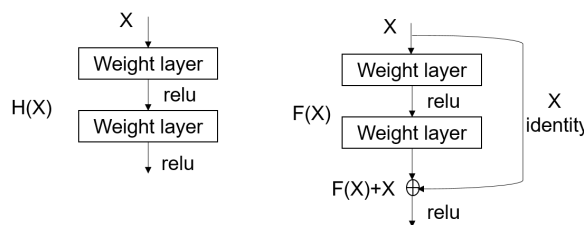
Figure 4. Normal Convolutional Neural Network (left); shortcut connections of ResNet architecture
(right) (He et al., 2016).

198
199 The Inception-ResNet-v2 network (Szegedy et al., 2017) is considered as state-of-the-art for ImageNet
200 recognition. Residual connections include shortcuts in models that allow researchers to train even deeper
201 networks to achieve a higher performance. This has also allowed Inception blocks to be further simplified.
202 *Fisher Vector Representations* The above standard CNN-based methods can be interpreted as extract-
203 ing local convolutional features and pooling them in a global image representation by the Fully-Connected
204 (FC) layers. In (Cimpoi et al., 2015), FC pooling was replaced by Fisher Vector pooling (FV) in the
205 pre-trained VGG network (on ImageNet ILSVRC data) for general texture representation. This allowed
206 for state-of-the-art performance across various texture databases and an efficient method benefit in trans-
207 ferring features from one domain to another. Motivated by their work, we further applied Fisher Vector
208 pooling to the filter features of the last convolutional layer of the above learned standard CNNs. Vector
209 representations generated in this way are quite suitable for tasks such as classification with Support
210 Vector Machine (SVM). They proved more transferable from one domain to another and can be applied
211 out-of-the-box without extensive fine-tuning, with only a few labeled images needed to train SVM.
212 Our framework (see Figure 5) first extracts deep convolutional features from the last convolutional
213 layers of CNNs trained on our regular texture dataset. Fisher Vector pooling (FV) maps a sequence
214 $\mathscr{F} = (\mathbf{f}_1, ..., \mathbf{f}_n), \mathbf{f}_i \in \mathbb{R}^D$ of local convolutional descriptors to a feature vector, where D is the dimensionality
215 of descriptors. Local descriptors are first assigned to elements in a visual dictionary, constructed by a K
216 modes Gaussian Mixture Model (GMM) $(\pi_k, \mu_k, \Sigma_k), k = 1, ..., K$, where $\pi_k \in \mathbb{R}$ is the prior probability of
217 the component, $\mu_k \in \mathbb{R}^D$ the Gaussian mean and $\Sigma_k \in \mathbb{R}^{D \times D}$ the Gaussian covariance. The assignments
218 $\eta(\mathbf{f}_i)$ are given by the posterior probability of each GMM component. Rather than storing visual word
219 occurrences only, these representations store a statistics of the difference between dictionary elements
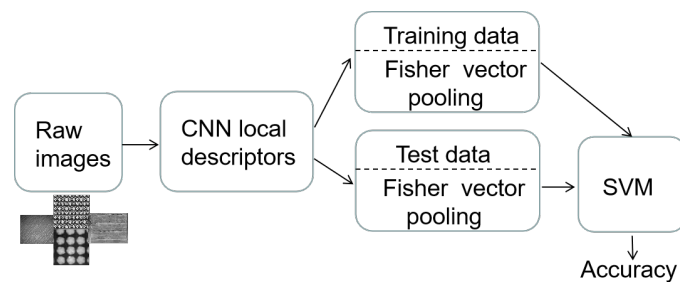
Figure 5. Framework for Fisher vector representations of CNN features, combined with SVM for texture classification.

220   and pooled local features. The FV descriptor encoder $\eta \text{FV}(\mathbf{f}_i)$ includes both first order $\Sigma_k^{-\frac{1}{2}}(\mathbf{f}_i - \mu_k)$ and
221   second order $\Sigma_k^{-1}(\mathbf{f}_i - \mu_k) \odot (\mathbf{f}_i - \mu_k) - \mathbf{1}$ statistics [see (Perronnin and Dance, 2007) for more details].
222   The dimensionality of the Fisher representation is $2KD$. Learning uses a standard non-linear SVM solver.
223   We denote this set of methods as FV-CNNreg.

Experimental Setup

225   In our experiments, we evaluate the performance of standard CNNs and FV-CNNreg for distinguishing
226   regular and irregular textures. We set a 7:1:2 ratio in our database, resulting in 860, 120, and 250 texture
227   images for training, validation and testing for each category. We used 10-fold cross-validation and the
228   averages of 10 splits for performance comparison. Input images were pre-processed to standardized size
229   and transformed to grey-scale. All the experiments involving CNNs training were implemented on the
230   Keras platform (Chollet, 2015). The VLFeat library in MATLAB was used for the computation of FV and
231   SVM solvers.

The training of these networks in Keras requires the adjustment of hyperparameters including learning rate, weight decay, momentum etc. The initial learning rate is often the most important hyperparameter. Its value is usually less than 1 and greater than 1e-6. Usually, 0.01 is used as a typical value. Keras provides the Stochastic Gradient Descent (SGD) class that implements the stochastic gradient descent optimizer with a learning rate decay and momentum. The learning rate is calculated at the end of each mini-batch as follows (Brownlee, 2020):

$$\varepsilon_i = \varepsilon_0 \times \frac{1}{(1 + \lambda \times i)} \tag{4}$$

232   Where ε_i is the learning rate for the current iteration i; ε_0 is the intial learning rate specified as an argument
233   to SGD and λ is the decay rate which is greater than zero. We investigated the effect of a set of different
234   values of the learning rate decay for CNNs in the Results section. We also compared the effect of using
235   data augmentation features such as the shift, flip and rotation from Keras.

RESULTS

237   *In the first experiment,* we evaluated the performance of standard CNN models. We report the classifica-
238   tion accuracy in Table 2 for three state-of-the-art classical CNN models, InceptionV3, ResNet-50 and
239   Inception-ResNet-v2. We also evaluated several combinations of the parameters of the neural networks to
240   be tuned. The hyperparameters finally used for offering this best results are presented in Table 3. The
241   initial learning rate $\varepsilon_0 = 0.01$ and the momentum $v = 0.9$ are typical choices (Brownlee, 2020).

Inception-ResNet-v2 appears to be the most robust model as seen in Table 2. This is mainly attributed to the beneficial combination of the Inception architecture with residual connections creating a large-scale architecture with more layers and parameters than other CNNs. Figure 6 shows the improvement of the accuracy during the training phase of the Inception-ResNet-v2 network using the validation images of our database. It also demonstrated how the associated cost reduced during training in one experiment. The number of images misclassified by Inception-ResNet-v2 is shown in Table 4.

To better understand the performance benefits of Inception-ResNet-v2, we analyzed correctly classified images (with class probabilities $p \geq 0.50$, see Figure 7), and misclassified images (with class probabilities

Table 2. Comparison of accuracy (%) for standard CNN models for 10-fold cross-validation experiments.

Folds	ResNet-50	InceptionV3	Inception-ResNet-v2
1	94.00	96.80	97.40
2	91.00	96.00	97.00
3	91.00	96.80	98.20
4	93.80	96.80	98.80
5	93.10	96.20	96.60
6	95.60	97.60	98.40
7	96.20	96.20	97.60
8	95.20	96.40	97.80
9	97.40	97.60	97.40
10	96.40	97.80	98.60
Mean	94.37	96.82	97.78
St.dev.	2.20	0.65	0.71

Table 3. Hyperparameters used in fine-tuning the three standard CNNs.

Model	Initial Learning Rate	Learning Rate Decay	Batch Size	Epoch
Inception-ResNet-v2	0.01	0.01	32	200
InceptionV3	0.01	0.1	32	200
ResNet-50	0.01	10	32	200

Table 4. The number of images misclassified by Inception-ResNet-v2. Type1 denotes the number of regular images classified as irregular. Type2 denotes the number of irregular images classified as regular. The test data has 250 regular and 250 irregular textures.

Folds	1	2	3	4	5	6	7	8	9	10
Type1 No.	8	4	2	2	7	5	3	2	3	4
Type2 No.	5	11	7	4	10	3	9	9	10	3
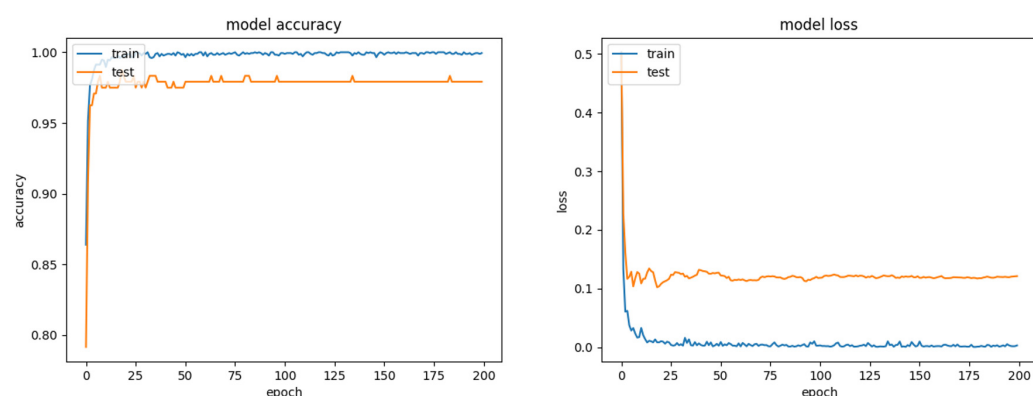
Figure 6. Training and validation accuracy of Inception-ResNet-v2 and associated cost during training.

$p < 0.50$, see Figure 8). Images are numbered in row-major order. The predicted class probabilities are shown below each image. We observed that the CNN correctly recognized most regular patterns: wallpapers (images 2-7), store shells (images 8 and 9), architectural surfaces (images 10-13), floor patterns (images 15-18), stone wall (image 19-21), fabric patterns (images 22-25). We can see that these patterns
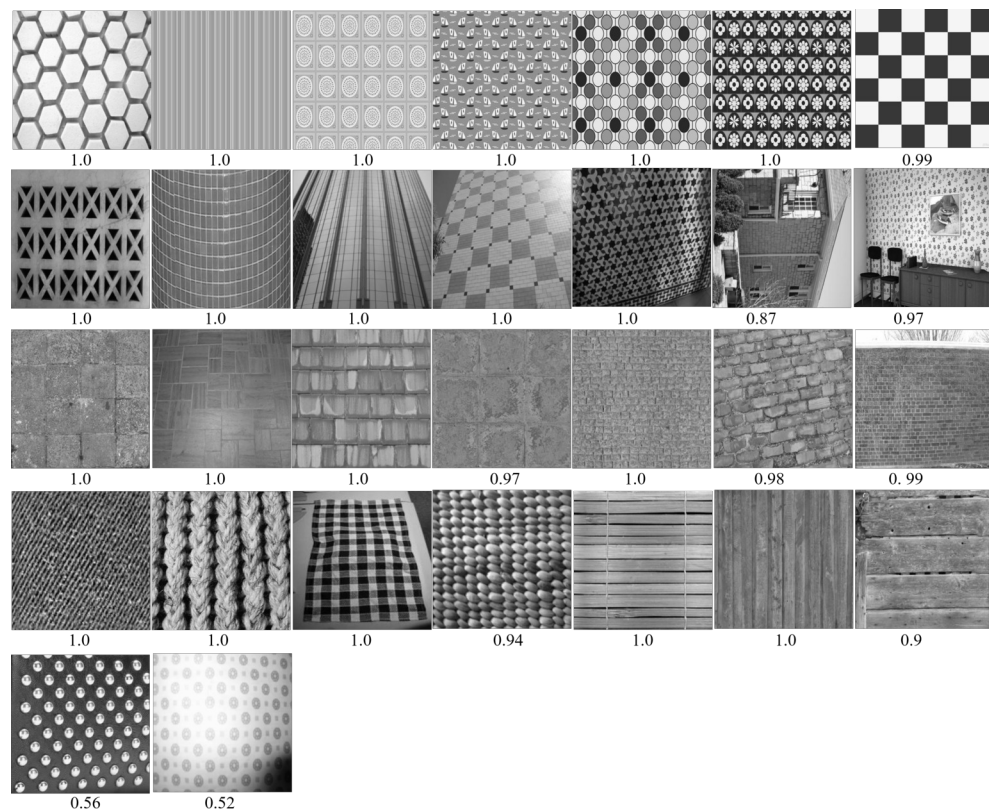
Figure 7. For regular textures, examples of correctly classified by Inception-ResNet-v2. Images are numbered from 1 to 30 in row-major order. Images were scaled to 256×256 for visualization. Predicted class probabilities are listed under each image.
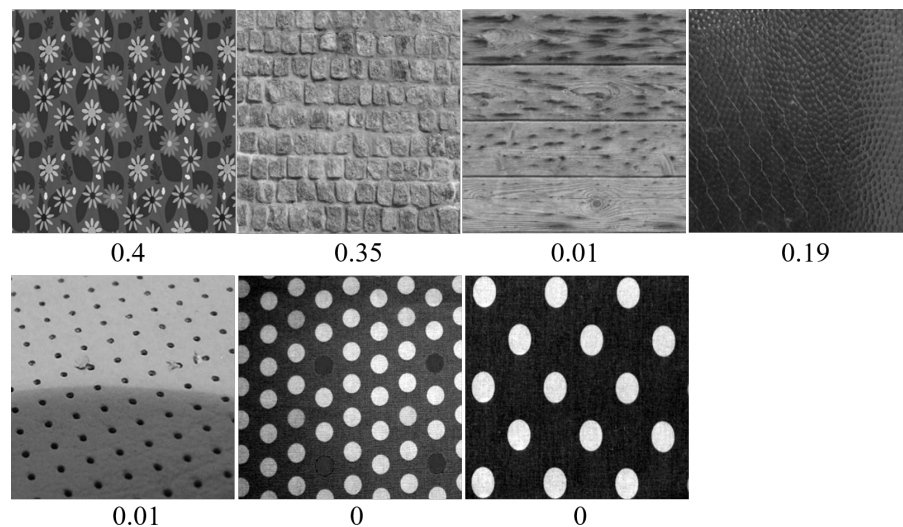
Figure 8. For regular textures, examples of wrongly classified images by Inception-ResNet-v2. Images are numbered from 1 to 7 in row-major order. The predicted class probabilities are shown below each image.

254 vary in shape. These easy cases include dotted, striped, lattice, chequered and honeycombed textons.
255 Regular patterns with different viewing angles are also correctly recognized (images 10-14). We observed
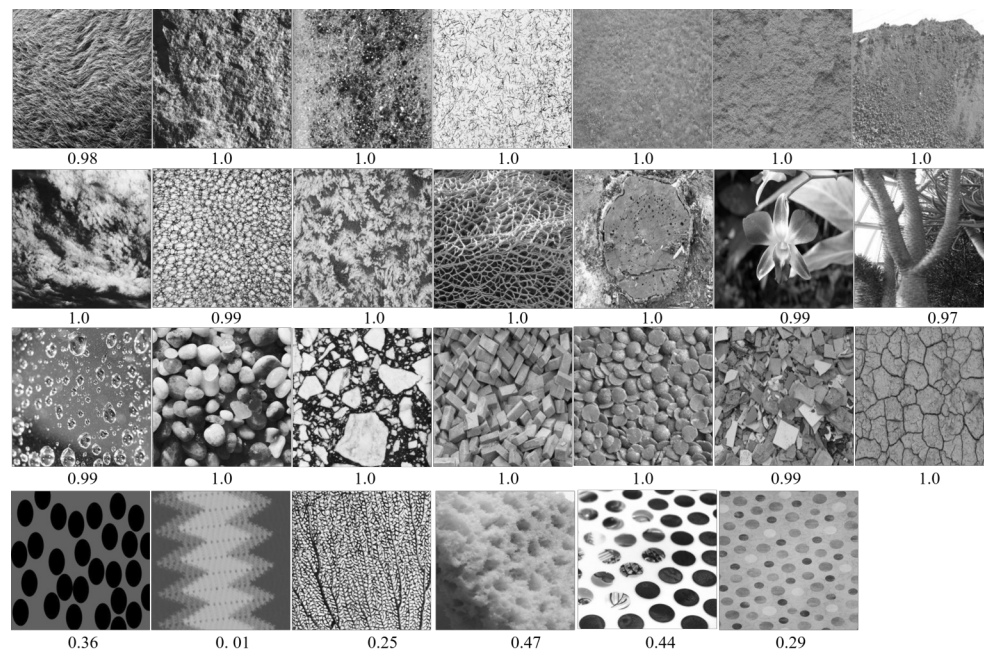
Figure 9. For irregular textures, examples of correctly classified (top three rows) and all wrongly classified images by Inception-ResNet-v2. Images are numbered from 1 to 27 in row-major order. The predicted class probabilities are shown below each image.

₂₅₆ that near-regular textures usually have lower probabilities.

₂₅₇ As shown in Figure 1 and Figure 7, our regular texture database contains many heterogeneous images
₂₅₈ from a variety of scenes. The high accuracy achieved by Inception-ResNet-v2 suggests CNNs are highly
₂₅₉ generalizable. This also suggests that the database we built is reasonable for regular texture recognition
₂₆₀ tasks. We believe CNNs have the potential to be a powerful tool for building general methods for
₂₆₁ recognizing regular textures.

₂₆₂ The wrongly classified examples are all shown in Figure 8. These images have low predicted
₂₆₃ probabilities ($p < 0.50$). It is important to take a close examination of these images, in order to deeply
₂₆₄ understand the performance of the trained CNN on recognizing regular textures. The first wallpaper
₂₆₅ textons have a regular layout, but very complex patterns. This can be seen as an outlier as most patterns
₂₆₆ are relatively simple. The stonewall (image 2) is a borderline case, with an accuracy of 0.35. We found
₂₆₇ that images with light edges are not easily recognizable. The planks (image 3) is misclassified, with an
₂₆₈ accuracy of 0.01. We can see that the each pile has a noisy surface (stochastic), and only the gaps between
₂₆₉ piles has a regular pattern. This shows the CNNs donot not use long-range regular structures. Similarly
₂₇₀ for other misclassified images. They both show high levels of noise within the larger regular patterns.

₂₇₁ Figure 9 shows the results for the irregular textures. The wrongly classified examples are shown
₂₇₂ in the bottom row. For images 22, 23, 24, 25, 26 and 27, potential causes are the examples possess
₂₇₃ a considerable amount of sharp edges, and no stochastic random texture contained in those irregular
₂₇₄ textures. This misled the CNN to only consider local structures, which made it wrongly classify them
₂₇₅ as regular textures. Possible solutions include adding similar samples in the training irregular images.
₂₇₆ Texture synthesis is a good way to generate irregular images by breaking regularity of existed regular
₂₇₇ textures. This provide a future direction to improve our irregular samples.

₂₇₈ In summary, standard CNNs such as Inception-ResNet-v2 achieved a high accuracy of 97.78% for
₂₇₉ classifying regular textures and irregular textures. The CNN correctly recognized most regular patterns,
₂₈₀ but had lower accuracy on borderline cases (near-regular textures) and hybrid textures (e.g. tiles of noisy
₂₈₁ textures).

₂₈₂ *In the second experiment*, we investigated the tuning of learning rate decay for the three standard
₂₈₃ CNN models using fold 1 data. We set the batch size and epoch as 32 and 200, and evaluated a set of the
₂₈₄ learning rate decays for the three CNN models. With the chosen model configuration, the results suggested

285 a moderate learning rate decay of 0.01, 0.1 and 10 resulted in good performance for Inception-ResNet-v2,
286 InceptionV3 and ResNet-50, respectively. Figure 10 shows oscillations in behavior for the large learning
287 rate of 0.1 for Inception-ResNet-v2, and the inability of the model to learn with the small learning rate of
288 0.01 for InceptionV3 and 0.1 for ResNet-50.
289   In the above experiments, we used data argumentation provided by Keras including horizontal and
290 vertical flip, rotation range of 10, width and height shift range of 0.05. Figure 11 compares the training
291 process between using the data argumentation and not. It can be observed that when the data argumentation
292 was used (left), the model took much less time to achieve convergence and had higher training accuracy.
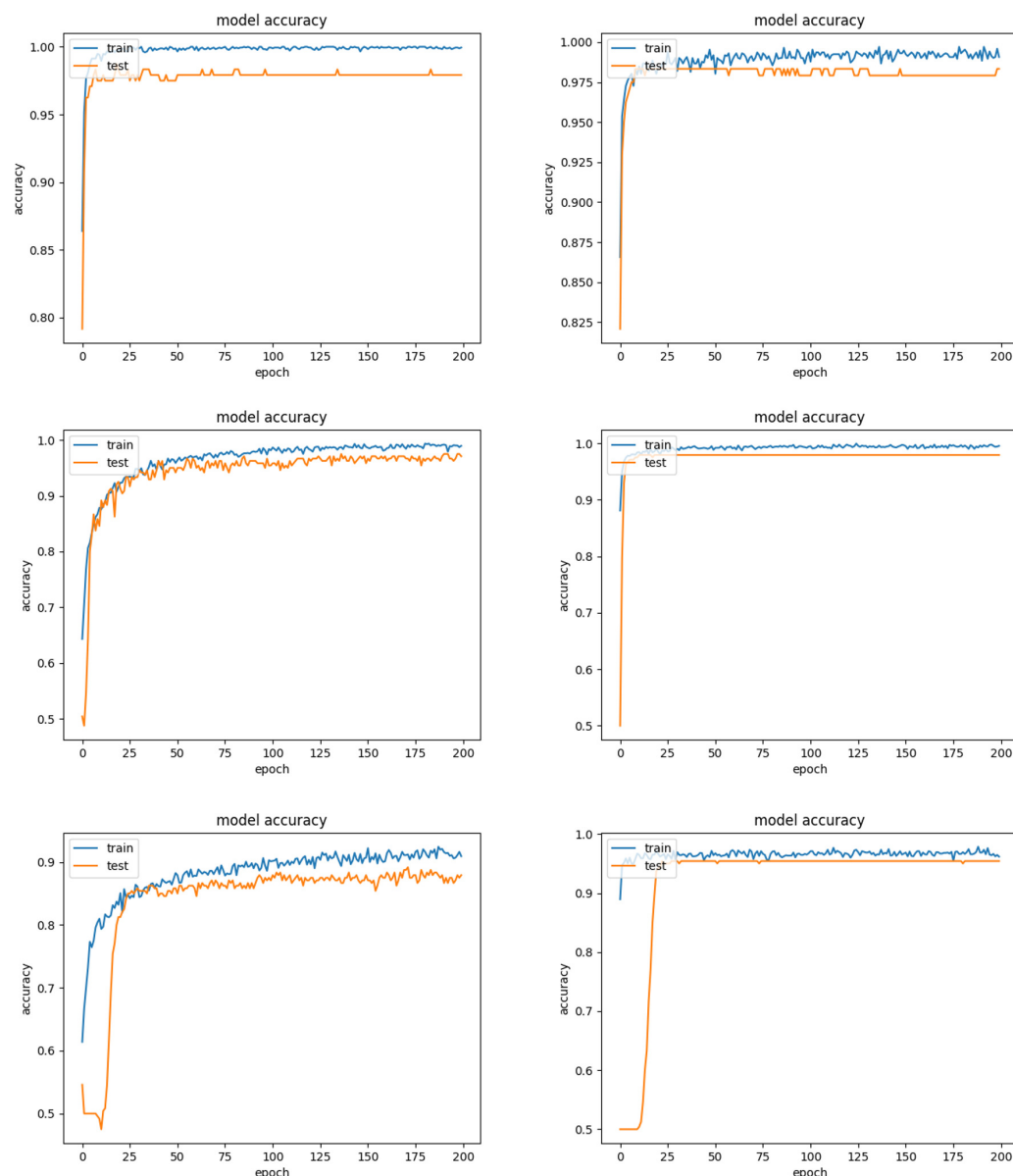
Figure 10. Train and validation accuracy for different learning rate decays ε. Top: Inception-ResNet-v2 ($\varepsilon = 0.01$ vs. $\varepsilon = 0.1$). Middle: InceptionV3 ($\varepsilon = 0.01$ vs. $\varepsilon = 0.1$). Bottom: ResNet-50 ($\varepsilon = 0.1$ vs. $\varepsilon = 10$).

293   *In the third experiment*, we evaluated the performance of the further generated Fisher representations of
294 the above trained CNNs for classifying regular and irregular textures. Specifically, we transformed the filter
295 features of the last convolutional layer of the trained Inception-ResNet-v2 into Fisher vector representation.
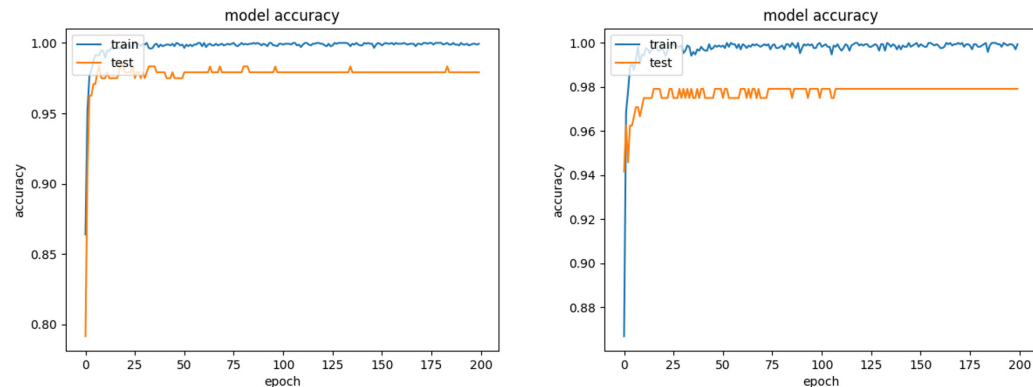
Figure 11. Training accuracies of Inception-ResNet-v2 between using data argumentation (left) and not (right).

A standard SVM solver was used as classifier. We denoted this method as FV-CNNreg-IncepResv2, as introduced in the Methods section. The last layer of convolutional filters of Inception-ResNet-v2 outputs features of dimension $8 \times 8 \times 1536$. Dimensionality reduction (PCA) was further applied to the CNN local descriptors before FV pooling, reducing the output features dimensions to 512 and providing the best performance in our test. Gaussian Mixture Model (GMM) was set to $K = 64$ modes. The traditional SIFT features were also compared. Three kernels were exploited for the SVM classifier, including linear kernel, Hellinger's kernel (short for Hell) and χ^2 kernel (short for Chi2) (Maji et al., 2008; Schölkopf et al., 2002). The SVM solvers was implemented using the VLFeat library. The hyperparameters were set as follows. Let n denote the total number of training samples (both positive and negative), regularization coefficient was $\frac{1}{n}$, stopping criterion epsilon 0.001 and maximum iterations $n \times 200$. Results are shown in Table 5.

Table 5. Comparison of accuracy (%) for Fisher vector representations for 10-fold cross-validation experiments. FV-CNNreg-IncepResv2 applies the Fisher pooling to the last convolutional layer of the trained Inception-ResNet-v2. FV-SIFT denotes the Fisher pooling of SIFT features. Note that linear means linear kernel, Hell denotes the Hellinger's kernel, and Chi2 represents χ^2 kernel.

Model/folds	FV-SIFT			FV-CNNreg-IncepResv2		
Kernel	Linear	Hell	Chi2	Linear	Hell	Chi2
1	89.60	89.40	89.40	98.00	98.00	98.00
2	90.4	90.00	89.20	97.20	96.40	96.80
3	90.00	87.60	90.40	97.80	98.40	98.40
4	91.20	90.20	90.00	97.80	98.00	98.20
5	92.40	91.60	91.60	96.20	96.60	96.60
6	91.00	90.00	90.00	98.20	97.80	97.80
7	90.00	89.20	92.20	97.40	97.40	97.80
8	91.20	89.40	91.00	97.40	97.40	97.60
9	91.20	92.30	91.00	98.00	98.60	97.80
10	90.60	88.60	90.20	98.20	98.40	98.40
Mean	90.76	89.96	90.50	97.62	97.70	97.71
St.dev.	0.82	1.42	0.95	0.61	0.75	0.61

Fisher vector representation, FV-CNNregIncepresV2, gave a good result of 97.71%. This proves the usefulness of transforming deep CNNs feature into Fisher vector representations. In comparison, Fisher vector representation of traditional SIFT features has a worse accuracy (90.76%) than all the CNN based methods.

DISCUSSIONS

Regularity is an important high-level characteristic of textures toward perception understanding. The main challenge of modelling the regularity of textures is that it is not obvious what kind of features should be extracted. We utilize the deep convolutional neural networks to automatically learn features of regular textures. Textures form existed databases were classified into regular and irregular; each category consisted of a variety of textures. To a large extent, We evaluated the ability of deep CNNs for modelling the texture regularity. The results obtained can provide useful guidance for concrete analysis of regular textures. The classic CNN model Inception-ResNet-v2 used in a standard way and its more transferable Fisher vector representation both show a high classification result of 98%, which is much higher than 90% obtained by the traditional SIFT features. The trained CNN can correctly recognize most regular patterns, but has lower accuracy on borderline cases, e.g. near-regular textures and hybrid textures. Beyond the impressive result, there are still several limitations. First, 2460 images may be still insufficient to evaluate the methods. More images are needed. In addition, this experiment shows the architectures of current CNNs do not use long-range structures and thus decreases their ability to model complex features such as texture regularity. In the future study we will investigate the way to learn the long range features by CNN models.

CONCLUSIONS

In this paper we have introduced a new regular texture database. It supplements state-of-the-art texture analysis databases towards perception understanding. Based on the created regular texture databases, we proposed a generalized regular texture modelling and recognition framework. The trained deep CNN models and further generated Fisher representations were robust to different texture layouts, pattern complexity, texton variability and viewing angles, and proved transferable from one domain to another. Our experiments showed that both methods reached remarkable accuracy, with a best performance of 98% for general regular texture classification.

To further understand and improve the performance of our proposed network, an ablation study has been presented. Although convolutional neural networks take spatial relations into account by pooling local features into a global representation using fully-connected layers, they are known to perform sub-optimally when learning long-range patterns. In our future work, we will focus on applying this generalized regular texture modelling to efficient repetitive elements detection in real-life applications.

REFERENCES

Adu-Gyamfi, Y. O., Asare, S. K., Sharma, A., and Titus, T. (2017). Automated vehicle recognition with deep convolutional neural networks. *Transportation Research Record*, 2645(2645):113–122.

Aksoy, S., Yalniz, I. Z., and Tasdemir, K. (2012). Automatic detection and segmentation of orchards using very high resolution imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 50(8):3117–3131.

Bayless, J. (2019). 20 free subtle textures for backgrounds. `https://designwoop.com/free-subtle-textures`.

Brownlee, J. (2020). Understand the impact of learning rate on neural network performance. `https://machinelearningmastery.com/`.

Cai, Y. and Baciu, G. (2011). Detection of repetitive patterns in near regular texture images. In *2011 IEEE 10th IVMSP Workshop: Perception and Visual Signal Analysis*, pages 60–65. IEEE.

Chollet, F. (2015). Keras. `https://keras.io`.

Chugai, D. (2019). Texturelib texture. `http://texturelib.com/`.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., , and Vedaldi, A. (2014). Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cimpoi, M., Maji, S., Kokkinos, I., and Vedaldi, A. (2015). Deep filter banks for texture recognition, description, and segmentation. *International Journal of Computer Vision*.

Cui, H., Yuan, G., Liu, N., Xu, M., and Song, H. (2020). Convolutional neural network for recognizing highway traffic congestion. *Journal of Intelligent Transportation Systems*, 24(10):1–11.

Dana, K. J., Van Ginneken, B., Nayar, S. K., and Koenderink, J. J. (1999). Reflectance and texture of real-world surfaces. *ACM Transactions On Graphics (TOG)*, 18(1):1–34.

Francv, M. (2010). Texturelibrary texture. `https://textures.forrest.cz/`.

362 Hays, J., Leordeanu, M., Efros, A. A., and Liu, Y. (2006). Discovering texture regularity as a higher-order
363   correspondence problem. In *Proceedings of the European Conference on Computer Vision, Volume*
364   *Part II*, pages 522–535.

365 He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In
366   *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages
367   770–778.

368 Hermano, S. D. (2017). Machine learning regularity representation from biological patterns: a case study
369   in a drosophila neurodegenerative model.

370 Hettiarachchi, R., Peters, J., and Bruce, N. (2014). Fence-like quasi-periodic texture detection in images.
371   *Theory and Applications of Mathematics & Computer Science*, 4(2):123–139.

372 Hossain, S. and Serikawa, S. (2013a). Deep learning for texture and dynamic texture analysis. *Pattern*
373   *Recognition Letters*, 34(15):2007–2022.

374 Hossain, S. and Serikawa, S. (2013b). Texture databases–a comprehensive survey. *Pattern Recognition*
375   *Letters*, 34(15):2007–2022.

376 Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional
377   neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

378 Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (2001). Gradient-based learning applied to document
379   recognition. *Intelligent Signal Processing*, pages 306–351.

380 Leung, T. K. and Malik, J. (1996). Detecting, localizing and grouping repeated scene elements from an
381   image. In *Proceedings of the European Conference on Computer Vision-Volume I*, pages 546–555.

382 Lin, W.-C., Hays, J., Wu, C., Liu, Y., and Kwatra, V. (2006). Quantitative evaluation of near regular
383   texture synthesis algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern*
384   *Recognition (CVPR)*, volume 1, pages 427–434.

385 Litjens, G. J. S., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Laak, J. A. W. M.
386   V. D., Ginneken, B. V., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis.
387   *Medical Image Analysis*, 42:60–88.

388 Liu, S., Ng, T.-T., Sunkavalli, K., Do, M. N., Shechtman, E., and Carr, N. (2015). Patchmatch-based auto-
389   matic lattice detection for near-regular textures. In *Proceedings of the IEEE International Conference*
390   *on Computer Vision*, pages 181–189.

391 Liu, Y., Hel-Or, H., Kaplan, C. S., and Gool, L. J. V. (2008). Computational symmetry in computer vision
392   and computer graphics. *Foundations and Trends in Computer Graphics and Vision*, 5:1–195.

393 Liu, Y., Lin, W.-C., and Hays, J. (2004). Near-regular texture analysis and manipulation. *ACM Transac-*
394   *tions on Graphics (TOG)*, 23(3):368–376.

395 Llamas, J., M Lerones, P., Medina, R., Zalama, E., and Gómez-García-Bermejo, J. (2017). Classification
396   of architectural heritage images using deep learning techniques. *Applied Sciences*, 7(10):992.

397 Maji, S., Berg, A. C., and Malik, J. (2008). Classification using intersection kernel support vector
398   machines is efficient. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8.
399   IEEE.

400 MidCenturyStyles (2015). Textures from flickr. `https://flickr.com/photos/`
401   `midcenturystyles/`.

402 Mnih, V. and Hinton, G. E. (2012). Learning to label aerial images from noisy data. In *Proceedings of the*
403   *29th International conference on machine learning (ICML-12)*, pages 567–574.

404 Murean, H. and Oltean, M. (2018). Fruit recognition from images using deep learning. *Acta Universitatis*
405   *Sapientiae: Informatica*, 10(1):26–42.

406 Park, M., Brocklehurst, K., Collins, R., and Liu, Y. (2009). Deformed lattice detection in real-world
407   images using mean-shift belief propagation. *IEEE Transactions on Pattern Analysis and Machine*
408   *Intelligence*, 31(10):1804–1816.

409 Perronnin, F. and Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In
410   *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

411 Petrovska, B., Zdravevski, E., Lameski, P., Corizzo, R., Štajduhar, I., and Lerga, J. (2020). Deep
412   learning for feature extraction in remote sensing: A case-study of aerial scene classification. *Sensors*,
413   20(14):3906.

414 PSU-NRTDB (2005). `http://vivid.cse.psu.edu/`.

415 Randen, T. (2012). Brodatz texture. `http://www.ux.uis.no/~tranden/brodatz.html`.

416 Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in

the brain. *Psychological Review*, 65(6):386–408.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). *Learning representations by back-propagating errors*, volume 323.

Saedi, S. I. and Khosravi, H. (2020). A deep neural network approach towards real-time on-branch fruit recognition for precision horticulture. *Expert Systems with Applications*, 159:113594.

Sarafraz, A. (2011). Kylberg texture. `https://computervisiononline.com/dataset/1105138658`.

Schaffalitzky, F. and Zisserman, A. (1998). Geometric grouping of repeated elements within images. In *British Machine Vision Conference 1998*, pages 1–10.

Schölkopf, B., Smola, A. J., Bach, F., et al. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singh, A., Sengupta, S., and Lakshminarayanan, V. (2020). Explainable deep learning models in medical image analysis. *Journal of Imaging*, 6(6):52.

SketchupTextureClub (2010). Sketchup texture. `https://www.sketchuptextureclub.com/textures/architecture/paving-outdoor/concrete/blocks-damaged`.

Sun, H.-C., Baker, C. L., St-Amand, D., et al. (2021). Visual perception of texture regularity: conjoint measurements and a wavelet response-distribution model. *bioRxiv*.

Sun, H.-C., Kingdom, F. A., and Baker, C. L. (2019). Perceived regularity of a texture is influenced by the regularity of a surrounding texture. *Scientific reports*, 9(1):1–11.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2016a). Inception-v4, inception-resnet and the impact of residual connections on learning.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI Conference on Artificial Intelligence*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.

Szegedy, C., Toshev, A., and Erhan, D. (2013). Deep neural networks for object detection. In *Advances in Neural Information Processing Systems 26*, volume 26, pages 2553–2561.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016b). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Tan, T., Wang, L., Huang, Y., and Wu, Z. (2014). Gait recognition method based on deep learning.

Wang, J., Zhu, H., Wang, S.-H., and Zhang, Y.-D. (2021). A review of deep learning on medical image analysis. *Mobile Networks and Applications*, 26(1):351–380.

Widrow, B. and Lehr, M. (1990). 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442.

Yu, W., Ai, T., Liu, P., and Cheng, X. (2017). The analysis and measurement of building patterns using texton co-occurrence matrices. *International Journal of Geographical Information Science*, 31(6):1079–1100.

Zou, Q., Wang, Y., Wang, Q., Zhao, Y., and Li, Q. (2020). Deep learning-based gait recognition using smartphones in the wild. *IEEE Transactions on Information Forensics and Security*, 15:3197–3212.