International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 4 (2017) pp. 532-539

Art Painting Identification using Convolutional Neural Network

Yiyu Hong[1] and Jongweon Kim[2]

[1]Research Scholar, Department of Copyright Protection, Sangmyung University, Seoul, South Korea.

[2] Professor, Department of Contents and Copyright, Sangmyung University, Seoul, South Korea.

[1]ORCID: 0000-0001-6138-7083, [2]ORCID: 0000-0002-8916-6431

Abstract

Convolutional Neural Network (CNN) applications have been suggested for many multimedia processing tasks and achieved great success. In this paper, we present a methodology about how to apply CNN for art painting identification. Each art painting image is distorted by various operations, such as lens distortion, scaling, rotation, etc., to simulate potential situation that it would be appeared on TV programs or photographs or movies. The distortions are carried out randomly by computer to make large datasets for CNN to training. And then, we studied about how the architecture and parameters of CNN effects the identification accuracy. In the end, we compare our method with the Scale-Invariant Feature Transform (SIFT), which is a state-of-the-art hand-crafted feature, and the result shows the proposed method outperform SIFT by 13.6%.

Keywords: Convolutional Neural Network; Art Painting Identification; Image Distortion.

INTRODUCTION

With the development of digital technique and network communication, art painting images can be easily found in daily life. There would be some law issues arising when copyrighted art paintings are showed on TV programs or movies without licenses from the copyright holders. And in many cases, this kind of situation happened by careless of the content producers. Thus, after the contents are made, if there is a computer program that automatically detect those copyrighted art paintings from the contents would significantly benefit the content producers. The detection process often divided into two parts: (1) the video frame or the photograph, which may contain art paintings, are segmented [1] or using sliding window scheme [2-4] to extract the region of interest, (2) then using image identification methods to the region to identify whether it is a copyrighted art paintings. In this paper, our research effort has focused on the second part. We trained a CNN on the art painting datasets, which are simulated possible situations for the art painting images would appear on the contents like movies. If input an image to the CNN, it will output a decision of the identity of the image (Fig. 1).

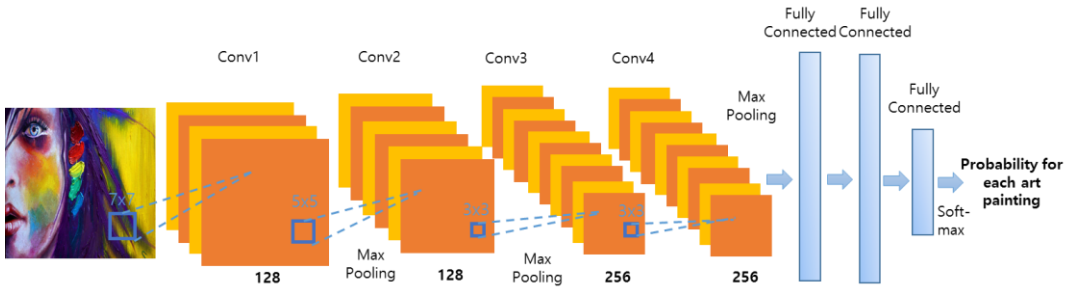

Figure 1: Our best performed CNN architecture for art painting identification

Deep learning is a branch of machine learning, and CNN is a type of deep learning algorithm. Recently, CNN has become a hot research topic in many scientific fields, such as image recognition [5], face recognition [6] and speech recognition [7], etc., because it can achieve state-of-the-art performance in these fields. In more than ten years ago, deep learning algorithms had not brought great improvement, as it cost too many resources like hardware and time. With the development of computing power, especially accessibility to multiple core GPUs to implement efficient parallelization, have shorten the deep learning networks' training time of more than 50 times compared to CPU-only implementation [8]. In 2012, Krizhevsky et al. [5] trained a CNN on large datasets [9] with highly-optimized GPU implementation showed significant improvement on image classification accuracy, which has arisen great interest of CNN in science fields.

RELATED WORKS

1. Convolutional Neural Network

Our paper was inspired by the strong performance of CNN in image classification tasks. So, here we give a brief introduction about architectures of CNN and their functions. A CNN mainly consists of a number of convolutional layer, pooling layer and fully-connected layer. A full CNN architecture can be formed by stacking these layers in an order.

Input layer: The raw pixel values of the image will be hold in input layer. And the value of the data is always normalized to zero-centered and range from 1 to -1 by subtracting mean value of the whole datasets or applying PCA and whitening process.

Convolutional layer: Convolutional layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input. Neurons that belong to the same layer share the same weights. Weight sharing increases learning efficiency by greatly reducing the number of free parameters being learnt. The constraints on the model enable CNN to achieve better generalization on vision problems.

Activation function: Neural networks have to implement complex mapping functions hence they need activation functions that are non-linear in order to bring in the much-needed non-linearity property that enables them to approximate any function. Activation function is applied after convolutional layer and full-connected layer. The most commonly used type of activation functions (Fig.2) are sigmoid, tanh, and relu function [5].

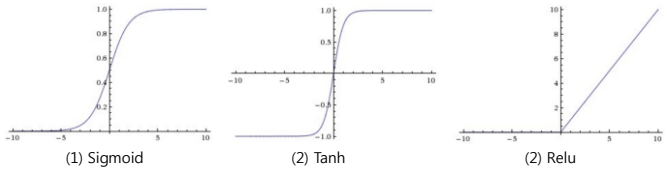

Figure 2: Activation functions

Pooling layer: The function of pooling layer is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network. Max pooling and average pooling are typical types. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2.

Fully-connected layer: The output from the convolutional and pooling layers represent high-level features of the input images. The purpose of the fully-connected layer is to use these features for classifying the input image into various classes based on the training dataset.

Softmax layer: The softmax layer takes a vector of arbitrary real-valued scores from the output layer of the fully connected layer and squashes it to a vector of output probabilities between zero and that sum to one.

The training process contains two steps: forward propagation and backward propagation. The forward propagation means that training images goes through a series of convolution, activation function, pooling operations and fully-connected layer to output probabilities for each class. After calculating the total error between target probability and output probability, backpropagation is used to calculate the gradients of the error with respect to all weights in the CNN and use gradient descent to update the trainable parameters to minimize the output error.

2. Scale-Invariant Feature Transform

David Lowe developed SIFT [10], which is an image descriptor for image-based matching and recognition. This descriptor is used for a large number of purposes in computer vision related to point matching between different views of a 3-D scene and view-based object recognition. The SIFT descriptor is designed to invariant to translations, rotations and scaling transformations in the image domain and robust to moderate perspective transformations and illumination variations. Experimentally, the SIFT descriptor has been proven to be very useful in practice for image matching and object recognition under real-world conditions, and also performs best among human-crafted image descriptors [11].

The SIFT descriptor was computed from the image intensities around interesting locations in the image domain which can be referred to as interest points. These interest points are obtained from scale-space extrema of differences-of-Gaussians within a difference-of-Gaussians pyramid. At each interest point, an image descriptor is computed. The SIFT descriptor can be seen as a position-dependent histogram of local gradient directions around the interest point. To obtain scale invariance of the descriptor, the size of this local neighborhood needs to be normalized in a scale-invariant manner. To obtain rotational invariance of the descriptor, a dominant orientation in this neighborhood is determined from the orientations of the gradient vectors in this neighborhood and is used for orienting the grid over which the position-dependent histogram is computed with respect to this dominant orientation to achieve rotational invariance.

Dataset

In order to train a CNN to identify art paintings, it need at least thousands of related images. So, we first downloaded 100 art painting images from google by searching keyword “art painting”, some of them are showed in Fig. 3.

Figure 3. Some of downloaded art paintings

And then, these downloaded 100 images are randomly distorted by various operations, such as projection, rotation, scaling, etc., to simulate situation when the regions of the interest of the art paintings would be captured from videos or photographs. Each image has 300 randomly distorted versions, thus, for 100 art painting images we generated 30000 distorted images. We use 25000 of the total distorted images for training and reserve 5000 for testing purposes. All images are resized to width=256 pixels and height=256 pixels. In the following, we will describe about what kind of image distortions and corresponding distortion strength that we used for generate distorted images.

Lens distortion: The images of real cameras suffer from more or less lens distortion. The most prevalent form of this effect is the barrel and pincushion distortion [12]. In barrel distortion, image magnification decreases with distance from the optical axis. The apparent effect is that of an image which has been mapped around a sphere (or barrel). In pincushion distortion, image magnification increases with the distance from the optical axis. The visible effect is that lines that do not go through the center of the image are bowed inwards, towards the center of the image, like a pincushion. We used the following lens distortion model:

$$R_d = R_o \times (1 + l \times R_o^2) \quad (1)$$

Where R_o and R_d are the polar distance (range 0 to 1) from the center of distortion in the undistorted and distorted images respectively, and l is the distortion parameter, barrel distortion will have a positive term for l whereas pincushion distortion will have a negative value. The Fig.4 (1) shows when the art painting image distorted by lens distortion with $l = 1.5$ (left picture) and -0.5 (right picture).

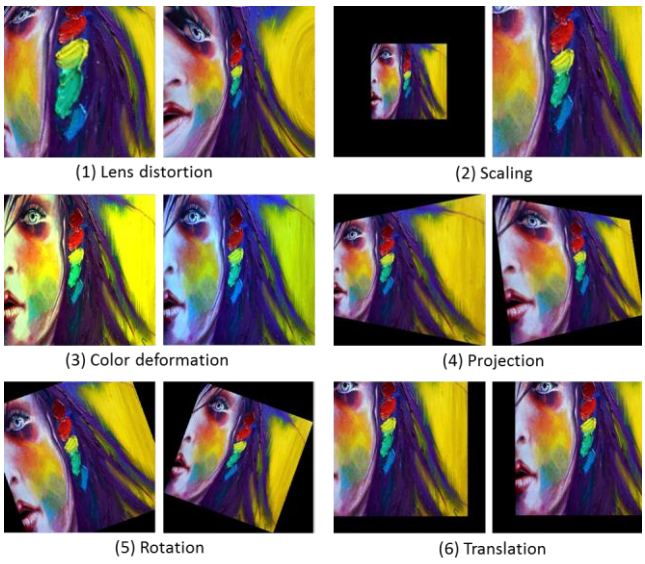

Figure 4: Image Distortions

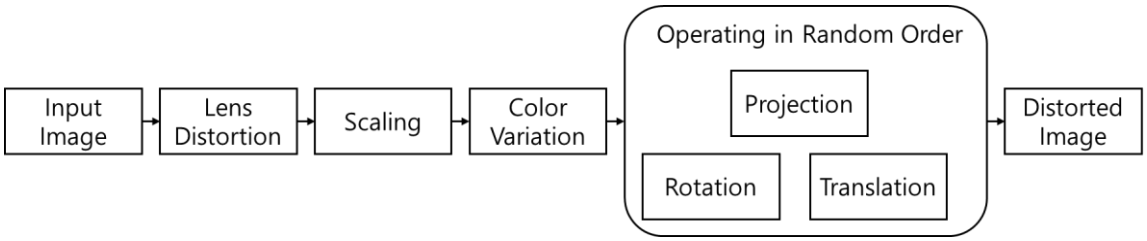

Figure 5: Overall process of image distortion

Scaling: Scaling distorted image is s times the size of the original image. Due to the size of the input image for CNN need to be fixed, when s is smaller than 1, the border of scaling distorted image is padded with zero to have same size of the original image, and when s is larger than 1, around the border of the scaling distorted image is cropped and the center part is remained. The Fig.4 (2) shows when the art painting is scaled by $s = 0.5$ (left picture) and 1.5 (right picture).

Color variation: Color variation is done by multiply $c = [c_r, c_g, c_b]$ to each RGB channel of the image. The Fig.4 (3) shows when the art painting is multiplied by $c = [1.2,\ 1.4,\ 0.8]$ (left picture) and $[0.8,\ 1.1,\ 1.3]$ (right picture).

Projection: Image projection can be done by mapping a square to a quadrilateral with a projective transformation. We first set up a coordinate system so that the original image fills the unit square with vertices [(0, 0), (1, 0), (1, 1), (0, 1)], which are respect to left-up, right-up, right-bottom, left-bottom corner of the image. And then transform to a quadrilateral with vertices $[(p_1, p_2), (1-p_3, p_4), (1-p_5, 1-p_6), (p_7, 1-p_8)]$. The Fig.4 (4) shows when the art painting is projected to vertices [(0, 0.2), (1, 0), (1, 1), (0, 0.8)] (left picture) and [(0.16, 0.13), (0.85, 0.28), (0.76, 0.8), (0.28, 0.87)] (right picture).

Rotation: We rotate original image by r degrees in a counterclockwise direction around its center point. Negative r will rotate the image in a clockwise direction. The rotated image will be larger than the original size, thus, here we propose two operations to make the rotated image fit the original size. One way is cropping the border of the rotated image and remain the center part, just like the operation carried out after scaling up an image. The other way is scaling down the rotated image to fit the original size. These two operations are carried out randomly when the distorted images are generated. The Fig.4 (5) shows the art painting rotated by 20 degrees with "crop" operation (left picture) and rotated -20 degrees with "resize" operation (right picture).

Translation: A translation operation shifts an image by a specified number of pixels in either x or y direction, or both. We here define t_x and t_y to mean image translated by pixels in x-direction and y-direction respectively. The Fig.4 (6) shows the art painting translated when t_x = 20, t_y = 20 (left picture) and $t_x = -40$, t_y = 30 (right picture).

The distortion strength for each distortion is set as follows: $l = -0.5$ or $1.5, s \in [0.5, 0.7] \cup [1.3, 1.5],\ c_{r,g,b} \in [0.8, 1.4], p_i \in [0.1, 0.3]\ where\ i = 1, 2, \ldots 8, r \in [-20, 20], t_{x,y} \in [-40, -20] \cup [20, 40]$. All distortion strength are generated randomly within the limited ranges.

The order of the image distortion operations that we used for making distorted images are illustrated in Fig.5. The input image first goes through lens distortion, scaling and color variation, then projection, rotation, and translation are operated in random order to eventually output distorted image. The Fig. 6 shows some distorted images by our method.

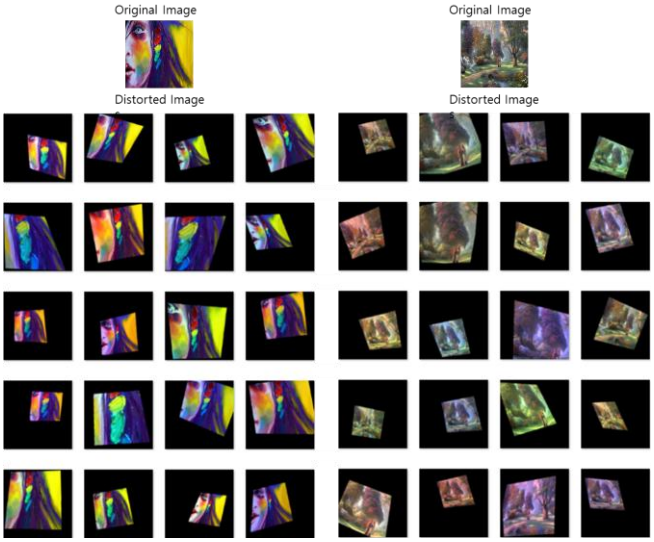

Figure 6. Distorted images

CNN Architecture and Experiments

During training, all input images are transformed into 256×256 grayscale images, and then the pixel value of the images are scaled to range from -1 to 1. In the following experiments in this section, if there is no notice for parameters: the weights in each convolutional layer is initialized from a zero-mean Gaussian distribution with standard deviation 0.01 and the biases are initialized with the constant 0.01; The ReLU non-linearity is applied to the output of every convolutional and fully-connected layer; Spatial pooling is carried out by max-pooling layers; Adam Optimizer [13] is used to minimize the

loss with the authors proposed default parameters of 0.9 for β_1, 0.999 for β_2 and 10^{-8} for ϵ. Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments, they showed empirically that Adam works well in practice and compares favorably to other adaptive learning-method algorithms; The learning rate is set to 0.0001 and batch size is set to 128; Zero-padding is carried out to preserve the input and output width and height are the same; The output of the last fully-connected layer is fed to a 100-way softmax, each way for an art paintings. Our implementation is used the TensorFlow (http://www.tensorflow.org), which is an open source software library for machine intelligence.

Table 1: Three architectures of CNN tested on our datasets

A			B			C		
Type	**Filter Size/ Stride**	**Output Size**	**Type**	**Filter Size/ Stride**	**Output Size**	**Type**	**Filter Size/ Stride**	**Output Size**
Conv	$11 \times 11/4$	$64 \times 64 \times 96$	Conv	$7 \times 7/3$	$86 \times 86 \times 64$	Conv	$7 \times 7/3$	$86 \times 86 \times 128$
Pool	$3 \times 3/2$	$32 \times 32 \times 96$	Conv	$7 \times 7/2$	$43 \times 43 \times 64$	Pool	$2 \times 2/2$	$43 \times 43 \times 128$
Conv	$5 \times 5/1$	$32 \times 32 \times 256$	Pool	$2 \times 2/2$	$22 \times 22 \times 128$	Conv	$5 \times 5/2$	$22 \times 22 \times 128$
Pool	$3 \times 3/2$	$16 \times 16 \times 256$	Conv	$5 \times 5/1$	$22 \times 22 \times 128$	Pool	$2 \times 2/2$	$11 \times 11 \times 128$
Conv	$3 \times 3/1$	$16 \times 16 \times 384$	Conv	$5 \times 5/1$	$22 \times 22 \times 192$	Conv	$3 \times 3/1$	$11 \times 11 \times 256$
Conv	$3 \times 3/1$	$16 \times 16 \times 384$	Pool	$2 \times 2/2$	$11 \times 11 \times 192$	Conv	$3 \times 3/1$	$11 \times 11 \times 256$
Conv	$3 \times 3/1$	$16 \times 16 \times 256$	Conv	$3 \times 3/1$	$11 \times 11 \times 256$	Pool	$2 \times 2/2$	$6 \times 6 \times 256$
Pool	$3 \times 3/2$	$8 \times 8 \times 256$	Conv	$3 \times 3/1$	$11 \times 11 \times 256$	FC		$1 \times 1 \times 2048$
FC		$1 \times 1 \times 4096$	Pool	$2 \times 2/2$	$6 \times 6 \times 256$	FC		$1 \times 1 \times 1024$
FC		$1 \times 1 \times 4096$	FC		$1 \times 1 \times 2048$	FC		$1 \times 1 \times 100$
FC		$1 \times 1 \times 100$	FC		$1 \times 1 \times 1024$			
			FC		$1 \times 1 \times 100$			

We evaluated three CNN architectures in this paper, which are outlined in Table.1 and the corresponding test error rates until 100 epochs training are showed in the Fig. 7. The A architecture is derived from AlexNet [5]. We first tested using the architecture A on the datasets, but the test error rates (Fig. 7) did not decrease after 40 epochs. Then we construct B architecture which is inspired by VGG Net [14], a stack in order of two convolutional layers followed by a pooling layer repeatedly. And the architecture C is a smaller version of B. Compared to A, the B and C used smaller filter size (11→7) in the beginning and less neurons in fully-connected layers. From the results in Fig. 7, we can see that test error rate of C architecture is decreased to around 40% at the 100 epochs, which works best among the three architectures.

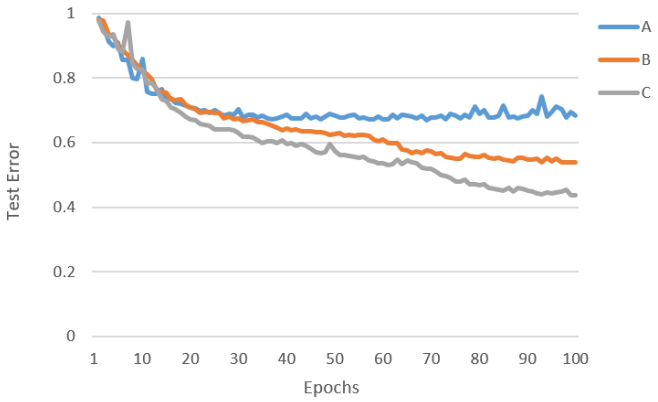

Figure 7: Plot of test error rates for the three CNN architectures

The table 2 shows how the initialization of the weight and bias influence the training on the architecture C with our datasets. We tested with three kinds of weight and bias combinations. The first combination is that (1) the weights are initialized from a zero-mean Gaussian distribution with standard deviation 0.1 (weight → Stdev 0.1) and the bias are initialized with the constant 0.01 (bias → Const 0.01), (2) weight → Stdev 0.01 and bias → Const 0.01, (3) weight → Stdev 0.001 and bias → Const 0.001. The entries of the table 2 contain the test error rates along with the training epochs. The table shows the CNN can be trained only when weight → Stdev 0.01 and bias → Const 0.01.

The table 3 shows how the initialization of the learning rate for Adam Optimizer influence the training on the architecture C with our datasets. We tested when the learning rates are $10^{-2}, 10^{-4}, 10^{-6}$, and the results shows the CNN can be trained only when the learning rate is set as 10^{-4}.

Table 2: Test errors for the different initialization of the weights and biases

Parameters / Epochs	Weight→Stdev 0.1 Bias → Const 0.1	Weight→Stdev 0.01 Bias → Const 0.01	Weight→Stdev 0.001 Bias → Const 0.001
1	0.989	0.977	0.989
10	0.990	0.783	0.989
20	0.989	0.668	0.989
30	0.989	0.618	0.989
40	0.990	0.599	0.990
50	0.989	0.563	0.990
60	0.989	0.532	0.990
70	0.989	0.510	0.990
80	0.989	0.470	0.989
90	0.989	0.447	0.989
100	0.989	0.437	0.990

Table 3: Test errors for the different learning rates

Parameters / Epochs	Learning rate= 10^{-2}	Learning rate= 10^{-4}	Learning rate= 10^{-6}
1	0.990	0.977	0.990
10	0.989	0.783	0.989
20	0.989	0.668	0.989
30	0.989	0.618	0.989
40	0.990	0.599	0.990
50	0.990	0.563	0.990
60	0.990	0.532	0.990
70	0.990	0.510	0.990
80	0.989	0.470	0.989
90	0.989	0.447	0.989
100	0.990	0.437	0.989

The graph in the Fig. 8 shows test error rates along with training epochs when architecture C trained by parameters that we described in the beginning of this section. The training was stopped at 900 epochs, because the test error rates are nearly unchanged after 850 epochs. The test error rates at 900 epochs is 2%

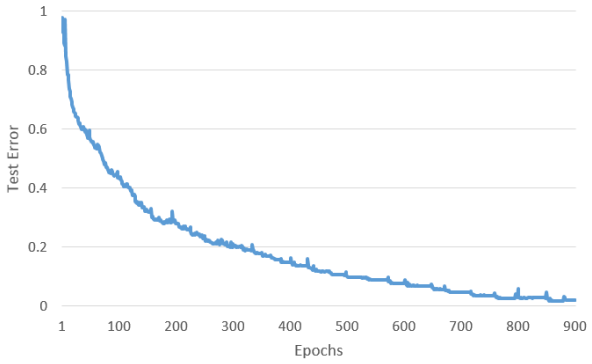

Figure 8: Plot of test error rates for the architecture C until 900 epochs

In the end of this section, we compare our proposed CNN-based method with SIFT which is the state-of-the-art hand-crafted image descriptor. The parameters for extracting and matching SIFT descriptors were used the default values of vlfeat [15]. The Fig. 9 shows two examples of SIFT key points matching. One (Fig. 9 (1)) is between distorted image and its original version which has 51 matched key points, the other (Fig. 9 (2)) is between two different images which has 8 matched key points.

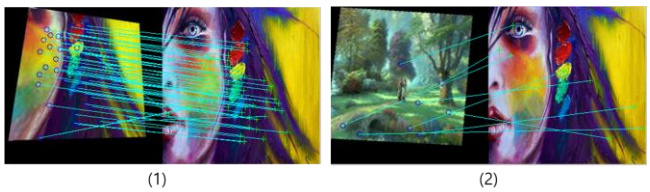

Figure 9: SIFT key points matching (1) Key points matching between distorted image and its original version (2) Key points matching between two different images

We calculate the test error rate for SIFT using following method: Each test image is matched with 100 original images, and the matched key points are counted as the similarity between two images. If the image which has the most number of matched key points is not the original image, which was distorted to generate the test image, we regard this case as a mismatch. The test error rate is calculated by counting the mismatches for all test images and divide the number of mismatch by the number of test images (5000). From the table 4, we can know our CNN-based method is outperform SIFT by 13.6%.

Table 4: Comparing test error rate between proposed CNN-based method and SIFT

	CNN-based	SIFT
Test error rate	2%	15.6%

CONCLUSION

In this paper, we proposed a methodology about how to apply CNN for art painting identification. We first generated distorted art painting images which could be various kinds of possible situation about the art painting could be captured. Then the images are fed to CNN to trained an art painting identifier. Three CNN architectures and related parameters were tested, and the results showed small change in the CNN architectures or parameters would significantly influence the performance. Furthermore, the proposed CNN-based method is outperforming the state-of-the-art hand-crafted image descriptor SIFT by 13.6%. In the future, we plan to detect and identify art painting images in a scene of movies or TV programs.

ACKNOWLEDGMENT

This research is supported by Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2016.

REFERENCES

[1] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers and A. W. M. Smeulders, “Segmentation as selective search for object recognition”, Proceedings of the IEEE 13th International Conference on Computer Vision, Barcelona, Spain, pp. 1879-1886, 2011.

[2] C. H. Lampert, M. B. Blaschko, T. Hofmann, “Beyond sliding windows: Object localization by efficient subwindow search”, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska, USA, pp. 1-8, 2008.

[3] H. A. Rowley, S. Baluja, and T. Kanade, “Neural network based face detection”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, pp. 23-38, 1998.

[4] H. Harzallah, F. Jurie, and C. Schmid. “Combining efficient object localization and image classification”. Proceedings of the IEEE 12th International Conference on Computer Vision, Kyoto, Japan, pp. 237-244, 2009.

[5] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural

networks", In Advances in Neural Information Processing Systems 25, pp. 1106-1114, 2012.

[6] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification", Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1701-1708, 2014.

[7] O. Abdel-Hamid, A. -R. Mohamed, H. Jiang, L. Deng, G. Penn and D. Yu, "Convolutional Neural Networks for Speech Recognition", IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), vol. 22, pp.1533-1545, 2014.

[8] J. Schmidhuber, "Deep learning in neural networks: An overview", Neural Networks, vol. 61, pp. 85-117, 2015.

[9] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. ILSVRC-2012, 2012. URL http://www.image-net.org/challenges/LSVRC/2012/.

[10] G. Vass and T. Perlaki, "Applying and removing lens distortion in post production", Proceedings of the 2 Hungarian Conference on Computer Graphics and Geometry, Budapest, Hungary, pp. 9-16, 2003,

[11] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", Proceedings of the 3rd International Conference on Learning Representations, San Diego, USA, 2015.

[12] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", In International Conference on Learning Representations, 2015.

[13] D. G. Lowe, "Distinctive image features from scale-invariant key-points", International Journal of Computer Vision, vol. 60, no. 2, pp. 91-110, 2004.

[14] K. Mikolajczyk and C. Schmid, "A Performance Evaluation of Local Descriptors", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 10, pp.1615-1630, 2005.

[15] A. Vedaldi, B. Fulkerson, VLFeat: An open and portable library of computer vision algorithms, http://www.vlfeat.org/.S. M. Metev and V. P. Veiko, Laser Assisted Microtechnology, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.