Fine-tuning deep convolutional neural networks for distinguishing illustrations from photographs

Gota Gando[a,*], Taiga Yamada[b,*], Haruhiko Sato[c,*], Satoshi Oyama[d], Masahito Kurihara[d]

[a] *Alt,Inc., Aomi 2-7-4, Koto-ku, Tokyo, Japan*
[b] *Panasonic System Design Co., Ltd., Shinyokohama 3-1-9, Kohoku-ku, Kanagawa, Japan*
[c] *Department of Electronics and Information Engineering, Faculty of Engineering, Hokkai-Gakuen University, Asahimachi 4-1-40, Toyohira-ku, Sapporo, Japan*
[d] *Graduate School of Information Science and Technology, Hokkaido University, Kita 14 Nishi 9 Kita-ku, Sapporo, Japan*

Abstract

Systems for aggregating illustrations require a function for automatically distinguishing illustrations from photographs as they crawl the network to collect images. A previous attempt to implement this functionality by designing basic features that were deemed useful for classification achieved an accuracy of only about 58 %. On the other hand, Deep Learning methods had been successful in computer vision tasks, and convolutional neural networks (CNNs) had performed good at extracting such useful image features automatically. We evaluated alternative methods to implement this classification functionality with focus on Deep Learning methods. As the result of experiments, the method that fine-tuned deep convolutional neural network (DCNN) acquired 96.8% accuracy, outperforming the other models including the custom CNN models that were trained from scratch. We conclude that DCNN with fine-tuning is the best method for implementing a function for automatically distinguishing illustrations from photographs.

Keywords: aggregation systems, machine learning, deep learning, illustrations

1. Introduction

Illustrations are available on a number of websites such as Pixiv and deviantART, but users need to move among such sites to locate desired illustrations. One solution is to create a system that automatically aggregates illustrations from such websites and recommends them to the user, as shown in Figure 1. Such a system would need to be able to distinguish illustrations from other types of images, including photographs. Several groups have investigated the ways to distinguish paintings or computer graphics from photographs, but most of these efforts included the use of 'hand-made' features to perform the classification, meaning that those features had to be selected or designed by human experts rather than by machine expert systems. A recent advance has been the use of deep learning to automate feature extraction for several domains. In particular, methods that use a convolutional neural network (CNN) model have achieved state-of-the-art results in computer vision tasks. Methods using a CNN model have also shown good performance in image style classification as well when pre-trained models are sufficiently fine-tuned.

The purpose of our research is to design a function for distinguishing illustrations from photographs in the target illustration aggregation system. This functionality could also be used when the system crawls the network, collecting illustrations by using the APIs of other image aggregation websites. Among the various genres of illustrations (such as line drawings, paintings, and 3D graphics), here we focus on images of paintings.

2. Related work

The problem of classifying photographs and other types of non-photographic images including paintings and computer graphics images has been explored by several groups. Athitsos et al. (1997) were the first to address the problem of finding images on the web that have a specific type like clip art. They used several metrics, including a color histogram, to distinguish photographs from graphics. Cutzu et al. (2003) proposed using handcrafted features given to a neural network to distinguish paintings from photographs. They achieved accuracy greater than 90% for a large set of images. Distinguishing various kinds of digital images is similar to classifying image styles in that both use high-level features. Karayev et al. (2013) presented a method that uses a convolutional neural network to classify image styles such as painting drawing styles. For the WikiPaintings dataset, which has 20 classes per art style, their classifier achieved 70% to 90% accuracy. Gatys et al.

*This work was done while the first, second, and third authors were at Hokkaido University.

Email addresses: gando@complex.ist.hokudai.ac.jp (Gota Gando), t-ymd@complex.ist.hokudai.ac.jp (Taiga Yamada), haru@complex.ist.hokudai.ac.jp (Haruhiko Sato), oyama@complex.ist.hokudai.ac.jp (Satoshi Oyama), kurihara@complex.ist.hokudai.ac.jp (Masahito Kurihara)

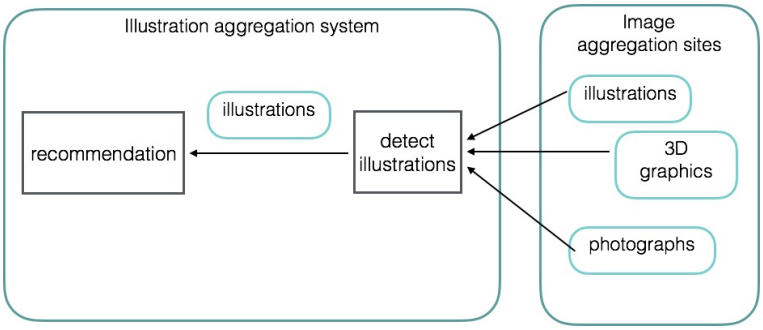


Figure 1: Overview of the target system. In this paper we design the detection process of the system.

(2015) used the VGG-network to classify and reconstruct the drawing styles of paintings.

3. Handcrafted features

We identified two features that could be useful for classification. First, we observed that many illustrations have dark outlines around objects or characters while natural images usually do not. We call this feature "outline detection." We also observed that the colors in illustrations tend not to change in a small region while those in even a very small region of a photograph do tend to change. We call this feature "color intensity." Each feature determines whether the input image is an illustration based on a hyperparameter threshould. We have developed two algorithms for extracting each feature.

3.1. Outline detection

The algorithm for outline detection first detects the edges of the objects in the input image by applying a Sobel filter:

$$\boldsymbol{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \boldsymbol{A}, \boldsymbol{G_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \boldsymbol{A}, \quad (1)$$

where $\boldsymbol{A}$ is the source image, and "*" is a convolution operator. $\boldsymbol{G_x}$ and $\boldsymbol{G_y}$ are images, and the final output of this operation is computed by combining them as follows:

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \quad (2)$$

where G(x, y) is an arbitrary pixel in the output image.

The algorithm then binarizes the output image using Otsu's method (Otsu (1979)) so that darker colors are converted to black and lighter colors are converted to white. The image output by this process is labeled S_1. Next, it takes the original input image again and detects regions that have colors close to black. The CIELAB color space (Connolly and Fliess (1997)) is used to detect the colors that are close to black. Their lightness values are checked to see if they are lower than the threshold. If the distance between a pixel and black is less than threshold θ, the pixel is considered to be black. The image output by this process is labeled S_2. Finally, images S_1, S_2 are compared, and the number of matching pixels is determined. The input image is classified as an illustration if the number is above the hyperparameter threshold. Example of images S_1 and S_2 are shown in Figure 2.

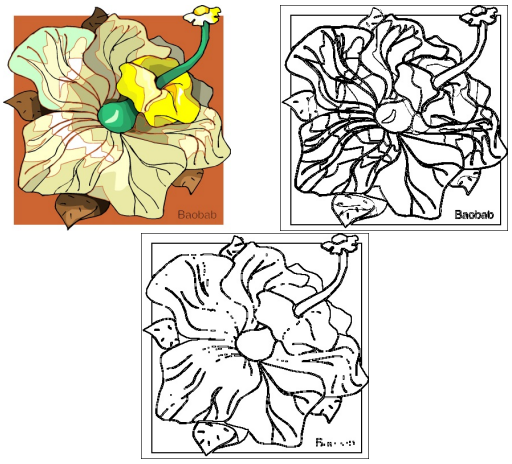


Figure 2: Example of outline detection. Top left image is original input. Top right image (S_1) was obtained by applying a Sobel filter and Otsu's method. Bottom image (S_2) was obtained by detecting the color black.

a_1	a_2	a_3
a_4	a_i	a_5
a_6	a_7	a_8

Figure 3: Example of the window when the size is three.

3.2. Color intensity

The algorithm for color intensity considers a small window in the input image that has an odd size such as 3×3 and compares the distances between the center pixel a_i and the marginal pixels a_j. An example of the window is illustrated in Figure 3. The neighborhood of a pixel i is denoted by $\sigma(i)$, and the marginal pixels are denoted by $a_j \in \sigma(i)$. The sum of the distances is given by

$$d_i = \sum_{j \in \sigma(i)} \min_i |a_i - a_j| \tag{3}$$

The value of d_i is added to another variable D as the window slides across the input image. The value that divides the sum by the number of pixels N in the image is used for the classification.

$$D = \frac{\sum_j^N d_i}{N} \tag{4}$$

We expect D to be small for illustration images, and larger for real world images.

4. Deep Learning

Bengio (2009) called neural networks that have many hidden layers a "deep architecture." Training deep neural networks is difficult due to their tendency to have many local optima. Nair and Hinton (2010) addressed this problem by pre-training the deep model. This method is called "greedy layerwise training." In this method, each layer is considered to be a restricted Boltzmann machine. This method is performed as pre-training. Training with supervised learning and backpropagation is then carried out as for normal neural networks. This second stage of learning is called "fine-tuning."

4.1. Batch normalization (BN)

Several methods have been proposed for training deep networks. Srivastava et al. (2014) proposed using a method called Dropout that disconnects neurons randomly during training to alleviate overfitting. Ioffe and Szegedy (2015) developed a BN algorithm that addresses the internal covariate shift problem by normalizing layer inputs for each mini-batch. Each activation of a mini-batch is transformed so that it has a zero mean and unit variance during stochastic gradient descent (SGD) training, and learned offset and scale factors are applied.

4.2. Convolutional neural networks (CNNs)

Convolutional neural networks (Lecun et al. (1998)) have been proven to be successful at solving computer vision tasks. The basic ideas of CNNs were originally described by Fukushima (1980). He derived his concept from the work of Hubel and Wiesel (1968). CNNs automatically extract useful features from input images by repeatedly applying convolutions and subsampling operations. The "local receptive fields" structure in a convolution layer differs from that in a traditional multi-layer perceptron layer, where each neuron is connected to another neuron in the previous layer. In a convolution layer, each neuron is connected to a part of the neurons in the previous layer. This enables CNNs to recognize patterns that have different coordinates in input images.

4.3. Deep Convolutional Neural Network

CNNs that have a deep architecture are called "deep convolutional neural networks (DCNNs)." Here we call CNNs that have three or more convolution layers "DCNNs." DCNNs have achieved much success in the computer vision field including in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). There are several models for deep network architectures, such as the AlexNet model (Krizhevsky et al. (2012)) shown in Figure 4, GoogLeNet (Szegedy et al. (2015)), and VGG-Net (Simonyan and Zisserman (2014)). We evaluated AlexNet-based architectures.

4.4. Transfer learning

The early layers of a DCNN that is trained with a large dataset can extract generic features, as shown by Zeiler and Fergus (2013) and Donahue et al. (2013), so we use methods that fine-tune a pre-trained model. We used an AlexNet architecture trained on the ImageNet dataset (Deng et al. (2009)) as the pre-trained model. The network architecture used for fine-tuning is shown in Table 1. As our dataset was relatively small (20,000 images) compared to the ImageNet dataset, we hypothesized that fine-tuning the last layer of the AlexNet rather than the earlier layers would improve performance. We fine-tuned the FC7 layer of the AlexNet and initialized the FC8 layer to enable training from scratch for binary classification.

5. Benchmarked Algorithms

5.1. Handcrafted features

We used the two features described in Section 3 to benchmark the outline detection and color intensity algorithms. The both method is applied with a threshould to determine whether the input is an illustration.

5.2. Color histograms

Chapelle et al. (1999) proposed using a color histogram-based support vector machine (SVM) for image classification. This approach uses the hue, saturation, and value (HSV) color space and fixes the number of bins per color component to 16. The dimension of each histogram is $16^3 = 4096$. We also tested red-green-blue (RGB) color space based histograms for comparison purposes in our experiments.

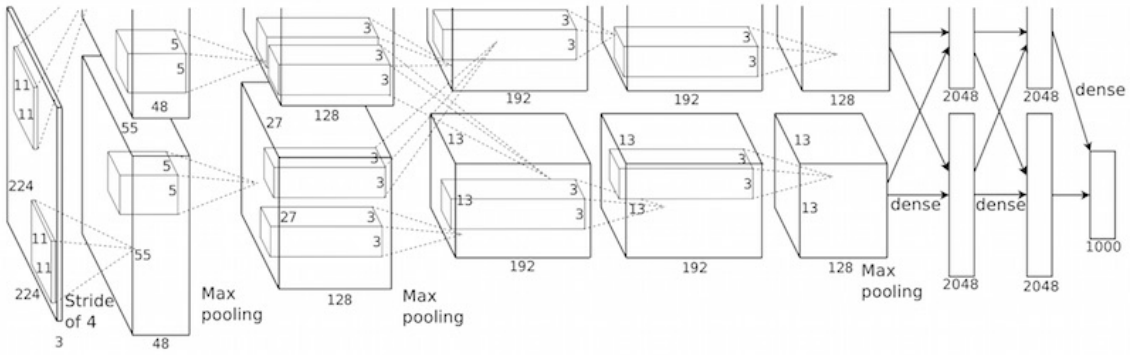


Figure 4: Network architecture of AlexNet model (Krizhevsky et al. (2012)).

Table 1: Network architecture of AlexNet.

Layer	Channels / Nodes	Filter size	Pooling size	Stride
1	96	11	3	4
2	256	5	3	1
3	384	3	N/A	1
4	384	3	N/A	1
5	256	3	3	1
6	4096	N/A	N/A	N/A
7	4096	N/A	N/A	N/A
8	2	N/A	N/A	N/A

5.3. Bag of words (BoW)

The bag of words (BoW) method considers images as sets of local features and uses the histogram of local features as the feature of images. One of the simplest methods is the one Csurka et al. (2004) proposed. They used the scale-invariant feature transform (SIFT) as a local feature and used an SVM as a classifier. The BoW method has several variants, such as the ones that use latent Dirichlet allocation (LDA), and spatial pyramid matching, and it has six steps:

1. Select images for creating vocabulary, and generate interest points for all training images.
2. Represent interest points using local-level features like the SIFT local feature.
3. Form vocabulary using a clustering algorithm such as K-means clustering.
4. Generate intermediate-level representations such as histograms for each image with vocabulary.
5. Train a classifier on the intermediate feature vectors.
6. For predicting, apply the trained classifier to the extracted BoW feature vector of the target image.

We used the SIFT feature for local-level features and an SVM with the radial basis function kernel for the classifier.

5.4. Deep Learning

We evaluated Krizhevsky et al.'s AlexNet and customized CNN/DCNN models with different numbers of convolution layers to compare accuracy and training time. The customized models have two fully connected layers and one softmax layer. They were trained using two types of algorithms: one that used fine-tuning and one that is trained from scratch. The AlexNet was trained using both algorithms while the customized models were trained using only the second algorithm. The customized CNN/DCNN models were also trained using BN to improve their performance. The original AlexNet with the Dropout was compared with one that had BN layers. Table 1 lists the specifications for the convolution and fully connected layers. The softmax classifier, which uses the cross-entropy loss (Bishop (1995)), is configured as the last layer.

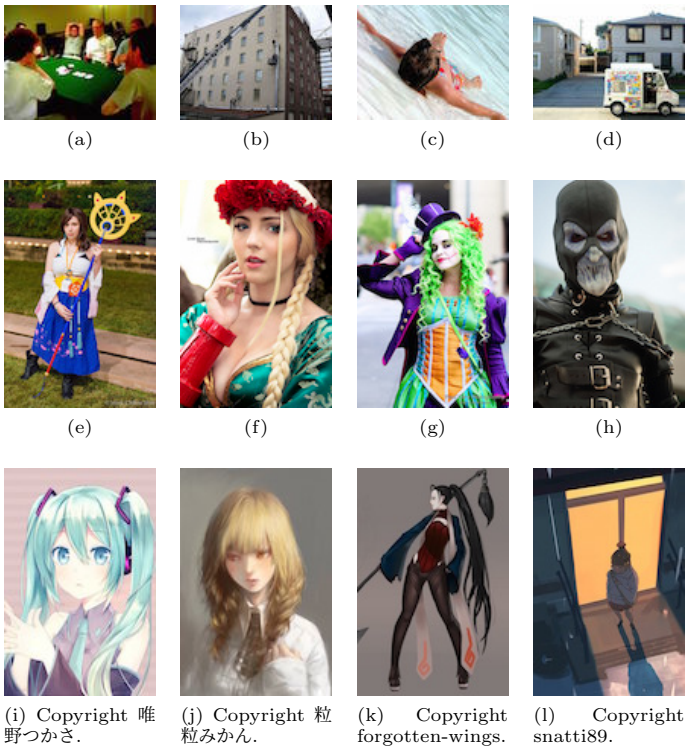
In the pre-processing, the images were resized to a fixed resolution of 256×256. In the fine-tuned model, they were randomly cropped to 227×227 for data augmentation purposes.

6. Datasets

We constructed two datasets for use in evaluating the models. Example images are shown in Figure 5.

6.1. Illustrations-Photographs dataset

One dataset consisted simply of 10,000 illustrations and 10,000 general photographs. The illustrations were randomly taken from anime-pictures.net and consisted of characters drawn in various styles. The photographs were randomly taken from the Flickr30k dataset (Young et al. (2014)).



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

(e) (f) (g) (h)

(i) Copyright 唯野つかさ. (j) Copyright 粒粒みかん. (k) Copyright forgotten-wings. (l) Copyright snatti89.

Figure 5: Example images used for evaluation. Those in first row are everyday images taken from Flickr30k dataset, those in second row are costume images taken from Flickr, and those in third row are illustrations taken from anime-pictures.net.

6.2. Illustrations-Cosplays dataset

The second dataset consisted of 10,000 anime illustrations and 10,000 costume images. To collect natural images that were similar to the illustrations, images of people in costume were taken from Flickr using keywords such as "cosplay" and "cosplayer" using Flickr 's search API.

7. Experimental setups

We evaluated model performance over the two datasets. Each dataset was split into 15,000 training images and 5,000 test images.

7.1. Illustration-Photographs dataset

We first evaluated the handcrafted features and DCNN with fine-tuning on the illustrations-photographs dataset. The Caffe (Jia et al. (2014)) was used to implement the DCNN model. We used SGD for fine-tuning. The hyperparameter settings used for SGD are shown in Table 2.

7.2. Illustrations-Costume images dataset

We next evaluated all models on the illustrations-costume images dataset. We fine-tuned the DCNN model using the same settings as for the first dataset and trained the customize models using the AdaDelta adaptive learning rate method (Zeiler (2012)) to shorten the training time. The

Table 2: Hyperparameters used for SGD

Parameter	Value
base_lr	0.0005
lr_policy	step
momentum	0.9
weight_decay	0.0005
gamma	0.1
batch size	16

Table 3: Hyperparameters used for AdaDelta

Parameter	Value
decay	0.9
batch size	32

hyperparameter settings used for AdaDelta are shown in Table 3. We used Caffe to fine-tune the DCNN model and used the Nervana Systems' deep learning framework "neon," which has a BN implementation, for the customized CNN/DCNN models. The settings used for the convolution layers and fully connected layers in the customized networks are shown in Table 4. As neon's back-end is implemented in a nondeterministic way, we used the average for five experiments.

8. Results

8.1. Illustrations-Photographs dataset

The results for the illustrations-photographs dataset are shown in Figure 6. The fine-tuned DCNN model achieved substantially higher accuracy (96.8%) after 20,000 iterations than either the outline detection or color intensity models, which used handcrafted features.

8.2. Illustrations-Costume images dataset

The results for the illustrations-costume images dataset are listed in Table 5. The training loss curves for the fine-tuned DCNN model and some of the CNN models are shown in Figures 9 and 11 to 14. The validation accuracies for the corresponding models are also shown in Figures 10 and 15 to 18.

The DCNN model had the highest accuracy (96.8%) after 50 epochs of training, and the CNN models that had more than two convolution layers outperformed the other models. The training loss curve and the validation accuracy curve for the DCNN model are shown in Figures 9 and 10. The model using the RGB histogram and SVM achieved 84.2% accuracy while the BoW model had the lowest accuracy.

The AlexNet model with BN converged at around 20-30 epochs while the training curve of the AlexNet model without BN became unstable after 20-40 epochs (Figures 13 and 14). The training loss for the fine-tuned DCNN model also converged at around 20-30 epochs.

Table 4: Network architecture used in our custom models.

Layer	Channels / Nodes	Filter size	Pooling size	Stride
1	64	2	N/A	2
2	128	3	2	2
3	256	3	2	2
4	512	3	2	2
5	1024	3	2	2
6	512	N/A	N/A	N/A
7	512	N/A	N/A	N/A
8	2	N/A	N/A	N/A

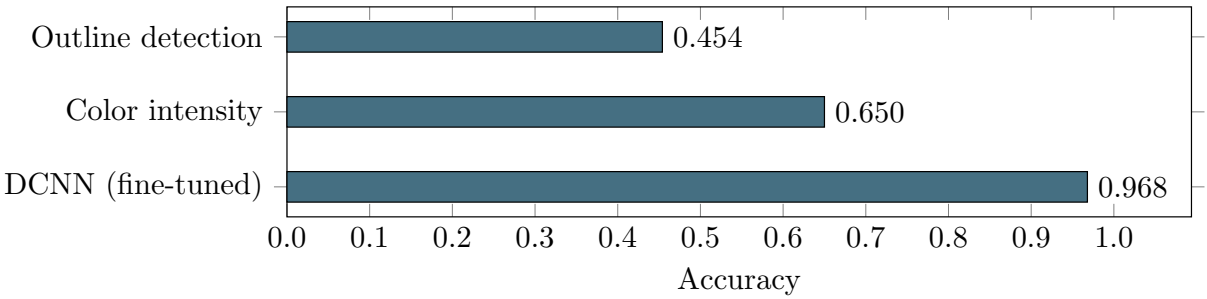


Figure 6: Results for illustrations-photographs dataset.

Table 5: Results for all models on the Illustrations-Costume images dataset

Model	Accuracy
Color intensity	0.577
Outline detection	0.437
Color histogram (RGB) + SVM	0.842
Color histogram (HSV) + SVM	0.499
Bag of Words (SIFT + SVM)	0.500
1-layer CNN	0.863
2-layer CNN	0.908
3-layer CNN	0.892
4-layer CNN	0.911
5-layer CNN	0.930
AlexNet (without BN)	0.832
AlexNet (with BN)	0.920
AlexNet (fine-tuned)	0.968

Among the CNN models, the ones with more convolution layers tended to have higher accuracy and a longer training time. The training loss curves for the 1-layer and 5-layer CNN models are shown in Figures 11 and 12. The model with five convolution layers had the best accuracy (93.0%) of the customized models.

The results for the CNN models are shown in Figures 7 and 8. The training time is the time it took the model to finish 100 epochs in the training.

9. Discussion

Our finding that the models using handcrafted features performed worse than the other models (Figure 6 and Table 5) suggests that these features are not useful for distinguishing illustrations from photographs. Using the outline detection feature produced particularly poor results, apparently because many illustrations, especially photo-realistic illustrations, do not have dark outline colors. Using the color intensity feature produced slightly better results, but still lower than most of the other models. This indicates that natural images do not have as much "scattered" coloring at the pixel level as we expected.

For the illustrations-costume dataset, using the RGB histogram model produced much better results than the BoW model. Since the BoW model uses SIFT, which does not include any color-related information, this finding indicates that recognizing color patterns is essential for distinguishing illustrations from natural images. The results plotted in Figure 7 show that networks with more layers perform better. We believe that CNN models that have more layers are better able to extract hierarchical patterns from images. The finding that the 5-layer CNN model outperformed the AlexNet model indicates that it is also possible that the number of filters in a convolution layer is an important factor in performance. The AlexNet model had fewer filters than the 5-layer CNN model while the customized models had more filters in the later layers (Table 4).

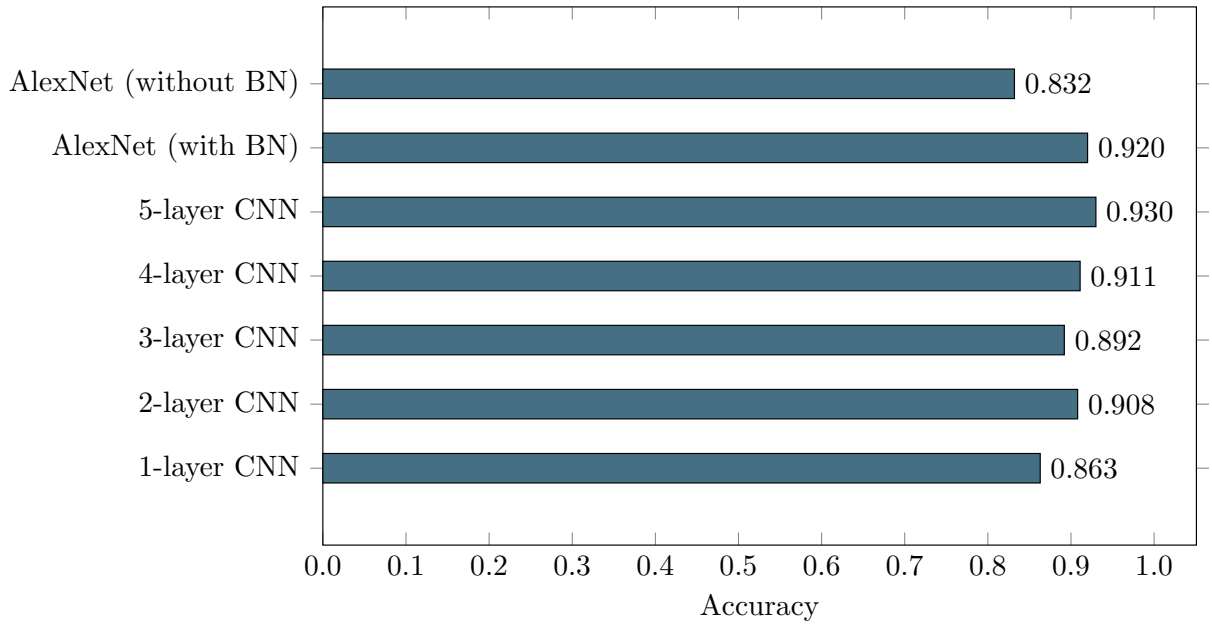


Figure 7: Accuracy of CNN models on illustrations-photographs dataset.

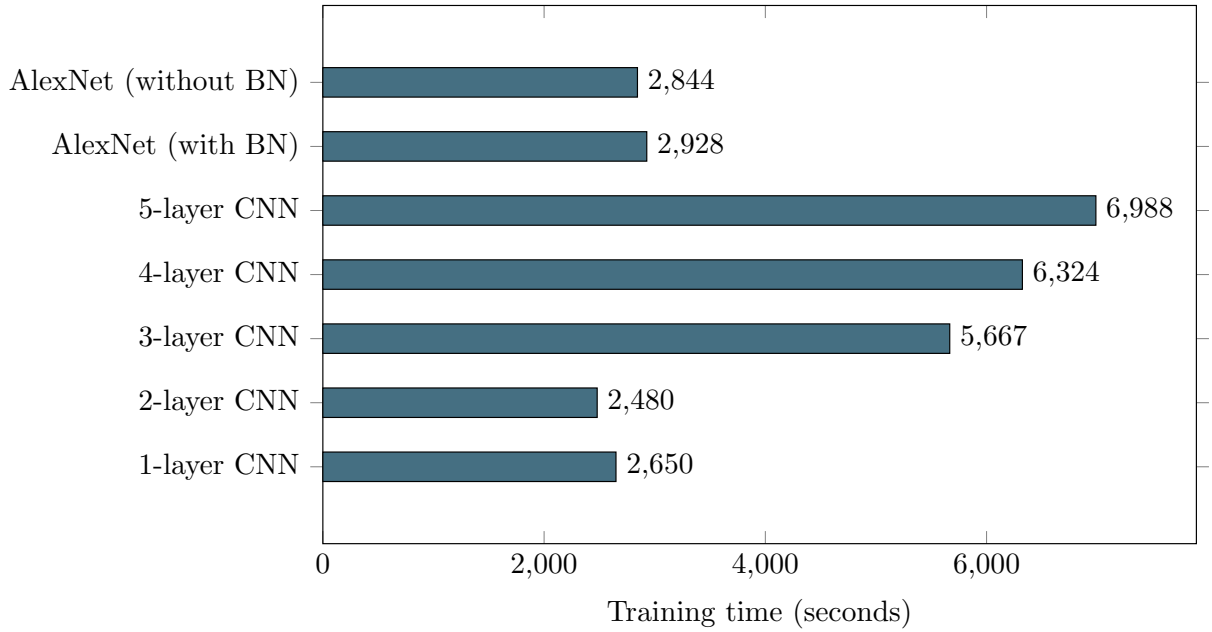


Figure 8: Training time of the customized models on illustrations-photographs dataset.

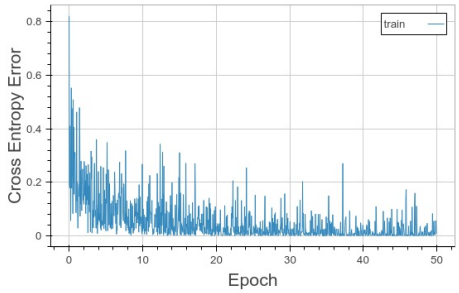


Figure 9: Traning loss curve of fine-tuned DCNN.

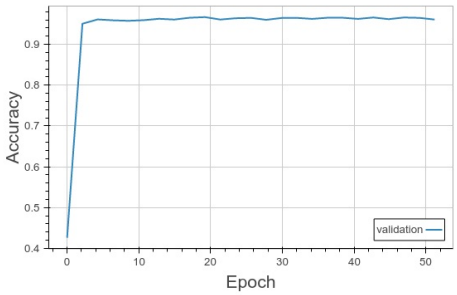


Figure 10: Validation accuracy curve of fine-tuned DCNN.

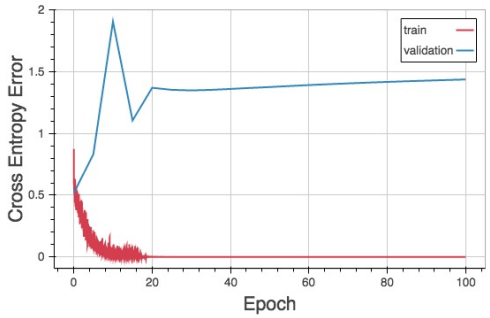


Figure 11: Training loss curve of 1-layer CNN.

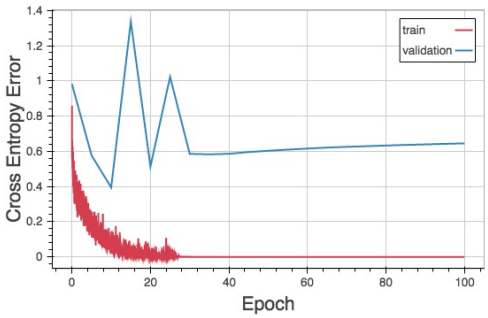


Figure 12: Training loss curve of 5-layer CNN.

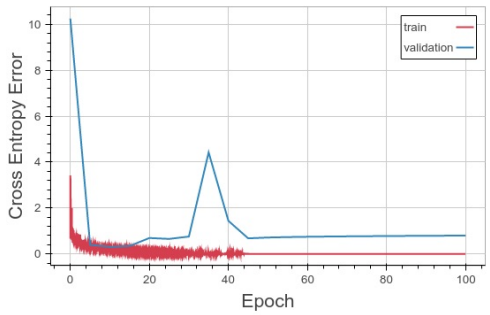


Figure 13: Training loss curve of AlexNet(with BN).

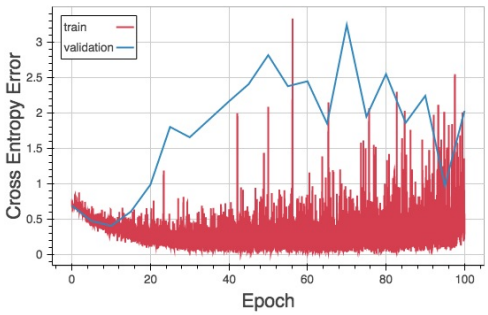


Figure 14: Training loss curve of AlexNet(without BN).

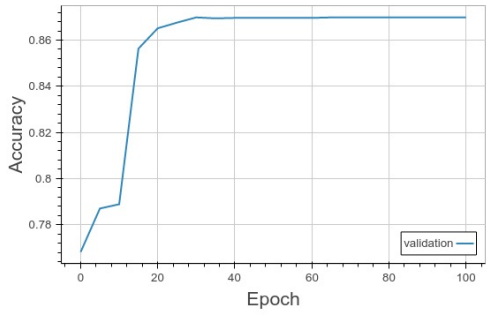


Figure 15: Validation accuracy of 1-layer CNN.

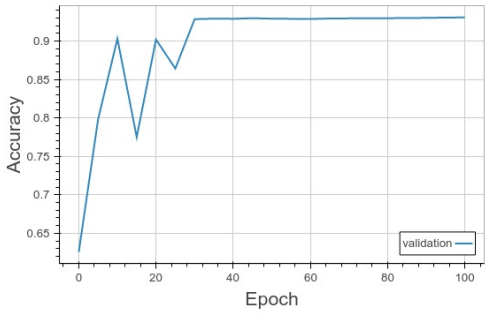


Figure 16: Validation accuracy of 5-layer CNN.

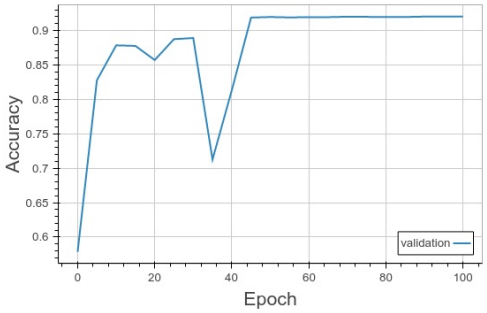


Figure 17: Validation accuracy of AlexNet(with BN).

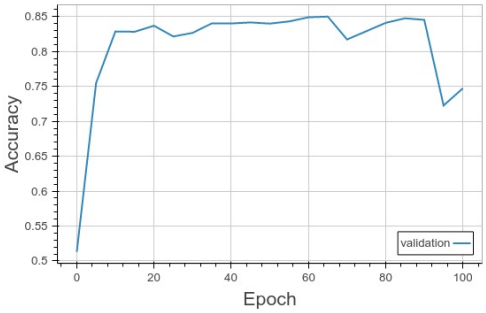


Figure 18: Validation accuracy of AlexNet(without BN).

10. Conclusion

We found that the DCNN model trained using fine-tuning was the most effective of the evaluated models. Its high accuracy and reasonable training time make the DCNN model with fine-tuning appropriate for implementing the illustration aggregation function in our target system.

Future work may include expanding the datasets so that the system can handle a wider variety of image types such as sketches and 3D graphics images. Also, it may be possible to use the results of this work to study a more general version of this task (applying deep learning models to the general paintings-photographs classification problem, etc.). Finally, it may be interesting to try improving the performance further by tuning the number of feature maps or adjusting the number of fully connected layers.

References

Athitsos, V., Swain, M., Frankel, C., June 1997. Distinguishing photographs and graphics on the world wide web. In: Content-Based Access of Image and Video Libraries, 1997. Proceedings. IEEE Workshop on. pp. 10–17.

Bengio, Y., Jan. 2009. Learning deep architectures for ai. Found. Trends Mach. Learn. 2 (1), 1–127.
URL http://dx.doi.org/10.1561/2200000006

Bishop, C. M., 1995. Neural Networks for Pattern Recognition. Oxford University Press, Inc., New York, NY, USA.

Chapelle, O., Haffner, P., Vapnik, V. N., Sep. 1999. Support vector machines for histogram-based image classification. Trans. Neur. Netw. 10 (5), 1055–1064.
URL http://dx.doi.org/10.1109/72.788646

Connolly, C., Fliess, T., 1997. A study of efficiency and accuracy in the transformation from RGB to CIELAB color space. IEEE Trans. Image Processing 6 (7), 1046–1048.
URL http://dx.doi.org/10.1109/83.597279

Csurka, G., Dance, C. R., Fan, L., Willamowski, J., Bray, C., 2004. Visual categorization with bags of keypoints. In: In Workshop on Statistical Learning in Computer Vision, ECCV. pp. 1–22.

Cutzu, F., Hammoud, R., Leykin, A., 2003. Estimating the photorealism of images: Distinguishing paintings from photographs. cvpr 02.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T., 2013. Decaf: A deep convolutional activation feature for generic visual recognition. CoRR abs/1310.1531.
URL http://arxiv.org/abs/1310.1531

Fukushima, K., 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36, 193–202.

Gatys, L. A., Ecker, A. S., Bethge, M., 2015. A neural algorithm of artistic style. CoRR abs/1508.06576.
URL http://arxiv.org/abs/1508.06576

Hubel, D. H., Wiesel, T. N., 1968. Receptive fields and functional architecture of monkey striate cortex. Journal of Physiology (London) 195, 215–243.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167.
URL http://arxiv.org/abs/1502.03167

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T., 2014. Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the 22Nd ACM International Conference on Multimedia. MM '14. ACM, New York, NY, USA, pp. 675–678.
URL http://doi.acm.org/10.1145/2647868.2654889

Karayev, S., Hertzmann, A., Winnemoeller, H., Agarwala, A., Darrell, T., 2013. Recognizing image style. CoRR abs/1311.3715.
URL http://dblp.uni-trier.de/db/journals/corr/corr1311.html\#KarayevHWAD13

Krizhevsky, A., Sutskever, I., Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems.

Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. pp. 2278–2324.

Nair, V., Hinton, G. E., 2010. Rectified linear units improve restricted boltzmann machines. In: Fürnkranz, J., Joachims, T. (Eds.), Proceedings of the 27th International Conference on Machine Learning (ICML-10). Omnipress, pp. 807–814.
URL http://www.icml2010.org/papers/432.pdf

Otsu, N., 1979. A Threshold Selection Method from Gray-level Histograms. IEEE Transactions on Systems, Man and Cybernetics 9 (1), 62–66.
URL http://dx.doi.org/10.1109/TSMC.1979.4310076

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15, 1929–1958.
URL http://jmlr.org/papers/v15/srivastava14a.html

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: CVPR 2015.
URL http://arxiv.org/abs/1409.4842

Young, P., Lai, A., Hodosh, M., Hockenmaier, J., 2014. From image descriptions to visual denotations: New similarity metrics for se-

mantic inference over event descriptions. TACL 2, 67–78.
URL https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/229

Zeiler, M. D., 2012. ADADELTA: an adaptive learning rate method. CoRR abs/1212.5701.
URL http://arxiv.org/abs/1212.5701

Zeiler, M. D., Fergus, R., 2013. Visualizing and understanding convolutional networks. CoRR abs/1311.2901.
URL http://arxiv.org/abs/1311.2901