A FRAMEWORK OF EXTRACTING MULTI-SCALE FEATURES USING MULTIPLE CONVOLUTIONAL NEURAL NETWORKS

Kuan-Chuan Peng and Tsuhan Chen

School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14850, USA
{kp388, tsuhan}@cornell.edu

ABSTRACT

Most works related to convolutional neural networks (CNN) use the traditional CNN framework which extracts features in only one scale. We propose multi-scale convolutional neural networks (MSCNN) which can not only extract multi-scale features but also solve the issues of the previous methods which use CNN to extract multi-scale features. With the assumption of label-inheritable (LI) property, we also propose a method to generate exponentially more training examples for MSCNN from the given training set. Our experimental results show that MSCNN outperforms both the state-of-the-art methods and the traditional CNN framework on artist, artistic style, and architectural style classification, supporting that MSCNN outperforms the traditional CNN framework on the tasks which at least partially satisfy LI property.

Index Terms— Convolutional neural networks, multi-scale, label-inheritable

1. INTRODUCTION

Convolutional neural networks (CNN) have received a great deal of focus in recent studies [1, 2, 3, 4, 5]. This trend started when Krizhevsky et al. [6] used CNN to achieve extraordinary performance in ImageNet [7] classification. Furthermore, Donahue et al. [8] show that the CNN used in Krizhevsky's work [6] can serve as a generic feature extractor, and that the extracted features outperform the state-of-the-art methods on standard benchmark datasets.

Although recent researchers have shown breakthrough performance by using CNN, there are two constraints in the traditional CNN framework: 1: CNN extracts the features in only one scale without explicitly leveraging the features in different scales. 2: The performance of CNN highly depends on the amount of training data, which is shown in recent works [4, 8]. Using either Caltech-101 [9] or Caltech-256 [10], these works [4, 8] modify the number of training examples per class and record the corresponding accuracy. Both works find that satisfactory performance can be achieved only when enough training examples are used. In this paper, we want to solve the above two issues of the traditional CNN framework. We propose multi-scale convolutional neural networks (MSCNN), a framework of extracting multi-scale features using multiple CNN. The details of MSCNN are explained in Sec. 3. We also propose a method to generate exponentially more training examples for MSCNN from the given training set based on the assumption that the task of interest is label-inheritable (LI). The concept of LI is explained in Sec. 2.

In recent studies extracting multi-scale features using CNN, Gong et al. [2] propose multi-scale orderless pooling, but He et al. [3] propose spatial pyramid pooling. Both works show experimental results which support the argument that using multi-scale features outperforms using features in only one scale. In both works, the multi-scale features are extracted by using special pooling methods in the same CNN, so the CNN parameters used to extract multi-scale features are the same before the pooling stage. However, we argue that the features in different scales should be extracted with different sets of CNN parameters learned from the training examples in different scales, which is explicitly modeled in our proposed MSCNN.

In the applications using multiple CNN, Lu et al. [11] predict pictorial aesthetics by using double-column convolutional neural network (DCNN) to extract features in two different scales, global view and fine-grained view. They also show that using DCNN outperforms using features in only one scale. The concept of using multiple CNN is similar in both DCNN and our proposed MSCNN. Nevertheless, the input images for the scale of fine-grained view in DCNN are generated by randomly cropping from the images in the scale of global view. Lu's approach [11] has two drawbacks: 1: The randomly cropped image may not well represent the entire image in the scale of fine-grained view. 2: By using only the cropped images in the scale of fine-grained view, the information which is not in the cropped area is ignored. In MSCNN, instead of throwing away information, we extract the features from every portion of the input image in each scale, which solves the above two drawbacks of DCNN.

In this paper, we make the following contributions:

- We propose multi-scale convolutional neural networks (MSCNN) which can extract the features in different scales using multiple convolutional neural networks.

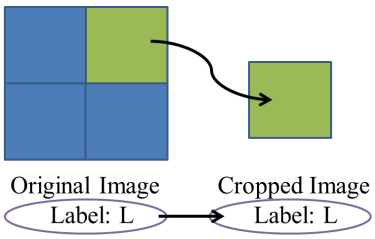

Fig. 1. The illustration of label-inheritable (LI) property. Given an image dataset D associated with a task T, if any cropped version of any image I from D can take the same label as that of I, we say that T satisfies LI property. In this figure, we only show the case when the cropped image is the upper right portion of the original image. In fact, the cropped image can be any portion of the original image.

We show that MSCNN outperforms not only the state-of-the-art performance on three tasks (architectural style [12], artist, and artistic style classification [13]) but also the traditional CNN framework which only extracts features in one scale.

- We also propose a method to generate exponentially more training examples for MSCNN from the given training set. Under the label-inheritable (LI) assumption, our method generates exponentially more training examples without the need to collect new training examples explicitly. Although our method and MSCNN are designed under LI assumption, our experimental results support that our proposed framework can still outperform the traditional CNN framework on the tasks which only partially satisfy LI assumption.

2. LABEL-INHERITABLE PROPERTY

Our proposed MSCNN is designed for the tasks satisfying label-inheritable property which is illustrated in Fig. 1. Given an image dataset D associated with a task T, if any cropped version of any image I from D can take the the same label as that of I, we say that T is label-inheritable (LI). In other words, if the concept represented by the label of any image I in the given dataset D is also represented in each portion of I, the corresponding task T satisfies LI property.

Fig. 2 shows three examples from three different tasks which satisfy LI property in different degrees. We also list the corresponding dataset, task, and label under each example image. For any image from Painting-91 [13], each portion of that image is painted by the same artist (Picasso for the example image), so the task "artist classification" is LI. For the images from arcDataset [12], we can recognize the architectural style from different parts of the architecture (for example, the roof and pillars). However, if the cropped image does not contain any portion of the architecture, the cropped image cannot represent the architectural style of the original image, which is the reason why LI property is only partially satisfied for architectural style classification. For the images from Caltech-101 [9], the LI property is satisfied when the cropped image contains the entire object which the label represents. If the cropped image contains only a portion of the object, the cropped image may not be able to take the same label. For the example image, the portion of the mouth cannot totally represent the label "faces." Therefore, we think object classification is mostly not LI.

Dataset	Painting-91 [13]	arcDataset [12]	Caltech-101 [9]
Task	Artist Classification	Architectural Style Classification	Object Classification
Label	Picasso	Baroque Architecture	Faces
Label-inheritable	Yes	Partial	Mostly No

Fig. 2. Example images from three different tasks which satisfy LI property in different degrees. The corresponding dataset, task, label, and the extent that LI property is satisfied are shown under each image.

In this work, we apply our proposed framework, MSCNN, to the tasks satisfying LI property in different degrees, showing that MSCNN outperforms the traditional CNN framework on the tasks satisfying LI property. We also show experimental results supporting that MSCNN can still outperform the traditional CNN framework on the tasks which only partially satisfy LI property. The details of the datasets and tasks used in this work are presented in Sec. 3.1, and the experimental results are shown in Sec. 4.

3. EXPERIMENTAL SETUP

3.1. Datasets and Tasks

Conducting experiment on 4 tasks from 3 datasets, we summarize all the datasets and tasks used in this work in Table 1, where their properties and related statistics are also shown. At the bottom of Table 1, we list the experimental setting associated with each task. In this paper, we refer to each task by its task ID listed in Table 1, where "training/testing split" indicates whether the training/testing splits are randomly generated or specified by the work proposing the dataset/task, and "number of fold(s)" shows that how many different training/testing splits are used for the task. To be consistent with prior works, we use the same experimental settings for these tasks as those used in the references listed at the bottom of Table 1.

Table 1. The tasks and associated datasets used in this work along with their properties. In this paper, we refer to each task by the corresponding task ID listed under each task. The experimental setting for each task is provided at the bottom of the table.

dataset	Painting-91 [13]	Painting-91 [13]	arcDataset [12]	Caltech-101 [9]
task	artist classification	artistic style classification	architectural style classification	object classification
task ID	ARTIST-CLS	ART-CLS	ARC-CLS	CALTECH101
number of classes	91	13	10 / 25	101
number of images	4266	2338	2043 / 4786	8677
image type	painting	painting	architecture	object
label-inheritable	yes	mostly yes	partial	mostly no
examples of class labels	Rubens, Picasso.	Baroque, Cubbism.	Georgian, Gothic.	chair, cup.
number of training images	2275	1250	300 / 750	1515 / 3030
number of testing images	1991	1088	1743 / 4036	3999 / 2945
training/testing split	specified [13]	specified [13]	random	random
number of fold(s)	1	1	10	5
evaluation metric	accuracy	accuracy	accuracy	accuracy
reference of the above setting	[13]	[13]	[12]	[4]

3.2. MSCNN Architecture

The architecture of our proposed multi-scale convolutional neural networks (MSCNN) is illustrated in Fig. 3. MSCNN consists of $m+1$ CNN which extract features in $m+1$ different scales (from scale 0 to scale m, one CNN for each scale), and the structures of the $m+1$ CNN are exactly the same. We only show $m=1$ in Fig. 3 for clarity. In each scale, we use Krizhevsky's CNN [6] as the feature extractor which takes an image as input and outputs a 4096-d feature vector. The 4096-d feature vector is retrieved from the output of the topmost fully connected layer of Krizhevsky's CNN [6].

Given an input image, we extract its multi-scale features according to the following steps:

1. In scale i, we place a $2^i \times 2^i$ grid on the input image and generate 4^i cropped images.
2. We resize each of the 4^i cropped images in scale i to a pre-defined image size $k \times k$.
3. Each of the 4^i cropped resized images takes turn (in the order of left to right, top to bottom) to be the input of the CNN in scale i. In other words, the CNN in scale i extracts features 4^i times with one of the 4^i cropped resized images as input at each time. Afterward we generate the feature vector in scale i by cascading those 4^i 4096-d feature vectors in order.
4. The final multi-scale feature vector is generated by cascading all the feature vectors in all $m+1$ scales in order, and the dimension of the multi-scale feature vector is $\left[\left(4^{m+1}-1\right)/3\right] \times 4096$.

For the CNN in each scale, we use the Caffe [14] implementation of Krizhevsky's CNN [6] except that the number of the nodes in the output layer is set to the number of classes in each task listed in Table 1. When using the Caffe [14] implementation, we adopt its default training parameters for training the CNN for ImageNet [7] classification unless otherwise specified. The pre-defined image size $k \times k$ is set to 256×256 according to the Caffe [14] implementation. The details of the training approach and how to make prediction are explained in Sec. 3.3.

3.3. Training Approach

Before training, we generate a training set S_i for each scale i from the original training set S of the task of interest T with the assumption that T satisfies LI property. We take the following steps:

1. We place a $2^i \times 2^i$ grid on each image of S, crop out 4^i sub-images, and resize each cropped image to $k \times k$.
2. Due to LI property, each cropped resized image is assigned the same label as that of the original image from which it is cropped.

After the above two steps, the generated training set S_i consists of $4^i \times |S|$ labeled images ($|S|$ is the number of images in S), and the size of each image is $k \times k$. We follow the Caffe [14] implementation, using $k = 256$.

In training phase, adopting the Caffe reference model [14] (denoted as M_{ImageNet}) for ImageNet [7] classification, we train the CNN in scale i using the training set S_i. We follow the descriptions and setting of supervised pre-training and fine-tuning used in Agrawal's work [1], where pre-training with M_{D} means using a data-rich auxiliary dataset D to initialize the CNN parameters and fine-tuning means that all the CNN parameters can be updated by continued training on

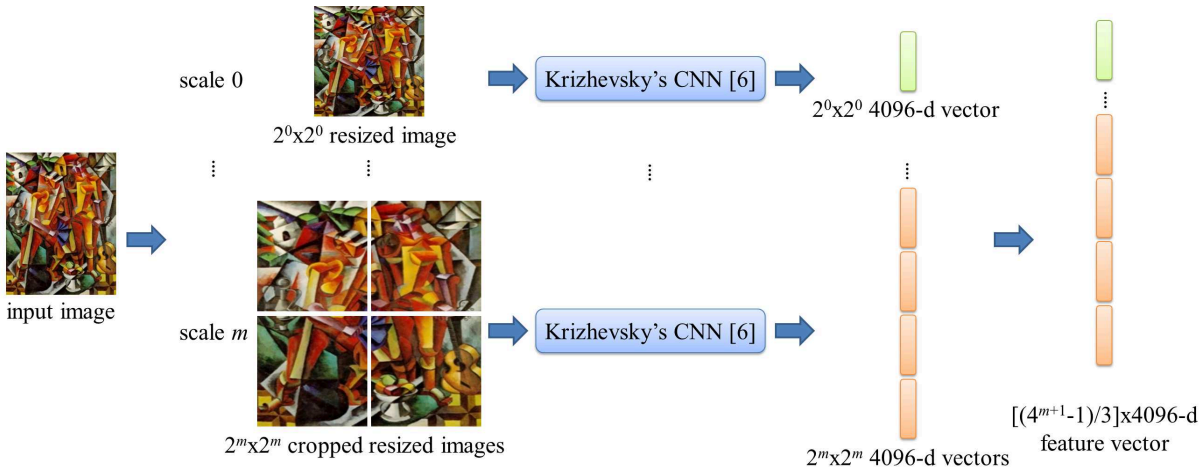

Fig. 3. The illustration of multi-scale convolutional neural networks (MSCNN) which consists of $m+1$ CNN (one for each of the $m+1$ different scales). We use each CNN as a feature extractor in that scale. We only show $m=1$ here for clarity. In scale i, we crop 4^i sub-images from the input image and resize each of them to 256×256. The CNN in scale i takes each of the 4^i cropped resized images as input at each time and outputs a 4096-d vector for each input. The final multi-scale feature vector of the input image is generated by cascading all the 4096-d vectors in $m+1$ scales. The details of the MSCNN architecture and the training approach are explained in Sec. 3.2 and Sec. 3.3 respectively.

the corresponding training set. For the CNN in each scale i, we pre-train it with M_{ImageNet} and fine-tune it with S_i. After finishing training all $m+1$ CNN in $m+1$ scales, we extract the $\left[\left(4^{m+1}-1\right)/3\right] \times 4096$-d multi-scale feature vector of each training image in S using the method described in Sec. 3.2. With these multi-scale feature vectors for training, we use support vector machine (SVM) to train a linear classifier supervised by the labels of the images in S. In practice, we use LIBSVM [15] to do so with the cost (parameter C in SVM) set to the default value 1. Trying different C values, we find that different C values produce similar accuracy, so we just use the default value.

In testing phase, given a testing image, we generate its $\left[\left(4^{m+1}-1\right)/3\right] \times 4096$-d multi-scale feature vector using the method described in Sec. 3.2. After that, we feed the feature vector of the testing image as the input of the trained SVM classifier, and the output of the SVM classifier is the predicted label of the testing image.

Compared with the traditional CNN ($m=0$), MSCNN has $m+1$ times of the number of parameters to train. However, under LI assumption, we can generate exponentially more training examples ($4^i \times |S|$ training examples in scale i) from the original training set S without the need to explicitly collect new training examples. For any two training examples I_i and I_j from S_i and S_j respectively ($i<j$), the overlapping content cannot exceed a quarter of the area of I_i. Therefore, under LI assumption, our training approach not only generates exponentially more training examples but also guarantees certain diversity among the generated training examples.

In terms of the training time, since S_i contains $4^i \times |S|$ training examples, the time needed to train the CNN in scale i is $4^i \times t_0$, where t_0 is the training time of the traditional CNN framework. The training time of the linear SVM classifier is far less than that of CNN, so it is negligible in the total training time. If we train the $m+1$ CNN in $m+1$ different scales in parallel, the total training time of MSCNN will be $4^m \times t_0$.

When training CNN, the traditional method implemented by Caffe [14] resizes each training image to a pre-defined resolution because of the hardware limitation of GPU. Due to the resizing, partial information in the original high-resolution training image is lost. In MSCNN, more details in the original high-resolution training image are preserved in scale i ($i>0$) because all the training examples in S_i ($\forall i \in \{0,1,\cdots,m\}$) are resized to the same resolution. The extra information preserved in scale i ($i>0$) makes MSCNN outperform the traditional CNN framework on the tasks satisfying LI property, which is shown in Sec. 4.

Our proposed MSCNN extracts the features in different scales with scale-dependent CNN parameters. In other words, the CNN parameters in scale i are specifically learned from the training images in scale i (the images in S_i). This training approach solves the issue of previous works [2, 3] which simply generate multi-scale features from the same set of CNN parameters that may not be the most suitable one for each scale. In addition, MSCNN extracts the features from every portion of the image in each scale, which solves the drawback of Lu's work [11] that does not fully utilize the information of the entire image in the scale of fine-grained view. Applying our proposed MSCNN to the tasks listed in Table 1, we report the performance in Sec. 4.

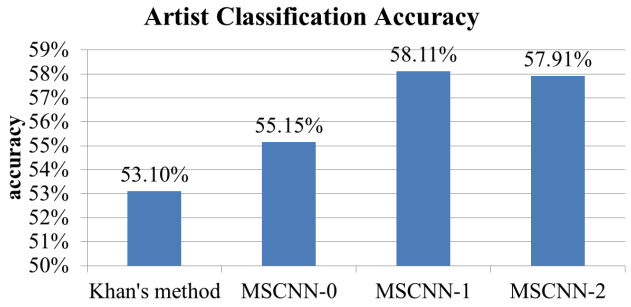

Fig. 4. The performance comparison for the task ARTIST-CLS. The results show that our proposed methods (MSCNN-1 and MSCNN-2) outperform both Khan's method [13] and the traditional CNN framework (MSCNN-0), which supports that MSCNN outperforms the traditional CNN framework on the tasks satisfying LI property.

4. EXPERIMENTAL RESULTS

Using the experimental setting specified in Table 1, we compare the performance of MSCNN with that of the traditional CNN framework ($m = 0$) and that of the state-of-the-art methods. We use the notation "MSCNN-m" to represent the MSCNN framework consisting of $m + 1$ CNN in $m + 1$ different scales (scale 0 to scale m). MSCNN-0 represents the traditional CNN framework which extracts the features in only one scale. The results of each task are explained in the following paragraphs.

ARTIST-CLS: Fig. 4 compares the performance of Khan's method [13], the traditional CNN framework (MSCNN-0) and our proposed methods (MSCNN-1 and MSCNN-2), showing that our proposed methods outperform both the current state-of-the-art method and the traditional CNN framework. Since ARTIST-CLS satisfies LI property, this result supports that MSCNN outperforms the traditional CNN framework on the tasks which are LI. Fig. 4 also shows that MSCNN-2 performs worse than MSCNN-1, which indicates that increasing the number of scales in MSCNN may not always increase the accuracy. This is reasonable because the training images in scale i (the images in S_i) are less likely to contain useful features as i increases. Therefore, the accuracy increases the most when we increase the number of scales in MSCNN to a certain level. Although MSCNN-2 performs worse than MSCNN-1, both MSCNN-1 and MSCNN-2 outperform MSCNN-0, which supports that using multi-scale features is better than using the features in one scale for the tasks satisfying LI property.

ART-CLS: Fig. 5 compares the performance of Khan's method [13] and MSCNN-0 to MSCNN-3, showing that our proposed methods (MSCNN-1 to MSCNN-3) outperform both the current state-of-the-art method and the traditional CNN framework (MSCNN-0). Because ART-CLS mostly satisfies LI property, this result supports that MSCNN outperforms the traditional CNN framework on the tasks satisfying LI property. Fig. 5 also supports the following two arguments: 1: The accuracy increases the most when we increase the number of scales in MSCNN to a certain level. 2: Using multi-scale features outperforms using the features in one scale for the tasks which are LI.

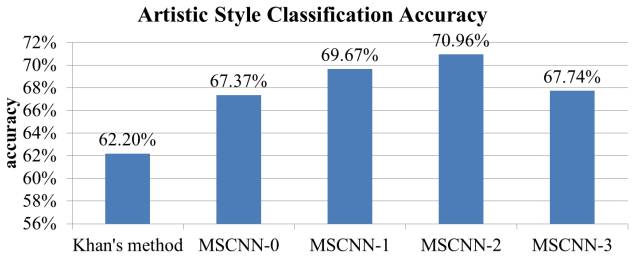

Fig. 5. The performance comparison for the task ART-CLS. The results show that our proposed methods (MSCNN-1 to MSCNN-3) outperform both Khan's method [13] and the traditional CNN framework (MSCNN-0), which supports that MSCNN outperforms the traditional CNN framework on the tasks which are LI.

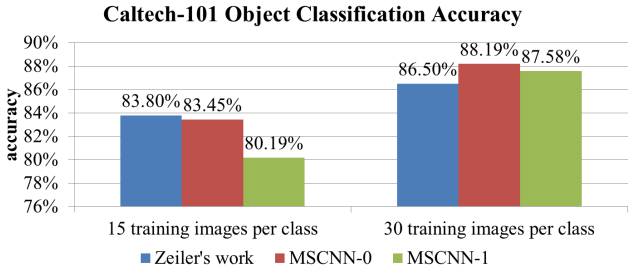

Fig. 6. The performance comparison for the task CALTECH101. The results show that our proposed method (MSCNN-1) performs worse than the traditional CNN framework (MSCNN-0) because CALTECH101 is mostly not LI, which indicates that MSCNN does not work for the tasks which are not LI.

CALTECH101: Fig. 6 compares the performance of Zeiler's work [4], MSCNN-0, and MSCNN-1, showing that Zeiler's work [4] and MSCNN-0 are comparable in accuracy. However, Fig. 6 shows that MSCNN-1 performs worse than MSCNN-0. This result is not surprising because CALTECH101 is mostly not LI and our proposed framework is based on LI assumption. Fig. 6 indicates that MSCNN does not work for the tasks which are not LI.

ARC-CLS: Fig. 7 compares the performance of Xu's method [12] and MSCNN-0 to MSCNN-2, showing that our proposed methods (MSCNN-1 and MSCNN-2) outperform both the current state-of-the-art method and the traditional CNN framework (MSCNN-0). Since ARC-CLS

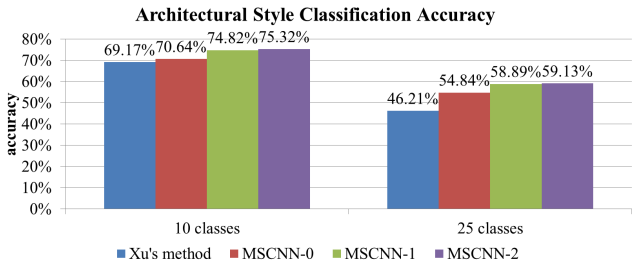

Fig. 7. The performance comparison for the task ARC-CLS. The results show that our proposed methods (MSCNN-1 and MSCNN-2) outperform both Xu's method [12] and the traditional CNN framework (MSCNN-0), which supports that MSCNN still outperforms the traditional CNN framework on the tasks which are partially LI.

only partially satisfies LI property, this result supports that MSCNN can still outperform the traditional CNN framework on the tasks which are only partially LI. Fig. 7 also supports that using multi-scale features outperforms using the features in only one scale for the tasks which are partially LI.

Based on our experimental results, we show that MSCNN outperforms the traditional CNN framework on the tasks which are LI. Even when LI property is only partially satisfied, MSCNN can still outperform the traditional CNN framework. In addition, our results indicate that for a task satisfying LI property, there is a most suitable number of scales for MSCNN such that the accuracy can increase the most.

5. CONCLUSION

We addressed the label-inheritable (LI) property of the classification tasks and proposed a novel framework, multi-scale convolutional neural networks (MSCNN), for the tasks satisfying LI property. We also proposed a training method for MSCNN under LI assumption such that exponentially more training examples can be generated without the need to collect new training data. MSCNN not only solves the issues of the previous methods which use CNN to extract multi-scale features but also outperforms both the state-of-the-art methods and the traditional CNN framework on two LI tasks, artist and artistic style classification. Our experimental results also show that MSCNN can still outperform both the state-of-the-art method and the traditional CNN framework even on architectural style classification which is only partially LI.

6. REFERENCES

[1] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *ECCV*, 2014, pp. 329–344.

[2] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *ECCV*, 2014, pp. 392–407.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *ECCV*, 2014, pp. 346–361.

[4] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV*, 2014, pp. 818–833.

[5] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based R-CNNs for fine-grained category detection," in *ECCV*, 2014, pp. 834–849.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. 2012.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009, pp. 248–255.

[8] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A deep convolutional activation feature for generic visual recognition," *CoRR*, vol. abs/1310.1531, 2013.

[9] F.-F. Li, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *CVPRW*, 2004.

[10] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[11] X. Lu, Z. Lin, H. Jin, J. Yang, and J. Z. Wang, "RAPID: rating pictorial aesthetics using deep learning," in *ACMMM*, 2014.

[12] Z. Xu, D. Tao, Y. Zhang, J. Wu, and A. C. Tsoi, "Architectural style classification using multinomial latent logistic regression," in *ECCV*, 2014, pp. 600–615.

[13] F. S. Khan, S. Beigpour, J. V. D. Weijer, and M. Felsberg, "Painting-91: a large scale database for computational painting categorization," *Machine Vision and Applications*, vol. 25, pp. 1385–1397, 2014.

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[15] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.