# CROSS-LAYER FEATURES IN CONVOLUTIONAL NEURAL NETWORKS FOR GENERIC CLASSIFICATION TASKS

*Kuan-Chuan Peng and Tsuhan Chen*

Cornell University
School of Electrical and Computer Engineering
Ithaca, NY 14850

## ABSTRACT

Recent works about convolutional neural networks (CNN) show breakthrough performance on various tasks. However, most of them only use the features extracted from the topmost layer of CNN instead of leveraging the features extracted from different layers. As the first group which explicitly addresses utilizing the features from different layers of CNN, we propose cross-layer CNN features which consist of the features extracted from multiple layers of CNN. Our experimental results show that our proposed cross-layer CNN features outperform not only the state-of-the-art results but also the features commonly used in the traditional CNN framework on three tasks – artistic style, artist, and architectural style classification. As shown by the experimental results, our proposed cross-layer CNN features achieve the best known performance on the three tasks in different domains, which makes our proposed cross-layer CNN features promising solutions for generic tasks.

***Index Terms*—** Convolutional neural networks (CNN), cross-layer features, generic classification tasks

## 1. INTRODUCTION

Convolutional neural networks (CNN) have shown breakthrough performance on various datasets in recent studies [1, 2, 3, 4, 5]. This widespread trend of using CNN starts when the CNN proposed by Krizhevsky et al. [6] outperforms the previous best results of ImageNet [7] classification by a large margin. In addition, Donahue et al. [8] adopt the CNN proposed by Krizhevsky et al. [6], showing that the features extracted from the CNN outperform the state-of-the-art results of standard benchmark datasets. Encouraged by these previous works, more researchers start to use Krizhevsky's CNN [6] as a generic feature extractor in their domains of interest.

Although extraordinary performance by using CNN has been reported in recent literature [1, 2, 3, 9, 10], there is one major constraint in the traditional CNN framework: the final output of the output layer is solely based on the features extracted from the topmost layer. In other words, given the features extracted from the topmost layer, the final output is independent of all the features extracted from other non-topmost layers. At first glance, this constraint seems to be reasonable because the non-topmost layers are implicitly considered in the way that the output of the non-topmost layers is the input of the topmost layer. However, we believe that the features extracted from the non-topmost layers are not explicitly and properly utilized in the traditional CNN framework where partial features generated by the non-topmost layers are ignored during training. Therefore, we want to relax this constraint of the traditional CNN framework by explicitly leveraging the features extracted from multiple layers of CNN. We propose cross-layer features based on Krizhevsky's CNN [6], and we show that our proposed features outperform Krizhevsky's CNN [6] on three different classification tasks – artistic style [11], artist [11], and architectural style [12]. The details of cross-layer CNN features, the experimental setup and the results are presented in Sec. 2, Sec. 3, and Sec. 4 respectively.

In recent studies analyzing the performance of multi-layer CNN [1, 4], both works extract features from Krizhevsky's CNN [6] and evaluate the performance on different datasets. These works achieve a consistent conclusion that the features extracted from the topmost layer have the best discriminative ability in classification tasks compared with the features extracted from other non-topmost layers. However, both works [1, 4] only evaluate the performance of the features extracted from one layer at a time without considering the features from multiple layers at once. Unlike [1, 4], in Sec. 4, we show that our proposed cross-layer CNN features outperform the features extracted from the topmost layer of Krizhevsky's CNN [6] (the features used in [1, 4]).

In previous works [11, 12] studying the three classification tasks (artistic style, artist, and architectural style) involved in this paper, the state-of-the-art results are achieved by the traditional handcrafted features (for example, SIFT and HOG) without considering CNN-related features. In Sec. 4, we show that our proposed cross-layer CNN features outperform the state-of-the-art results on the three tasks. Another related prior work is the double-column convolutional neural

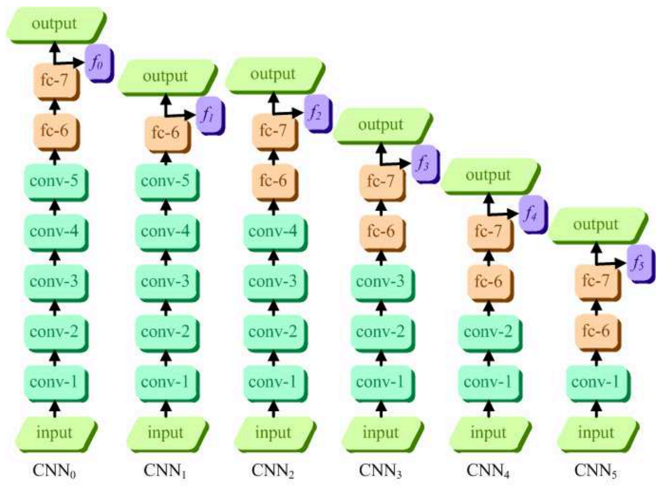

**Fig. 1**. The six CNN structures adopted in this work. CNN$_0$ represents Krizhevsky's CNN [6], and CNN$_1$ to CNN$_5$ are the same as CNN$_0$ except that some layers are removed. We use each CNN$_i$ $(i = \{0, 1, \cdots, 5\})$ as a feature extractor which takes an image as input and outputs a feature vector $f_i$ from the topmost fully connected layer. These $f_i$s are cascaded to form our proposed cross-layer features according to the definition in Table 1.

network (DCNN) proposed by Lu et al. [13]. Using DCNN to predict pictorial aesthetics, Lu et al. [13] extract multi-scale features from multiple CNNs with the multi-scale input data generated from their algorithm. In contrast, our work focuses on the cross-layer CNN features extracted from multiple layers without the need to generate multi-scale input.

In this paper, our main contribution is the concept of utilizing the features extracted from multiple layers of convolutional neural networks (CNN). Based on this concept, we propose cross-layer CNN features extracted from multiple layers and show that our proposed features outperform not only the state-of-the-art performance but also the results of the traditional CNN framework in artistic style [11], artist [11], and architectural style [12] classification. To the best of our knowledge, this is the first paper explicitly utilizing the features extracted from multiple layers of CNN, which is a strong departure from most CNN-related works which use only the features extracted from the topmost layer.

## 2. CROSS-LAYER CNN FEATURES

Fig. 1 and Table 1 illustrate the involved CNN structures in this work and how we form our proposed cross-layer CNN features respectively. There are 6 different CNN structures in Fig. 1, where CNN$_0$ represents the CNN proposed in Krizhevsky's work [6], and CNN$_1$ to CNN$_5$ are the "sub-CNNs" of CNN$_0$ (they are the same as CNN$_0$ except that some layers are removed). We use the same notation of convolutional layers (conv-1 to conv-5) and fully connected layers (fc-6 and fc-7) as that used in [1] to represent the corresponding layers in Krizhevsky's CNN [6]. In Fig. 1, in addition to the input and output layers, we only show the convolutional and fully connected layers of Krizhevsky's CNN [6] for clarity. Instead of using the output from the output layer of each CNN$_i$ $(i = \{0, 1, \cdots, 5\})$, we treat CNN$_i$ as a feature extractor which takes an image as input and outputs a $k$-d feature vector $f_i$ from the topmost fully connected layer, which is inspired by [8]. We follow the specification of Krizhevsky's CNN [6] and use $k = 4096$ in our experiment.

| feature ID | cross-layer features (Fig. 1) | dimension |
|---|---|---|
| $F_0$ (baseline) | $f_0$ | $k$ |
| $F_1$ | $f_0 + f_1$ | $2k$ |
| $F_2$ | $f_0 + f_2$ | $2k$ |
| $F_3$ | $f_0 + f_2 + f_3$ | $3k$ |
| $F_4$ | $f_0 + f_2 + f_3 + f_4$ | $4k$ |
| $F_5$ | $f_0 + f_2 + f_3 + f_4 + f_5$ | $5k$ |

**Table 1**. The summary of the cross-layer CNN features used in this work. Serving as a baseline, $F_0$ represents the features extracted from the topmost layer in the traditional CNN framework. $F_1$ to $F_5$ are our proposed cross-layer CNN features which are formed by cascading $f_i$s $(i = \{0, 1, \cdots, 5\})$ defined in Fig. 1. We follow the specification of Krizhevsky's CNN [6] and use $k = 4096$.

In Fig. 1, $f_0$ represents the features extracted from the output of the fc-7 layer in Krizhevsky's CNN [6], $f_1$ represents the features extracted from the fc-6 layer, and $f_2$ to $f_5$ represent the features derived from different combinations of the convolutional layers. $f_i$ $(i = \{0, 1, \cdots, 5\})$ is extracted from the topmost fully connected layer of CNN$_i$, not from the intermediate layer of CNN$_0$ because the features extracted from the topmost layer have the best discriminative ability according to [1, 4]. As the features learned from CNN$_0$ and its sub-CNNs, $f_i$s implicitly reflect the discriminative ability of the corresponding layers of CNN$_0$. Most CNN-related works use only $f_0$ and ignore the intermediate features ($f_1$ to $f_5$), but we explicitly extract them as part of our proposed cross-layer CNN features which are explained in the following paragraph.

Using the feature vectors ($f_i$s) defined in Fig. 1, we cascade these $f_i$s and form our proposed cross-layer CNN features. We summarize these cross-layer CNN features ($F_1$ to $F_5$) in Table 1, where how the features are formed and their dimensions are specified. $F_0$ represents the features extracted from the topmost layer in the traditional CNN framework without cascading the features from other layers. The feature IDs listed in Table 1 are used to refer to the corresponding cross-layer CNN features when we report the experimental results in Sec. 4, where we compare the performance of $F_i$ $(i = \{0, 1, \cdots, 5\})$ on three different tasks.

| dataset | Painting-91 [11] | Painting-91 [11] | arcDataset [12] |
|---|---|---|---|
| task | artist classification | artistic style classification | architectural style classification |
| task ID | ARTIST-CLS | ART-CLS | ARC-CLS |
| number of classes | 91 | 13 | 10 / 25 |
| number of images | 4266 | 2338 | 2043 / 4786 |
| image type | painting | painting | architecture |
| examples of class labels | Rubens, Picasso. | Baroque, Cubbism. | Georgian, Gothic. |
| number of training images | 2275 | 1250 | 300 / 750 |
| number of testing images | 1991 | 1088 | 1743 / 4036 |
| training/testing split | specified [11] | specified [11] | random |
| number of fold(s) | 1 | 1 | 10 |
| evaluation metric | accuracy | accuracy | accuracy |
| reference of the above setting | [11] | [11] | [12] |

**Table 2**. 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. For the task ARC-CLS, we conduct our experiment using two different experimental settings (the same as those used in [12]).

## 3. EXPERIMENTAL SETUP

### 3.1. Datasets and Tasks

We conduct experiment on three tasks (artistic style, artist, and architectural style classification) from two different datasets (Painting-91 [11] and arcDataset [12]). We summarize these datasets and tasks in Table 2, where their properties and related statistics are shown. We also provide the experimental settings associated with each task at the bottom of Table 2. When reporting the results in Sec. 4, we use the task ID listed in Table 2 to refer to each task. In Table 2, "training/testing split" represents whether the training/testing splits are randomly generated or specified by the literature proposing the dataset/task, and "number of fold(s)" lists the number of different splits of training/testing sets used for the task. To evaluate different methods under fair comparison, we use the same experimental settings for these three tasks as those used in the references listed at the bottom of Table 2. For the task ARC-CLS, there are two different experimental settings (10-way and 25-way classification) provided by [12], and we do both in our experiment.

### 3.2. Training Approach

In our experiment, we use the Caffe [14] implementation to train the 6 CNN$_i$ $(i = \{0, 1, \cdots, 5\})$ in Fig. 1 for each of the three tasks in Table 2. For each task, CNN$_i$ is adjusted such that the number of the nodes in the output layer is set to the number of classes of that task. When using the Caffe [14] implementation, we adopt its default training parameters for training Krizhevsky's CNN [6] for ImageNet [7] classification unless otherwise specified. Before training CNN$_i$ for each task, all the images in the corresponding dataset are resized to 256$\times$256 according to the Caffe [14] implementation.

In training phase, adopting the Caffe reference model provided by [14] (denoted as $M_{\text{ImageNet}}$) for ImageNet [7] classification, we train CNN$_i$ $(i = \{0, 1, \cdots, 5\})$ in Fig. 1 for each of the three tasks in Table 2. We follow the descriptions and setting of supervised pre-training and fine-tuning used in Agrawal's work [1], where pre-training with $M_{\text{D}}$ 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 the corresponding training set. For each CNN$_i$ for each of the three tasks in Table 2, we pre-train it with $M_{\text{ImageNet}}$ and fine-tune it with the training set of that task. After finishing training CNN$_i$, we form the cross-layer CNN features $F_i$ $(i = \{0, 1, \cdots, 5\})$ according to Table 1. With these cross-layer CNN feature vectors for training, we use support vector machine (SVM) to train a linear classifier supervised by the labels of the training images in the corresponding dataset. Specifically, one linear classifier is trained for each $F_i$ $(i = \{0, 1, \cdots, 5\})$ for each task (a total of 6 classifiers per task). 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 result in similar accuracy, so we just use the default value.

In testing phase, we use the given testing image as the input of the trained CNN$_i$ $(i = \{0, 1, \cdots, 5\})$ and generate $f_i$s. The cross-layer CNN features $F_i$ $(i = \{0, 1, \cdots, 5\})$ are formed by cascading the generated $f_i$s according to Table 1. After that, we feed each feature vector ($F_i$) of the testing image as the input of the corresponding trained SVM classifier, and the output of the SVM classifier is the predicted label of the testing image.

## 4. EXPERIMENTAL RESULTS

Using the training approach described in Sec. 3.2, we evaluate the performance of $F_i$ $(i = \{0, 1, \cdots, 5\})$ defined in Table 1 on the three tasks listed in Table 2. The experimental results are summarized in Table 3, where the numbers represent the classification accuracy (%) and the **bold** numbers represent the best performance for that task. We compare the performance of our proposed cross-layer CNN features ($F_1$ to $F_5$) with the following two baselines: 1: The current known best performance of that task provided by the references listed in Table 3. 2: The performance of $F_0$, which represents the commonly used features in the traditional CNN framework.

The results in Table 3 show that all of our proposed cross-layer CNN features ($F_1$ to $F_5$) outperform the two baselines on the three tasks, which supports our claim that utilizing the features extracted from multiple layers of CNN is better than using the traditional CNN features which are only extracted from the topmost fully connected layer. Furthermore, we find that the types of layers (either fully connected or convolutional layer) we remove from $\text{CNN}_0$ to form the sub-CNNs (and hence $f_i$ and $F_i$) do not influence the fact that the classification accuracy will increase as long as the features from multiple layers are considered simultaneously. Specifically, the cross-layer CNN features $F_1$ are formed by cascading the features from different combinations of the fully connected layers, but $F_2$ to $F_5$ are formed by cascading the features from different combinations of the convolutional layers. All of our proposed cross-layer CNN features ($F_1$ to $F_5$) outperform the two baselines on the three tasks because we explicitly utilize the features from multiple layers of CNN, not just the features extracted from the topmost fully connected layer.

Table 3 also shows that CNN-based features ($F_0$ to $F_5$) outperform the classical handcrafted features (for example, SIFT and HOG) used in the prior works [11, 12], which is consistent with the findings of the recent CNN-related literature [3, 4, 5, 8, 13]. In addition, our proposed cross-layer CNN features are generic features which are applicable to various tasks, not just the features specifically designed for certain tasks. As shown in Table 3, these cross-layer CNN features are effective in various domains from artistic style classification to architectural style classification, which makes our proposed cross-layer CNN features promising solutions for other tasks which future researchers are interested in.

| task ID | ARTIST-CLS | ART-CLS | ARC-CLS |
|---|---|---|---|
| prior work | 53.10 | 62.20 | 69.17 / 46.21 |
| reference | [11] | [11] | [12] |
| $F_0$ (baseline) | 55.15 | 67.37 | 70.64 / 54.84 |
| $F_1$ | 55.40 | 68.20 | **71.34 / 55.57** |
| $F_2$ | 56.25 | 68.29 | 70.94 / 55.44 |
| $F_3$ | **56.40** | 68.57 | 70.73 / 55.35 |
| $F_4$ | 56.35 | **69.21** | 70.68 / 55.32 |
| $F_5$ | 56.35 | **69.21** | 70.68 / 55.31 |

**Table 3**. The summary of our experimental results. The numbers represent the classification accuracy (%), and the **bold** numbers represent the best performance for each task. The results show that for all the three tasks, our proposed cross-layer CNN features ($F_1$ to $F_5$) outperform not only the best known results from prior works but also the features commonly used in the traditional CNN framework ($F_0$).

## 5. CONCLUSION

In this work, we mainly focus on the idea of utilizing the features extracted from multiple layers of convolutional neural networks (CNN). Based on this idea, we propose the cross-layer CNN features, showing their efficacy on artistic style, artist, and architectural style classification. Our proposed cross-layer CNN features outperform not only the state-of-the-art results of the three tasks but also the CNN features commonly used in the traditional CNN framework. Furthermore, as the first group advocating that we should leverage the features from multiple layers of CNN instead of using the features from only a single layer, we point out that our proposed cross-layer CNN features are promising generic features which can be applied to various tasks.

## 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] L. Kang, P. Ye, Y. Li, and D. Doermann, "A deep learning approach to document image quality assessment," in *ICIP*, 2014.

[10] A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *ICIP*, 2013.

[11] 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.

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