Multimed Tools Appl (2018) 77:29407–29434
https://doi.org/10.1007/s11042-018-5691-4

Fusion that matters: convolutional fusion networks for visual recognition

Yu Liu[1] · **Yanming Guo**[1] · **Theodoros Georgiou**[1] · **Michael S. Lew**[1]

Received: 30 April 2017 / Revised: 16 November 2017 / Accepted: 19 January 2018 / Published online: 27 February 2018

Abstract In recent years, deep learning has been successfully applied to diverse multimedia research areas, with the aim of learning powerful and informative representations for a variety of visual recognition tasks. In this work, we propose convolutional fusion networks (CFN) to integrate multi-level deep features and fuse a richer visual representation. Despite recent advances in deep fusion networks, they still have limitations due to expensive parameters and weak fusion modules. Instead, CFN uses 1 × 1 convolutional layers and global average pooling to generate side branches with few parameters, and employs a locally-connected fusion module, which can learn adaptive weights for different side branches and form a better fused feature. Specifically, we introduce three key components of the proposed CFN, and discuss its differences from other deep models. Moreover, we propose fully convolutional fusion networks (FCFN) that are an extension of CFN for pixel-level classification applied to several tasks, such as semantic segmentation and edge detection. Our experiments demonstrate that CFN (and FCFN) can achieve promising performance by consistent improvements for both image-level and pixel-level classification tasks, compared to a plain CNN. We release our codes on https://github.com/yuLiu24/CFN. Also, we make a live demo (goliath.liacs.nl) using a CFN model trained on the ImageNet dataset.

✉ Yu Liu
y.liu@liacs.leidenuniv.nl

Yanming Guo
y.guo@liacs.leidenuniv.nl

Theodoros Georgiou
t.k.georgiou@liacs.leidenuniv.nl

Michael S. Lew
m.s.lew@liacs.leidenuniv.nl

[1] Leiden Institute of Advanced Computer Science, Leiden University, Niels Bohrweg 1, Leiden,
The Netherlands

Keywords Deep fusion networks · Locally-connected fusion · Image-level recognition · Pixel-level recognition · Transferring deep features

1 Introduction

In the past few years, deep convolutional neural networks (CNNs) have been successfully applied in the multimedia community, with the aim of learning powerful visual representations for a variety of multimedia research areas, for example image recognition [21, 31, 53, 58], object detection [15, 16, 20, 48], image retrieval [6, 40, 46], face recognition [50, 56]. Additionally, more extensive efforts have been made to adapt CNN models to address the pixel-level classification problems, such as semantic segmentation [8, 42, 67], edge detection [38, 52, 62], and saliency detection [23, 32]. Up to date, CNNs based approaches are leading state-of-the-art performance on many frequently benchmarked datasets such as PASCAL VOC [14], ImageNet [49], MSCOCO [37], and Place [68].

An area of significant progress on CNN architectures has been increasing their depth to learn more powerful visual representations. In particular, the depth has increased from several layers (e.g., LeNet [10] and Alexnet [31]) to several tens of layers (e.g., VGGnet [53] and GoogLeNet [58]). Nevertheless, training a very deep network is extremely difficult because of vanishing gradients and degradation. To overcome this challenge, recent work in both Highway networks [55] and ResNet [21] proposes to add shortcut connections between neighboring convolutional layers, which are able to alleviate the vanishing gradient issue and ease the training stage. As a result, they promote the study on constructing much deeper neural networks (e.g. hundreds of layers) and exploring their potential bottleneck that may limit the learning capabilities. Extended variants [22, 57, 66] based on ResNet provide more insights towards delving into the residual learning mechanism. Moreover, other attempts have applied the residual learning mechanism to other vision tasks related to image classification [11, 29, 61]. Nevertheless, it is nontractable to optimize very deep neural networks due to their large amount of parameters and the amount of physical memory they require.

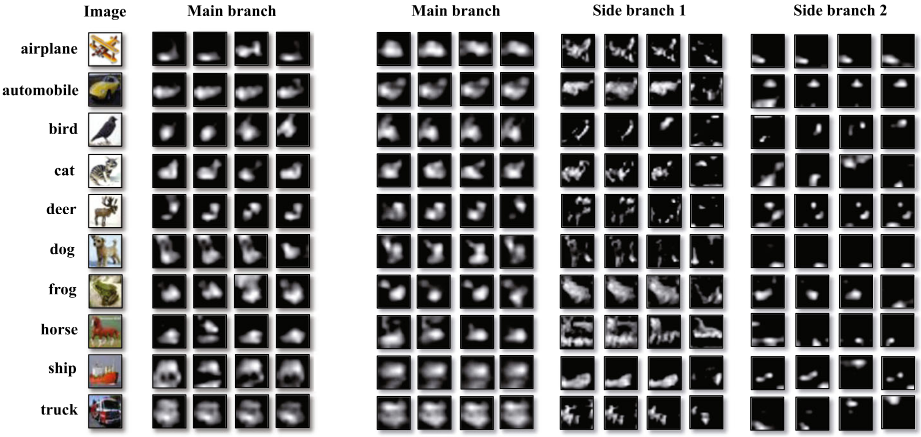
An alternative is to have greater integration in the existing intermediate layers in a deep neural network, rather than deepening the network with new layers. Commonly, the topmost activations in deep networks (i.e. fully-connected layers) often serve as discriminative visual representations to describe the image content. However, it is important to note that intermediate activations (i.e. convolutional layers) can also provide informative and complementary clues about images, including low-level textures, boundaries, and local parts. Therefore, researchers [1, 39, 64] have given greater attention to intermediate layers, and evaluated their contributions regarding image recognition performance. In addition, a large number of approaches [3, 39, 60, 65] have leveraged sophisticated encoding schemes (e.g., BoW, VLAD and Fisher Vector) to further encode intermediate feature activations. These approaches extract deep features from off-the-shelf CNNs without training new networks.

Driven by the significant results of the aforementioned methods, extensive research efforts [51, 63] have turned to explicitly training deep fusion networks where multi-level intermediate layers are fused together by adding new side branches. As a result, the deep fused representation allows us to integrate the strengths of individual layers and generate superior prediction. Although these deep fusion networks have achieved promising performance, they still require a large number of additional parameters required for generating the side branches. In addition, their fusion modules (e.g. sum-pooling) do not consider the importance of different side branches.

In this work, we propose convolutional fusion networks (CFN), which is a new fusion architecture to integrate intermediate layers with adaptive weights. To be specific, CFN mainly consists of three key components: (1) *Efficient side outputs*: we use efficient 1 × 1 convolution and global average pooling [36] to generate side branches from intermediate layers and as a result it has a small number of additional parameters. (2) *Early fusion and late prediction*: it can not only provide a richer representation, but also reduce the number of parameters, compared to the "early prediction and late fusion" strategy [63]. (3) *Locally-connected fusion*: we propose to adapt a locally-connected layer to act as a fusion module. It allows us to learn adaptive weights (a.k.a. importance) for different side outputs and generate a better fused representation. Figure 1 visually compares the feature activations learned in CNN and CFN, respectively. It can be seen that aggregating multi-level intermediate layers is essential to integrate their individual information.

Overall, our contributions in this paper can be summarized as follows:

- We propose a new fusion architecture (CFN) which can provide promising insights towards how to efficiently exploit and fuse multi-level features in deep neural networks. In particular, to the best of our knowledge, this is the first attempt to use a locally-connected layer as a fusion module.
- We introduce CFN models to address the image-level classification task. The results on the CIFAR and ImageNet datasets demonstrate that CFN can achieve promising improvements over the plain CNN. In addition, we transfer the trained CFN model to three new tasks, including scene recognition, fine-grained recognition and image retrieval. The results with the transferred CFN model have consistent performance improvements on these tasks.
- We further develop fully convolutional fusion networks (FCFN), which are able to perform pixel-level classification tasks such as semantic segmentation and edge detection.



(a) CNN activations (b) CFN activations

Fig. 1 Illustration of the features activations learned in (**a**) CNN and (**b**) CFN. The CIFAR-10 images are used here. Compared with CNN, CFN can learn complementary clues in the side branches to the full depth main branch. Specifically, the side branch 1 mainly learns the boundaries or shapes around objects, and the side branch 2 focuses on some semantic "parts" that fire strong near the objects

As a result, FCFN, as a fully convolutional extension, reveals the strong generalization capabilities of CFN for a variety of visual recognition tasks.

A preliminary version of this work appeared in [41]. This paper extends the earlier work [41] as follows. (1) We review more related work involved in taking advantage of intermediate layers in deep neural networks (Section 2). (2) We offer more details about constructing CFN and provide deeper analysis about its relationships with other deep models (Section 3.4). (3) An extended model–FCFN, is proposed to address pixel-level classification tasks (Section 4). Specifically, We build the FCFN models based on the representative fully convolutional networks (FCN) used for semantic segmentation and edge detection. (4) We conduct more comprehensive experiments to verify the effectiveness of CFN (Section 5). In addition, we give two promising directions towards improving CFN in future work.

The remainder of this paper is organized as follows. Section 2 summarizes recent works in terms of using intermediate features in deep neural networks. Section 3 introduces the details of constructing the proposed CFN for image-level classification problem. In addition, we compare and highlight the differences of CFN from other deep models. The FCFN model for pixel-level classification is described in Section 4. Section 5 presents experimental results that demonstrate the performance of CFN and FCFN on various visual recognition tasks. Also, we point out two promising directions in the future. Finally, Section 6 concludes this work.

2 Related work

In this section, we summarize related work that focuses on the use of intermediate convolutional layers in the following three aspects, and highlight our differences from them.

Employment of intermediate layers In recent decades, turning the focus of research to CNNs has been a leading and promising trend in the multimedia community. CNNs can explore high-level visual concepts in images by employing the top activations (i.e. fully-connected layers). However, intermediate convolutional layers can also provide informative and complementary image features. The approaches [3, 9, 39, 43, 60, 64, 65] are able to achieve competitive performance with the fully-connected layers, by taking advantage of intermediate layers. To obtain richer image representations, it is beneficial to extract dense CNN features from local patches or region proposals in one image. The local CNN features are used to construct a visual codebook, based on which an encoder scheme(e.g., BoW, VLAD and Fisher Vector) aggregates them and produces a more powerful deep representation. For example, Mohedano et al. [43] used the BoW model to encode the local convolutional activations, and Ng et al. [65] integrated the intermediate layers with the VLAD scheme. Similarly, Cimpoi et al. [9] and Wei et al. [60] made use of Fisher Vectors to encode intermediate activations. Moreover, Liu et al. [39] and Babenko et al. [3] aggregated several intermediate activations and generated a more discriminative and expensive image descriptor.

Intermediate supervision In a plain CNN model, a supervision signal, for instance the ground-truth categorial label for the input image, is only associated with the top-most prediction. As a result, the effects of the loss cost on the intermediate layers are implicit and weak. To guide the intermediate layers explicitly, some works attempt to impose additional supervision on the intermediate layers. DSN [34] was the first approach to use earlier and

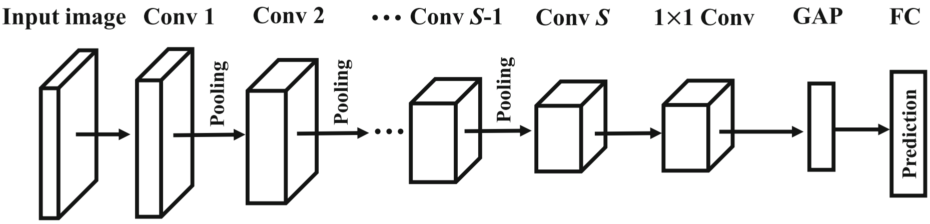


Fig. 2 The general pipeline of a plain CNN model. Note that one 1 × 1 convolutional layer and global average pooling are used on the top layers

more supervision signals, rather than the standard approach of only supervising the final prediction. Likewise, GoogLeNet [58] created two extra branches from the intermediate layers and supervised them jointly with the full-depth main branch. Sun et al. [56] built a robust face recognition network learned with multiple identification-verification supervisory signals. In summary, these approaches only aim to supervise intermediate layers better, but not to integrate their outputs as a fused feature.

Deep fusion networks In order to incorporate intermediate outputs explicitly, multi-layer fusion networks are presented in recent literature [42, 51, 62, 63]. Briefly speaking, they begin to create the side branches from the intermediate layers and then fuse their outputs together to make a more accurate prediction. A similar work in [63] built a DAG-CNNs model that summed up the multi-scale outputs from intermediate layers. However, DAG-CNNs required adding a large number of parameters, and its fusion module (i.e. sum-pooling) failed to consider the importance of side branches. Different from DAG-CNNs, the proposed CFN can learn adaptive weights for fusing side branches, while adding few parameters. Apart from image recognition, multi-layer fusion is also well-established for pixel-level classification. For example FCN [42] summed low-resolution, high layers with high-resolution, low layers for semantic segmentation. In contrast to simple sum computation, HED [62] employed a 1 × 1 convolution to learn shared weights. However, our FCFN is able to learn more adaptive weights than [42, 62].

3 Convolutional fusion networks

In this section, we introduce the details of building CFN and its training procedure. In addition, we compare its differences from other deep models.

3.1 Overview

First, we show a general architecture of a plain CNN model. As illustrated in Fig. 2, it mainly comprises of successive convolutional layers and pooling layers. In particular, a 1 × 1 convolutional layer and global average pooling is used in the last due to their efficiency, similar to [21, 36, 58]. Based on this plain CNN, we can develop the proposed CFN by adding new side branches from intermediate layers and aggregating them in a locally-connected fusion module. Figure 3 illustrates the architecture of the proposed CFN.

3.2 Architecture

Overall, CFN mainly consists of following three key components.

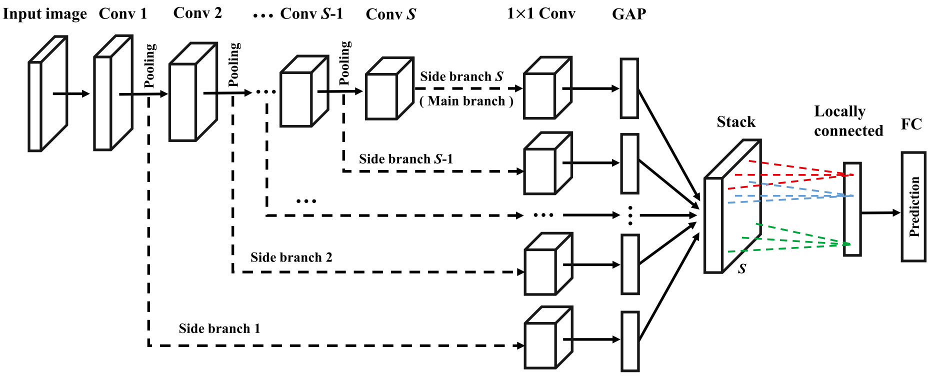


Fig. 3 The general pipeline of the proposed CFN. First, the side branches start from the pooling layers and consist of a 1 × 1 convolution layer and global average pooling. Then, all side outputs are stacked together. A locally-connected layer is used to learn adaptive weights for the side outputs (drawn in different color). Finally, the fused feature is fed to the following fully-connected layer that is able to make a better prediction. (best viewed in zoom in)

Efficient side outputs Prior work often added new fully-connected (FC) layers in the side branch [63], but this way may severely increase the number of parameters. Instead, CFN is able to efficiently create the side branches from the intermediate layers by adding few parameters. First, the side branches are grown from the pooling layers (see Fig. 3). These side branches use a 1 × 1 convolution like the main branch. All 1 × 1 convolutional layers must have the same number of channels so that they can be integrated together. Then, global average pooling (GAP) is performed over the 1 × 1 convolutional maps so as to obtain one-dimensional feature vector, called GAP feature. As a result, the side branches have the similar top layers (1 × 1 conv and GAP) as the full-depth main branch. One difference is that the 1 × 1 conv in the main branch follows a convolutional layer but not a pooling layer. For concise formulation, we consider the full-depth main branch as another side branch.

Assume that there are S side branches in total and the last side branch (i.e. S-th) indicates the main branch. We notate $h_{i,j}^{(s)}$ as the input of 1 × 1 convolution in the s-th side branch, where $s = 1, 2, \ldots, S$ and (i, j) is the spatial location over feature maps. As 1 × 1 convolution has K channels, its output associated with the k-th kernel is denoted as $f_{i,j,k}^{(s)}$, where $k = 1, \ldots, K$. Let $H^{(s)}$ and $W^{(s)}$ be the height and width of features maps derived from the s-th 1 × 1 convolution. Then, global average pooling performed over the feature map $f_k^{(s)}$ is calculated by

$$g_k^{(s)} = \frac{1}{H^{(s)}W^{(s)}} \sum_{i=1}^{H^{(s)}} \sum_{j=1}^{W^{(s)}} f_{i,j,k}^{(s)}, \qquad (1)$$

Where $g_k^{(s)}$ is the k-th element in the s-th GAP feature vector. Thus, we can notate $g^{(s)} = [g_1^{(s)}, \ldots, g_K^{(s)}]$, a 1 × K dimensional vector, as the whole GAP feature from the s-th side branch. Recall that $g^{(S)}$ represents the GAP feature from the full-depth main branch.

Early fusion and late prediction Considering how to incorporate the side branches, some work [42, 62, 63] used an "early prediction and late fusion" (EPLF) strategy. In Fig. 4a, EPLF computes a prediction from the GAP feature using a fully-connected layer and then fuses side predictions together to make the final prediction. In contrast to EPLF [63], in which a couple of FC layers are added, we present an opposite strategy called "early fusion and late prediction" (EFLP). EFLP first fuses the GAP features from the side

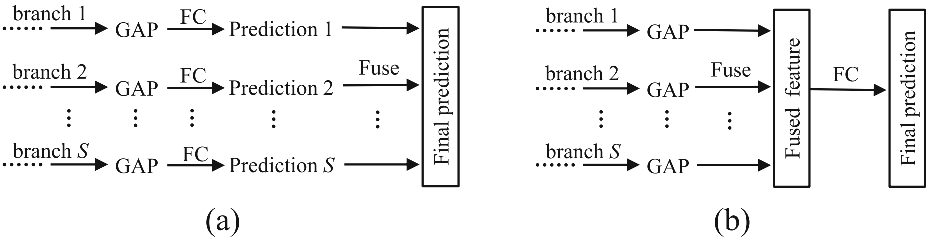


Fig. 4 Comparison between EPLF and EFLP. **a** The schematic pipeline of EPLF strategy; **b** The schematic pipeline of EFLP strategy

branches and obtains a fused feature. Then, a fully-connected layer following the fused feature is used to estimate the final prediction. As seen in Fig. 4b, EFLP has fewer parameters due to using only one fully-connected layer. Assume that each fully-connected layer has C units that correspond to the number of object categories. The fusion module has W_{fuse} parameters. Quantitatively, we can compare the number of parameters between EFLP and EPLF using the following formula:

$$W_{EFLP} = K(C+1) + W_{fuse} < W_{EPLF} = SK(C+1) + W_{fuse}. \tag{2}$$

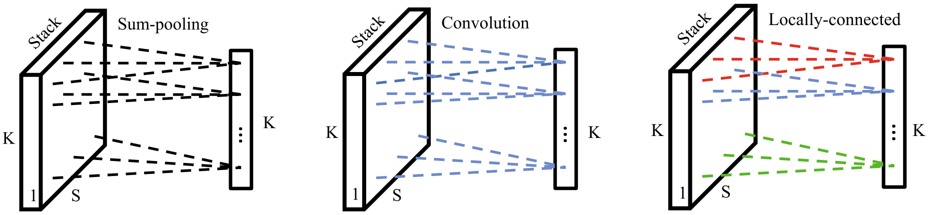
Hence, we make use of EFLP to fuse intermediate features earlier due to its efficiency. More importantly, the fused feature in EFLP is able to act as a richer image representation, compared with the widely-used fc6 and fc7 [31, 53]. In addition, the fused feature could be transferred to other vision domains due to its significant generalization capabilities. However, EPLF is less capable to generate such a rich image representation. We also observe that EFLP can achieve the same accuracy as EPLF, though EPLF contains more parameters.

Locally-connected fusion Another significant component in CFN is that it employs a locally-connected (LC) layer to fuse the side branches. Owing to its no-sharing filters over spatial dimensions, LC layer can learn different weights in each local field [18]. For example, DeepFace [56] used the LC filters to learn more discriminative face representations instead of spatially-sharing convolutional filters. Differently, our aim is to adapt a LC layer to learn adaptive weights (or importance) for different side branches, and generate a better fused feature. As we know, this is the first attempt to apply a locally-connected layer to a fusion module. Formally, its details are introduced as follows.

At first, we stack all GAP features together from $g^{(1)}$ to $g^{(S)}$, and form a layer G with size of $1 \times K \times S$, see Fig. 3. For example, the s-th feature map of G is $g^{(s)}$. Then, one LC layer which has K of no-sharing filters is convolved over G. Each filter has $1 \times 1 \times S$ kernel size. Since LC is able to learn adaptive weights for different elements in the GAP features which represent the importance of the side branches, it is able to produce a better fused feature. Finally, the fused feature convolved by LC also has $1 \times K$ shape, denoted as $g^{(f)}$. The i-th element in $g^{(f)}$ can be expressed by

$$g_i^{(f)} = \sigma\left(\sum_{j=1}^{S} W_{i,j}^{(f)} \cdot g_i^{(j)} + b_i^{(f)}\right), \tag{3}$$

where $i = 1, 2, \dots, K$; σ indicates the activation function (i.e. ReLU). $W_{i,j}^{(f)}$ and $b_i^{(f)}$ represent the weights and bias for fusing the i-th elements of GAP features that are from different



(a) No weights (b) Sharing weights (c) No-sharing weights

Fig. 5 Comparison of three fusion modules (best viewed in color). Left: Sum-pooling fusion has no weights; Middle: Convolution fusion learns sharing weights over spatial dimensions, as drawn in the same color; Right: Locally-connected fusion learns no-sharing weights over spatial dimensions, as drawn in different colors. To learn element-wise weights, we use 1 × 1 local field

side branches. The number of parameters in the LC fusion is $K \times (S + 1)$. Using these additional parameters benefits adaptive fusion while does not require any manual tuning.

To clearly demonstrate the advantage of the LC fusion module, Fig. 5 compares LC fusion with other fusion methods. In Fig. 5a, the sum-pooling fusion simply sums up the side outputs together without learning any weights, whereas this way treats each side branch equally and fails to consider their different importance. In Fig. 5b, the convolutional fusion can learn only one sharing filter over the whole spatial dimensions (as drawn with the same blue color). In contrast, LC enables the fusion module to learn independent weights over each local field (i.e. 1 ×1× S size) (drawn in different colors in Fig. 5c). Although LC fusion consumes more parameters than the sum-pooling fusion (no weights) and the convolutional fusion $(S + 1)$, these parameters are a negligible proportion of the total number of the network parameters (see Table 1).

3.3 Training

CFN has a similar training procedure as a standard CNN, including forward pass and backward propagation. Assume a training dataset which contains N images: $\{x^{(i)}, y^{(i)}\}$, where

Table 1 Two plain CNN models built for classification on CIFAR-10/100

CNN-A	CNN-B
Input 32 × 32 RGB image	
5 × 5 × 64 conv, ReLU	3 × 3 × 96 conv, ReLU
	3 × 3 × 96 conv, ReLU
3 × 3 max-pooling, stride 2. Dropout ratio 0.5	
5 × 5 × 64 conv, ReLU	3 × 3 × 192 conv, ReLU
	3 × 3 × 192 conv, ReLU
3 × 3 average-pooling, stride 2. Dropout ratio 0.5	
5 × 5 × 64 conv, ReLU	3 × 3 × 192 conv, ReLU
	3 × 3 × 192 conv, ReLU
1 × 1 × 192 conv, ReLU	
8 × 8 global average pooling. Dropout ratio 0.5	
10 or 100-way fully-connected layer	
Softmax classifier	

$x^{(i)}$ is the i-th input image and $y^{(i)}$ is its ground-truth class label. W indicates the set of all parameters learned in the CFN (including the LC fusion weights), Therefore, the objective function in the network is to minimize the total loss cost $\mathcal{L}$

$$\underset{W}{argmin}\ \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(f(x^{(i)};W),y^{(i)}), \tag{4}$$

where $f(x^{(i)};W)$ indicates the predicted class of $x^{(i)}$. We use the softmax loss function to compute the cost $\mathcal{L}$. To minimize the loss cost, the partial derivatives of the loss cost with respect to any weight are recursively computed by the chain rule during the backward propagation [10]. Since the main parts in the CFN model are the side branches, we will induce the detail computations of their partial derivatives. For notational simplicity, we independently consider each image in the following.

First, we compute the gradient of the loss cost with respect to the outputs of the side branches. As an example of the s-th side branch, the gradient of $\mathcal{L}$ with respect to the side output $g^{(s)}$ can be formulated as below

$$\frac{\partial\mathcal{L}}{\partial g^{(s)}}=\frac{\partial\mathcal{L}}{\partial g^{(f)}}\cdot\frac{\partial g^{(f)}}{\partial g^{(s)}}, s=1,2,\ldots,S. \tag{5}$$

Second, we formulate the gradient of $\mathcal{L}$ with respect to the inputs of the side branches. Let $a^{(s)}$ be the input of the s-th side branch. As depicted in Fig. 3, $a^{(s)}$ corresponds to the pooling layer. However, the input of the main branch, denoted as $a^{(S)}$, refers to the last convolutional layer (i.e. conv S). It is important to note that the gradient of $a^{(s)}$ depends on several side branches. To be more specific, the gradient of $a^{(1)}$ is influenced by S branches; the gradient of $a^{(2)}$ need to consider the gradient from the 2-th to S-th branch; but the gradient of $a^{(S)}$ is updated by only the main branch. Then, the gradient of $\mathcal{L}$ with respect to the side input $a^{(s)}$ can be computed via

$$\frac{\partial\mathcal{L}}{\partial a^{(s)}}=\sum_{i=s}^{S}\frac{\partial\mathcal{L}}{\partial g^{(i)}}\cdot\frac{\partial g^{(i)}}{\partial a^{(i)}}, \tag{6}$$

where i indexes the related branch that contributes to the gradient of $a^{(s)}$. It needs to sum up the gradients from several side branches. As is common practice, we employ a standard stochastic gradient descent (SGD) algorithm with mini-batch [31] to train the entire CFN model.

3.4 Discussion

To get more insights about CFN, we compare it with other deep models.

Relationship with CNN As CFN is built upon a plain CNN, we therefore compare their differences as described below. Normally, a plain CNN only estimates a final prediction based on the topmost layer; as a result, the effects of intermediate layers towards the prediction are implicit and indirect. In contrast, CFN connects the intermediate layers using additional side branches, and fuses them to jointly make the final predictions. In this way, CFN allows us to take advantage of intermediate layers explicitly and directly. This advantage explains why CFN is able to achieve more accurate prediction than a plain CNN.

Relationship with DSN DSN [34] is the first model to add extra supervision to intermediate layers for earlier guidance. As a result, it can improve the directness and transparency of learning a deep neural network. Therefore, we can view DSN as a “**loss fusion**” model. Instead, CFN still uses one supervision towards the final prediction derived from the fused representation, however it is able to increase the effects of the loss cost on the intermediate layers without adding more supervision signals. In a word, we clarify that CFN is a “**feature fusion**” model. It is important to note that there is no technical conflict between CFN and DSN, so it is promising to combine these two models together in the future.

Relationship with ResNet ResNet [21] addresses the vanishing gradient problem by adding densely “linear” shortcut connections. Due to being inspired by a distinct motivation, CFN has three main differences with ResNet: (1) The side branches in CFN are not shortcut connections. They start from pooling layers and merge into a fusion module together. (2) In contrast to adding a “linear” branch, we still use non-linear ReLU in side branches. (3) CFN employs a sophisticated fusion module to generate a richer feature, rather than using the simple summation as in ResNet. As mentioned in the ResNet work, when the network is not overly deep (e.g., 11 or 18 layers), ResNet may obtain few improvements over a plain CNN. However, CFN can obtain some considerable gains over CNN. Hence, CFN can serve as an alternative for improving the discriminative capabilities of not-very-deep models, instead of purely increasing the depth. In summary, ResNet tells us that “depth that matters”, but CFN concludes that “fusion that matters”.

4 Fully convolutional fusion networks

Deep neural networks allow us to narrow the gap between different vision tasks. More specifically, CNN models for image-level classification can be well-adapted to other pixel-level classification tasks which aim to generate a per-pixel prediction in images. As a common practice, it is essential to cast traditional convolutional neural networks to their corresponding fully convolutional networks (FCNs) by replacing the fully-connected layers with more convolutional layers. FCNs are able to infer any size of images without requiring specific input dimensionality. In this section, we introduce fully convolutional fusion networks (FCFN), which are used for two representative pixel-level classification tasks: semantic segmentation and edge detection. Based on the FCN models, FCFN models are built to learn better pixel representations from a fusion module.

4.1 Semantic segmentation

Semantic segmentation intends to predict a category label for spatial pixels in an image. The FCN-8s [42] model was the first attempt to apply CNNs to semantic segmentation, and yielded significant improvements in comparison with non-deep-learning approaches. First, FCN-8s is fine-tuned from the VGG-16 model [53] pre-trained on the ImageNet dataset [49]. Then, it adds two side branches with the full-depth main branch; as a result, it allows to integrate both coarse-level and fine-level pixel predictions together that benefit to improve the semantic segmentation performance. Particularly, FCN-8s uses a simple sum-pooling to fuse the multi-level predictions. Inspired by FCN-8s, we extend the above CFN model and build the FCFN for generating fused pixel features. From FCN-8s to CFN-8s, we use two locally-connected layers in a two-stage fusion fashion, as illustrated in Fig. 6.

Recall that the LC fusion module is able to learn independent weights for each spatial pixel in an image. We need extend the formulations in Section 3.2 to be suitable for the LC fusion module in FCFN. In the first fusion module, two branches which have both

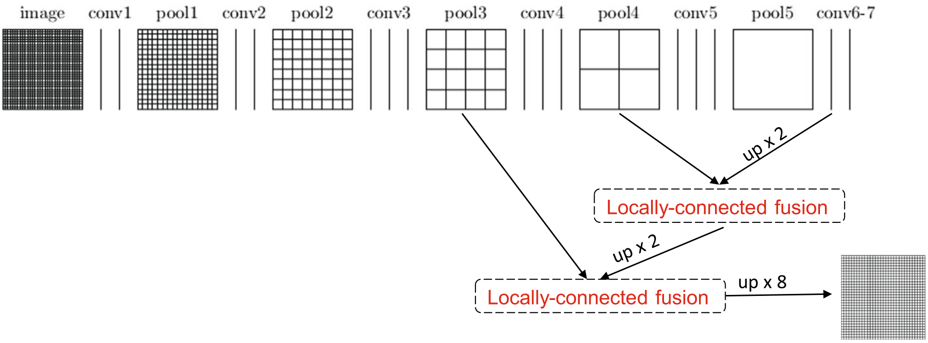


Fig. 6 Illustration of the proposed CFN-8s model semantic segmentation, which is built upon the baseline FCN-8s [42]. Two locally-connected layers are used in a two-stage fusion fashion

K channels of feature maps are taken as input. Note that the top layers are unsampled 2 times to retain the same spatial dimensions as the bottom layers. We consider the adaptive weights of each channel separately and reshape each two-dimensional feature map to a one-dimensional feature vector. For example, $g_{k,i}^{(1)}$ indicates the feature activation of the i-th pixel of the k-th channel in the first branch, and $g_{k,i}^{(2)}$ is the corresponding activation in the second branch, where $i = 1, \dots, H \times W$ and $k = 1, \dots, K$. Therefore, the fused pixel feature can be computed by

$$g_{k,i}^{(f)} = \sigma\left(\sum_{j=1}^{2} W_{k,i,j}^{(f)} \cdot g_{k,i}^{(j)} + b_{k,i}^{(f)}\right), \tag{7}$$

The parameters in this fusion module are $H \times W \times 21 \times 2$.

Moreover, the second fusion module integrates coarser feature maps with the output of the first fusion module. Let $g_{k,i}^{'(1)}$ be the activation in the coarser layer. For notational simplicity, the activation $g_{k,i}^{(f)}$ from the output of the first fusion module, is renamed to $g_{k,i}^{'(2)}$. Likewise, the computation in the second LC fusion is expressed via

$$g_{k,i}^{'(f)} = \sigma\left(\sum_{j=1}^{2} W_{k,i,j}^{'(f)} \cdot g_{k,i}^{'(j)} + b_{k,i}^{'(f)}\right), \tag{8}$$

where $g_{k,i}^{'(f)}$ represents the final fused feature by using the two-stage fusion approach.

Considering the computation of the loss cost with respect to the ground-truth, we still employ the softmax loss function and accumulate the loss of all pixels together. For brevity, we give the loss computation for one image here. It is easy to extend to a mini-batch size of images.

$$\mathcal{L} = -\sum_{i=1}^{H \times W} \sum_{k=1}^{K} \mathbf{h}(y_i = k) \log p_{k,i}, \tag{9}$$

where y_i is the ground-truth pixel label. $\mathbf{h}(i = j)$ is the Kronecker delta response. The predicted pixel probability is normalized with the softmax function, where $p_{k,i} = \frac{\exp(g_{k,i}^{'(f)})}{\sum_{k=1}^{K} \exp(g_{k,i}^{'(f)})}$. As introduced in Section 3.3, we still use the SGD with mini-batch to train the entire FCFN model.

4.2 Edge detection

The problem of edge detection is to extract semantically meaningful edges in images, Typically, edge detection acts as a low-level task, but has significant contributions to other high-level visual tasks, such as object detection and image segmentation. Driven by the increasing developments of deep learning, edge features have moved from carefully-engineered descriptors such as Canny [5], gPb [2] and Structured Edges (SE) [13]), to discriminative deep features [4, 52, 62]. In particular, HED [62] is the first work to use FCNs for end-to-end edge detection, and leads state-of-the-art performance on well-known benchmarks. HED integrates the strengths of high efficiency from end-to-end FCNs [42] and using additional deep supervision as in DSN [34].

In this section, we develop a FCFN model for edge detection based on HED. Figure 7 shows the overview architecture of FCFN. In contrast to HED, which uses a convolutional fusion module, FCFN fuses five intermediate side branches with a locally-connected layer. To be specific, one side branch generates a feature map where the activations measure the probabilities of pixels being edges. Five feature maps from side branches stack together, and are reshaped from $(H, W, 5)$ to $(H \times W, 5)$. Similarly, we can compute the fused prediction $g_i^{(f)}$ $(i = 1, \dots, H \times W)$ by

$$g_i^{(f)} = \sigma\left(\sum_{j=1}^{5} W_{i,j}^{(f)} \cdot g_i^{(j)} + b_i^{(f)}\right), \tag{10}$$

Then, the loss cost based on the sigmoid cross-entropy function is expressed via

$$\mathcal{L}_{fuse} = -\sum_{i=1}^{H\times W}\left[\beta_i \log g_i^{(f)} + (1-\beta_i)\log(1-g_i^{(f)})\right], \tag{11}$$

where the parameter β_i regulates the importance of edge and non-edge pixels, as mentioned in [62]. It is important to note that we also impose the intermediate supervision on the side branches similar to [34, 62], to discard the negative edges in the earlier intermediate layers. The loss cost in the k-th side branch (i.e. $k = 1, \dots, 5$) is represented by the following formala

$$\mathcal{L}_{side}^{(k)} = -\sum_{i=1}^{H\times W}\left[\beta_i \log g_i^{(k)} + (1-\beta_i)\log(1-g_i^{(k)})\right], \tag{12}$$

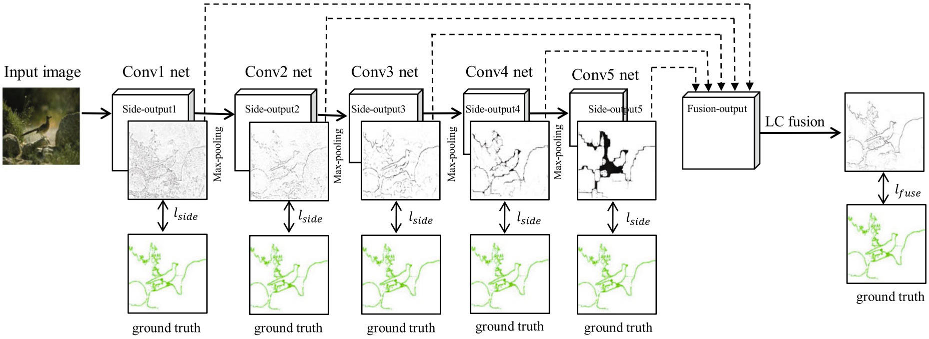


Fig. 7 Illustration of the proposed FCFN model built for edge detection, which is bulit upon the baseline HED [62]. Five side branches are fused together in a LC fusion module

where $g_i^{(k)}$ accounts for the predicted probability of the i-th pixel being an edge point. Finally, the overall loss cost in FCFN integrates a fused loss term and five intermediate loss terms together:

$$\mathcal{L} = \sum_{k=1}^{5} \mathcal{L}_{side}^{(k)} + \mathcal{L}_{fuse}. \tag{13}$$

This edge detection network is also fine-tuned from the VGG-16 model and updated with the SGD algorithm with mini-batch end-to-end.

5 Experiments

This experimental section evaluates the performance of the proposed CFN for image-level classification and FCFN for pixel-level classification. First, we trained the CFN models on the CIFAR-10/100 [30] and ImageNet 2012 [49]. Then, we transferred the trained CFN model to three new tasks, including scene recognition, fine-grained recognition and image retrieval. Moreover, we trained the specific FCFN models for semantic segmentation on the PASCAL dataset [14], and edge detection on the BSDS dataset [2], respectively. All experiments were conducted using the Caffe library [27] on a NVIDIA TITAN X card with 12 GB memory.

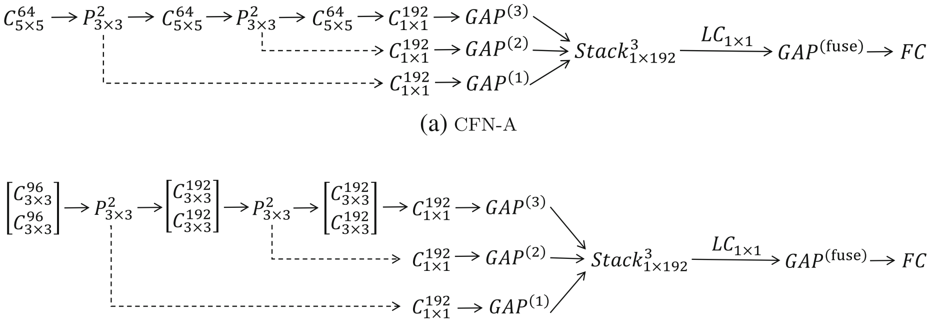
5.1 CIFAR dataset

Both CIFAR-10 [30] and CIFAR-100 [30] consist of 50,000 training images and 10,000 testing images. They define 10 and 100 object categories, respectively. We preprocessed their RGB images by global contrast normalization [17], and randomly shuffle the training set.

Models We employ two plain CNNs as baselines and build their CFN counterparts. Table 1 describes the two CNN models used for CIFAR-10/100, called CNN-A and CNN-B. (1) CNN-A is a shallow network similar to the Caffe-Quick model [27]. It has three 5 × 5 convolutions and a 1 × 1 convolution. The global average pooling is performed over the 1 × 1 convolutional maps. Finally, a fully-connected layer with 10 or 100 units is used to predict object categories; (2) CNN-B replaces the 5 × 5 convolution in CNN-A by two 3 × 3 convolutional layers. This is inspired by the advantageous observation in VGGnet [53]. In addition, CNN-B utilizes more feature channels than CNN-A. Note that, when training the CNN-B model on the CIFAR-100 dataset, the first and second convolutional layer also use 192 channels instead of 96 channels. Correspondingly, the CFN-A and CFN-B models are built upon CNN-A and CNN-B respectively, by constructing two additional side branches after the pooling layers, as depicted in Fig. 8a and b.

We use the same hyper-parameters to train CNN and CFN, for example, a weight decay of 0.0001, a momentum of 0.9, and a mini-batch size of 100. The learning rate is initialized with 0.1 and is divided by 10 after 10×10^4 iterations. The whole training will be terminated after 12×10^4 iterations. As for CFN, the initialized weights in the LC fusion module are set to 0.333, as there are three side branches in total (including the full-depth main branch).

Results Table 2 shows the results on CIFAR-10/100 test sets. We can analyze the results from the following three aspects:



(a) CFN-A

(b) CFN-B

Fig. 8 Illustration of CFN built for CIFAR dataset. For the convolutional layers (denoted as C), the right lower numbers indicate the kernel size; the right upper number indicates the number of channels. For the pooling layers (denoted as P), the right lower numbers indicate the window size; the right upper number is the size of strides

(1) CFN achieves about 1% improvements on the performance compared to the plain CNNs (both CNN-A and CNN-B). For example on the CIFAR-10 dataset, CFN-A and CFN-B obtains 14.73 and 8.27 error rates that are better than the results of CNN-A and CNN-B (15.57 and 9.28). The comparison between CFN and CNN demonstrates the effectiveness of fusing multi-level intermediate layers. Additionally, CFN is able to improve the expressive capabilities of deep neural networks, to learn superior visual representation.

(2) In order to analyze the advantage of using the LC fusion, we implement the sum-pooling fusion and convolutional fusion that are denoted as CNN-Sum and CNN-Conv. By comparing CFN with either CNN-Sum or CNN-Conv, it can be seen that the LC fusion outperforms the other two fusion ways by a considerable margin. Hence, learning adaptive weights is essential to generate a better fused feature.

(3) Moreover, we compute the number of parameters in the models to estimate their efficiency. In the second column of the table, the additional parameters for extra side branches and the LC fusion are significantly fewer than the number of basic parameters in the models. Although the LC fusion consumes more parameters compared to the sum-pooling fusion and convolutional fusion, these parameters have minimal increase to the network complexity. Furthermore, we further compare the training time

Table 2 Test error (%) on CIFAR-10/100 dataset (without data augmentation)

Model	#parameters	CIFAR-10	CIFAR-100
CNN-A	0.224M (basic)	15.57	40.62
CNN-Sum-A	0.224M (basic) + 0.025M (extra branches) + 0 (fusion)	15.33	40.32
CNN-Conv-A	0.224M (basic) + 0.025M (extra branches) + 4 (fusion)	15.19	40.15
CFN-A	0.224M (basic) + 0.025M (extra branches) + 768 (fusion)	**14.73**	**39.54**
CNN-B	1.287M (basic)	9.28	31.89
CNN-Sum-B	1.287M + 0.074M (extra branches) + 0 (fusion)	8.84	31.42
CNN-Conv-B	1.287M + 0.074M (extra branches) + 4 (fusion)	8.68	31.16
CFN-B	1.287M + 0.074M (extra branches) + 768 (fusion)	**8.27**	**30.68**

For both Model A and B, the proposed CFN can achieve better results shown in bold face

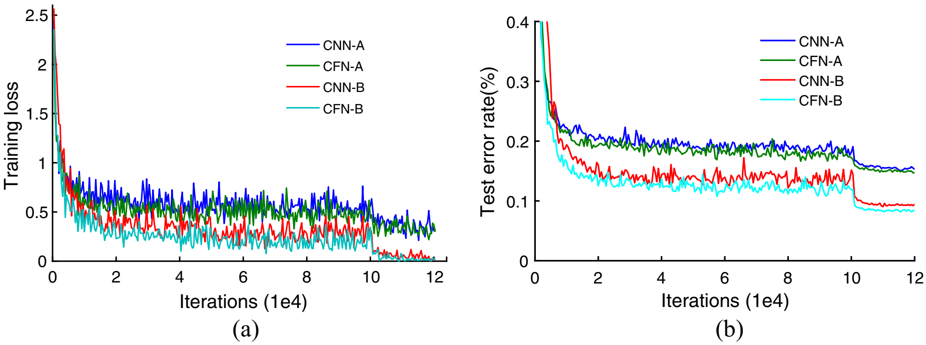


Fig. 9 Comparison between CFN and CNN on the CIFAR-10 dataset. **a** The training loss when training CFN and CNN. **b** The test error rates along with the increasing iterations

between CNN and CFN. For example on the CIFAR-10 dataset, CNN-B and CFN-B take about 1.67 and 2.08 h, respectively.

Figure 9 shows the training loss and the test accuracy while training CFN and CNN. Along with the training iterations, both CFN-A and CFN-B models have less training loss and lower test error rates than the corresponding CNN models. In addition, Fig. 10a presents the adaptive weights learned in the LC fusion of CFN-B. Recall that LC learns 192 filters (each filter is 1 × 3 size) and each filter has 1 × 3 weights. We compute the average weight in each branch, and estimate its fluctuation. As we can see in the figure, the side branch 3 (a.k.a. the full-depth main branch) plays a core role, while the other two side branches are complementary to the main branch. After a large amount of training iterations, the adaptive weights tend to be stable. Moreover, in Fig. 1, we visualize and compare the learned feature maps in CNN-B and CFN-B. We select ten images from the CIFAR-10 dataset. The feature maps in the 1 × 1 convolutional layer of three side branches are extracted. We rank the feature maps by averaging spatial activations and select the top-4 maps to visualize. One can see that CFN can learn improtantly complementary clues in the side branches, while retaining the necessary information in the main branch.

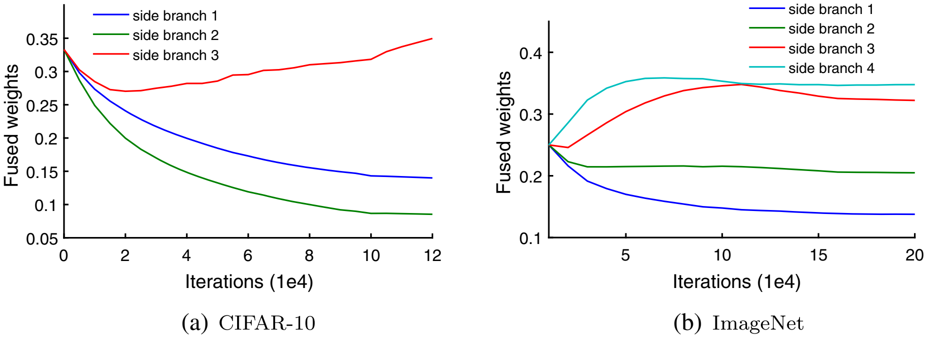


Fig. 10 Illustration of adaptive weights of the side branches learned in the LC fusion. All side branches are initialized with the same weights before training. During the training stage, the top branches have larger weights than the bottom branches

Table 3 Test error on CIFAR-10/100 to compare CFN-B with recent state-of-the-art

Method	#Layers	CIFAR-10	CIFAR-10*	CIFAR-100
Maxout Networks [17]	5	11.68%	9.38%	38.57%
NIN [36]	9	10.41%	8.81%	35.68%
DSN [34]	9	9.69%	7.97%	34.54%
ALL-CNN [54]	9	9.08%	7.25%	33.71%
RCNN-160 [35]	6	8.69%	7.09%	31.75%
NIN + SReLU [28]	9	8.41%	6.98%	31.10%
CNN (baseline)	8	9.28%	7.34%	31.89%
CFN (ours)	8	**8.27%**	**6.77%**	**30.68%**

A superscripted * indicates to use the standard data augmentation [34]

CFN can achieve superior results which are bolded

Comparison with other approaches Table 3 reports recent results on CIFAR datasets. For fair comparison, we compare CFN-B with other not-very-deep models. Notably, ”not-very-deep” is a relative concept. We use it to emphasis the differences between the models in Table 3 and other ResNet-like models. In Table 3, our method (CFN) and the compared methods develop less than 10 layers' models to evaluate their effectiveness. These models certainly belong to deep neural networks, however, they are not very deep, compared to the ResNets that mostly have hundreds of layers on the datasets like CIFAR-10/100. Also, we show the depth of these models for clear comparison and analysis. In summary, CFN obtains comparative results and outperforms these compared methods. In this work, we aim to investigate the potential of integrating multiple intermediate layers, and the results in Table 3 verify the effectiveness of CFN. Also, building CFN on top of a much deeper model (e.g. ResNet [21]) will be our main future work.

5.2 ImageNet 2012

The ImageNet 2012 dataset [49] consists of 1.2 million training images, 50,000 validation images and 100,000 test images. We developed a basic 11-layer plain CNN (called CNN-11) where the channels of convolutional layers range from 64 to 1024. This baseline model is inspired by prior widely-used deep models [21, 53, 58]. Based on this CNN, we built its CFN counterpart (called CFN-11) as illustrated in Fig. 11. Note that we create three extra side branches from the intermediate pooling layers (excluding the first pooling layer).

The training setup in our implementation follows the empirical practice in existing literature [21, 31, 53, 58]. To be more specific, the original image is resized to 256 × 256. In training phase, a 224 × 224 crop is randomly sampled from the resized image or its flip. The cropped input image is subtracted with per-pixel mean. We initialize the weights and

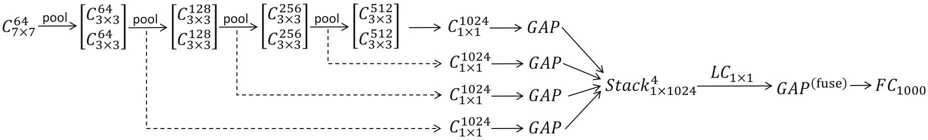


Fig. 11 Overview of the CFN-11 architecture built on top of CNN-11. Three additional side branches are generated from the pooling layers, and fused together with the full-depth main branch (a.k.a the last side branch)

Table 4 Error rates (%) on the ImageNet 2012 validation set

Method	AlexNet	CNN-11	DSN-11	ResNet-11	CFN-11	CNN-19	CFN-19
Top-1	42.90	43.11	42.24	43.02	**41.96**	36.99	**35.47**
Top-5	19.80	19.91	19.24	19.85	**19.09**	14.74	**13.93**

Best results are in bold face

biases following GoogLeNet [58], for example a weight decay of 0.0001, and a momentum of 0.9. Batch normalization (BN) [24] is added after one convolutional layer. The learning rate starts from 0.01 and decreases to 0.001 at 10×10^4 iterations, and to 0.0001 at 15×10^4 iterations. The whole training will be terminated after 20×10^4 iterations. Notably, LC weights in the fusion module are initialized with 0.25 due to four side branches in total. We use SGD to optimize the models with a mini-batch size of 64.

Results Table 4 compares the results on the validation set. The following will analyze these results from several aspects.

(1) CNN-11 is able to achieve competitive results as compared to AlexNet [31], however, it consumes much fewer parameters (∼6.3 millions) than Alexnet (∼60 millions). This results from replacing the fully-connected layers with global average pooling.

(2) CFN-11 obtains about 1% improvement over CNN-11, while adding few parameters (∼0.5 millions). It verifies its consistent improvements on the performance for a large-scale dataset. Moreover, for fair comparison with other deep models, we implement the DSN-11 and ResNet-11 based on the plain CNN-11, which are shown in Figs. 12 and 13, respectively. It can be seen that, CFN-11 can still achieve better accuracy than DSN-11 and ResNet-11. Therefore, we can view CFN as an alternative to promote the discriminative capacity of such a not-overly deep CNN model, rather than increasing the depth as in ResNet. Notably, CFN-11 can improve CNN-11, but ResNet-11 cannot. But it does not conclude that CFN may be better than ResNet, as the networks are not very deep. Our primary purpose is to evaluate the superiority of CFN over CNN, and the results are consistent with our motivation.

(3) To test the generalization of CFN to deeper networks, we build a 19-layer model following the similar principle of the 11-layer model. Likewise, CFN-19 outperforms CNN-19 with about 1% gains for both top-1 and top5 performance. For simplicity, we did not use the same implementation in ResNet [21], such as scale augmentation, large mini-batch size, multi-scale test. Therefore, our results of CNN-19 and CFN-19

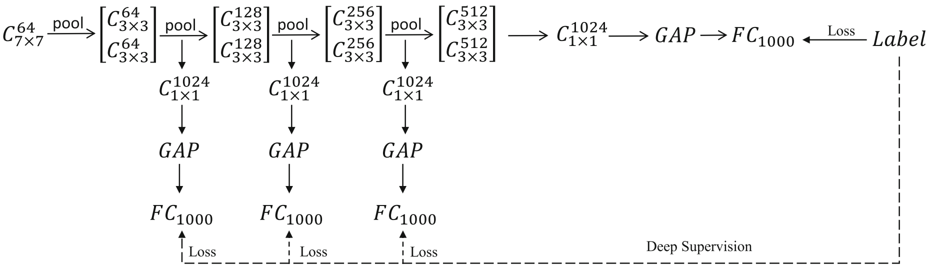

$C^{64}_{7\times7}$
pool
$C^{64}_{3\times3}$ $C^{64}_{3\times3}$
$C^{128}_{3\times3}$ $C^{128}_{3\times3}$
$C^{256}_{3\times3}$ $C^{256}_{3\times3}$
$C^{512}_{3\times3}$ $C^{512}_{3\times3}$
$C^{1024}_{1\times1}$
GAP
FC_{1000}
Loss
$Label$
Deep Supervision

Fig. 12 Overview of the DSN-11 architecture built on top of CNN-11. DSN-11 creates three side branches that can provide intermediate predictions for the input image. The ground-truth label is also used to guide these intermediate predictions, to enhance the discriminative abilities of hidden layers

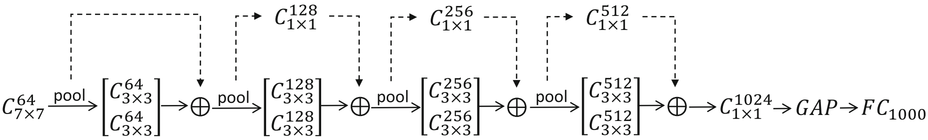


Fig. 13 Overview of the ResNet-11 architecture built on top of CNN-11. There are four residual connections in total. Due to inconsistent numbers of channels, 1x1 convolution layers are needed in the residual connections, but they are not followed by ReLU to make sure linear transformation

are not as high as CNN-18 and ResNet-18 in [21]. We believe that our results can raise awareness of the potential of building deep multi-layer fusion networks. If needed, in the future we will develop 50 or 100 layers of networks to test the effectiveness of CFN, apart from 11 and 19 layers.

Similar to CIFAR-10, Fig. 10b illustrates the adaptive weights learned in the LC fusion of CFN-11. It is important to note that, the top branches (i.e. 3 and 4) have larger weights than the bottom branches (i.e. 1 and 2). Additionally, we extract the feature activations of the 1 × 1 convolutional layer in one side branch. Figure 14 show and compare the feature maps from different side branches.

5.3 Transferring deep fused features

To evaluate the generalization of CFN, we transferred the trained ImageNet model (i.e. CFN-11) to three new tasks: scene recognition, fine-grained recognition and image retrieval. Each task is evaluated on two widelly-used datasets: Scene-15 [33] and Indoor-67 [47], Flower [44] and Bird [59], and Holidays [26] and UKB [45]. The configurations of these six datasets are summarized in Table 5. Also, image examples are shown in Fig. 15.

The AlexNet [31] acts as a baseline that uses the fc7 layer (4096-Dim) to provide an image representation. For CNN-11, we use the output of the global average pooling (1024-Dim) as image feature. Notably, CFN-11 allows us to utilize a fused feature (1024-Dim) that integrates multiple intermediate layers. For scene and fine-grained recognition, linear

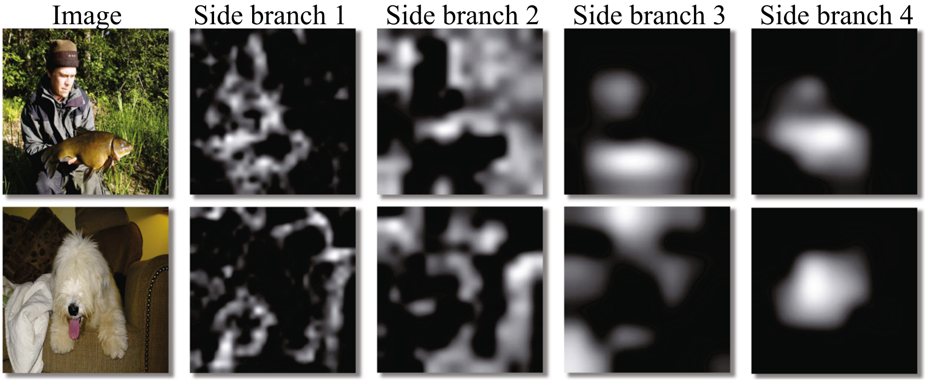


Fig. 14 Illustration of feature maps in the four side branches. On one hand, the side branch 1 and 2 can capture some low-level clues about images, such as boundaries and textures. On the other hand, side branch 3 and 4 aim to obtain more abstract features that fire strong around objects. Therefore, CFN can incorporate multi-layer intermediate features explicitly and adaptively so as to improve visual representation

Table 5 Configurations of six datasets for scene recognition, fine-grained recognition and image retrieval

	Scene 15	Indoor 67	Flower	Bird	Holidays	UKB
#categories	15	67	102	200	–	–
#train images	1500	5360	2040	5994	991	10200
#test images	2985	1340	6149	5794	500	10200

SVM [7] is trained to compute the classification accuracy. For image retrieval, we use KNN to compute the mAP on Holidays and the N-S score on UKB.

Results Table 6 reports the transfer learning results on six datasets. Although it is challenging to generalize a deep model to diverse visual tasks, we can still summarize these results to provide more insights about CFN.

(1) Overall, CFN-11 obtains consistent improvements for the three tasks on all datasets, compared with the baseline CNN-11. In addition, CFN-11 outperforms the Alexnet while using a much lower dimensional feature vector. These results reveal that learning fused deep representations is beneficial for not only image classification, but also a variety of visual tasks, even though the images in these tasks have large differences.

(2) Notably, the improvements on these three tasks are more significant than those on the ImageNet itself. In particular, CFN-11 yields about 6% accuracy gains on the Flower dataset for fine-grained recognition. On other datasets, about 2% accuracy gains are obtained as well (Note that the UKB uses the N-S score which is different from precision accuracy). We believe that fine-tuning the models on the target datasets will further improve the results.

5.4 Semantic segmentation

We conduct the semantic segmentation experiment on the PASCAL VOC 2012 segmentation dataset [14] that consists of 20 foreground object classes and a background class. The original dataset contains 1464 training images, 1449 validation images, and 1456 test images. When evaluating the val set, we use a merged training dataset with the original training images and the augmented training images as in [19]. Since there are validation

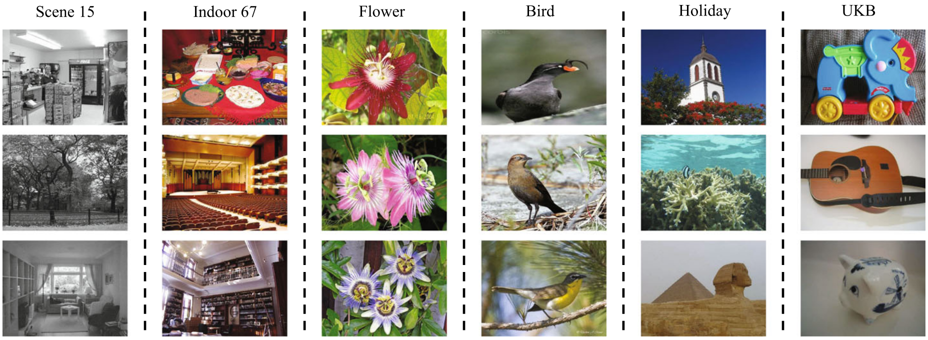


Fig. 15 Image examples from six datasets about scene recognition, fine-grained recognition and image retrieval. We can see their significant differences with respect to the image content

Table 6 Results on transferring the ImageNet model to three target tasks

Method	Dim	Scene recognition		Fine-grained recognition		Image retrieval	
		Scene 15	Indoor 67	Flower	Bird	Holidays	UKB
AlexNet [31]	4096	83.99	58.28	78.68	45.79	76.77	3.45
CNN-11	1024	84.32	60.45	76.79	45.98	78.33	3.47
CFN-11	1024	**86.83**	**62.24**	**82.57**	**48.12**	**80.32**	**3.54**

For each dataset, better results are in bold face

images included in the merged training set, we need to pick the non-intersecting set of 904 images [42] as a new validation set.

For fair comparison, we used the same parameters to train both the baseline FCN-8s [42] and our proposed FCFN, including a fixed learning rate of 10^{-4}, a weight decay of 0.0001, a momentum of 0.9, and a mini-batch size of 1. The training stage will be terminated after 100K iterations. It is worth mentioning that, we fine-tune FCN-8s directly from the VGG-16 model, without pre-training FCN-32s and FCN-16s. FCFN undergoes the same training procedure. The segmentation performance is measured in terms of pixel intersection-over-union (IoU).

Results Table 7 reports the mean IoU accuracy and the detailed results of 20 object classes. The proposed FCFN achieves 1.6% gains on the mean IoU performance compared to the

Table 7 Semantic segmentation results (IoU accuracy) on the PASCAL VOC 2012 val set. For the 20 object classes, better results are in bold

	FCN-8s [42]	FCFN
aero	**75.5**	75.2
bike	**34.5**	33.8
bird	69.5	**72.0**
boat	**56.7**	53.3
bottle	59.7	**63.8**
bus	68.7	**71.2**
car	**70.3**	69.2
cat	73.4	**75.0**
chair	23.8	**24.0**
cow	53.0	**63.4**
table	39.7	**40.7**
dog	63.3	**65.6**
horse	46.3	**57.6**
mbike	**75.2**	74.5
person	73.9	**75.4**
plant	**42.2**	40.2
sheep	59.7	**62.3**
sofa	27.0	**30.3**
train	73.4	**74.0**
tv	**58.7**	55.7
mean	**57.1**	**60.3**

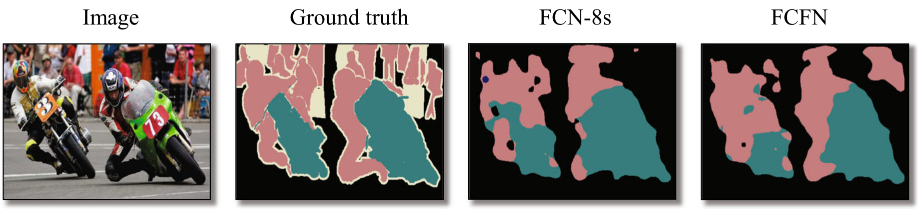


Fig. 16 Comparison of a semantic segmentation example between the baseline FCN-8s and the proposed FCFN

baseline FCN-8s. In addition, more object classes obtain superior results based on FCFN, rather than FCN-8s. Figure 16 shows a visual example to highlight the segmentation details between the two models. We clarify that FCFN is a fundamental architecture that can be integrated with other sophisticated techniques such as CRF [8] and Recurrent Neural Networks (RNN) [67], in order to further recover the segmentation details.

5.5 Edge detection

We evaluate the edge detection performance on the BSDS500 dataset [2] that consists of 200 training, 100 validation, and 200 testing images. One image is manually annotated by five human annotators on average. The validation set is used to fine-tune the hyperparameters, similar to HED [62]. For example, we use a momentum of 0.9 and a weight decay of 0.0002. Also, the initialization of theside-output filters is set to 0, and the initialization of the LC fusion filter is set to 0.2 due to fusing five side branches in total. The training images are resized to 400 × 400 and the batch size is 8. The learning rate is initialized with 1e-6, and the training is terminated after 25 epoches. The performance measurements for edge detection include fixed contour threshold (ODS), per-image best threshold (OIS) and average precision (AP).

Results Table 8 provides a comparison of edge detection results on the BSDS dataset. First, we can see that deep learning approaches (in the lower group) largely push the state-of-the-art performance compared to the hand-crafted edge detection approaches (in the upper group). In addition, the proposed FCFN outperforms the baseline HED with considerable improvements. This shows the advantage of learning adaptive weights in the locally-connected fusion module. Moreover, we illustrate the precision/recall curves of

Table 8 Edge detection results on the BSDS dataset. The upper group lists some representative approaches without using deep learning. The lower group gives the deep learning based approaches

Method	ODS	OIS	AP
Canny [5]	.600	.630	.580
gPb-owt-ucm [2]	.726	.757	.696
SE-Var [13]	.746	.767	.803
DeepEdge [4]	.753	.772	.807
DeepContour [52]	.757	.776	.790
HED [62]	0.780	0.802	0.786
FCFN	**0.784**	**0.806**	**0.788**

Best results are in bold face

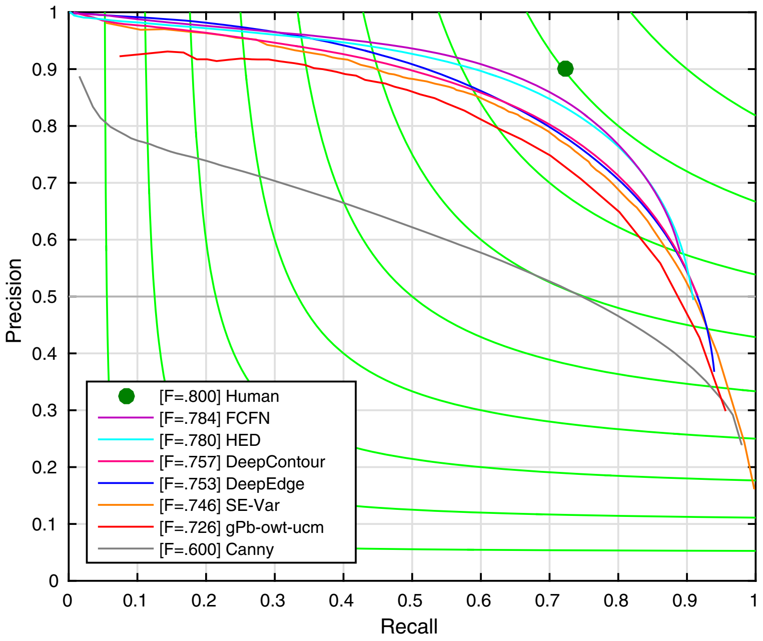


Fig. 17 Precision and recall curves on the BSDS500 dataset. These methods are ranked according to their best F-score (ODS). FCFN achieves superior result as compared with other approaches

these approaches in Fig. 17. Figure 18 shows an edge detection example and compares the visual details between FCFN and HED.

5.6 Future work

Recall that the proposed CFN (and FCFN) model is a general extension of a plain CNN, and can be applied to a variety of visual recognition tasks. Therefore, it is potentially possible to promote CFN by introducing more techniques in future work. In the following, we provide two promising directions.

(1) While computing adaptive weights in the LC fusion module, we use a 1×1 kernel filter to independently consider each spatial location in the feature maps. A potential attempt is to utilize larger kernel sizes such as 1×2 and 1×3, which can incorporate the contextual information in the feature maps.

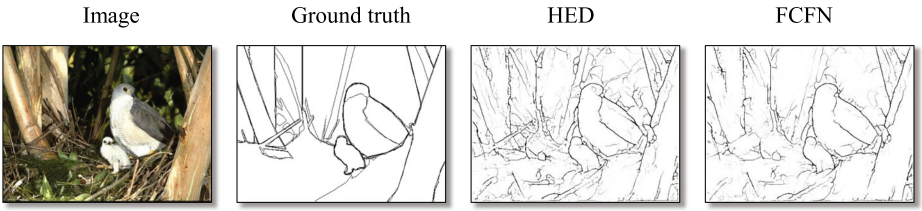


Fig. 18 Comparison of an edge detection example between the baseline HED and the proposed FCFN. The FCFN results look more similar with the ground-truth annotations than the HED results

(2) The adaptive weights are learned with the training images and will be directly applied to the test images for inference. It may be beneficial to learn input-specific weights to decrease the variance between images. Jaderberg et al. [25] proposed a new learnable module, called the Spatial Transformer, that can perform explicit spatial transformations of features within CNNs. Similarly, Brabandere et al. [12] proposed a Dynamic Filter Network (DFN), where filters are dynamically generated conditioned on an input image. Driven by these works, CFN can also learn dynamical filters conditioned on an input image.

6 Conclusions

This work presents a deep fusion architecture (CFN) built on top of plain CNNs. It allows to aggregate intermediate layers with adaptive weights, and generates a discriminative feature representation. We conducted comprehensive experiments to evaluate its effectiveness for both image-level and pixel-level classification tasks. To summarize, several remarks and insights can be obtained based on the performance from the experiments:

(1) On the CIFAR and ImageNet datasets, the CFN models achieved considerable improvements while adding few parameters, even though these models are not very deep. CFN is a simple yet efficient architecture that has potential to be adapted to both deep (e.g. 10 layers) and much deeper (e.g. 100 layers). In future work, we will build CFN on top of other deeper networks.

(2) CFN shows promising results when it is transferred to three different tasks, since CFN inherits the generalization capabilities from CNN. Additionally, CFN yields remarkable gains over CNN in the Flower dataset for fine-grained recognition. We find that it is quite important and necessary to make use of intermediate features to describe fine-grained attributes of objects.

(3) Although the FCFN models need to learn more adaptive weights in the fusion module, there are significant benefits due to the improvements for semantic segmentation and contour detection. In addition, a great many details related to objects (e.g. boundary) can be obtained from the intermediate layers instead of the top layers.

(4) Although we have evaluated CFN for several classification tasks, it is promising to apply CFN to other visual applications such as object detection and visual tracking.

Acknowledgments This work was supported mainly by the LIACS Media Lab at Leiden University and in part by the China Scholarship Council. We would like to thank NVIDIA for the donation of GPU cards.

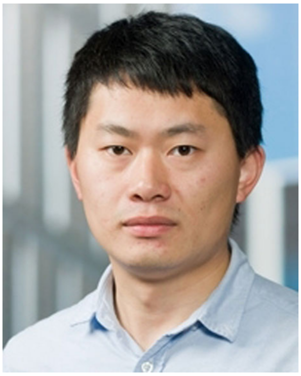
References

1. Agrawal P, Girshick R, Malik J (2014) Analyzing the performance of multilayer neural networks for object recognition
2. Arbelaez P, Maire M, Fowlkes C, Malik J (2011) Contour detection and hierarchical image segmentation. IEEE Trans Pattern Anal Mach Intell 33(5):898–916

3. Babenko A, Lempitsky VS (2015) Aggregating deep convolutional features for image retrieval. In: Proceedings of the international conference on computer vision (ICCV)
4. Bertasius G, Shi J, Torresani L (2015) Deepedge: a multi-scale bifurcated deep network for top-down contour detection. In: The IEEE conference on computer vision and pattern recognition (CVPR)
5. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 8(6):679–698
6. Cao L, Gao L, Song J, Shen F, Wang Y (2017) Multiple hierarchical deep hashing for large scale image. https://doi.org/10.1007/s11042-017-4489-0
7. Chang CC, Lin CJ (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2:27:1–27:27
8. Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2015) Semantic image segmentation with deep convolutional nets and fully connected crfs. In: International conference on learning representations (ICLR)
9. Cimpoi M, Maji S, Vedaldi A (2015) Deep filter banks for texture recognition and segmentation. In: Proceedings of the ieee conference on computer vision and pattern recognition (CVPR)
10. Cun L, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1990) Handwritten digit recognition with a back-propagation network. In: Neural information processing systems (NIPS)
11. Dai J, Li Y, He K, Sun J (2016) R-FCN: object detection via region-based fully convolutional networks. In: Neural information processing systems (NIPS)
12. De Brabandere B, Jia X, Tuytelaars T, Van Gool L (2016) Dynamic filter networks. In: Neural information processing systems (NIPS)
13. Dollár P, Zitnick CL (2015) Fast edge detection using structured forests. IEEE Trans Pattern Anal Mach Intell 37(8):1558–1570
14. Everingham M, Eslami SMA, Van Gool L, Williams CKI, Winn J, Zisserman A (2015) The pascal visual object classes challenge: a retrospective. Int J Comput Vis 111(1):98–136
15. Girshick R (2015) Fast R-CNN. In: Proceedings of the International conference on computer vision (ICCV)
16. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
17. Goodfellow IJ, Warde-Farley D, Mirza M, Courville AC, Bengio Y (2013) Maxout networks. In: ICML
18. Gregor K, LeCun Y (2010) Emergence of complex-like cells in a temporal product network with local receptive fields. CoRR arXiv:1006.0448
19. Hariharan B, Arbelaez P, Bourdev L, Maji S, Malik J (2011) Semantic contours from inverse detectors. In: The IEEE international conference on computer vision (ICCV)
20. He K, Zhang X, Ren S, Sun J (2014) Spatial pyramid pooling in deep convolutional networks for visual recognition. In: Proceedings of the European conference on computer vision (ECCV)
21. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the ieee conference on computer vision and pattern recognition (CVPR)
22. Huang G, Sun Y, Liu Z, Sedra D, Weinberger K (2016) Deep networks with stochastic depth. In: Proceedings of the european conference on computer vision (ECCV)
23. Huang X, Shen C, Boix X, Zhao Q (2015) Salicon: reducing the semantic gap in saliency prediction by adapting deep neural networks. In: The IEEE international conference on computer vision (ICCV)
24. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: ICML
25. Jaderberg M, Simonyan K, Zisserman A, Kavukcuoglu K (2015) Spatial transformer networks. In: Neural information processing systems (NIPS)
26. Jegou H, Douze M, Schmid C (2008) Hamming embedding and weak geometric consistency for large scale image search. In: Proceedings of the European conference on computer vision (ECCV)
27. 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: ACM multimedia
28. Jin X, Xu C, Feng J, Wei Y, Xiong J, Yan S (2016) Deep learning with s-shaped rectified linear activation units. In: AAAI
29. Kim JH, Lee SW, Kwak DH, Heo MO, Kim J, Ha JW, Zhang BT (2016) Multimodal residual learning for visual qa. In: Neural information processing systems (NIPS)
30. Krizhevsky A (2009) Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto
31. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Neural information processing systems (NIPS)
32. Kuen J, Wang Z, Wang G (2016) Recurrent attentional networks for saliency detection. In: The IEEE conference on computer vision and pattern recognition (CVPR)

33. Lazebnik S, Schmid C, Ponce J (2006) Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
34. Lee C, Xie S, Gallagher P, Zhang Z, Tu Z (2015) Deeply-supervised nets. In: AISTATS
35. Liang M, Hu X (2015) Recurrent convolutional neural network for object recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
36. Lin M, Chen Q, Yan S (2014) Network in network. In: International conference on learning representations (ICLR)
37. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: Common objects in context. In: Proceedings of the European conference on computer vision (ECCV)
38. Liu Y, Lew MS (2016) Learning relaxed deep supervision for better edge detection. In: The IEEE conference on computer vision and pattern recognition (CVPR)
39. Liu L, Shen C, van den Hengel A (2015) The treasure beneath convolutional layers: cross convolutional layer pooling for image classification. In: proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
40. Liu Y, Guo Y, Wu S, Lew MS (2015) Deepindex for accurate and efficient image retrieval. In: Proceedings of the 5th ACM on international conference on multimedia retrieval (ICMR)
41. Liu Y, Guo Y, Lew SM (2017) On the exploration of convolutional fusion networks for visual recognition. In: International conference on multimedia modeling (MMM)
42. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
43. Mohedano E, McGuinness K, O'Connor NE, Salvador A, Marques F, Giro-i Nieto X (2016) Bags of local convolutional features for scalable instance search. In: Proceedings of ACM on international conference on multimedia retrieval (ICMR)
44. Nilsback ME, Zisserman A (2008) Automated flower classification over a large number of classes. In: Indian conference on computer vision, graphics and image processing
45. Nister D, Stewenius H (2006) Scalable recognition with a vocabulary tree. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
46. Ou X, Ling H, Liu S, Lei J (2016) Hierarchical deep semantic hashing for fast image retrieval. Multimed Tools Appl 76(20):21281–21302
47. Quattoni A, Torralba A (2009) Recognizing indoor scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
48. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: towards real-time object detection with region proposal networks. In: Neural information processing systems (NIPS)
49. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet large scale visual recognition challenge. IJCV 115(3):211–252
50. Schroff F, Kalenichenko D, Philbin J (2015) FaceNet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
51. Sermanet P, Chintala S, LeCun Y (2012) Convolutional neural networks applied to house numbers digit classification. In: International conference on pattern recognition (ICPR)
52. Shen W, Wang X, Wang Y, Bai X, Zhang Z (2015) Deepcontour: a deep convolutional feature learned by positive-sharing loss for contour detection. In: The IEEE conference on computer vision and pattern recognition (CVPR)
53. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: International conference on learning representations (ICLR)
54. Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M (2015) Striving for simplicity: the all convolutional net. In: International conference on learning representations (ICLR)
55. Srivastava RK, Greff K, Schmidhuber J (2015) Training very deep networks. In: Neural information processing systems (NIPS)
56. Sun Y, Wang X, Tang X (2015) Deeply learned face representations are sparse, seletive, and robust. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
57. Szegedy C, Ioffe S, Vanhoucke V (2016) Inception-v4, inception-resnet and the impact of residual connections on learning. CoRR arXiv:1602.07261
58. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
59. Wah C, Branson S, Welinder P, Perona P, Belongie S (2011) The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001

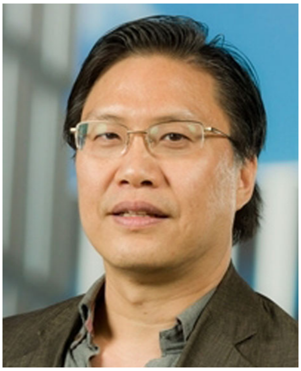
60. Wei XS, Gao BB, Wu J (2015) Deep spatial pyramid ensemble for cultural event recognition. In: Proceedings of the international conference on computer vision (ICCV) workshops
61. Wu S, Zhong S, Liu Y (2017) Deep residual learning for image steganalysis. Multimed Tools Appl. https://doi.org/10.1007/s11042-017-4440-4
62. Xie S, Tu Z (2015) Holistically-nested edge detection. In: Proceedings of the international conference on computer vision (ICCV)
63. Yang S, Ramanan D (2015) Multi-scale recognition with DAG-CNNs. In: Proceedings of the international conference on computer vision (ICCV)
64. Yoo D, Park S, Lee JY, Kweon IS (2015) Multi-scale pyramid pooling for deep convolutional representation. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), DeepVision workshop
65. Yue-Hei Ng J, Yang F, Davis LS (2015) Exploiting local features from deep networks for image retrieval. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), Deep Vision workshops
66. Zagoruyko S, Komodakis N (2016) Wide residual networks. In: British machine vision conference (BMVC)
67. Zheng S, Jayasumana S, Romera-Paredes B, Vineet V, Su Z, Du D, Huang C, Torr P (2015) Conditional random fields as recurrent neural networks. In: The IEEE international conference on computer vision (ICCV)
68. Zhou B, Lapedriza A, Xiao J, Torralba A, Oliva A (2014) Learning deep features for scene recognition using places database. In: Neural information processing systems (NIPS)



Yu Liu received the B.S. degree and M.S degree from School of Software Technology, Dalian University of Technology, Dalian, China, in 2011 and 2014, respectively. He is currently a Ph.D. in Leiden Institute of Advanced Computer Science (LIACS), Leiden University. His current research interests include computer vision and deep learning, especially, image classification, image retrieval and multi-modal matching. He obtained the Best Paper Award in MMM 2017.

Yanming Guo received the B.S. degree in information system engineering, the M.S degree in operational research from the National University of Defense Technology, Changsha, China, in 2011 and 2013, respectively. He is currently a Ph.D. in Leiden Institute of Advanced Computer Science (LIACS), Leiden University. His current research interests include image classification, object detection and image retrieval.

Theodoros Georgiou received the B.S. degree and M. S degree from Leiden University, The Netherlands. He is currently a Ph.D. in Leiden Institute of Advanced Computer Science (LIACS), Leiden University. His current research interests include high-dimensional data mining and deep learning.



Michael S. Lew is co-head of the Imagery and Media Research Cluster at LIACS and director of theLIACS Media Lab. He received his doctorate from University of Illinois at Urbana-Champaign and then became a postdoctoral researcher at Leiden University. One year later he became the first Leiden University Fellow which was a pilot program for tenure track professors. In 2003, he became a tenured associate professor at Leiden University and was invited to serve as a chair full professor in computer science at Tsinghua University (the MIT of China). He has published over 100 peer reviewed papers with three best paper citations in the areas of computer vision, content-based retrieval, and machine learning. Currently (September 2014), he has the most cited paper in the history of the ACM Transactions on Multimedia. In addition, he has the most cited paper from the ACM International Conference on Multimedia Information Retrieval (MIR) 2008 and also from ACM MIR 2010. He has served on the organizing committees for over a dozen ACM and IEEE conferences. He served as the founding the chair of the ACM ICMR steering committee and had served as chair for both the ACM MIR and ACM CIVR steering committees. In addition he is the Editor-in-Chief of the International Journal of Multimedia Information Retrieval (Springer) and a member of the ACM SIGMM Executive Board which is the highest and most influential committee of the SIGMM.