DualNet: Learn Complementary Features for Image Recognition

Saihui Hou, Xu Liu and Zilei Wang
Department of Automation, University of Science and Technology of China
{saihui, liuxu91}@mail.ustc.edu.cn, zlwang@ustc.edu.cn

Abstract

In this work we propose a novel framework named DualNet aiming at learning more accurate representation for image recognition. Here two parallel neural networks are coordinated to learn complementary features and thus a wider network is constructed. Specifically, we logically divide an end-to-end deep convolutional neural network into two functional parts, i.e.*, feature extractor and image classifier. The extractors of two subnetworks are placed side by side, which exactly form the feature extractor of DualNet. Then the two-stream features are aggregated to the final classifier for overall classification, while two auxiliary classifiers are appended behind the feature extractor of each subnetwork to make the separately learned features discriminative alone. The complementary constraint is imposed by weighting the three classifiers, which is indeed the key of DualNet. The corresponding training strategy is also proposed, consisting of iterative training and joint finetuning, to make the two subnetworks cooperate well with each other. Finally, DualNet based on the well-known CaffeNet, VGGNet, NIN and ResNet are thoroughly investigated and experimentally evaluated on multiple datasets including CIFAR-100, Stanford Dogs and UEC FOOD-100. The results demonstrate that DualNet can really help learn more accurate image representation, and thus result in higher accuracy for recognition. In particular, the performance on CIFAR-100 is state-of-the-art compared to the recent works.*

1. Introduction

Recent years have witnessed the bloom of deep convolutional neural network (DCNN), which has remarkably boosted the performance of various visual assignments [14, 29, 5]. The success of DCNN is largely attributed to its deep architecture and end-to-end learning approach, which can learn hierarchical representation of the input. And the fundamental research on DCNN is to develop advanced networks and corresponding training algorithms, with the aims of extracting more discriminative features for recognition.

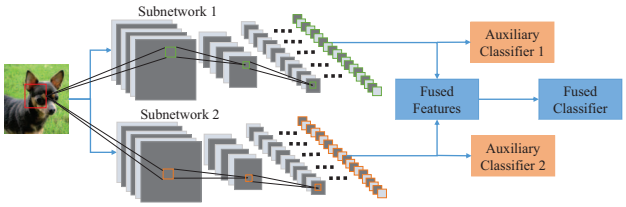

Figure 1. **Illustration of the proposed DualNet**. The input images are fed into two subnetworks, which are coordinated to learn complementary features. The two-stream features are then fused to form a unified representation, and passed into the *Fused Classifier* for overall classification. The auxiliary classifiers are appended to keep the separately learned features discriminative.

There have been considerable interests in enhancing DCNN with greater capacity, in which the networks are generally designed to be *deeper* or *wider*. For example, He *et al*. [13] propose a 152-layer ResNet, which is $8\times$ *deeper* than VGGNet [31] and achieves state-of-the-art performance in several ILSVRC2015 tasks. On the other hand, going *wider* has also been proven applicable [30, 37, 36]. For instance, the recent WRN [36] decreases depth and increases width of ResNet and achieves comparable performance. In fact, according to neurobiology, human visual system only activates several neurons of cortex for a certain input pattern, but the details of visual stimulus can be perfectly perceived in the V_1 zone, especially for the area of central fovea [9]. Such powerful capability implies that there exist plenty of neurons in visual cortex to represent the details of input stimulus, although each of them just has a simple response pattern. Inspired by such a mechanism, we particularly propose to construct a wider network to learn richer features for image recognition, *i.e*., assigning more sibling nodes in each layer. However, developing innovative networks is nontrivial, which needs expertise in neuroscience and labor-intensive parameter tuning.

In this work, we present a framework named DualNet to effectively learn more accurate representation for image recognition, as illustrated in Figure 1. The core idea of DualNet is to coordinate two parallel DCNNs to learn features complementary to each other, and thus richer features can be extracted from the raw images. Specifically, we consider

2380-7504/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCV.2017.62

an end-to-end DCNN to be composed of two logical parts, *i.e*., feature extractor and image classifier, although they are integrated without explicit division. Then a network of double width is constructed by placing the feature extractors of two subnetworks side by side, which exactly form the feature extractor of DualNet. Consequently, two streams of features can be extracted for an input image, which are then aggregated to form a unified representation to the final classifier for overall classification. Meanwhile, two auxiliary classifiers are appended behind the feature extractor of each subnetwork to make the separately learned features discriminative alone. And the complementary constraint is imposed by weighting the three involved classifiers, which is indeed the key of DualNet. Besides, the corresponding training strategy is proposed to make two subnetworks cooperate well, which consists of *iterative training* and *joint finetuning*. Compared to straightly doubling the layer width[1], our method is practically feasible without introducing too much memory cost, and is able to bring significant improvement for image recognition.

Finally, we thoroughly investigate the proposed DualNet framework based on the well-known CaffeNet [19], VGGNet [31], NIN [22] and ResNet [13], and experimentally evaluate its effectiveness on multiple datasets including CIFAR-100 [18], Stanford Dogs [17] and UEC FOOD-100 [26]. The results show that DualNet performs well with different DCNN architectures and datasets of diverse domains, and reports promising improvement compared to the baselines. In particular, the performance achieved by DualNet on CIFAR-100 is state-of-the-art, with less computation cost introduced compared to the recent works [35, 27]. To the best of our knowledge, this work is the first to focus on the cooperation of multiple DCNNs, with the same input and only simple fusion method considered, such as SUM, Max and Concat.

The following paper is organized as follows. In Section 2, we review the related works on image recognition and DCNN. Section 3 presents the details of DualNet, and Section 4 provides the experimental evaluation, which is further discussed in Section 5. Finally, the whole work is concluded in Section 6.

2. Related Work

Image recognition is a basic issue of computer vision, in which adopted features are critical in determining the classification performance. While traditional methods are usually based on hand-crafted features, such as SIFT [24] and HOG [7], more recent works [19, 31, 34, 13] have resorted to the Deep Convolutional Neural Network (DCNN), which is able to automatically learn discriminative features from millions of labelled images. A typical DCNN consists of a number of convolution and pooling layers optionally followed by fully connected layers [19].

The convolutional neural network had its earlier root in [8, 20]. But the real milestone was not set until recent years by AlexNet [19], whose massive improvement shown on ILSVRC2012 rekindled people's interests in DCNN. Due to the availability to large training data and GPU accelerated computation, multiple efforts have been taken to enhance DCNN for greater capacity, *e.g*., increased depth [31, 34, 13], enlarged width [30, 37, 36], smaller stride in convolution or pooling [31, 37], new nonlinear activations [25, 12], novel layers [11, 34], effective regulations [15] and so on. These improvements have turned out to be helpful first on generic classification and then applied to other visual assignments. However, the design of innovative networks is of high complexity which needs expertise in neuroscience [34], and the parameter tuning is labor-intensive [31, 34, 13]. Different from these previous works, in DualNet, we do not redesign a certain component in DCNN and instead exploit the potentials of existing networks. We assemble multiple DCNNs to learn complementary features and thus form a wider network to extract more accurate image representation for recognition.

More related works lie in [28, 23]. Compared to the multi-stream learning [28] in which parallel streams have different parameter numbers and receptive field sizes, the subnetworks in DualNet have the same architectures. Besides, the motivation and optimization are totally different. As for [23], it specially deals with fine-grained categorization, while DualNet aims at generic classification. And our work further differs from [23] in design philosophy, network architecture and computation cost. Firstly, in [23], the output of two DCNNs are multiplied to assemble the information of each location, while DualNet is designed to make each unit of high layers to describe its corresponding image patch more accurately. Secondly, the bilinear pooling is taken as the fusion method in [23], but in DualNet only simple SUM is adopted. In fact, the bilinear model that performs best in [23] is eventually implemented with a single DCNN because of weight sharing, however, DualNet holds two DCNNs which do not share weights and are complementary to each other. At the same time, the auxiliary classifiers are introduced in DualNet and the training process is quite different. Thirdly, the features after the bilinear pooling are of high dimension, which can model the subtle difference of fine-grained categories better but requires much more training complexity and memory cost. In contrast, DualNet is rather computationally efficient. Furthermore, DualNet is not a specific network but a fairly generic framework which can generalize multiple DCNN architectures. Particularly, the performance achieved by DualNet is state-of-the-art on CIFAR-100 with SUM as the fusion method.

[1] To train a VGGNet with double neurons at each layer without decreasing the mini-batch size (32) will exceed the memory limit of a Tesla K40 GPU.

3. Our Approach

The extraction of visual features is usually treated as the most important design choice in computer vision tasks, including image recognition. Despite of the great improvement brought by DCNN, the top-1 accuracy for image recognition is still not satisfying enough for practical applications, *e.g.*, 19.38% top-1 error on ImageNet validation set with ResNet-152 [13]. Hence it is still vital and necessary to develop advanced models to learn more accurate image representation. So far DCNN is considered to be the most competitive approach for feature extraction, which is able to abstract hierarchical features ranging from edges to entire objects [37]. In practice, DCNN is trained by optimizing the objective loss function, *i.e.*, the training is driven by the errors generated at the highest layer according to back propagation (BP). Consequently, in the optimization of single network, some distinctive details of the objects, which are low-level but essential to discriminate the classes of strong similarity, are likely to be dropped in the middle layers or overwhelmed by massive useless information, since the loss signals received by shallow layers for parameter update have been filtered by multiple upper layers. In other words, it is difficult for single network to learn the whole details of input images.

To deal with this issue, in this work, we propose a novel DualNet framework consisting of two parallel networks. The highlight of DualNet is to coordinate two networks to learn complementary features from input images, *i.e.*, one network is able to learn details about the objects of interest which are missing in the other, such that after fusion richer and more accurate image representation can be extracted for recognition. Particularly, in the design of DualNet, we follow the principles listed below:

P1 The features after fusion are expected to be the most discriminative compared to the features extracted by each subnetwork, which exactly indicates the complementary learning embedded in DualNet.

P2 The framework should be fairly generic to perform well with most of typical DCNNs, *e.g.*, VGGNet and ResNet, and popular datasets, *e.g.*, CIFAR-100.

P3 In terms of computation cost, the networks should be efficient as much as possible for training and test, and compatible for a Tesla K40 GPU (12 GB memory limit) without decreasing the mini-batch size.

P4 Only simple fusion methods, such as SUM, MAX and Concat, are considered to ensure the generalization ability and computation efficiency, and the focus is the cooperation and complementarity of two subnetworks.

From another perspective, DualNet can be also viewed to provide an approach to construct a wider network. By effectively assembling the feature extractors of two subnetworks, we finally acquire a network with double neurons at each layer such that the input patterns can be more fully represented. In the following, we will respectively elaborate on the architecture of DualNet and the corresponding training strategy.

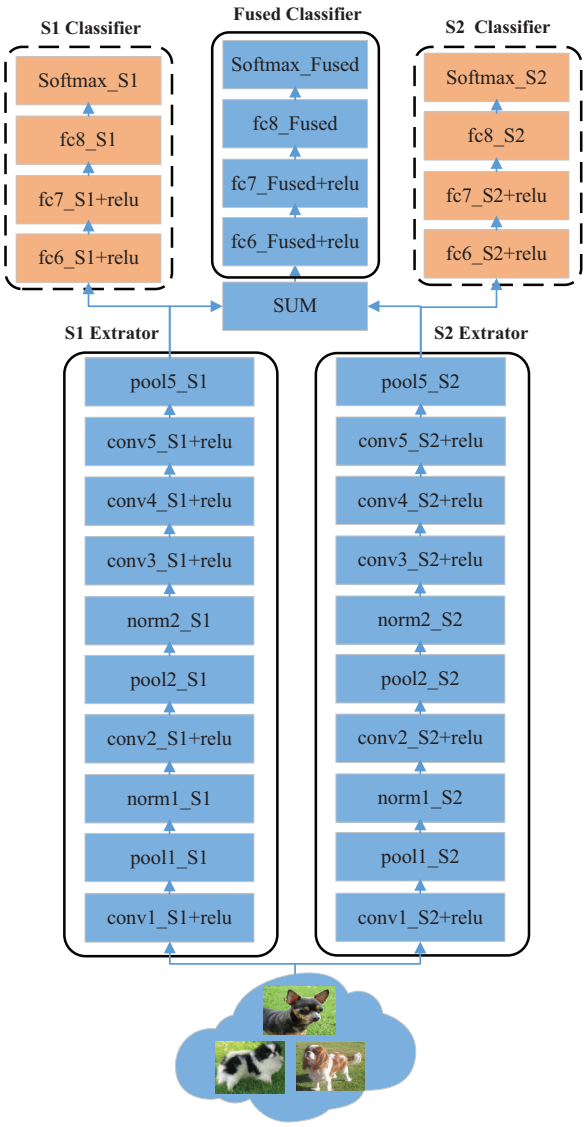

Figure 2. **The architecture of DNC.** The *dropout* layers following *fc6* and *fc7* in the {*Fused, S1, S2*} *Classifier* are omitted. On the whole, the feature maps of the *S1 Extractor* and *S2 Extractor* are summed into the *Fused Classifier*, which is expected to achieve higher accuracy for recognition by coordinating two extractors to learn complementary features.

3.1. Network Architecture

In DualNet, two identical DCNNs are adopted for the complementary learning, as illustrated in Figure 1. The Subnetwork 1 and Subnetwork 2 can be any existing mod-

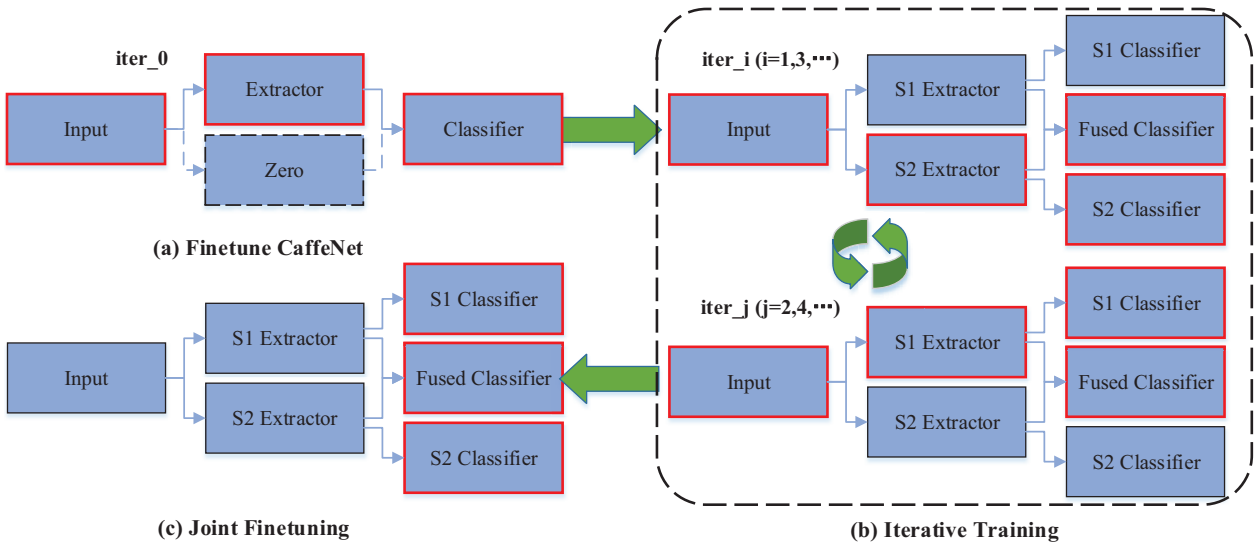

Figure 3. **Illustration of the training strategy of DNC.** Caffenet is first finetuned on specific dataset for the initialization of DNC. Then the *S1 Extractor* and *S2 Extractor* are trained in an iterative way through the *iterative training*. Finally, the last fully connected layers of the *Fused Classifier*, *S1 Classifier*, *S2 Classifier* are jointly finetuned. At each stage, only the components surrounded by the red rectangle are finetuned with the rest fixed. Best viewed electronically.

els, and here we evaluate DualNet based on the well-known CaffeNet, VGGNet, NIN and ResNet[2] (referring to P2). They are coordinated to learn complementary features from input images, which are then aggregated to build richer and more accurate representation for recognition compared to single network.

An example architecture of the DualNet From CaffeNet (DNC) is shown in Figure 2, where the two involved Caffenet are denoted as *S1* and *S2* for simplicity. Particularly, we logically divide an end-to-end CaffeNet into two functional parts, *i.e.*, feature extractor and image classifier. The division is not explicitly specified and theoretically can occur at any layer, *e.g.*, *pool5* here. On the whole, DNC has a symmetrical architecture, in which the *S1 Extractor* and *S2 Extractor* are placed side by side and the feature maps produced by them are integrated into the *Fused Classifier*. The auxiliary *S1 Classifier* and *S2 Classifier* are appended to make the features produced by each feature extractor discriminative alone. And the complementary constraint is imposed by weighting the three classifiers. In fact, the key component of DualNet is the *Fused Classifier*, which can assemble two extractors to describe the objects of interest from different aspects, and thus result in higher accuracy for recognition (referring to P1).

The fusion layer *pool5* is empirically selected for the following reasons. Each activation unit in *pool5* corresponds to a 32×32 patch in the input image, while the one in fully connected layers sees the entire scene. We expect that one of the image extractors can learn more specific characteristics about the objects complementary to the other, and these details are usually presented in small regional areas. So it is implied that the fusion is better performed on the local patch, not the full image scope. In addition, fusing at *pool5* also has computational benefit when performing test (referring to P3). SUM is chosen as the fusion method for simplicity, and also for transferring the parameters from the last fully connected layer, *i.e.*, classifier, of the original CaffeNet. And the coefficients are fixed as $\{0.5, 0.5\}$. Further discussion about the selection of fusion method is provided in Section 5 according to P2, P3 and P4.

The same strategy is applied to the 16-layer VGGNet to construct the DualNet From VGGNet (DNV). Please refer to [31] for the details of VGGNet. The *S1 Extractor* and *S2 Extractor* are comprised of the layers before *pool5* in VGGNet. The feature maps from two extractors are averaged and sent into the following *Fused Classifier* for overall classification. The auxiliary classifier is appended after each extractor to keep the features discriminative alone.

For NIN [22] and ResNet [13] in which there does not exist fully connected layers and the input size is much smaller (32×32), following the same philosophy, we choose to construct the DualNet From NIN (denoted as DNI instead of DNN to avoid confusion) and DualNet From ResNet (DNR) by averaging the features maps of two subnetworks at the penultimate convolution layer (*e.g.*, *cccp5* in NIN), while the last convolution layer is for prediction[3].

[2]All using the public version in the Caffe Model Zoo [1]. For NIN, the settings of NIN-CIFAR10 is adopted. And for ResNet, since there is no complete model in the Caffe Model Zoo (only testing code), we implement it on CIFAR-10 with Caffe, according to [13, 14] and the third-party implementation available at [2].

[3]The network architectures of DNV, DNI and DNR are illustrated in the supplementary material.

3.2. Training Strategy

The training strategy plays vital roles in coordinating the two extractors to learn complementary features, in which the whole network is not just globally finetuned (referrring to P1, P3). Taking the training of DNC as example, as illustrated in Figure 3, it mainly consists of two aspects: the *iterative training* between the *S1 Extractor* and *S2 Extractor* to make them cooperate well, and the *joint finetuning* of the *Fused Classifier*, *S1 Classifier* and *S2 Classifier* for further performance improvement.

3.2.1 Iterative Training

The *iterative training* means, between the *S1 Extractor* and *S2 Extractor*, fixing one of them and finetuning the other in an iterative way. On the one hand, it is out of the consideration of conserving GPU memory (referring to P3) and reducing overfitting. On the other hand, we hope that this way would explicitly make one extractor learn complementary features to the other during each iteration, thus yielding more discriminative fused features (referring to P1). Particularly, the domain-specific finetuning of CaffeNet can be treated as a special case of the *iterative training* for DNC, *i.e.*, *iter_0*, in which another extractor is assigned with zero and the auxiliary classifiers are omitted for the moment. Then the finetuned CaffeNet is utilized to initialize the *S1 Extractor* and *Fused Classifier* for the next iteration, *i.e.*, *iter_1*, and meanwhile the *S1 Classifier* is also initialized but not involved in *iter_1*.

While training the *S2 Extractor* (in *iter_i, i=1,3,...*), the parameters of the *S1 Extractor* are fixed. Appending the *S2 Classifier* at the top of *pool5_S2* can prevent that, the *S2 Extractor* moves towards the same weights as the *S1 Extractor* during training[4], and thus will have little effect on the fused features. Specifically, the modules including the *S2 Extractor*, *S2 Classifier* and *Fused Classifier* are optimized according the loss function defined as:

$$L_1 = L_{Fused} + \lambda_{S2} L_{S2} \tag{1}$$

where L_{Fused} and L_{S2} are both cross entropy loss computed by the *Softmax_Fused* and *Softmax_S2*. The loss weight λ_{S2} is empirically set to 0.3. To some extent, the second term in the loss plays as the regularization for training, and $\lambda_{S2} < 1$ is to inform the *S2 Extractor* that the *Fused Classifier* is more important in the optimization.

And while the *S2 Extractor* is fixed (in *iter_j, j=2,4,...*), the *S1 Classifier* is appended for the finetuning of the *S1 Extractor* (as well as the *Fused Classifier*) and the loss function is defined as:

$$L_2 = L_{Fused} + \lambda_{S1} L_{S1} \tag{2}$$

[4]For example, in *iter_1*, without the *S2 Classifier*, the modules to train will be the *S2 Extractor* and *Fused Classifier*, which are basically the same as CaffeNet in in *iter_0*. The ablation study is taken in Section 4.4

where L_{Fused} and L_{S1} are cross entropy loss computed by the *Softmax_Fused* and *Softmax_S1*. The loss weight λ_{S1} is also set to 0.3.

The maximum iteration (denoted as *max_iter*) is set according to the cross validation. Generally speaking, it is no more than 2 to gain the satisfying improvement. When testing, the output of the *Fused Classifier* is taken for a fair comparison with the base model, *e.g.*, CaffeNet. Certainly, we can also assemble the predictions of three classifiers, *i.e.*, the probability of each class is computed as:

$$score = score_{Fused} + \lambda_{S2} score_{S2} + \lambda_{S1} score_{S1} \tag{3}$$

where $score_{Fused}$, $score_{S2}$, $score_{S1}$ are the output of the *Fused Classifier*, *S2 Classifier* and *S1 Classifier* at testing time. Then $score$ is taken to evaluate for the recognition.

3.2.2 Joint Finetuning

There are three classifier modules, *i.e.*, the *S1 Classifier*, *S2 Classifier* and *Fused Classifier*, involved in DualNet, but in the above methods their abilities have not been fully exploited. Here an alternative integration method is proposed to further boost the performance.

Since global finetuning of the whole network, *e.g.*, DNV, is time-consuming and requires large GPU memory (referring to P3), we instead choose to jointly finetune the last fully connected layer of three classifier modules (*e.g.*, *fc8* in DNC, *cccp6* in DNI) with the following loss function:

$$L_3 = L_{Fused} + \lambda_{S2} L_{S2} + \lambda_{S1} L_{S1} \tag{4}$$

where as above L_{Fused}, L_{S2}, L_{S1} are all cross entropy loss output by the *Fused Classifier*, *S2 Classifier* and *S1 Classifier* respectively. The loss weights λ_{S2} and λ_{S1} keep as 0.3. Correspondingly, in the testing phase, the prediction for each image is obtained according to Equation (3).

4. Experiment

In this section, we evaluate the DualNet From CaffeNet (DNC), DualNet From VGGNet (DNV), DualNet From NIN (DNI), DualNet From ResNet (DNR) on multiple widely-used datasets, including CIFAR-100 [18], Stanford Dogs [17] and UEC FOOD-100 [26] (referring to P2). The hyper-parameters for the *iterative training* are identical to those of finetuning the standard deep models on specific datasets. For the *joint finetuning*, the base learning rate is reduced by a factor of 10 for a few additional iterations. All the networks are implemented with Caffe [16] and trained/tested on a Tesla K40 GPU (referring to P3) [5].

[5]The implementation with the parameters for training each model, as well as the pretrained models, is available at https://github.com/ustc-vim/dualnet.

Table 1. **The top-1 accuracy on CIFAR-100 achieved by the standard deep models (NIN&ResNet) and DualNet (DNI&DNR).** The first row shows the results of NIN&ResNet and the rest are all achieved by DNI&DNR. After the *iterative training*, we respectively evaluate the performance of the *Fused Classifier* and the weighted average of three classifiers, while the latter one can validate the necessity of the *joint finetuning*. *w/o aug*-without data augmentation, *w/ aug*-with data augmentation.

Model \ Training	DNI (*w/o aug*)	DNR (*w/ aug*)
standard deep model (NIN&ResNet)	66.91%	69.09%
iterative training (*Fused Classifier*)	69.01%	71.93%
iterative training (*classifier average*)	69.51%	72.29%
joint finetuning (*classifier average*)	**69.76%**	**72.43%**

4.1. CIFAR-100

CIFAR-100 [18] consists of 60000 32×32 natural images in 100 classes, which are split into 50000 for training and 10000 for test. The dataset is pre-processed using global contrast normalization and ZCA whitening following [10, 22, 35, 27], and then is taken to evaluate DNI and DNR here.

NIN [22] is chosen as the base model because it yields one of the best performances on CIFAR-100, and the recent works [35, 27] are also built upon it. We follow the network setting of NIN-CIFAR10 available in the Caffe Model Zoo and change the output number of the last convolution layer to 100. The results achieved by the standard NIN and our DNI are shown in Table 1, and reported in the top-1 accuracy. After the *iterative training*, the *Fused Classifier* of DNI achieves 69.01% testing accuracy, which improves the performance of NIN by more than 2%. At this stage, *i.e.*, before the *joint finetuing*, we also evaluate the weighted average of three classifiers (computed according to Equation (3) and denoted as *classifier average*) and get 69.51%. Finally, the performance is further improved to 69.76% when the *joint finetuning* is done. It is worth noticing that, *max_iter* is set to 1 for the *iterative training*, and thus the computation cost of training DNI is not heavy.

Since data augmentation for CIFAR-100 is not standardized and it is hard to isolate the impact of data augmentation from the methods, following [27], DNI is trained on CIFAR-100 without augmentation to enable a fair comparison with the existing literatures. Table 2 shows the performance comparison of different methods. There are some works, *e.g.*, [32, 14, 36, 6], which are not listed here because they report the results only with data augmentation. Directly comparable to our DNI are [22, 35, 27], which are also built upon NIN. Note that we reproduce higher accuracy with NIN than the original version [22] that was implemented with cuda-convnet [3], and some parameters have been updated. HD-CNN [35] actually adopts the cropping strategy and 10 view testing. It is listed here because it is one of the most representative works and also taken for comparison in [27]. To the best of our knowledge, DDN [27] reports the best result on CIFAR-100 without augmentation before this work. And our DNI performs better than DDN [27] by 1.41%.

Table 2. **Performance comparison of DNI with the existing methods on CIFAR-100 without data augmentation.** The results are all reported in the top-1 accuracy. *-with cropping strategy and 10 view testing [35].

Method	Test accuracy
Maxout Network [10]	61.43%
Tree based priors [33]	63.15%
Network In Network [22]	64.32%
DSN [21]	65.43%
NIN+LA units [4]	65.60%
HD-CNN* [35]	67.38%
DDN [27]	68.35%
DNI (ours)	**69.76%**

Next, in order to evaluate DNR, we first follow the description in [13, 14] to implement a 20-layer ResNet (denoted as ResNet-20) with Caffe. And it achieves the top-1 accuracy of 90.55% on CIFAR-10 without any data augmentation, which basically agrees with the result in [13]. But when the same setting is used for CIFAR-100, the performance (only 60.82%) is much worse than NIN, which is probably caused by data scarcity. After all, the number of images in each class of CIFAR-100 is only one tenth of CIFAR-10. In that case, we augment the training data with padding and randomly changing contrast and brightness, and then get 69.09% on CIFAR-100 with ResNet-20[6]. Then DNR is constructed and also trained on the augmented data. As shown in Table 1, the *iterative training* (*max_iter* is also set to 1) improves the performance to 71.93%. After the *joint finetuning*, DNR finally achieves 72.43% on CIFAR-100, which is 3.34% higher than the base model.

4.2. Stanford Dogs

Since the image size of CIFAR-100 (32×32) is much smaller than the input size of CaffeNet (227×227) and VGGNet (224×224), it is not proper to resize the images of CIFAR-100 to train them. So another dataset, *i.e.*, Stanford Dogs [17], is chosen to evaluate DNC and DNV. The dataset is made up of 120 classes and 20580 images. Throughout the experiments, no bounding box annotation or data augmentation is involved, except that the flip is randomly executed on the images before being input into the network. We follow the standard split way attached in the dataset for training and test, *i.e.*, for each class there are 100 images used for training and the rest for test. At testing time, the evaluation is done with one single center crop of input images. For simplicity, we only report the results of DNC and DNV when all the training is done. And *max_iter* is set to

[6]The network architecture of ResNet-20 and the parameters for data augmentation are provided in the supplementary material.

Table 3. **The top-1 accuracy on Stanford Dogs and UEC FOOD-100 achieved by the standard deep models (CaffeNet&VGGNet) and DualNet (DNC&DNV).** The results are reported using the weighted average of three classifiers when all training is done. The performance comparison demonstrates that the proposed DualNet performs well with CaffeNet and VGGNet.

Method \ Dataset	Stanford Dogs	UEC FOOD-100
CaffeNet	66.84%	39.92%
DNC (From CaffeNet)	**67.94%**	**41.11%**
VGGNet	74.11%	47.40%
DNV (From VGGNet)	**77.56%**	**49.19%**

1 for the *iterative training*. According to Table 3, DNC and DNV perform well and both achieve higher accuracy than the corresponding baselines, *i.e.*, CaffeNet and VGGNet.

4.3. UEC FOOD-100

UEC FOOD-100 [26] is selected to further evaluate DNC and DNV since it is a totally fresh domain which differs from CIFAR-100 and Stanford Dogs (referring to P2), and *max_iter* is set to 2 on it according to the cross validation. There are 100 food categories in the dataset with more than 100 images for each category. There is no split way provided in it, so for each class we randomly pick 100 images for training with the rest for test. The identical experimental settings on Stanford Dogs are followed, *i.e.*, neither bounding box annotation nor data augmentation for training, and one single center crop for test. The performance comparison of DNC and DNV with their base models on UEC FOOD-100 is shown in Table 3. Besides, we further evaluate DNV on the dataset after each training procedure, *i.e.*, the domain-specific finetuning of VGGNet, each iteration of the *iterative training* (*iter_i, i=1,2*), and the *joint finetuning*. As shown in Figure 4, the top-1 accuracy is improved step by step in the training process, indicating that each training procedure helps.

Note that the focus of experiments on Stanford Dogs and UEC FOOD-100 is on the behaviors of DualNet from CaffeNet and VGGNet on datasets of diverse domains, not reporting state-of-the-art results. So we only compare the performance of DNC and DNV with the base models, *i.e.*, CaffeNet and VGGNet, and do not apply any data augmentation [19] or utilize parts [38]. The philosophy of DualNet is to coordinate two DCNNs to learn complementary features, so it is natural to adopt the single DCNN as baselines. And this work makes sense by providing a generic framework to effectively integrate two DCNNs to extract more discriminative representation.

4.4. Experimental Analyses

In this section we take further experiments to analyze the performance achieved by DualNet. Here DNI is taken to illustrate the proof, and the results are reported on CIFAR-100 without data augmentation.

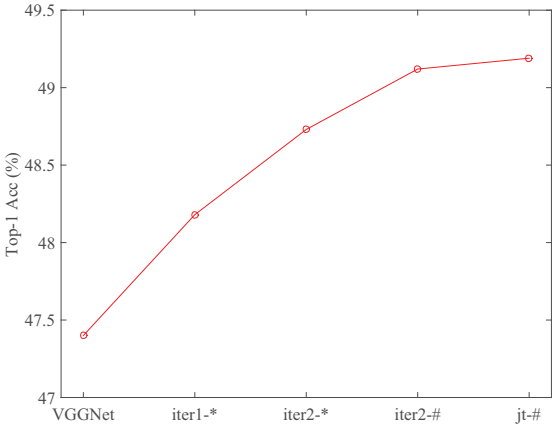

Figure 4. **The top-1 accuracy on UEC FOOD-100 by DNV after each training procedure.** *-*Fused Classifier*, #-*classifier average*, iter_i (i=1,2)-*iterative training*, jt-*joint finetuning*.

In Section 3.2.1, we have explained the importance of the auxiliary classifiers and here assess it experimentally through ablation study. Specifically, in the first iteration of *iterative training* for DNI, *i.e.*, *iter_1*, the identical settings are followed except that the *S2 Classifier* is removed. That is, the finetuned NIN is utilized to initialize the *S1 Extractor* and *Fused Classifier*, and then the *S2 Extractor* and *Fused Classifier* are trained without the *S2 Classifier*. When *iter_1* is done, the output of the *Fused Classifier* is taken for evaluation. In this way we get 67.70% on CIFAR-100, which is much lower than the 69.09% obtained in the same conditions but with the *S2 Classifier* appended. So introducing the auxiliary classifiers is indeed necessary.

We also have tried to remove both the auxiliary classifiers of DNI and then train the remaining network globally, including the *S1 Extractor*, *S2 Extractor* and *Fused Classifier*. In that case, there exist two initialization strategies. The first one is to train from scratch. Then the *Fused Classifier* only achieves 66.11% on CIFAR-100. The second is to initialize the *S1 Extractor*, *S2 Extractor* and *Fused Classifier* with NIN after being finetuned on CIFAR-100. Then we obtain 68.49% with the *Fused Classifier*. Both the results are lower than the 69.76% achieved by DNI.

Another attempt is to directly double the number of features maps in each layer, which is feasible with NIN on a Tesla K40 GPU (not feasible with VGGNet) and then train the network from scratch on CIFAR-100. Compared to the standard NIN (66.91%, also trained from scratch), the testing accuracy is improved to 68.87%. However, it still performs worse than DNI (69.76%), although more computation cost is introduced, such as time and GPU memory required for the optimization.

Besides, since the final accuray of DNI is reported by assembling the three classifiers after the *joint finetuning*, we further evaluate the performance of each individual classi-

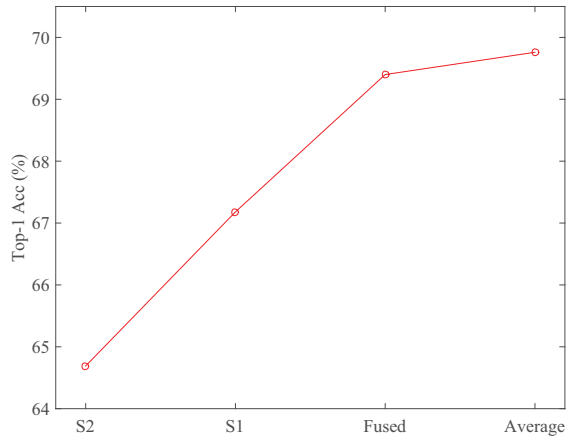

Figure 5. **The evaluation of each classifier after the *joint finetuning* of DNI on CIFAR-100 without data augmentation**. S2-*S2 Classifier*, S1-*S1 Classifier*, Fused-*Fused Classifier*, Average-*the weighted average of three classifiers*.

fier (*S1 Classifier*, *S2 Classifier* and *Fused Classifier*). According to Figure 5, among the three classifiers, the *Fused Classifier* performs best as expected (referring to P1), and the weighted average can further improve the performance.[7]

5. Discussion

Fusion method. In DualNet, we focus on the cooperation of two subnetworks to learn complementary features guided by P1-P4. Referring to P2 and P4, we deal with the issue considering only simple fusion methods, such as SUM, MAX and Concat, to ensure the generalization ability. Besides SUM, we have tried Concat but it does not perform well. The probable reason is that much more parameters are introduced. Max is not a better choice either because it does not make semantic sense for the fusion, in which two extractors are supposed to be complementary, not competitive. Besides, adopting SUM as the fusion method makes all our models, especially DNV, compatible to train on a Tesla K40 GPU without decreasing the mini-batch size (referring to P3). Certainly, other complex methods, *e.g.*, the bilinear pooling, can also be adopted for DualNet, which, however, will incur high computation cost and impair the generalization ability.

Computational effort. In the *iterative training* of DualNet, one of the feature extractor is fixed, and thus the memory consumption is less compared to global finetuning or directly doubling the width of each layer. And for the *joint finetuning*, only the last layer of three classifiers are jointly finetuned. Besides, *max_iter* is always no more than 2, and in most cases, it is in the first iteration that most of the improvement is gained (*e.g.*, DNI on CIFAR-100). More iterations for the *iterative training* and the *joint training* are both optional. Hence the cost of training DualNet is not heavy. Besides, we provide some complexity statistics for the test shown in Table 4. NIN and DNI are further evaluated on CIFAR-100 in terms of GPU memory consumption and time cost for the entire test set. The mini-batch size for test is set to 100. Through the comparison, it can be seen that the improvement brought by DNI is achieved without introducing too much computation cost (two times as much as NIN). By contrast, in HD-CNN [35] which is also built on NIN, the GPU memory consumption and testing time are three times and four times as much as the base model.

[7]More experimental analyses are provided in the supplementary material.

Table 4. **Comparison of GPU memory consumption and time cost between NIN and DNI for the test on CIFAR-100 (10000 images).** The first row shows the complexity statistics of NIN, while the rest are about DNI.

Training / Model	Memory (MB)	Time (sec.)
standard deep model (NIN)	515	9.3
iterative training (*Fused Classifier*)	1032	18.2
iterative training (classifier average)	1061	19.1
joint finetuning (classifier average)	1061	19.1

6. Conclusion

In this work, we deal with the issue of image recognition through providing a fairly generic framework named DualNet, in which two parallel DCNNs are coordinated to learn complementary features, thus constructing a wider network and yielding more discriminative representation. Following the principles P1-P4, two-stream features are extracted for an input image, which are then fused through SUM to form a unified representation. Apart from the overall classifier based on the fused features, two auxiliary classifiers are proposed to be appended behind each extractor to keep the separately learned features discriminative alone. And the complementary constraint is imposed through weighting the three classifiers. Finally, DualNet based on CaffeNet, VGGNet, NIN and ResNet are thoroughly investigated and experimentally evaluated on CIFAR-100, Stanford Dogs and UEC FOOD-100, which all achieve higher top-1 accuracy than the baselines. In particular, the performance on CIFAR-100 is state-of-the-art compared to the recent works. In the future, we plan to explore the utilization of more advanced fusion methods and apply the network compression techniques to further optimize DualNet.

Acknowledgment

This work is supported partially by the National Natural Science Foundation of China under Grant 61673362 and 61233003, Youth Innovation Promotion Association CAS, and the Fundamental Research Funds for the Central Universities.We are grateful for the generous donation of Tesla GPU K40 from the NVIDIA corporation.

References

[1] https://github.com/BVLC/caffe/wiki/Model-Zoo. 4

[2] https://github.com/twtygqyy/resnet-cifar10/tree/master/resnet20. 4

[3] https://code.google.com/p/cuda-convnet/. 6

[4] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014. 6

[5] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016. 1

[6] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR*, 2016. 6

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 2

[8] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980. 2

[9] M. Gazzaniga, R. Ivry, and G. Mangun. *Cognitive Neuroscience: The Biology of the Mind (Fourth Edition)*. W. W. Norton, 2013. 1

[10] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013. 6

[11] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 2

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 2

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 3, 4, 6

[14] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 1, 4, 6

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2

[16] 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. 5

[17] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshop*, 2011. 2, 5, 6

[18] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. 2, 5, 6

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2, 7

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[21] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets (2014). *arXiv preprint arXiv:1409.5185*. 6

[22] M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2014. 2, 4, 6

[23] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 2015. 2

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 2

[25] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. 2

[26] Y. Matsuda, H. Hoashi, and K. Yanai. Recognition of multiple-food images by detecting candidate regions. In *ICME*, 2012. 2, 5, 7

[27] V. N. Murthy, V. Singh, T. Chen, R. Manmatha, and D. Comaniciu. Deep decision network for multi-class image classification. In *CVPR*, 2016. 2, 6

[28] N. Neverova, C. Wolf, G. W. Taylor, and F. Nebout. Multi-scale deep learning for gesture detection and localization. In *ECCV Workshops*, 2014. 2

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1

[30] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 1, 2

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 2, 4

[32] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Ali, R. P. Adams, et al. Scalable bayesian optimization using deep neural networks. In *ICML*, 2015. 6

[33] N. Srivastava and R. R. Salakhutdinov. Discriminative transfer learning with tree-based priors. In *NIPS*, 2013. 6

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2

[35] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu. Hd-cnn: Hierarchical deep convolutional neural networks for large scale visual recognition. In *ICCV*, 2015. 2, 6, 8

[36] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016. 1, 2, 6

[37] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 1, 2, 3

[38] N. Zhang, J. Donahue, R. B. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *ECCV*, 2014. 7