Deep Layer Aggregation

Fisher Yu
UC Berkeley

Dequan Wang
UC Berkeley

Trevor Darrell
UC Berkeley

Abstract

Convolutional networks have had great success in image classification and other areas of computer vision. Recent efforts have designed deeper or wider networks to improve performance; as convolutional blocks are usually stacked together, blocks at different depths represent information at different scales. Recent models have explored 'skip' connections to aggregate information across layers, but heretofore such skip connections have themselves been 'shallow', projecting to a single fusion node. In this paper, we investigate new deep-across-layer architectures to aggregate the information from multiple layers. We propose novel iterative and hierarchical structures for deep layer aggregation. The former can produce deep high resolution representations from a network whose final layers have low resolution, while the latter can effectively combine scale information from all blocks. Results show that the our proposed architectures can make use of network parameters and features more efficiently without dictating convolution module structure. We also show transfer of the learned networks to semantic segmentation tasks and achieve better results than alternative networks with baseline training settings.

1 Introduction

Representation learning has received considerable attention in computer vision community and great progress has been made on major challenges (Russakovsky et al. 2015). Learned representations on one task can be transfered to other computer vision problems, which further encourages research in this area (Krizhevsky et al. 2012; Szegedy et al. 2015; Simonyan and Zisserman 2015; He et al. 2016; Zagoruyko and Komodakis 2016).

Deeper networks with bigger receptive fields typically yield better performance. But deeper networks are usually more difficult to optimize. To overcome this barrier, existing works have proposed different building blocks or modules that can reduce the number of parameters while maintaining effective convolutional kernel size, e.g., in the inception module with dimensionality reduction (Szegedy et al. 2015), or the propagation of meaningful gradients through hundreds of layers in the residual block architecture proposed by (He et al. 2016).

Nevertheless, scant attention has been paid to organization of those blocks or modules other than sequential layering. LeCun et al. (1998) stacks several convolutional layers followed by pooling or non-linear activation together to extract features for digit recognition. Krizhevsky et al. (2012) extended this idea to make breakthrough progress in large-scale image recognition. Since then, even though more modules based on convolutional layers have been proposed, they are still generally stacked together to construct the final networks. Various network visualization methods (Zeiler and Fergus 2014; Yu et al. 2017) show that deeper layers in this stacking structure can extract semantic features at larger scale. This may be good for representing object-centric images, but it may not suffice to represent natural images with multiple objects in which objects can appear at different scale. Although previous works usually assume that the network can learn scale-invariant representation, it may not be the case in practice, especially when the amount of training data is relatively small.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

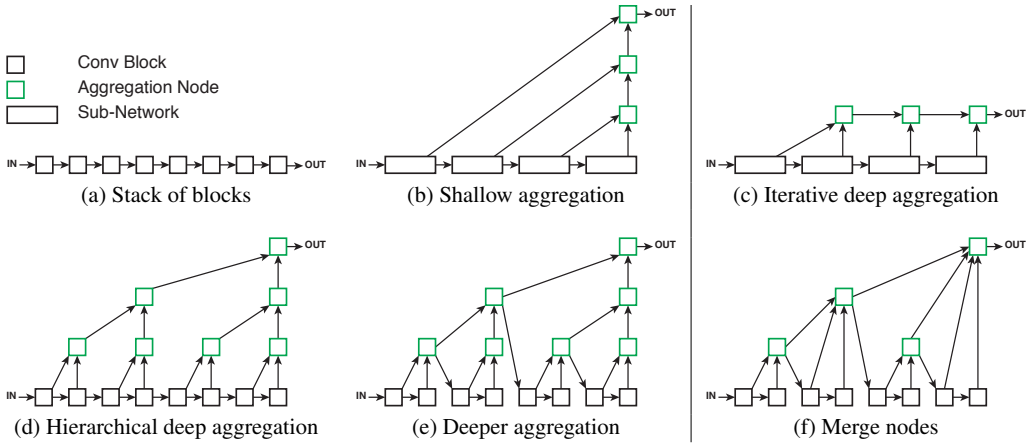


Figure 1: Different types of scale aggregation structures. (a) is used in most image classification networks such as GoogLeNet and ResNet. (b) is a generalization of skip connections commonly used in segmentation and object detection networks. Our proposed models: (c) reordered the connections in (b) to achieve deep aggregations. (d-f) shows changes to improve efficiency of hierarchical deep aggregation. (c) and (f) are used in this paper.

In this paper, we investigate how to aggregate multiple layer activations so that the final representation can incorporate scale information explicitly and the resulting network can be more compact. In contrast to the related works cited above, our layer aggregation connections can themselves be deep, being aggregated by learned intermediate units on the path to a final integration node. Instead of building new networks from scratch, we design a network framework in which the modules proposed in the previous works can be directly used and it can be compatible with future module design. Also, the network should be suitable for different computer vision problems and the learned representation can be transfered to different domains.

We propose two specific structures for deep layer aggregation: Iterative Deep Aggregation (IDA) and Hierarchical Deep Aggregation (HDA). IDA can produce deep high resolution representations from a network whose final layers have low resolution, while HDA can effectively combine layer information from all blocks. In HDA, the building modules are organized in a hierarchical tree structure and multi-layer information is aggregated by different levels of tree nodes. It is deep in the sense that the tree has multiple levels from the root of the hierarchy to the shallowest layer. The framework is general so that we can use different building modules in the same structure and better module can lead to better performance. In HDA, the aggregation depth of blocks grows with the log of the number of blocks. HDA is useful when the blocks have very similar scale information, while IDA is useful to aggregate subnetworks whose outputs have dramatic scale change.

In our experiments, we utilize residual blocks from ResNet (He et al. 2016) and ResNeXt (Xie et al. 2017) and conduct experiments on large-scale image classification, fine-grained classification and semantic image segmentation. Results show that our framework can achieve similar results as baseline ResNet or ResNext architectures with fewer parameters thanks to the layer aggregation, which enables more efficient and compact models. Our model achieves new state-of-the-art results for compact models. Without modification, our networks also obtain new state-of-the-art results on several fine-grained classification benchmarks. We further transfer the learned network to semantic image segmentation for CamVid (Brostow et al. 2009) and Cityscapes (Cordts et al. 2016), and beat the previous best method by 4 points with fewer parameters on CamVid dataset.

2 Deep Layer Aggregation

Our basic convolutional block is a small convolutional network. A block can be as simple as a convolutional layer followed by non-linear activation, as in AlexNet (Krizhevsky et al. 2012) and VGGNet (Simonyan and Zisserman 2015). It usually contains several layers and intra-block skip connections to facilitate gradient back propagation, such as the inception module in GoogLeNet (Szegedy et al. 2015) and the residual block in ResNet (He et al. 2016). Most existing networks can

be viewed as stacking of convolutional blocks, as shown in Figure 1(a). In this scheme, each block enlarges the receptive field; evidence has shown that more blocks capture semantically meaningful information at larger scale. In a conventional stacking structure, the output is generated by the last convolutional block, which doesn't have flexible scale information.

To be exact, we define aggregation as a network layer that learns to combine outputs of different layers. We call a group of aggregations *deep* if the output of lowest aggregated layer passes through multiple aggregations.

2.1 Iterative Deep Aggregation

We propose to aggregate the information at different layers across the network directly. Although this is rarely considered for image classification, people have proposed combining deep low resolution feature maps and shallow high resolution maps directly to produce high resolution prediction. Usually, the stack of convolutional blocks are divided into sub-networks by certain criterion such as sub-network output resolution. Then additional layers are added to combine layer outputs from larger to smaller scales. Prior work on 'shallow-skip' layers, e.g., U-Net (Ronneberger et al. 2015), FCN (Long et al. 2015) and FPN (Lin et al. 2017), provide connections that bypass traditional sequential layers. This concept is shown in Figure 1(b). However, in this scheme, the lowest aggregated layer only passes through one aggregation node, which leads to shallow aggregation for smallest scale.

We propose iterative deep aggregation shown in Figure 1(c). The aggregation starts from the smallest scale and the information is iteratively combined with bigger scale.

The iterative deep aggregation function I for a serious of inputs $\mathbf{x}_1, ..., \mathbf{x}_n$ with ascendantly deeper semantic information can be formally defined as

$$I(\mathbf{x}_1, ..., \mathbf{x}_n) = \begin{cases} \mathbf{x}_1 & \text{if } n = 1 \\ I(N(\mathbf{x}_1, \mathbf{x}_2), \mathbf{x}_3, ..., \mathbf{x}_n) & \text{otherwise,} \end{cases} \tag{1}$$

where N is the aggregation node.

2.2 Hierarchical Deep Aggregation

While iterative deep aggregation can combine outputs of sub-networks, it is less effective to combine individual blocks.

We propose another aggregation method that hierarchically combines the convolutional blocks as in Figure 1(d). In this structure, all the convolutional blocks get deep aggregation.

We can change the connection within the hierarchy to obtain deeper aggregation. Instead of sending the output of one block to another directly, we use the output of the substree as the input of block in the next substree, as shown in Figure 1(e). Then the inputs of the convolutional blocks are the aggregation of all previous blocks, which is richer than output of a single convolutional block.

Furthermore, we merge the parent aggregation node with its left child node, as shown in Figure 1(f). This can save the network parameters and the resulting aggregations are still deep based on our definition.

Our proposed hierarchical deep aggregation is formulated in Equation 2 below. T_n is the function for a hierarchy with depth n.

$$T_n(\mathbf{x}) = N(R^n_{n-1}(\mathbf{x}), R^n_{n-2}(\mathbf{x}), ..., R^n_1(\mathbf{x}), L^n_1(\mathbf{x}), L^n_2(\mathbf{x})), \tag{2}$$

where N is the aggregation node. R and L are defined as

$$L^n_2(\mathbf{x}) = B(L^n_1(\mathbf{x})), \quad L^n_1(\mathbf{x}) = B(R^n_1(\mathbf{x})),$$
$$R^n_m(\mathbf{x}) = \begin{cases} T_m(\mathbf{x}) & \text{if } m = n - 1 \\ T_m(R^n_{m+1}(\mathbf{x})) & \text{otherwise,} \end{cases}$$

where B represents the convolutional block.

3 Model elements

Aggregation Nodes The main function of an aggregation node is to combine and compress the information from their inputs. The nodes learn to select important information from their input layers and then project the representation to a lower dimension so that its output has the same dimension with the input from the last convolutional block before the node in the network. The number of inputs for IDA is always two in this paper, while HDA nodes have variable number of inputs depending on hierarchy depth.

Although any convolutional block can be used as an aggregation node for combination and compression, we choose to use a single convolutional layer as shown in Figure 2(b). This approach can avoid adding too much overhead to build the aggregation structures and it is easier to back propagate gradients through the single layer node during training. In image classification networks, all the nodes use 1×1 convolution. In semantic segmentation, we add additional iterative deep aggregation to upsample the dilated feature maps. 3×3 convolution is used in this case.

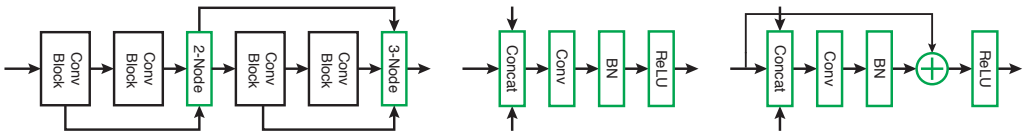
Residual connections are important to construct very deep convolutional networks (He et al. 2016). We can also add residual connections as shown in Figure 2(c). However, it is not clearly useful in the aggregation node. In HDA, the shortest path from any block to the root is at most the depth of the hierarchy, so diminishing or exploding gradients may not appear along the aggregation paths. In our experiments, we find that residual connection in node can help HDA when the deepest hierarchy has 4 levels or more, while it may hurt for networks with smaller hierarchy. Our base aggregation, i.e. N in Equation 1 and 2, is defined by:

$$N(\mathbf{x}_1, ..., \mathbf{x}_n) = \sigma(\text{BatchNorm}(\sum_i W_i \mathbf{x}_i + \mathbf{b})), \tag{3}$$

where σ is the non-linear activation, and $\mathbf{w}_i$ and $\mathbf{b}$ are the weights in the convolution. If residual connections are added, the equation becomes

$$N(\mathbf{x}_1, ..., \mathbf{x}_n) = \sigma(\text{BatchNorm}(\sum_i W_i \mathbf{x}_i + \mathbf{b}) + \mathbf{x}_n). \tag{4}$$

Note that the order of arguments for N matters and it should follow Equation 2.



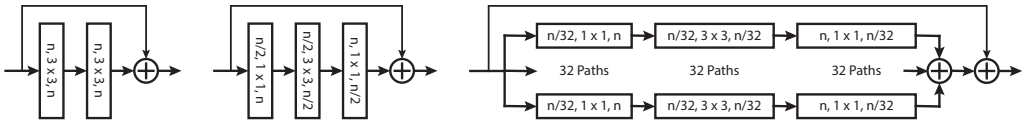
(a) 3-level hierarchical aggregation (b) Plain 3-node (c) Residual 3-node

Figure 2: Illustration of aggregation nodes. (a) shows another view of HDA. (b) and (c) show two general designs of nodes with 3 inputs. In (b), the inputs are concatenated and passed to a convolutional layer. In (c), after batch normalization, an additional residual connection adds input from the last block to the node output to facilitate gradient propagation for networks with more than 100 layers.

Convolutional Blocks Deep layer aggregation is a general architecture in the sense that it can be compatible with different kinds of convolutional blocks. It has no requirement on the internal block structure. In this paper, we study three kinds of convolutional blocks with residual connections. We reduce the ratio between output dimension and middle by half for both bottleneck blocks in He et al. (2016) and split block in Xie et al. (2017). In the split block, cardinality is 32. The blocks and their configurations are show in Figure 3.

4 Image Classification and Segmentation

We now discuss how to use the architectures and components proposed in the previous sections to build networks for image understanding tasks. To study the performance of our aggregated learned representation, we focus on image classification and semantic image segmentation, which only requires a single projection layer on top of the learned representation to make predictions.



(a) Basic (b) Bottleneck (c) Split

. We incorporate them within our DLA architectures.

Figure 3: Convolutional blocks used in this paper. Our aggregation architecture is as general as stacking layers, so we can use the building blocks of existing networks. The layer labels indicate output channels, kernel size and input channels. (a) and (b) are derived from He et al. (2016) and (c) from Xie et al. (2017).

4.1 Classification Networks

The classification networks can be divided into several levels and the layers with in each level have the same spatial resolution. We follow the structure in ResNet, where the final feature maps downsample the input images by 32 and a global average pooling is applied before softmax classification. As each downsampling shrinks the previous resolution by 2, there are 6 levels in total with the first level operating on the original image resolution. Each level is treated as a sub-network in this paper. We use hierarchical deep aggregation within each sub-network to connect the convolutional blocks and use iterative deep aggregation to connect the sub-networks. We can easily combine these two aggregations by sharing the aggregation nodes. In this case, we only need to change the root node at each hierarchy by combining Equation 1 and 2 to

$$T_n^r(\mathbf{x}) = N(S(\mathbf{x}), R_{n-1}^n(\mathbf{x}), R_{n-2}^n(\mathbf{x}), ..., R_1^n(\mathbf{x}), L_1^n(\mathbf{x}), L_2^n(\mathbf{x})), \tag{5}$$

where x is the output of the previous sub-network and S downsamples the spatial resolution of the input by 2. We use max pooling with kernel size 2 and stride 2 for downsampling. As in the DRN proposed by Yu et al. (2017), we used convolutional layers instead of max pooling in the first two levels. In the first level, a 7×7 convolution followed by a basic block (Figure 3(a)) is applied on the input image. The second level only has a basic block. For the other levels, we use the combination of HDA and IDA.

For a direct comparison of layers and parameters in different networks, we build networks with similar number of layers as ResNet-34, ResNet-50 and ResNet-101, although the exact same depth can not be achieved because of our new hierarchical structure. To test how efficient DLA can make use of parameters, we also build networks with much fewer parameters by shrinking the numbers of channels in the levels. The schemes of our tested networks are listed in Table 1.

4.2 Segmentation Networks

Semantic image segmentation can exploit the aggregation of multi-scale information since segmented objects can have dramatically different scales due to perspective distortion. We can easily adopt DLA for segmentation. Given a pretrained DLA, we can transform final feature map to higher resolution suitable for segmentation using dilated convolution (Yu and Koltun 2016). After dilation, the final feature maps downsample the original images by 8.

Further, we can use IDA to learn the upsampling of final feature map. We first project the outputs of Level 3 to Level 6 to 32 channels and then upsample the levels to the same resolution as Level 2. All the projection and upsampling parameters are learned together with the other network parameters. But upsampling is initialized by bilinear interpolation. Finally, we iteratively aggregate the features maps from Level 2 to Level 6 to learn a deep fusion of low and high level features. The aggregation nodes use 3×3 convolutions instead of 1×1. This is similar to hypercolumn classifier proposed by Hariharan et al. (2015). But hypercolumn classifier can be treated as shallow aggregation. Our aggregation also uses 3×3 convolution to combine pixel predictions in the neighbors. It is worth noting that we use two IDA in this case: one for connecting levels in the original classification network and the other for upsamping the predicted feature maps by 4 times.

5 Experiments

We test Deep Layer Aggregation networks on two tasks: image classification and semantic image segmentation. Our hypothesis is that for classification tasks, layer aggregation can combine informa-

Name	Block	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
DLA-36	Basic	16	32	1-64	2-128	2-256	1-512
DLA-48-B-s	Bottleneck	16	32	1-64	2-64	2-128	1-256
DLA-48-B-m	Bottleneck	16	32	1-64	2-128	2-256	1-512
DLA-62-B	Bottleneck	16	32	1-128	2-256	3-512	1-1024
DLA-104-B	Bottleneck	16	32	1-128	3-256	4-512	1-1024
DLA-48-S-s	Split	16	32	1-64	2-64	2-128	1-256
DLA-62-S-s	Split	16	32	1-64	2-64	3-128	1-256
DLA-62-S	Split	16	32	1-128	2-256	3-512	1-1024
DLA-104-S	Split	16	32	1-128	3-256	4-512	1-1024

Table 1: Deep Layer Aggregation networks used in this paper. The block structures are listed in Figure 3 and n is the channel number. Level 1 and 2 only show channel numbers. Level 3 to 4 show d-n and d is the depth of HDA in that level.

tion of different scales directly in general, which can make more efficient use of parameters and layer outputs than stacking of convolutional blocks. For fine grained classification, the DLA networks may learn better representation for detailed object difference through deep aggregation. On semantic segmentation, IDA can learn to upsample the pixel-level predictions, which may improve segmenting small objects with little additional parameters.

5.1 ImageNet Classification

We first train our networks on the ImageNet 2012 training set (Russakovsky et al. 2015). Similar to ResNet (He et al. 2016), training is performed by SGD for 120 epochs with momentum 0.9, weight decay 10^{-4} and batch size 256. We start the training with learning rate 0.1, which is reduced by 10 every 30 epochs. We use scale and aspect ratio augmentation (Xie et al. 2017), but not color perturbation. For fair comparison, we also train the ResNet models with the same training procedure. It leads to slightly better results than those reported in the original paper.

We evaluate the performance of trained models on the ImageNet 2012 validation set. The images are resized so that the shorter side has 256 pixels. Then central 224×224 crops are extracted from the images and fed into networks to measure prediction accuracy.

DLA vs ResNet compares DLA networks to ResNets with similar number of layers and same convolutional blocks as shown in Table 2. We find that DLA networks can achieve better performance than their baselines with fewer parameters. DLA-36 and ResNet-34 both use basic blocks, but DLA-36 has about 30% fewer parameters and about 1 point improvement in top-1 error rate. We usually expect diminishing returns of performances with deeper networks. Larger number of parameters can also learn the scale information in the networks with stacking structure. However, our results show that compared to ResNet-50 and ResNet-101, DLA networks can still outperform the baselines significantly with fewer parameters.

DLA vs ResNeXt shows that DLA is flexible enough to use different kind of convolutional blocks ans still have advantage in efficient parameter utilization as shown in Table 2. Our models based on the split blocks have much fewer parameters but they still have similar performance with ResNeXt models. For example, DLA-104-S has nearly the half number of parameters compared to ResNeXt-101, but the error rate difference is only 0.2%.

Compact models have received a lot of attention due to the limited hardware capability to run convolutional networks on consumer devices. We also test DLA networks with parameter number constraints to study how efficient DLA can make use of model parameters and layer outputs. We compare to SequeezeNet (Iandola et al. 2016) and MobileNet (Howard et al. 2017) in our experiments. The results are listed in Table 3. We find that DLA networks can outperform the other networks with same number of parameters. The performance is further improved with the split blocks proposed by Xie et al. (2017) are used. For a direct comparison with stacking structure, we also build and train ResNet-42-s and ResNet-54-s. They use the same convolutional blocks with DLA-48-B-s and similar number of parameters. The results show that DLA-48-B-s can still achieve much better classification accuracy.

	Top-1	Top-5	P
ResNet-34	26.0	8.4	21.8M
DLA-36	25.1	7.8	15.8M
ResNet-50	23.8	7.1	25.6M
DLA-62-B	22.8	6.4	22.3M
ResNet-101	22.5	6.3	44.5M
DLA-104-B	22.0	6.0	33.7M
ResNeXt-50	22.2	-	25.0M
DLA-62-S	22.2	6.2	17.7M
DLA-104-S	21.4	5.8	26.8M
ResNeXt-101	21.2	5.6	44.0M

Table 2: Comparison with ResNet and ResNeXt

	Top-1	Top-5	P
SqueezNet-A	42.5	19.7	1.2M
SqueezNet-B	39.6	17.5	1.2M
MobileNet	39.8	-	1.3M
ResNet-42-s	40.3	17.4	1.0M
ResNet-54-s	37.5	15.5	1.3M
DLA-48-B-s	36.1	14.3	1.3M
DLA-48-S-s	34.4	13.4	1.1M
DLA-62-S-s	32.8	12.4	1.3M

Table 3: Comparison with other small models

5.2 Fine-grained Categorization

We follow the Inception-style scale and aspect ration augmentation (Szegedy et al. 2015), AlexNet-style PCA-based noise(Krizhevsky et al. 2012) and the photometric distortions from (Howard 2013), in order to mitigate against over-fitting because of the limited size of fine-grained datasets. We use the same training procedure for all of fine-grained experiments. The training is performed by SGD with a mini-batch size of 64, while the learning rate starts from 0.01 and is then divided by 10 every 50 epochs, for 110 epochs in total. We use the same momentum, weight decay, mean, std, eigenvalues and eigenvectors as ImageNet Classification regardless of fine-grained datasets.

	#Class	#Train (per class)	#Test (per class)
Bird	200	5994 (30)	5794 (29)
Car	196	8144 (42)	8041 (41)
Plane	102	6667 (67)	3333 (33)
Food	101	75750 (750)	25250 (250)
NABirds	555	23929 (43)	24633 (44)
ILSVRC	1000	1,281,167 (1281)	100,000 (100)

Table 4: Statistic for fine-grained categorization datasets. Compared to large-scale generic classification datasets, they contain more fine-grained categories with fewer training samples.

	Bird	Car	Plane	Food	NABirds	P
DLA-48-B-s	76.4	88.1	86.5	82.4	72.7	1.3M
DLA-62-S-s	79.2	91.2	88.4	83.6	76.5	1.3M
DLA-48-B-m	79.4	91.6	88.0	84.5	77.1	4.5M
DLA-36	82.1	93.0	90.4	84.9	81.0	15.8M
DLA-62-B	82.8	93.4	90.8	86.9	81.6	22.3M
DLA-62-S	83.5	93.6	91.2	87.7	83.0	17.7M
V-Baseline	73.1	79.8	74.1	81.2	73.0	138.3M
V-Gao et al. (2016)	84.3	91.2	84.1	82.4	-	18.8M
V-Cui et al. (2017)	86.2	92.4	86.9	84.2	-	18.8M
R-Baseline	78.4	84.7	79.2	82.1	80.7	25.6M
R-Gao et al. (2016)	81.6	88.6	81.6	83.2	-	25.6M
R-Cui et al. (2017)	84.7	92.7	85.7	85.5	-	25.6M

Table 5: Comparison with state-of-the-art methods on fine-grained datasets. Image classification accuracy on Birds (Wah et al. 2011), Car (Krause et al. 2013), Plane (Maji et al. 2013), Food (Bossard et al. 2014), and NABirds (Van Horn et al. 2015). Higher is better. P is the number of parameters in each model. For fair comparison, we calculate the number of parameters for 1000-way classification. V- and R- indicate the base model as VGGNet-16 and ResNet-50, respectively.

We evaluate our model for fine-grained recognition on Birds (Wah et al. 2011), Car (Krause et al. 2013), Plane (Maji et al. 2013), Food (Bossard et al. 2014), and NABirds (Van Horn et al. 2015). The statistics of these datasets can be found in Table 4, while results are shown in Table 5. For fair comparison, we follow the experimental setup of Cui et al. (2017). We randomly crop 224×224 in resized 256×256 images for Bossard et al. (2014) and 448×448 in resized 512×512 for the rest of datasets, while keeping 224×224 input size for original VGGNet-16 (V-Baseline).

Most of our models are comparable to the state-of-the-art methods, without any additional annotations or specific module for fine-grained task. In particular, we establish new state-of-the-arts results on Car, Plane, Food, and NABirds datasets. In addition, our models with only several million parameters laos show very competitive performance. However, we do not get better results on Birds than the previous works, probably because it has fewer examples in each category than the other datasets do, as shown in Table 4.

5.3 Semantic Segmentation

In this section, we reports two experiments for urban scene understanding: using the CamVid dataset (Brostow et al. 2009), and the Cityscapes dataset (Cordts et al. 2016). We use the mean IoU score (Everingham et al. 2010) as evaluation metrics for both datasets. We only train our model on the training set without any other extra data. We do not use the validation set. We adopt the same optimization procedures with SGD for all the experiments. We start with learning rate 10^{-3} and momentum 10^{-4}. The learning rate is reduced to 10^{-4} after 800 epochs for CamVid and 200 epochs for Cityscapes. At each optimization step, we sample 8 images from the dataset and a crop is sampled from from each image in the batch with 50% chance mirroring. The crop size is 400 for CamVid and 896 for Cityscapes. We didn't use any other data augmentation so that we can compare the baseline performance for all the networks.

	Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Pole	Sidewalk	Bicyclist	mean IoU
Russell et al. (2009)	73.4	70.2	91.1	64.2	24.4	91.1	29.1	31.0	13.6	72.4	28.6	53.6
Tighe and Lazebnik (2010)	70.4	54.8	83.5	43.3	25.4	83.4	11.6	18.3	5.2	57.4	8.9	42.0
Liu and He (2015)	66.8	66.6	90.1	62.9	21.4	85.8	28.0	17.8	8.3	63.5	8.5	47.2
Badrinarayanan et al. (2015)	68.7	52.0	87.0	58.5	13.4	86.2	25.3	17.9	16.0	60.5	24.8	46.4
Chen et al. (2015)	81.5	74.6	89.0	82.2	42.3	92.2	48.4	27.2	14.3	75.4	50.1	61.6
Yu and Koltun (2016)	82.6	76.2	89.9	84.0	46.9	92.2	56.3	35.8	23.4	75.3	55.5	65.3
Kundu et al. (2016)	84.0	77.2	91.3	85.6	49.9	92.5	59.1	37.6	16.9	76	57.2	66.1
DLA-62-S-s	82.4	75.1	91.1	83.0	42.3	92.5	55.2	31.8	25.1	77.7	39.8	63.3
DLA-36	80.9	73.8	90.9	79.1	45.3	93.0	51.3	31.1	27.9	77.2	49.9	63.7
DLA-62-S-s†	83.1	76.6	92.4	82.6	47.4	93.7	57.4	31.3	33.5	80.9	46.4	66.0
DLA-36†	84.2	77.6	92.3	83.8	51.8	94.9	62.7	39.8	37.4	82.6	63.3	70.0

Table 6: Semantic segmentation results on the CamVid dataset. Methods marked with † used IDA to upsample the prediction results. IDA can improve the segmentation performance significantly with a small number of parameters.

	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	mean IoU	params
ResNet-50	96.9	77.4	90.3	35.8	42.8	59.0	66.8	74.5	91.6	57.0	93.4	78.7	55.3	92.1	43.2	59.5	36.2	52.0	75.2	67.3	25.6M
DRN-26	97.4	80.7	90.4	36.1	47.0	56.9	63.8	73.0	91.2	57.9	93.4	77.3	53.8	92.7	45.0	70.5	48.4	44.2	72.8	68.0	21.1M
DRN-42	97.7	82.2	91.2	40.5	52.6	59.2	66.7	74.6	91.7	57.7	94.1	79.1	56.0	93.6	56.0	74.3	54.7	50.9	74.1	70.9	31.2M
DLA-36	97.7	81.7	90.6	40.4	46.6	56.9	65.4	73.2	91.3	58.1	93.6	77.8	55.9	92.4	42.0	66.8	42.4	47.1	73.7	68.1	15.8M
DLA-36†	98.0	83.9	91.9	44.8	52.7	66.0	71.3	78.5	92.2	60.2	94.8	81.6	59.8	93.8	54.4	72.6	43.9	52.1	76.3	72.0	15.9M

Table 7: Performance of dilated residual networks on the Cityscapes validation set. Higher is better. Method marked with † used IDA to upsample the prediction results, while the other methods use bilinear interpolation to upsample the feature maps that are 8× smaller than input images in spatial resolution. The additional parameters for IDA upsampling is negligible. DRN-26 and DRN-42 are in DRN-C family proposed by Yu et al. (2017)

On CamVid (Sturgess et al. 2009), we use 367 training images, 100 validation images, and 233 test images with 11 semantic categories for CamVid. The results could be found in Table 6. The output prediction probability maps from vanilla dilated DLA networks downsample the input images by 8. We test two methods to upsample the predictions. One use $8\times$ bilinear interpolation and the other use LDA to upsample the prediction by 4 times followed by $2\times$ bilinear interpolation. The results are shown in Table 6. We find that DLA-36 with IDA upsampling can beat the previous methods by more than 4 points. DLA-62-S-s is also very competitive, although it only has 1.3 million parameters. It shows that DLA can make use of the small number of images and parameters in training more efficiently than the other networks.

We also test DLA-36 on Citycapes and the results are in Table 7. We use $2,975$ training images, 500 validation images, and $1,525$ test images with 19 semantic categories for Cityscapes. DLA-36 can perform similarly as DRN-26 proposed by Yu et al. (2017) with less parameters and without degriding layers proposed in that paper. IDA upsampling can also improve the performance significantly and beat the previous baseline methods.

6 Conclusion

In this paper, we study architecture designs to combine convolutional network layers directly with general deep structures and assess their benefits. We propose iterative deep aggregation (IDA), which combines outputs of subnetworks with dramatic scale difference, and hierarchical deep aggregation (HDA), which can organize network building blocks in a tree structure instead of connecting them sequentially. We use HDA and IDA together to build Deep Layer Aggregation (DLA) networks for image classification and segmentation. Experiments show that DLA networks can make more efficient use of parameters on large scale datasets. It can also perform very well on smaller datasets, probably due to direct combination of layer outputs. DLA networks establish new state-of-the-art results on several fine-grained classification datasets and on the CamVid image classification datasets.

References

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *ECCV*, 2014.

Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2009.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel pooling for convolutional neural networks. In *CVPR*, 2017.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *CVPR*, 2016.

Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 447–456, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Andrew G Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshops*, 2013.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun. Feature space optimization for semantic video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3168–3175, 2016.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

Buyu Liu and Xuming He. Multiclass semantic video segmentation with object-level active inference. In *CVPR*, 2015.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

Chris Russell, Pushmeet Kohli, Philip HS Torr, et al. Associative hierarchical crfs for object class image segmentation. In *ICCV*, 2009.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Paul Sturgess, Karteek Alahari, Lubor Ladicky, and Philip HS Torr. Combining appearance and structure from motion features for road scene understanding. In *BMVC*, 2009.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

Joseph Tighe and Svetlana Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. *ECCV*, 2010.

Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *CVPR*, 2015.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *CVPR*, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.