Deep Layer Aggregation

Fisher Yu Dequan Wang Evan Shelhamer Trevor Darrell
UC Berkeley

Abstract

Visual recognition requires rich representations that span levels from low to high, scales from small to large, and resolutions from fine to coarse. Even with the depth of features in a convolutional network, a layer in isolation is not enough: compounding and aggregating these representations improves inference of what and where. Architectural efforts are exploring many dimensions for network backbones, designing deeper or wider architectures, but how to best aggregate layers and blocks across a network deserves further attention. Although skip connections have been incorporated to combine layers, these connections have been "shallow" themselves, and only fuse by simple, one-step operations. We augment standard architectures with deeper aggregation to better fuse information across layers. Our deep layer aggregation structures iteratively and hierarchically merge the feature hierarchy to make networks with better accuracy and fewer parameters. Experiments across architectures and tasks show that deep layer aggregation improves recognition and resolution compared to existing branching and merging schemes.

1. Introduction

Representation learning and transfer learning now permeate computer vision as engines of recognition. The simple fundamentals of compositionality and differentiability give rise to an astonishing variety of deep architectures [23, 39, 37, 16, 47]. The rise of convolutional networks as the backbone of many visual tasks, ready for different purposes with the right task extensions and data [14, 35, 42], has made architecture search a central driver in sustaining progress. The ever-increasing size and scope of networks now directs effort into devising design patterns of modules and connectivity patterns that can be assembled systematically. This has yielded networks that are deeper and wider, but what about more closely connected?

More nonlinearity, greater capacity, and larger receptive fields generally improve accuracy but can be problematic for optimization and computation. To overcome these barriers, different blocks or modules have been incorporated to balance and temper these quantities, such as bottlenecks for dimensionality reduction [29, 39, 17] or residual, gated, and concatenative connections for feature and gradient propagation [17, 38, 19]. Networks designed according to these schemes have 100+ and even 1000+ layers.

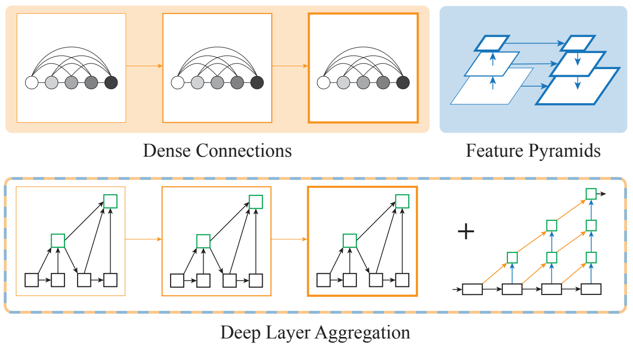

Figure 1: Deep layer aggregation unifies semantic and spatial fusion to better capture what and where. Our aggregation architectures encompass and extend densely connected networks and feature pyramid networks with hierarchical and iterative skip connections that deepen the representation and refine resolution.

Nevertheless, further exploration is needed on how to connect these layers and modules. Layered networks from LeNet [26] through AlexNet [23] to ResNet [17] stack layers and modules in sequence. Layerwise accuracy comparisons [11, 48, 35], transferability analysis [44], and representation visualization [48, 46] show that deeper layers extract more semantic and more global features, but these signs do not prove that the last layer is the ultimate representation for any task. In fact, skip connections have proven effective for classification and regression [19, 4] and more structured tasks [15, 35, 30]. Aggregation, like depth and width, is a critical dimension of architecture.

In this work, we investigate how to aggregate layers to better fuse semantic and spatial information for recognition and localization. Extending the "shallow" skip connections of current approaches, our aggregation architectures incorporate more depth and sharing. We introduce two structures for deep layer aggregation (DLA): iterative deep aggregation (IDA) and hierarchical deep aggregation (HDA). These structures are expressed through an architectural framework, independent of the choice of backbone, for compatibility with current and future networks. IDA focuses on fusing resolutions and scales while HDA focuses on merging features from all modules and channels. IDA follows the base hierarchy to refine resolution and aggregate scale stage-by-stage. HDA assembles its own hierarchy of tree-structured connections that cross and merge stages to aggregate different levels of representation. Our schemes can be combined to compound improvements.

Our experiments evaluate deep layer aggregation across standard architectures and tasks to extend ResNet [16] and ResNeXt [41] for large-scale image classification, fine-grained recognition, semantic segmentation, and boundary detection. Our results show improvements in performance, parameter count, and memory usage over baseline ResNet, ResNeXT, and DenseNet architectures. DLA achieve state-of-the-art results among compact models for classification. Without further architecting, the same networks obtain state-of-the-art results on several fine-grained recognition benchmarks. Recast for structured output by standard techniques, DLA achieves best-in-class accuracy on semantic segmentation of Cityscapes [8] and state-of-the-art boundary detection on PASCAL Boundaries [32]. Deep layer aggregation is a general and effective extension to deep visual architectures.

2. Related Work

We review architectures for visual recognition, highlight key architectures for the aggregation of hierarchical features and pyramidal scales, and connect these to our focus on deep aggregation across depths, scales, and resolutions.

The accuracy of AlexNet [23] for image classification on ILSVRC [34] signalled the importance of architecture for visual recognition. Deep learning diffused across vision by establishing that networks could serve as backbones, which broadcast improvements not once but with every better architecture, through transfer learning [11, 48] and meta-algorithms for object detection [14] and semantic segmentation [35] that take the base architecture as an argument. In this way GoogLeNet [39] and VGG [39] improved accuracy on a variety of visual problems. Their patterned components prefigured a more systematic approach to architecture.

Systematic design has delivered deeper and wider networks such as residual networks (ResNets) [16] and highway networks [38] for depth and ResNeXT [41] and FractalNet [25] for width. While these architectures all contribute their own structural ideas, they incorporated bottlenecks and shortened paths inspired by earlier techniques. Network-in-network [29] demonstrated channel mixing as a technique to fuse features, control dimensionality, and go deeper. The companion and auxiliary losses of deeply-supervised networks [27] and GoogLeNet [39] showed that it helps to keep learned layers and losses close. For the most part these architectures derive from innovations in connectivity: skipping, gating, branching, and aggregating.

Our aggregation architectures are most closely related to leading approaches for fusing feature hierarchies. The key axes of fusion are semantic and spatial. Semantic fusion, or aggregating across channels and depths, improves inference of what. Spatial fusion, or aggregating across resolutions and scales, improves inference of where. Deep layer aggregation can be seen as the union of both forms of fusion.

Densely connected networks (DenseNets) [19] are the dominant family of architectures for semantic fusion, designed to better propagate features and losses through skip connections that concatenate all the layers in stages. Our hierarchical deep aggregation shares the same insight on the importance of short paths and re-use, and extends skip connections with trees that cross stages and deeper fusion than concatenation. Densely connected and deeply aggregated networks achieve more accuracy as well as better parameter and memory efficiency.

Feature pyramid networks (FPNs) [30] are the dominant family of architectures for spatial fusion, designed to equalize resolution and standardize semantics across the levels of a pyramidal feature hierarchy through top-down and lateral connections. Our iterative deep aggregation likewise raises resolution, but further deepens the representation by nonlinear and progressive fusion. FPN connections are linear and earlier levels are not aggregated more to counter their relative semantic weakness. Pyramidal and deeply aggregated networks are better able to resolve what and where for structured output tasks.

3. Deep Layer Aggregation

We define aggregation as the combination of different layers throughout a network. In this work we focus on a family of architectures for the effective aggregation of depths, resolutions, and scales. We call a group of aggregations *deep* if it is compositional, nonlinear, and the earliest aggregated layer passes through multiple aggregations.

As networks can contain many layers and connections, modular design helps counter complexity by grouping and repetition. Layers are grouped into blocks, which are then grouped into stages by their feature resolution. We are concerned with aggregating the blocks and stages.

3.1. Iterative Deep Aggregation

Iterative deep aggregation follows the iterated stacking of the backbone architecture. We divide the stacked blocks of the network into stages according to feature resolution. Deeper stages are more semantic but spatially coarser. Skip connections from shallower to deeper stages merge scales

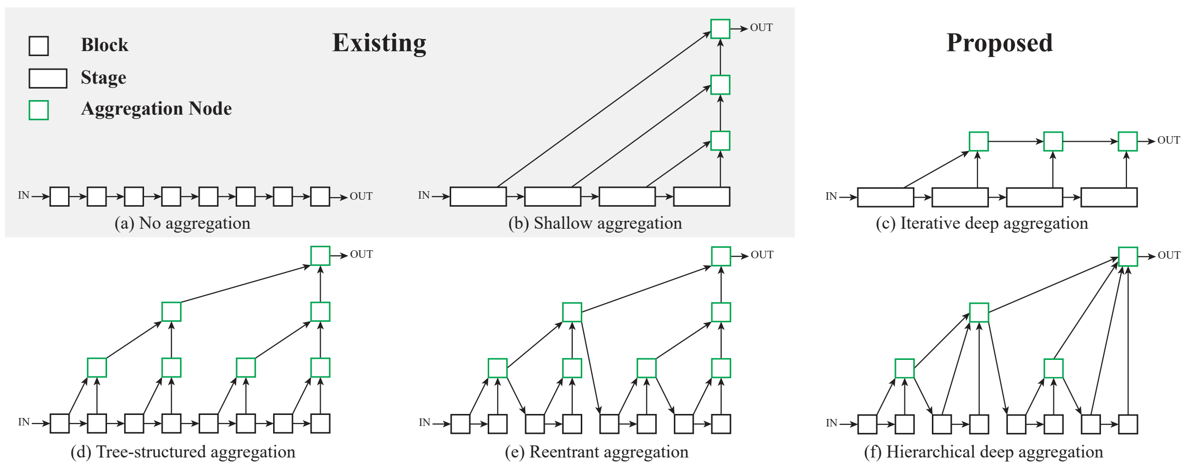
Block
Stage
Aggregation Node
Existing
Proposed
IN
OUT
(a) No aggregation
(b) Shallow aggregation
(c) Iterative deep aggregation
(d) Tree-structured aggregation
(e) Reentrant aggregation
(f) Hierarchical deep aggregation

Figure 2: Different approaches to aggregation. (a) composes blocks without aggregation as is the default for classification and regression networks. (b) combines parts of the network with skip connections, as is commonly used for tasks like segmentation and detection, but does so only shallowly by merging earlier parts in a single step each. We propose two deep aggregation architectures: (c) aggregates iteratively by reordering the skip connections of (b) such that the shallowest parts are aggregated the most for further processing and (d) aggregates hierarchically through a tree structure of blocks to better span the feature hierarchy of the network across different depths. (e) and (f) are refinements of (d) that deepen aggregation by routing intermediate aggregations back into the network and improve efficiency by merging successive aggregations at the same depth. Our experiments show the advantages of (c) and (f) for recognition and resolution.

and resolutions. However, the skips in existing work, e.g. FCN [35], U-Net [33], and FPN [30], are linear and aggregate the shallowest layers the least, as shown in Figure 2(b).

We propose to instead progressively aggregate and deepen the representation with IDA. Aggregation begins at the shallowest, smallest scale and then iteratively merges deeper, larger scales. In this way shallow features are refined as they are propagated through different stages of aggregation. Figure 2(c) shows the structure of IDA.

The iterative deep aggregation function I for a series of layers $\mathbf{x}_1, ..., \mathbf{x}_n$ with increasingly deeper and semantic information is formulated as

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

where N is the aggregation node.

3.2. Hierarchical Deep Aggregation

Hierarchical deep aggregation merges blocks and stages in a tree to preserve and combine feature channels. With HDA shallower and deeper layers are combined to learn richer combinations that span more of the feature hierarchy. While IDA effectively combines stages, it is insufficient for fusing the many blocks of a network, as it is still only sequential. The deep, branching structure of hierarchical aggregation is shown in Figure 2(d).

Having established the general structure of HDA we can improve its depth and efficiency. Rather than only routing intermediate aggregations further up the tree, we instead feed the output of an aggregation node back into the backbone as the input to the next sub-tree, as shown in Figure 2(e). This propagates the aggregation of all previous blocks instead of the preceding block alone to better preserve features. For efficiency, we merge aggregation nodes of the same depth (combining the parent and left child), as shown in Figure 2(f).

The hierarchical deep aggregation function T_n, with depth n, is formulated as

$$\begin{aligned} T_n(\mathbf{x}) = N(&R^n_{n-1}(\mathbf{x}), R^n_{n-2}(\mathbf{x}), ..., \\ &R^n_1(\mathbf{x}), L^n_1(\mathbf{x}), L^n_2(\mathbf{x})), \end{aligned} \quad (2)$$

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 a convolutional block.

3.3. Architectural Elements

Aggregation Nodes The main function of an aggregation node is to combine and compress their inputs. The nodes learn to select and project important information to maintain the same dimension at their output as a single input. In our architectures IDA nodes are always binary, while HDA nodes have a variable number of arguments depending on the depth of the tree.

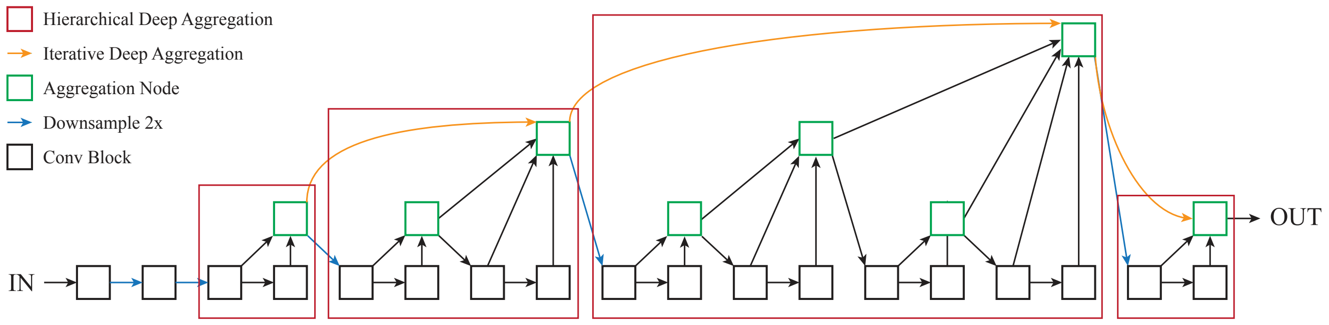

Figure 3: Deep layer aggregation learns to better extract the full spectrum of semantic and spatial information from a network. Iterative connections join neighboring stages to progressively deepen and spatially refine the representation. Hierarchical connections cross stages with trees that span the spectrum of layers to better propagate features and gradients.

Although an aggregation node can be based on any block or layer, for simplicity and efficiency we choose a single convolution followed by batch normalization and a nonlinearity. This avoids overhead for aggregation structures. In image classification networks, all the nodes use 1×1 convolution. In semantic segmentation, we add a further level of iterative deep aggregation to interpolate features, and in this case use 3×3 convolution.

As residual connections are important for assembling very deep networks, we can also include residual connections in our aggregation nodes. However, it is not immediately clear that they are necessary for aggregation. With 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 does matter and should follow Equation 2.

Blocks and Stages Deep layer aggregation is a general architecture family in the sense that it is compatible with different backbones. Our architectures make no requirements of the internal structure of the blocks and stages.

The networks we instantiate in our experiments make use of three types of residual blocks [17, 41]. Basic blocks combine stacked convolutions with an identity skip connection. Bottleneck blocks regularize the convolutional stack by reducing dimensionality through a 1×1 convolution. Split blocks diversify features by grouping channels into a number of separate paths (called the cardinality of the split). In this work, we reduce the ratio between the number of output and intermediate channels by half for both bottleneck and split blocks, and the cardinality of our split blocks is 32. Refer to the cited papers for the exact details of these blocks.

4. Applications

We now design networks with deep layer aggregation for visual recognition tasks. To study the contribution of the aggregated representation, we focus on linear prediction without further machinery. Our results do without ensembles for recognition and context modeling or dilation for resolution. Aggregation of semantic and spatial information matters for classification and dense prediction alike.

4.1. Classification Networks

Our classification networks augment ResNet and ResNeXT with IDA and HDA. These are staged networks, which group blocks by spatial resolution, with residual connections within each block. The end of every stage halves resolution, giving six stages in total, with the first stage maintaining the input resolution while the last stage is $32\times$ downsampled. The final feature maps are collapsed by global average pooling then linearly scored. The classification is predicted as the softmax over the scores.

We connect across stages with IDA and within and across stages by HDA. These types of aggregation are easily combined by sharing aggregation nodes. In this case, we only need to change the root node at each hierarchy by combining Equation 1 and 2. Our stages are downsampled by max pooling with size 2 and stride 2.

The earliest stages have their own structure. As in DRN [46], we replace max pooling in stages 1–2 with strided

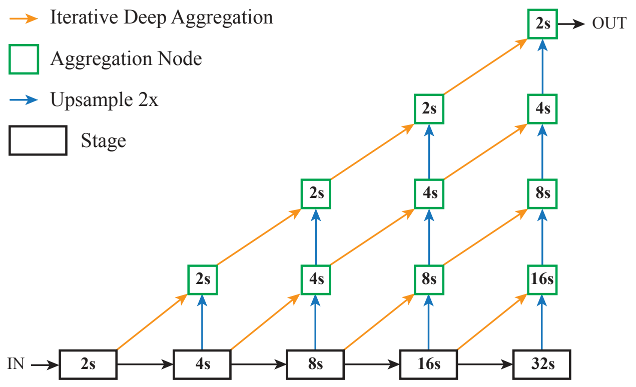

Figure 4: Interpolation by iterative deep aggregation. Stages are fused from shallow to deep to make a progressively deeper and higher resolution decoder.

convolution. The stage 1 is composed of a 7×7 convolution followed by a basic block. The stage 2 is only a basic block. For all other stages, we make use of combined IDA and HDA on the backbone blocks and stages.

For a direct comparison of layers and parameters in different networks, we build networks with a comparable number of layers as ResNet-34, ResNet-50 and ResNet-101. (The exact depth varies as to keep our novel hierarchical structure intact.) To further illustrate the efficiency of DLA for condensing the representation, we make compact networks with fewer parameters. Table 1 lists our networks and Figure 3 shows a DLA architecture with HDA and IDA.

4.2. Dense Prediction Networks

Semantic segmentation, contour detection, and other image-to-image tasks can exploit the aggregation to fuse local and global information. The conversion from classification DLA to fully convolutional DLA is simple and no different than for other architectures. We make use of interpolation and a further augmentation with IDA to reach the necessary output resolution for a task.

IDA for interpolation increases both depth and resolution by projection and upsampling as in Figure 4. All the projection and upsampling parameters are learned jointly during the optimization of the network. The upsampling steps are initialized to bilinear interpolation and can then be learned as in [35]. We first project the outputs of stages 3–6 to 32 channels and then interpolate the stages to the same resolution as stage 2. Finally, we iteratively aggregate these stages to learn a deep fusion of low and high level features. While having the same purpose as FCN skip connections [35], hypercolumn features [15], and FPN top-down connections [30], our aggregation differs in approach by going from shallow-to-deep to further refine features. Note that we use IDA twice in this case: once to connect stages in the backbone network and again to recover resolution.

5. Results

We evaluate our deep layer aggregation networks on a variety of tasks: image classification on ILSVRC, several kinds of fine-grained recognition, and dense prediction for semantic segmentation and contour detection. After establishing our classification architecture, we transfer these networks to the other tasks with little to no modification. DLA improves on or rivals the results of special-purpose networks.

5.1. ImageNet Classification

We first train our networks on the ImageNet 2012 training set [34]. Similar to ResNet [16], 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 [41] with color perturbation. For fair comparison, we also train the ResNet models with the same training procedure. This leads to slight improvements over the original results.

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 numbers of layers and the same convolutional blocks as shown in Figure 5. We find that DLA networks can achieve better performance with fewer parameters. DLA-34 and ResNet-34 both use basic blocks, but DLA-34 has about 30% fewer parameters and $\sim$ 1 point of improvement in top-1 error rate. We usually expect diminishing returns of performance of deeper networks. 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 convolutional blocks and still have advantage in accuracy and parameter efficiency as shown in Figure 5. Our models based on the split blocks have much fewer parameters but they still have similar performance with ResNeXt models. For example, DLA-X-102 has nearly the half number of parameters compared to ResNeXt-101, but the error rate difference is only 0.2%.

DLA vs. DenseNet compares DLA with the dominant architecture for semantic fusion and feature re-use. DenseNets are composed of dense blocks that aggregate all of their layers by concatenation and transition blocks that reduce dimensionality for tractability. While these networks can aggressively reduce depth and parameter count by feature re-use, concatenation is a memory-intensive fusion operation. DLA achieves higher accuracy with lower memory usage because the aggregation node fan-in size is log of the total number of convolutional blocks in HDA.

Name	Block	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6
DLA-34	Basic	16	32	1-64	2-128	2-256	1-512
DLA-46-C	Bottleneck	16	32	1-64	2-64	2-128	1-256
DLA-60	Bottleneck	16	32	1-128	2-256	3-512	1-1024
DLA-102	Bottleneck	16	32	1-128	3-256	4-512	1-1024
DLA-169	Bottleneck	16	32	2-128	3-256	5-512	1-1024
DLA-X-46-C	Split	16	32	1-64	2-64	2-128	1-256
DLA-X-60-C	Split	16	32	1-64	2-64	3-128	1-256
DLA-X-60	Split	16	32	1-128	2-256	3-512	1-1024
DLA-X-102	Split	16	32	1-128	3-256	4-512	1-1024

Table 1: Deep layer aggregation networks for classification. Stages 1 and 2 show the number of channels n while further stages show d-n where d is the aggregation depth. Models marked with "-C" are compact and only have $\sim$1 million parameters.

Compact models have received a lot of attention due to the limited capabilities of consumer hardware for running convolutional networks. We design parameter constrained DLA networks to study how efficiently DLA can aggregate and re-use features. We remove color perturbation and set the minimum cropping area to be 0.25 because small models do not have enough capacity for the aggressive data augmentation. We compare to SqueezeNet [20], which shares a block design similar to our own. Table 2 shows that DLA is more accurate with the same number of parameters. Furthermore DLA is more computationally efficient by operation count.

	Top-1	Top-5	Params	FMAs
SqueezNet-A	42.5	19.7	1.2M	1.70B
SqueezNet-B	39.6	17.5	1.2M	0.72B
DLA-46-C	35.1	13.3	1.3M	0.58B
DLA-X-46-C	34.0	13.0	**1.1M**	**0.53B**
DLA-X-60-C	**32.0**	**11.6**	1.3M	0.59B

Table 2: Comparison with compact models. DLA is more accurate at the same number of parameters while inference takes fewer operations (counted by fused multiply-adds).

5.2. Fine-grained Recognition

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. The other hyperparameters are fixed to their settings for ImageNet classification. In order to mitigate over-fitting, we carry out the following data augmentation: Inception-style scale and aspect ratio variation [39], AlexNet-style PCA color noise[23], and the photometric distortions of [18].

We evaluate our models on various fine-grained recognition datasets: Bird (CUB) [40], Car [22], Plane [31], and Food [5]. The statistics of these datasets can be found in Table 3, while results are shown in Figure 6. For fair comparison, we follow the experimental setup of [9]: we randomly crop 448×448 in resized 512×512 for all the datasets, while keeping 224×224 input size for original VGGNet.

Our results improve or rival the state-of-the-art without further annotations or specific modules for fine-grained recognition. In particular, we establish new state-of-the-arts results on Car, Plane, and Food datasets. Furthermore, our models are competitive while having only several million parameters. However, our results are not better than state-of-the-art on Birds, although note that this dataset has fewer instances per class so further regularization might help.

	#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)
ILSVRC	1000	1,281,167 (1281)	100,000 (100)

Table 3: Statistics for fine-grained recognition datasets. Compared to generic, large-scale classification, these tasks contain more specific classes with fewer training instances.

5.3. Semantic Segmentation

We report experiments for urban scene understanding on CamVid [6] and Cityscapes [8]. Cityscapes is a larger-scale, more challenging dataset for comparison with other methods while CamVid is more convenient for examining ablations. We use the standard mean intersection-over-union (IoU) score [12] as the evaluation metric for both datasets. Our networks are trained only on the training set without the usage of validation or other further data.

CamVid has 367 training images, 100 validation images, and 233 test images with 11 semantic categories. We start the training with learning rate 0.01 and divide it by 10 after 800 epochs. The results are shown in Table 5. We find that models with downsampling rate 2 consistently outperforms those downsampling by 8. We also try to augment the data

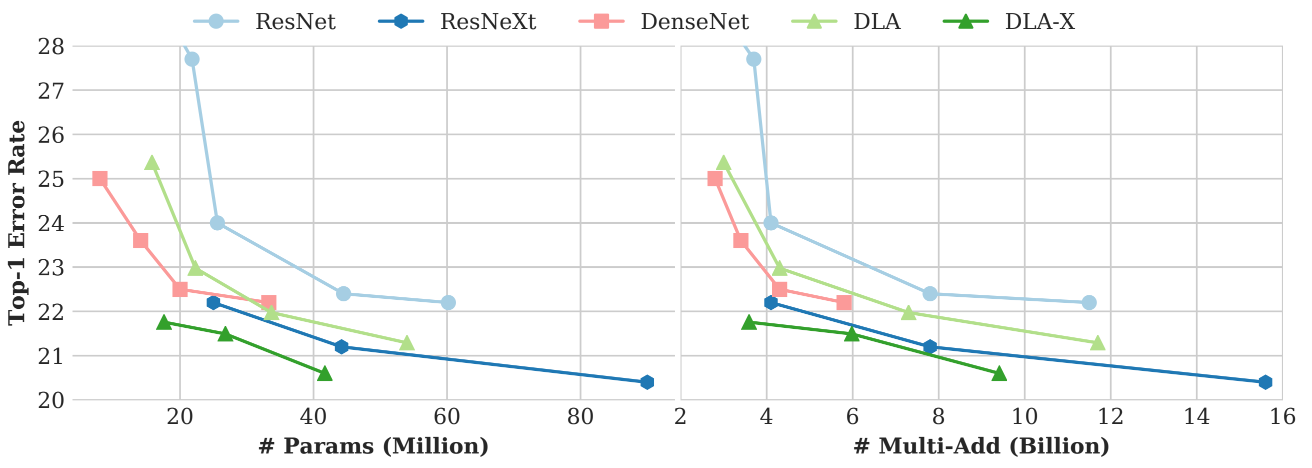

Figure 5: Evaluation of DLA on ILSVRC. DLA/DLA-X have ResNet/ResNeXt backbones respectively. DLA achieves the highest accuracies with fewer parameters and fewer computation.

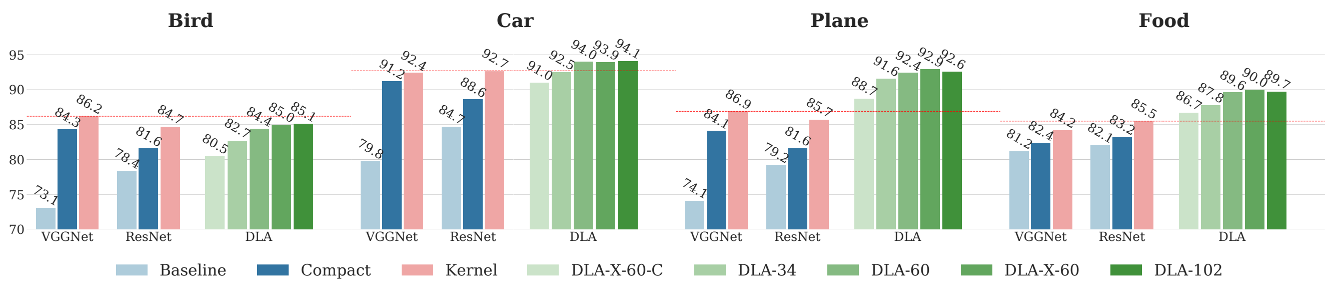

Figure 6: Comparison with state-of-the-art methods on fine-grained datasets. Image classification accuracy on Bird [40], Car [22], Plane [31], and Food [5]. Higher is better. P is the number of parameters in each model. For fair comparison, we calculate the number of parameters for 1000-way classification. V- and R- indicate the base model as VGGNet-16 and ResNet-50, respectively. The numbers of Baseline, Compact [13] and Kernel [9] are directly cited from [9].

by randomly rotating the images between [-10, 10] degrees and randomly scaling the images between 0.5 and 2. The final results are significantly better than prior methods.

Cityscapes has $2,975$ training images, 500 validation images, and $1,525$ test images with 19 semantic categories. Following previous works [49], we adopt the poly learning rate $(1 - \frac{\text{epoch}-1}{\text{total epoch}})^{0.9}$ with momentum 0.9 and train the model for 500 epochs with batch size 16. The starting learning rate is 0.01 and the crop size is chosen to be 864. We also augment the data by randomly rotating within 10 degrees and scaling between 0.5 and 2. The validation results are shown in 4. Surprisingly, DLA-34 performs very well on this dataset and it is as accurate as DLA-102. It should be noted that fine spatial details do not contribute much for this choice of metric. RefineNet [28] is the strongest network in the same class of methods without the computational costs of additional data, dilation, and graphical models. To make a fair comparison, we evaluate in the same multi-scale fashion as that approach with image scales of [0.5, 0.75, 1, 1.25, 1.5] and sum the predictions. DLA improves by 2+ points.

5.4. Boundary Detection

Boundary detection is an exacting task of localization. Although as a classification problem it is only a binary task of whether or not a boundary exists, the metrics require precise spatial accuracy. We evaluate on classic BSDS [1] with multiple human boundary annotations and PASCAL boundaries [32] with boundaries defined by instances masks of select semantic classes. The metrics are accuracies at different thresholds, the optimal dataset scale (ODS) and more lenient optimal image scale (OIS), as well as average precision (AP). Results are shown in for BSDS in Table 6 and the precision-recall plot of Figure 7 and for PASCAL boundaries in Table 7.

To address this task we follow the training procedure of HED [42]. In line with other deep learning methods, we take the consensus of human annotations on BSDS and only supervise our network with boundaries that three or more

Method	Split	mIoU
DLA-34 8s		73.5
DLA-34 2s	Val	75.1
DLA-102 2s		74.4
FCN-8s [35]		65.3
RefineNet-101 [28]	Test	73.6
DLA-102		75.3
DLA-169		**75.9**

Table 4: Evaluation on Cityscapes to compare strides on validation and to compare against existing methods on test. DLA is the best-in-class among methods in the same setting.

Method	ODS	OIS	AP
SE [10]	0.746	0.767	0.803
DeepEdge [3]	0.753	0.772	0.807
DeepContour [36]	0.756	0.773	0.797
HED [42]	0.788	0.808	0.840
CEDN [43]†	0.788	0.804	0.821
UberNet [21] (1-Task)†	0.791	0.809	**0.849**
DLA-34 8s	0.760	0.772	0.739
DLA-34 4s	0.767	0.778	0.751
DLA-34 2s	0.794	0.808	0.787
DLA-102 2s	**0.803**	**0.813**	0.781

Table 6: Evaluation on BSDS († indicates outside data). ODS and OIS are state-of-the-art, but AP suffers due to recall.

Method	Train	ODS	OIS	AP
SE [10]		0.541	0.570	0.486
HED [43]	BSDS	0.553	0.585	0.518
DLA-34 2s		0.642	0.668	0.624
DLA-102 2s		0.648	0.674	0.623
DSBD [32]		0.643	0.663	0.650
M-DSBD [32]	PASCAL	0.652	0.678	0.674
DLA-34 2s		0.743	0.757	**0.763**
DLA-102 2s		**0.754**	**0.766**	0.752

Table 7: Evaluation on PASCAL Boundaries. DLA is state-of-the-art.

annotators agree on. Following [43], we give the boundary labels 10 times weight of the others. For inference we simply run the net forward, and do not make use of ensembles or multi-scale testing. Assessing the role of resolution by comparing strides of 8, 4, and 2 we find that high output resolution is critical for accurate boundary detection. We also find that deeper networks does not continue improving the prediction performance on BSDS.

On both BSDS and PASCAL boundaries we achieve

Method	mIoU
SegNet [2]	46.4
DeepLab-LFOV [7]	61.6
Dilation8 [45]	65.3
FSO [24]	66.1
DLA-34 8s	66.7
DLA-34 2s	68.6
DLA-102 2s	**71.0**

Table 5: Evaluation on CamVid. Higher depth and resolution help. DLA is state-of-the-art.

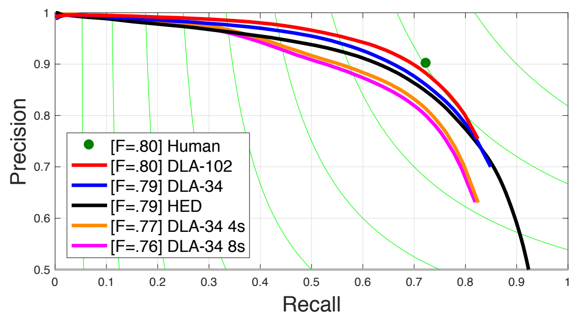

Figure 7: Precision-recall evaluation on BSDS. DLA is the closest to human performance.

state-of-the-art ODS and OIS scores. In contrast the AP on BSDS is surprisingly low, so to understand why we plot the precision-recall curve in Figure 7. Our approach has lower recall, but this is explained by the consensus ground truth not covering all of the individual, noisy boundaries. At the same time it is the closest to human performance. On the other hand we achieve state-of-the-art AP on PASCAL boundaries since it has a single, consistent notion of boundaries. When training on BSDS and transferring to PASCAL boundaries the improvement is minor, but training on PASCAL boundaries itself with $\sim 10\times$ the data delivers more than 10% relative improvement over competing methods.

6. Conclusion

Aggregation is a decisive aspect of architecture, and as the number of modules multiply their connectivity is made all the more important. By relating architectures for aggregating channels, scales, and resolutions we identified the need for deeper aggregation, and addressed it by iterative deep aggregation and hierarchical deep aggregation. Our models are more accurate and make more efficient use of parameters and computation than baseline networks. Our aggregation extensions improve on dominant architectures like residual and densely connected networks. Bridging the gaps of architecture makes better use of layers in aggregate.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 2011. 7

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015. 8

[3] G. Bertasius, J. Shi, and L. Torresani. DeepEdge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR*, 2015. 8

[4] C. M. Bishop. *Pattern recognition and machine learning*, page 229. Springer-Verlag New York, 2006. 1

[5] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101–mining discriminative components with random forests. In *ECCV*, 2014. 6, 7

[6] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2009. 6

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 8

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 6

[9] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, and S. Belongie. Kernel pooling for convolutional neural networks. In *CVPR*, 2017. 6, 7

[10] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 8

[11] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 1, 2

[12] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 6

[13] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *CVPR*, 2016. 7

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and segmentation. *TPAMI*, 2015. 1, 2

[15] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 1, 5

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 5

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

[18] A. G. Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013. 6

[19] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 1, 2

[20] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 6

[21] I. Kokkinos. Ubernet: Training auniversal'convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. *arXiv preprint arXiv:1609.02132*, 2016. 8

[22] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshops*, 2013. 6, 7

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2, 6

[24] A. Kundu, V. Vineet, and V. Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, 2016. 8

[25] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR*, 2017. 2

[26] 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. 1

[27] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570, 2015. 2

[28] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. In *CVPR*, 2017. 7, 8

[29] M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2014. 1, 2

[30] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1, 2, 3, 5

[31] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 6, 7

[32] V. Premachandran, B. Bonev, X. Lian, and A. Yuille. Pascal boundaries: A semantic boundary dataset with

a deep semantic boundary detector. In *WACV*, 2017. 2, 7, 8

[33] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015. 3

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 2, 5

[35] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *TPAMI*, 2016. 1, 2, 3, 5, 8

[36] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *CVPR*, 2015. 8

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[38] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. In *NIPS*, 2015. 1, 2

[39] 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. 1, 2, 6

[40] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 6, 7

[41] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 2, 4, 5

[42] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015. 1, 7, 8

[43] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang. Object contour detection with a fully convolutional encoder-decoder network. In *CVPR*, 2016. 8

[44] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014. 1

[45] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 8

[46] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *CVPR*, 2017. 1, 4

[47] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1

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

[49] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *arXiv preprint arXiv:1612.01105*, 2016. 7