CrossMark

Semantic Understanding of Scenes Through the ADE20K Dataset

Bolei Zhou[1] · Hang Zhao[2] · Xavier Puig[2] · Tete Xiao[3] · Sanja Fidler[4] · Adela Barriuso[2] · Antonio Torralba[2]

Abstract

Semantic understanding of visual scenes is one of the holy grails of computer vision. Despite efforts of the community in data collection, there are still few image datasets covering a wide range of scenes and object categories with pixel-wise annotations for scene understanding. In this work, we present a densely annotated dataset *ADE20K*, which spans diverse annotations of scenes, objects, parts of objects, and in some cases even parts of parts. Totally there are 25k images of the complex everyday scenes containing a variety of objects in their natural spatial context. On average there are 19.5 instances and 10.5 object classes per image. Based on ADE20K, we construct benchmarks for scene parsing and instance segmentation. We provide baseline performances on both of the benchmarks and re-implement state-of-the-art models for open source. We further evaluate the effect of synchronized batch normalization and find that a reasonably large batch size is crucial for the semantic segmentation performance. We show that the networks trained on ADE20K are able to segment a wide variety of scenes and objects.

1 Introduction

Semantic understanding of visual scenes is one of the holy grails of computer vision. The emergence of large-scale image datasets like ImageNet (Russakovsky et al. 2015), COCO (Lin et al. 2014) and Places (Zhou et al. 2014), along with the rapid development of the deep convolutional neural network (CNN) approaches, has brought great advancements to visual scene understanding. Nowadays, given a visual scene of a living room, a robot equipped with a trained CNN can accurately predict the scene category. However, to freely navigate in the scene and manipulate the objects inside, the robot has far more information to digest from the input image: it has to recognize and localize not only the objects like sofa, table, cup, and TV, but also their parts, e.g., a seat of a sofa or a handle of a cup, to allow proper manipulation, as well as to segment the stuff like floor, wall and ceiling for spatial navigation.

Recognizing and segmenting objects and stuff at pixel level remains one of the key problems in scene understanding. Going beyond the image-level recognition, the pixel-level scene understanding requires a much denser annotation of scenes with a large set of objects. However, the current datasets have a limited number of objects [e.g., COCO (Lin et al. 2014), Pascal (Everingham et al. 2010)] and in many cases those objects are not the most common objects that one encounters in the world (like frisbees or baseball bats), or the datasets only cover a limited set of scenes [e.g., Cityscapes (Cordts et al. 2016)]. Some notable exceptions are Pascal-Context (Mottaghi et al. 2014) and the SUN database (Xiao et al. 2010). However, Pascal-Context contains scenes still primarily focused on 20 object classes, while SUN has noisy labels at object level.

The motivation of this work is to construct a dataset that has densely annotated images (every pixel has a semantic label) with a large and unrestricted open-vocabulary. The

Communicated by Bernt Schiele.

✉ Bolei Zhou
  bzhou@ie.cuhk.edu.hk

[1] Department of Information Engineering, The Chinese University of Hong Kong, Shatin, China

[2] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, USA

[3] School of Electronic Engineering and Computer Science, Peking University, Beijing, China

[4] Department of Computer Science, University of Toronto, Toronto, Canada

∑ Springer

images in our dataset are manually segmented in great detail, covering a diverse set of scene, object and object part categories. The challenge for collecting such annotations is finding reliable annotators, as well as the fact that labeling is difficult if the class list is not defined in advance. On the other hand, open vocabulary naming also suffers from naming inconsistencies across different annotators. In contrast, our dataset was annotated by a single expert annotator, providing extremely detailed and exhaustive image annotations. On average, our annotator labeled 29 annotation segments per image, compared to 16 segments per image labeled by external annotators (like workers from Amazon Mechanical Turk). Furthermore, the data consistency and quality are much higher than that of external annotators. Figure 1 shows examples from our dataset.

The preliminary result of this work is published at Zhou et al. (2017). Compared to the previous conference paper, we include more description of the dataset, more baseline results on the scene parsing benchmark, the introduction of the instance segmentation benchmark and its baseline results, as well as the effect of synchronized batch norm and the joint training of objects and parts. We also include the contents of the Places Challenges that we hosted at ECCV'16 and ICCV'17 and the analysis on the challenge results.

The sections of this work are organized as follows. In Sect. 3 we describe the construction of the ADE20K dataset and its statistics. In Sect. 4 we introduce the two pixel-wise scene understanding benchmarks that we build upon ADE20K: scene parsing and instance segmentation. We train and evaluate several baseline networks on the benchmarks. We also re-implement and open-source several state-of-the-art scene parsing models and evaluate the effect of batch

normalization size. In Sect. 5 we introduce the Places Challenges at ECCV'16 and ICCV'17 based on the benchmarks of the ADE20K, as well as the qualitative and quantitative analysis on the challenge results. In Sect. 6 we train a network to jointly segment objects and their parts. Section 7 further explores the applications of the scene parsing networks for hierarchical semantic segmentation and automatic scene content removal. Section 8 concludes this work.

2 Related Work

Many datasets have been collected for the purpose of semantic understanding of scenes. We review the datasets according to the level of details of their annotations, then briefly go through the previous work of semantic segmentation networks.

Object Classification/Detection Datasets Most of the large-scale datasets typically only contain labels at the image level or provide bounding boxes. Examples include ImageNet (Russakovsky et al. 2015), Pascal (Everingham et al. 2010), and KITTI (Geiger et al. 2012). ImageNet has the largest set of classes, but contains relatively simple scenes. Pascal and KITTI are more challenging and have more objects per image, however, their classes and scenes are more constrained.

Semantic Segmentation Datasets Existing datasets with pixel-level labels typically provide annotations only for a subset of foreground objects [20 in PASCAL VOC (Everingham et al. 2010) and 91 in Microsoft COCO (Lin et al. 2014)]. Collecting dense annotations where all pixels are labeled is much more challenging. Such efforts include Pascal-
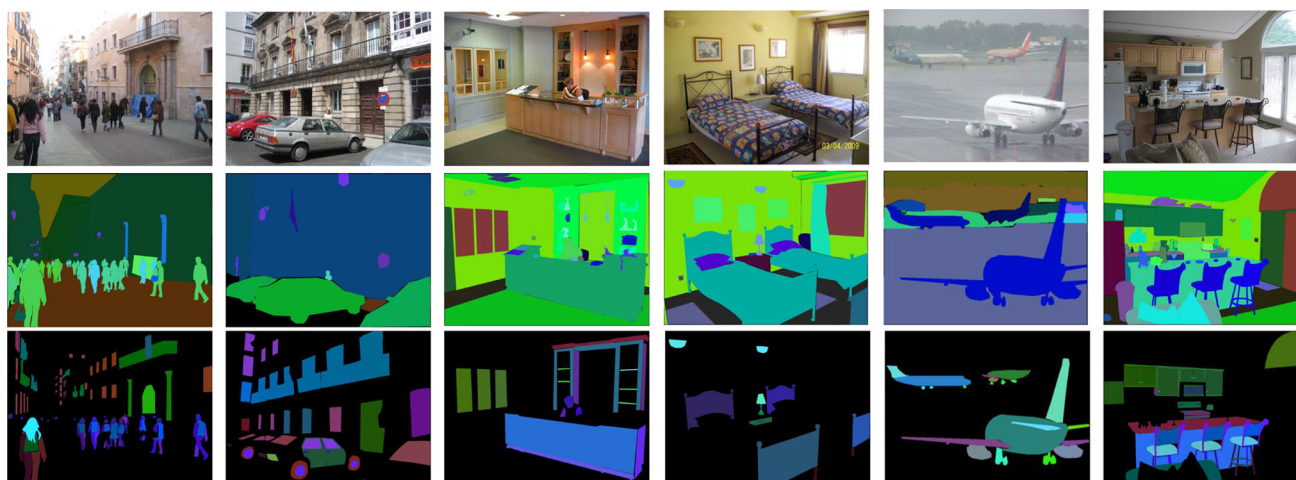

Fig. 1 Images in ADE20K dataset are densely annotated in detail with objects and parts. The first row shows the sample images, the second row shows the annotation of objects, and the third row shows the annotation of object parts. The color scheme encodes both the object categories and object instances, in which different object categories have large color difference while different instances from the same object category have small color difference (e.g., different person instances in first image have slightly different colors)

Context (Mottaghi et al. 2014), NYU Depth V2 (Nathan Silberman and Fergus 2012), SUN database (Xiao et al. 2010), SUN RGB-D dataset (Song et al. 2015), CityScapes dataset (Cordts et al. 2016), and OpenSurfaces (Bell et al. 2013, 2015). Recently COCO stuff dataset (Caesar et al. 2017) provides additional stuff segmentation complementary to the 80 object categories in COCO dataset, while COCO attributes dataset (Patterson and Hays 2016) annotates attributes for some objects in COCO dataset. Such a dataset with progressive enhancement of diverse annotations over the years makes great progress to modern development of image datasets.

Datasets with Objects, Parts and Attributes Two datasets were released that go beyond the typical labeling setup by also providing pixel-level annotation for the object parts, i.e., Pascal-Part dataset (Chen et al. 2014), or material classes, i.e., OpenSurfaces (Bell et al. 2013, 2015). We advance this effort by collecting very high-resolution images of a much wider selection of scenes, containing a large set of object classes per image. We annotated both stuff and object classes, for which we additionally annotated their parts, and parts of these parts. We believe that our dataset, ADE20K, is one of the most comprehensive datasets of its kind. We provide a comparison between datasets in Sect. 3.6.

Semantic Segmentation Models With the success of convolutional neural networks (CNN) for image classification (Krizhevsky et al. 2012), there is growing interest for semantic pixel-wise labeling using CNNs with dense output, such as the fully CNN (Long et al. 2015), deconvolutional neural networks (Noh et al. 2015), encoder-decoder SegNet (Badrinarayanan et al. 2017), multi-task network cascades (Dai et al. 2016), and DilatedVGG (Chen et al. 2016; Yu and Koltun 2016). They are benchmarked on Pascal dataset with impressive performance on segmenting the 20 object classes. Some of them (Long et al. 2015; Badrinarayanan et al. 2017; Zhao et al. 2017a) are evaluated on Pascal Context (Mottaghi et al. 2014) or SUN RGB-D dataset (Song et al. 2015) to show the capability to segment more object classes in scenes. Joint stuff and object segmentation is explored in Dai et al. (2015) which uses pre-computed superpixels and feature masking to represent stuff. Cascade of instance segmentation and categorization has been explored in Dai et al. (2016). A multiscale pyramid pooling module is proposed to improve the scene parsing (Zhao et al. 2017b). A recent multi-task segmentation network UperNet is proposed to segment visual concepts from different levels (Xiao et al. 2018).

3 ADE20K: Fully Annotated Image Dataset

In this section, we describe the construction of our ADE20K dataset and analyze its statistics.

3.1 Image Annotation

For our dataset, we are interested in having a diverse set of scenes with dense annotations of all the presenting visual concepts. The visual concepts can be: (1) discrete object which is a thing with a well-defined shape, e.g., car, person; (2) stuff which contains amorphous background regions, e.g., grass, sky; or (3) object part, which is a component of some existing object instance which has some functional meaning, such as head or leg. Images come from the LabelMe (Russell et al. 2008), SUN datasets (Xiao et al. 2010), and Places (Zhou et al. 2014) and were selected to cover the 900 scene categories defined in the SUN database. Images were annotated by a single expert worker using the LabelMe interface (Russell et al. 2008). Figure 2 shows a snapshot of the annotation interface and one fully segmented image. The worker provided three types of annotations: object and stuff segments with names, object parts, and attributes. All object instances are segmented independently so that the dataset could be used to train and evaluate detection or segmentation algorithms.

Segments in the dataset are annotated via polygons. Given that the objects appearing in the dataset are fully annotated, even in the regions where they are occluded, there are multiple areas where the polygons from different objects overlap. In order to convert the annotated polygons into a segmentation mask, these are sorted in every image by depth layers. Background classes like *sky* or *wall* are set as the farthest layer. The rest of depths of objects are set as follows: when a polygon is fully contained inside another polygon, the object from the inner polygon is given a closer depth layer. When objects only partially overlap, we look at the region of intersection between the two polygons, and set the closest object as the one whose polygon has more points in the region of intersection. Once objects have been sorted, the segmentation mask is constructed by iterating over the objects in decreasing depth to ensure that object parts never occlude whole objects and no object is occluded by its parts.

Datasets such as COCO (Lin et al. 2014), Pascal (Everingham et al. 2010) or Cityscapes (Cordts et al. 2016) start by defining a set of object categories of interest. However, when labeling all the objects in a scene, working with a predefined list of objects is not possible as new categories appear frequently (see Fig. 6d). Here, the annotator created a dictionary of visual concepts where new classes were added constantly to ensure consistency in object naming.

Object parts are associated with object instances. Note that parts can have parts too, and these associations are labeled as well. For example, the *rim* is a part of a *wheel*, which in turn is part of a *car*. A *knob* is a part of a *door* that can be part of a *cabinet*. A subset of the part hierarchy tree is shown in Fig. 3 with a depth of 4.
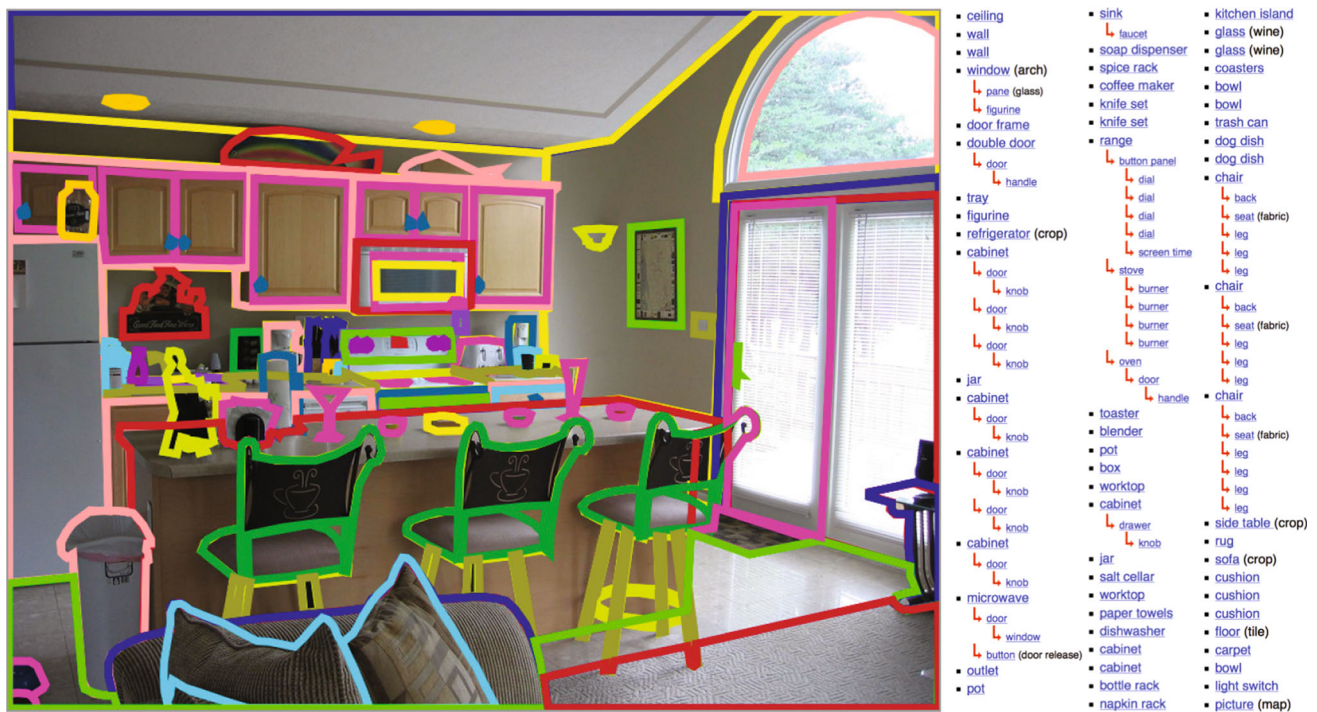
Fig. 2 Annotation interface, the list of the objects and their associated parts in the image
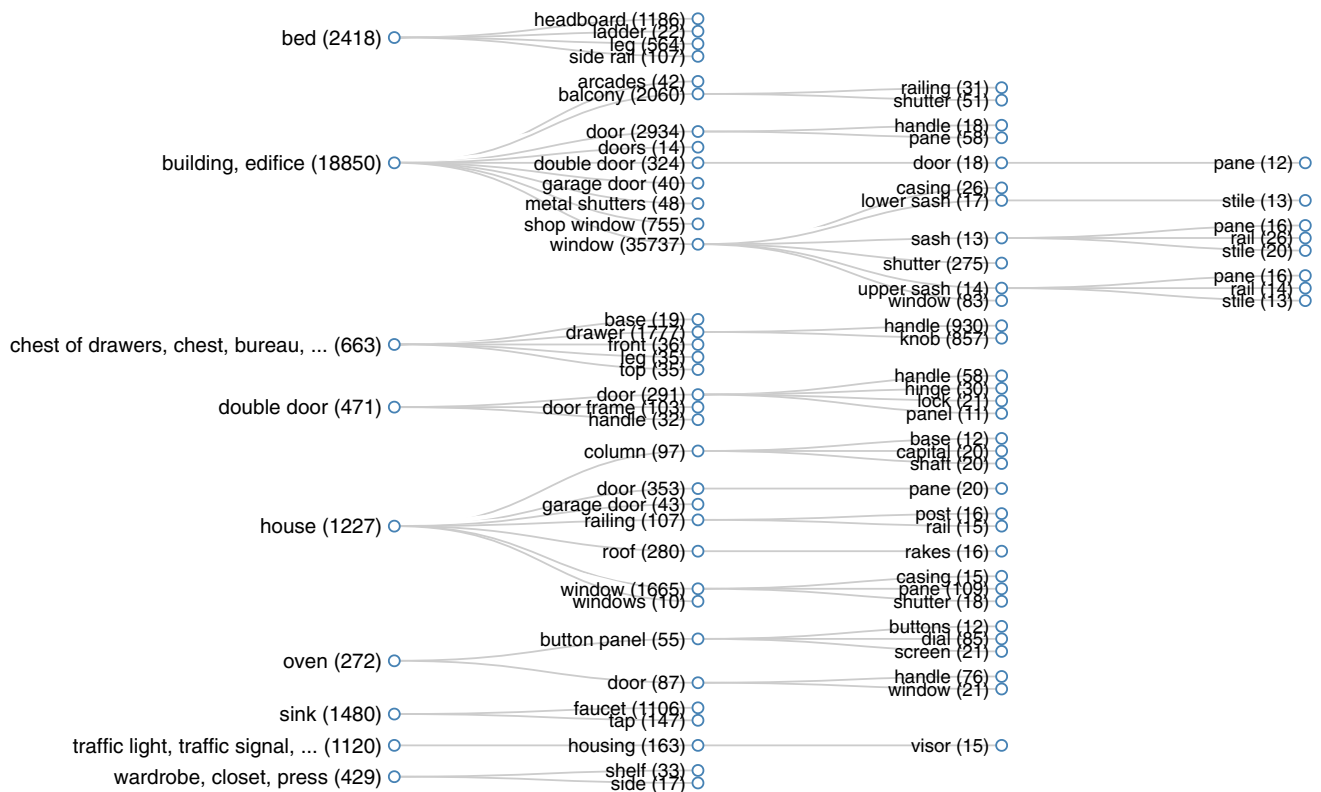

Fig. 3 Section of the relation tree of objects and parts for from dataset. Numbers indicate the number of instances for each object. The full relation tree is available at the dataset web-page
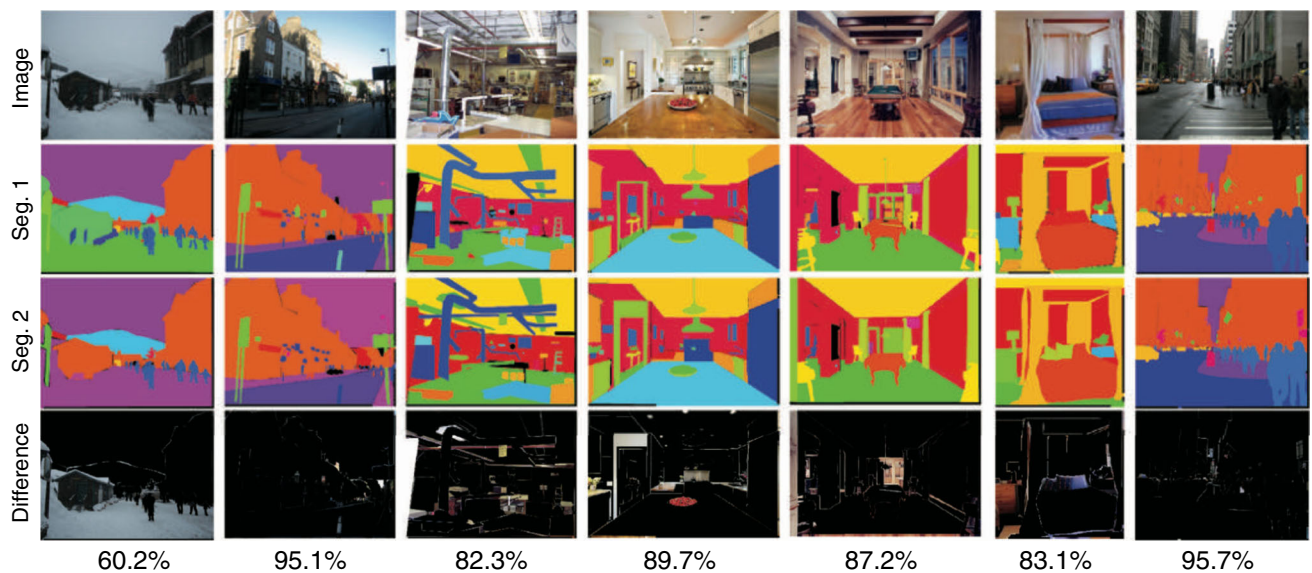
Fig. 4 Analysis of annotation consistency. Each column shows an image and two segmentations done by the same annotator at different times. Bottom row shows the pixel discrepancy when the two segmentations are subtracted, while the number at the bottom shows the percentage of pixels with the same label. On average across all reannotated images, 82.4% of pixels got the same label. In the example in the first column the percentage of pixels with the same label is relatively low because the annotator labeled the same region as *snow* and *ground* during the two rounds of annotation. In the third column, there were many objects in the scene and the annotator missed some between the two segmentations

3.2 Dataset Summary

After annotation, there are 20,210 images in the training set, 2000 images in the validation set, and 3000 images in the testing set. There are in total 3169 class labels annotated, among which 2693 are object and stuff classes while 476 are object part classes. All the images are exhaustively annotated with objects. Many objects are also annotated with their parts. For each object there is additional information about whether it is occluded or cropped, and other attributes. The images in the validation set are exhaustively annotated with parts, while the part annotations are not exhaustive over the images in the training set. Sample images and annotations from the ADE20K dataset are shown in Fig. 1.

3.3 Annotation Consistency

Defining a labeling protocol is relatively easy when the labeling task is restricted to a fixed list of object classes. However, it becomes challenging when the class list is open-ended. As the goal is to label all the objects within each image, the list of classes grows unbounded. Many object classes appear only a few times across the entire collection of images. However, those rare object classes cannot be ignored as they might be important elements for the interpretation of the scene. Labeling in these conditions becomes difficult because we need to keep a growing list of all the object classes in order to have a consistent naming across the entire dataset. Despite the best effort of the annotator, the process is not free from noise.

To analyze the annotation consistency we took a subset of 61 randomly chosen images from the validation set, then asked our annotator to annotate them again (there is a time difference of six months). One expects that there are some differences between the two annotations. A few examples are shown in Fig. 4. On average, 82.4% of the pixels got the same label. The remaining 17.6% of pixels had some errors for which we grouped into three error types as follows:

– *Segmentation quality* Variations in the quality of segmentation and outlining of the object boundary. One typical source of error arises when segmenting complex objects such as buildings and trees, which can be segmented with different degrees of precision. This type of error emerges in 5.7% of the pixels.
– *Object naming* Differences in object naming (due to ambiguity or similarity between concepts, for instance, calling a big car a *car* in one segmentation and a *truck* in the another one, or a palm tree a *tree*. This naming issue emerges in 6.0% of the pixels. These errors can be reduced by defining a very precise terminology, but this becomes much harder with a large growing vocabulary.
– *Segmentation quantity* Missing objects in one of the two segmentations. There is a very large number of objects in each image and some images might be annotated more thoroughly than others. For example, in the third column
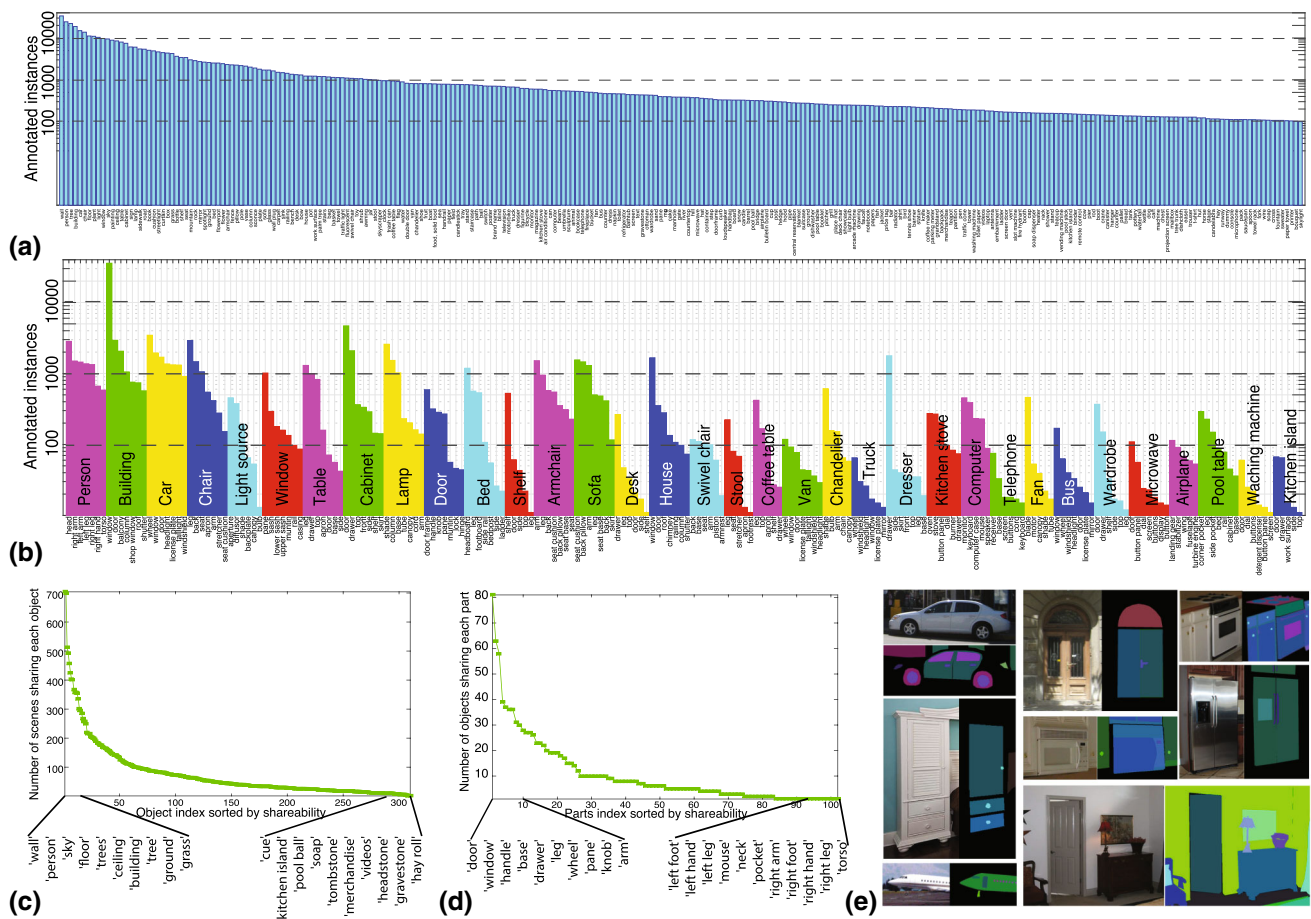
Fig. 5 **a** Object classes sorted by frequency. Only the top 270 classes with more than 100 annotated instances are shown. 68 classes have more than a 1000 segmented instances. **b** Frequency of parts grouped by objects. There are more than 200 object classes with annotated parts. Only objects with 5 or more parts are shown in this plot (we show at most 7 parts for each object class). **c** Objects ranked by the number of scenes of which they are part. **d** Object parts ranked by the number of objects of which they are part. **e** Examples of objects with doors. The bottom-right image is an example where the door does not behave as a part

of Fig. 4 the annotator missed some small objects in different annotations. Missing labels account for 5.9% of the error pixels. A similar issue existed in segmentation datasets such as the Berkeley Image segmentation dataset (Martin et al. 2001).

The median error values for the three error types are: 4.8%, 0.3%, and 2.6%, which shows that the mean value is dominated by a few images, and that the most common type of error is segmentation quality.

To further compare the annotation done by our single expert annotator and the AMT-like annotators, 20 images from the validation set are annotated by two invited external annotators, both with prior experience in image labeling. The first external annotator had 58.5% of inconsistent pixels compared to the segmentation provided by our annotator, and the second external annotator had 75% of the inconsistent pixels. Many of these inconsistencies are due to the poor

quality of the segmentations provided by external annotators [as it has been observed with AMT which requires multiple verification steps for quality control (Lin et al. 2014)]. For the best external annotator (the first one), 7.9% of pixels have inconsistent segmentations (just slightly worse than our annotator), 14.9% have inconsistent object naming and 35.8% of the pixels correspond to missing objects, which is due to the much smaller number of objects annotated by the external annotator in comparison with the ones annotated by our expert annotator. The external annotators labeled on average 16 segments per image while our annotator provided 29 segments per image.

3.4 Dataset Statistics

Figure 5a shows the distribution of ranked object frequencies. The distribution is similar to a Zipf's law and is typically found when objects are exhaustively annotated in images
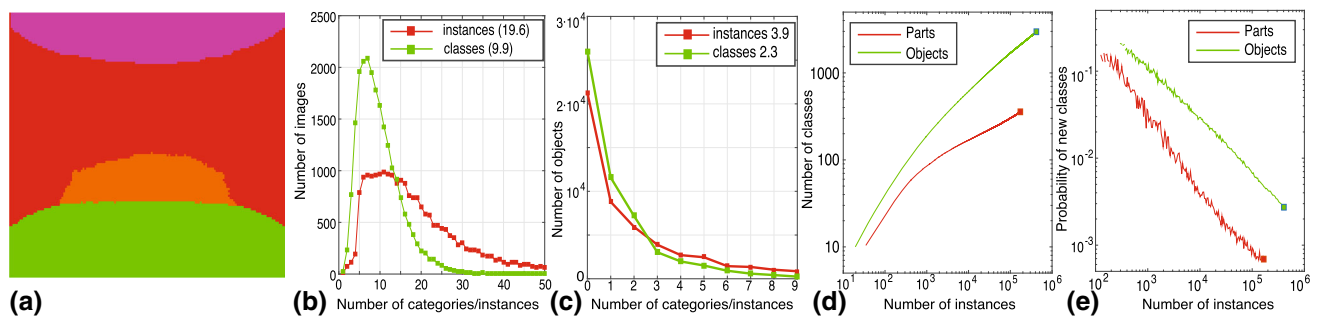
Fig. 6 **a** Mode of the object segmentations contains *sky*, *wall*, *building* and *floor*. **b** Histogram of the number of segmented object instances and classes per image. **c** Histogram of the number of segmented part instances and classes per object. **d** Number of classes as a function of segmented instances (objects and parts). The squares represent the current state of the dataset. **e** Probability of seeing a new object (or part) class as a function of the number of instances

Table 1 Comparison with existing datasets with semantic segmentation

	Images	Obj. inst.	Obj. classes	Part inst.	Part classes	Obj. classes per image
COCO	123,287	886,284	91	0	0	3.5
ImageNet[a]	476,688	534,309	200	0	0	1.7
NYU Depth V2	1,449	34,064	894	0	0	14.1
Cityscapes	25,000	65,385	30	0	0	12.2
SUN	16,873	313,884	4479	0	0	9.8
OpenSurfaces	22,214	71,460	160	0	0	N/A
PascalContext	10,103	$\sim$ 104,398[b]	540	181,770	40	5.1
ADE20K	22,210	434,826	2693	175,961	476	9.9

[a]Has only bounding boxes (no pixel-level segmentation). Sparse annotations
[b]PascalContext dataset does not have instance segmentation. In order to estimate the number of instances, we find connected components (having at least 150 pixels) for each class label

(Spain and Perona 2010; Xiao et al. 2010). They differ from the ones from datasets such as COCO or ImageNet where the distribution is more uniform resulting from manual balancing.

Figure 5b shows the distribution of annotated parts grouped by the objects to which they belong and sorted by frequency within each object class. Most object classes also have a non-uniform distribution of part counts. Figure 5c, d show how objects are shared across scenes and how parts are shared by objects. Figure 5e shows the variability in the appearances of the part *door*.

The mode of the object segmentations is shown in Fig. 6a and contains the four objects (from top to bottom): *sky*, *wall*, *building* and *floor*. When using simply the mode to segment the images, it gets, on average, 20.9% of the pixels of each image right. Figure 6b shows the distribution of images according to the number of distinct classes and instances. On average there are 19.5 instances and 10.5 object classes per image, larger than other existing datasets (see Table 1). Figure 6c shows the distribution of parts.

As the list of object classes is not predefined, there are new classes appearing over time of annotation. Figure 6d shows the number of object (and part) classes as the number of annotated instances increases. Figure 6e shows the probability that instance $n+1$ is a new class after labeling n instances. The more segments we have, the smaller the probability that we will see a new class. At the current state of the dataset, we get one new object class every 300 segmented instances.

3.5 Object-Part Relationships

We analyze the relationships between the objects and object parts annotated in ADE20K. In the dataset, 76% of the object instances have associated object parts, with an average of 3 parts per object. The class with the most parts is *building*, with 79 different parts. On average, 10% of the pixels correspond to object parts. A subset of the relation tree between objects and parts can be seen in Fig. 3.

The information about objects and their parts provides interesting insights. For instance, we can measure in what proportion one object is part of another to reason about how strongly tied these are. For the object *tree*, the most common parts are *trunk* or *branch*, whereas the least common are *fruit*, *flower* or *leaves*.

The object-part relationships can also be used to measure similarities among objects and parts, providing information about objects tending to appear together or sharing similar affordances. We measure the similarity between two parts as the common objects each one is part of. The most similar part to *knob* is *handle*, sharing objects such as *drawer*, *door* or *desk*. Objects can similarly be measured by the parts they have in common. As such, the most similar objects to *chair* are *armchair*, *sofa* and *stool*, sharing parts such as *rail*, *leg* or *seat base*.

3.6 Comparison with Other Datasets

We compare ADE20K with existing datasets in Table 1. Compared to the largest annotated datasets, COCO (Lin et al. 2014) and Imagenet (Russakovsky et al. 2015), our dataset comprises of much more diverse scenes, where the average number of object classes per image is 3 and 6 times larger, respectively. With respect to SUN (Xiao et al. 2010), ADE20K is roughly 35% larger in terms of images and object instances. However, the annotations in our dataset are much richer since they also include segmentation at the part level. Such annotation is only available for the Pascal-Context/Part dataset (Mottaghi et al. 2014; Chen et al. 2014) which contains 40 distinct part classes across 20 object classes. Note that we merged some of their part classes to be consistent with our labeling (e.g., we mark both *left leg* and *right leg* as the same semantic part *leg*). Since our dataset contains part annotations for a much wider set of object classes, the number of part classes is almost 9 times larger in our dataset.

An interesting fact is that any image in ADE20K contains at least 5 objects, and the maximum number of object instances per image reaches 273, and 419 instances, when counting parts as well. This shows the high annotation complexity of our dataset.

4 Pixel-Wise Scene Understanding Benchmarks

Based on the data of the ADE20K, we construct two benchmarks for pixel-wise scene understanding: scene parsing and instance segmentation:

- *Scene parsing* Scene parsing is to segment the whole image densely into semantic classes, where each pixel is assigned a class label such as the region of *tree* and the region of *building*.
- *Instance segmentation* Instance segmentation is to detect the object instances inside an image and further generate the precise segmentation masks of the objects. Its difference compared to the task of scene parsing is that in scene parsing there is no instance notion for the seg-

mented regions, instead in instance segmentation if there are three persons in a scene, the network is required to segment each one of the person regions.

We introduce the details of each task and the baseline models we train as below.

4.1 Scene Parsing Benchmark

We select the top 150 categories ranked by their total pixel ratios[1] in the ADE20K dataset and build a scene parsing benchmark of ADE20K, termed as *SceneParse150*. Among the 150 categories, there are 35 stuff classes (i.e., *wall, sky, road*) and 115 discrete object classes (i.e., *car, person, table*). The annotated pixels of the 150 classes occupy 92.75% of all the pixels of the dataset, where the stuff classes occupy 60.92%, and discrete object classes occupy 31.83%.

We map the WordNet synsets with each one of the object names, then build up a WordNet tree through the hypernym relations of the 150 categories shown in Fig. 7. We can see that these objects form several semantic clusters in the tree, such as the *furniture* synset node containing *cabinet*, *desk*, *pool table*, and *bench*, the *conveyance* node containing *car*, *truck*, *boat*, and *bus*, as well as the *living thing* node containing *shrub*, *grass*, *flower*, and *person*. Thus, the structured object annotation given in the dataset bridge the image annotation to a wider knowledge base.

As for baseline networks for scene parsing on our benchmark, we train several semantic segmentation networks: SegNet (Badrinarayanan et al. 2017), FCN-8s (Long et al. 2015), DilatedVGG, DilatedResNet (Chen et al. 2016; Yu and Koltun 2016), two cascade networks proposed in Zhou et al. (2017) where the backbone models are SegNet and DilatedVGG. We train these models on NVIDIA Titan X GPUs.

Results are reported in four metrics commonly used for semantic segmentation (Long et al. 2015):

- *Pixel accuracy* indicates the proportion of correctly classified pixels;
- *Mean accuracy* indicates the proportion of correctly classified pixels averaged over all the classes.
- *Mean IoU* indicates the intersection-over-union between the predicted and ground-truth pixels, averaged over all the classes.
- *Weighted IoU* indicates the IoU weighted by the total pixel ratio of each class.

---

[1] As the original images in the ADE20K dataset have various sizes, for simplicity we rescale the large-sized images to make their minimum heights or widths as 512 in the SceneParse150 benchmark.
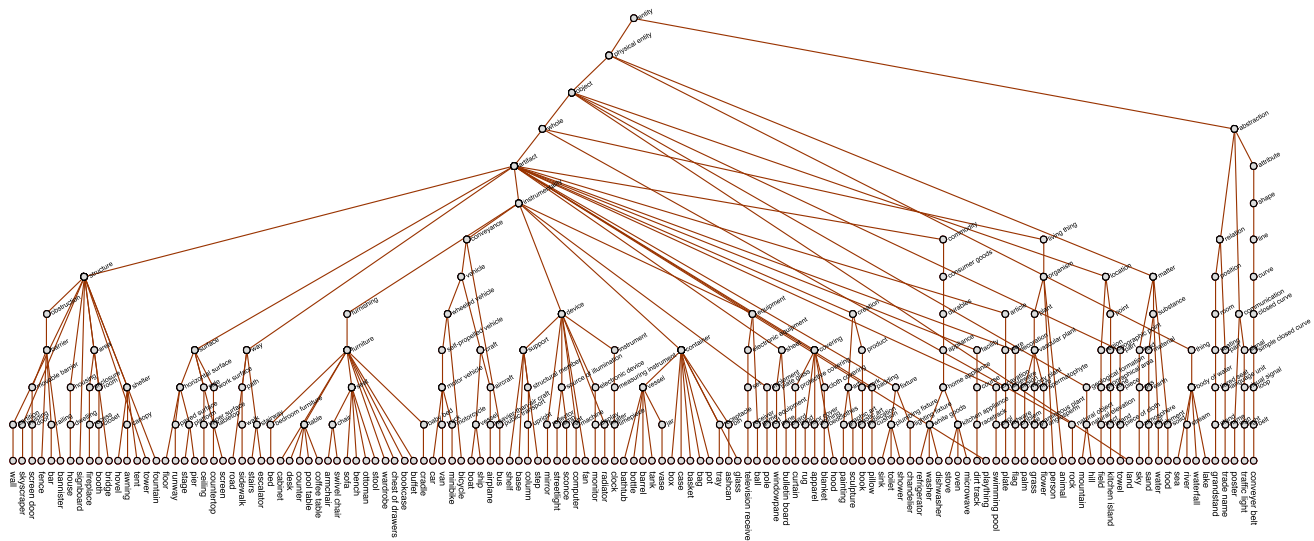
Fig. 7 Wordnet tree constructed from the 150 objects in the SceneParse150 benchmark. Clusters inside the wordnet tree represent various hierarchical semantic relations among objects

Table 2 Baseline performance on the validation set of SceneParse150

Networks	Pixel Acc. (%)	Mean Acc. (%)	Mean IoU	Weighted IoU
FCN-8s	71.32	40.32	0.2939	0.5733
SegNet	71.00	31.14	0.2164	0.5384
DilatedVGG	73.55	44.59	0.3231	0.6014
DilatedResNet-34	76.47	45.84	0.3277	0.6068
DilatedResNet-50	76.40	45.93	0.3385	0.6100
Cascade-SegNet	71.83	37.90	0.2751	0.5805
Cascade-DilatedVGG	74.52	45.38	0.3490	0.6108

Since some classes like *wall* and *floor* occupy far more pixels of the images, pixel accuracy is biased to reflect the accuracy over those few large classes. Instead, mean IoU reflects how accurately the model classifies each discrete class in the benchmark. The scene parsing data and the development toolbox are released in the Scene Parsing Benchmark website.[2]

The segmentation performance of the baseline networks on SceneParse150 is listed in Table 2. Among the baselines, the networks based on dilated convolutions achieve better results in general than FCN and SegNet. Using the cascade framework, the performance is further improved. In terms of mean IoU, Cascade-SegNet and Cascade-DilatedVGG outperform SegNet and DilatedVGG by 6% and 2.5%, respectively.

Qualitative scene parsing results from the validation set are shown in Fig. 8. We observe that all the baseline networks can give correct predictions for the common, large object and stuff classes, the difference in performance comes mostly from small, infrequent objects and how well they han-

dle details. We further plot the IoU performance of all the 150 categories given by the baseline model DilatedResNet-50 in Fig. 9. We can see that the best segmented categories are stuffs like *sky*, *building* and *road*; the worst segmented categories are objects that are usually small and have few pixels, like *blanket*, *tray* and *glass*.

4.2 Open-Sourcing the State-of-the-Art Scene Parsing Models

Since the introduction of SceneParse150 firstly in 2016, it has become a standard benchmark for evaluating new semantic segmentation models. However, the state-of-the-art models are in different libraries (Caffe, PyTorch, Tensorflow) while training code of some models are not released, which makes it hard to reproduce the original results reported in the paper. To benefit the research community, we re-implement several state-of-the-art models in PyTorch and open-source them.[3] Particularly, we implement: (1) the plain

---

[2] http://sceneparsing.csail.mit.edu.

[3] Re-implementation of the state-of-the-art models are released at https://github.com/CSAILVision/semantic-segmentation-pytorch.
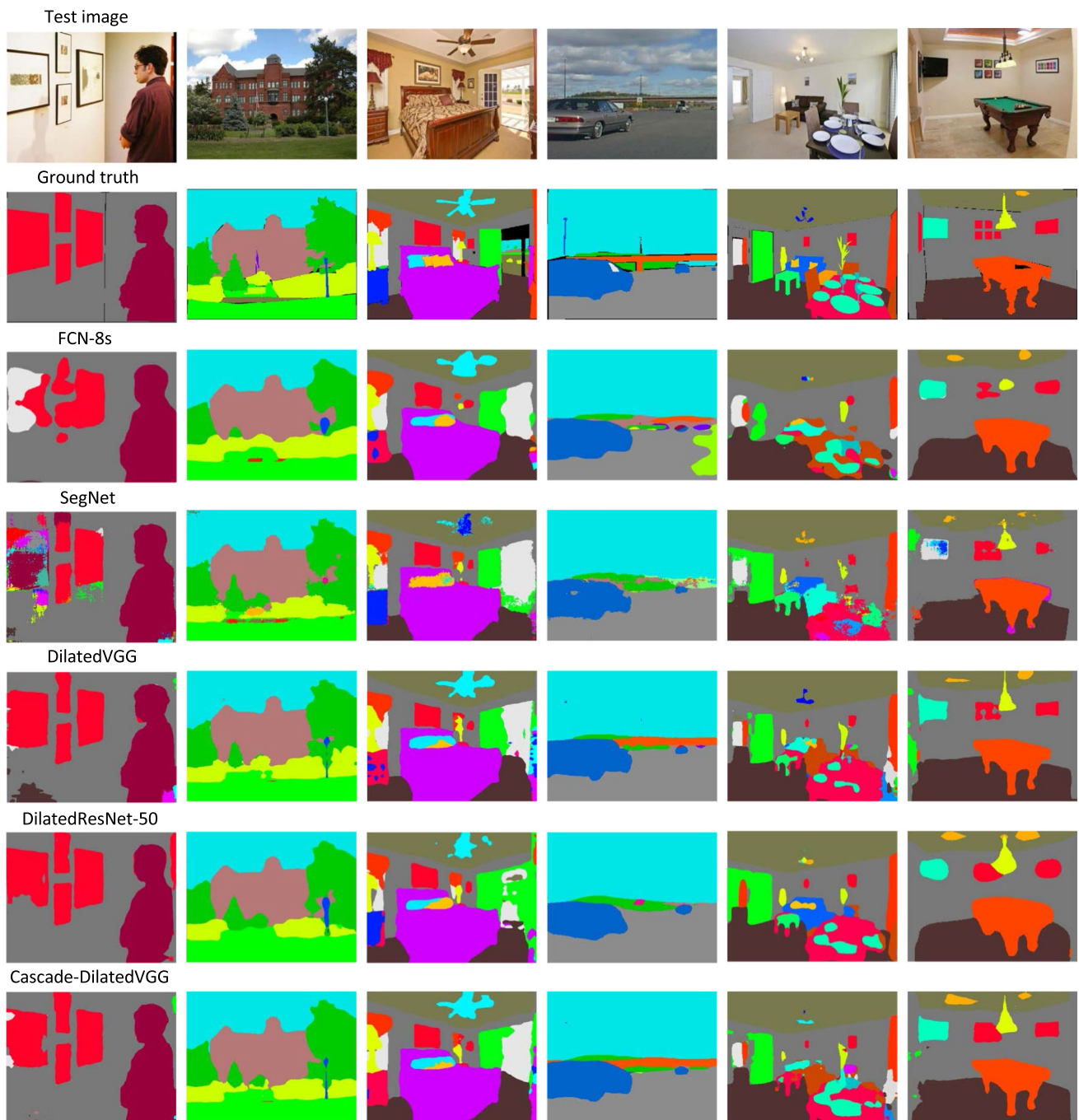
Fig. 8 Ground-truths, scene parsing results given by the baseline networks. All networks can give correct predictions for the common, large object and stuff classes, the difference in performance comes mostly from small, infrequent objects and how well they handle details

dilated segmentation network which uses the dilated convolution Yu and Koltun (2016); (2) PSPNet proposed in Zhao et al. (2017b), which introduces Pyramid Pooling Module (PPM) to aggregate multi-scale contextual information in the scene; (3) UPerNet proposed in Xiao et al. (2018) which adopts architecture like Feature Pyramid Network (FPN) (Lin et al. 2017) to incorporate multi-scale context

more efficiently. Table 3 shows results on the validation set of SceneParse150. Compared to plain DilatedResNet, PPM and UPerNet architectures improve mean IoU by 3–7%, and pixel accuracy by 1–2%. The superior performance shows the importance of context in the scene parsing task.
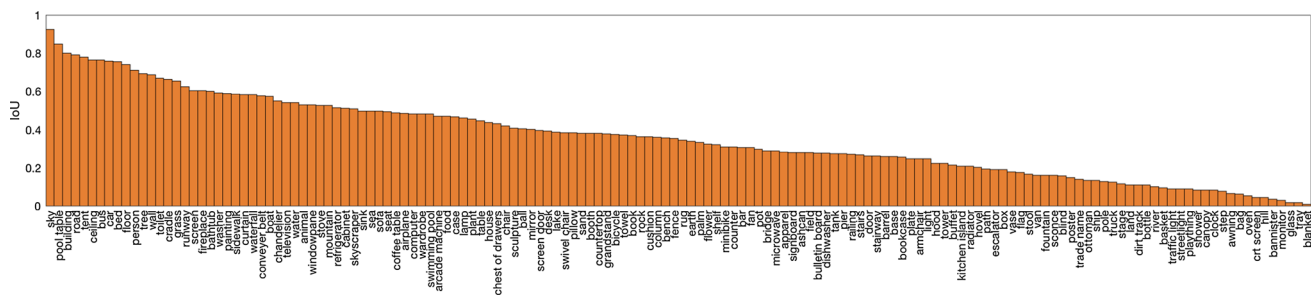
Fig. 9 Plot of scene parsing performance (IoU) on the 150 categories achieved by DilatedResNet-50 model. The best segmented categories are stuff, and the worst segmented categories are objects that are usually small and have few pixels

Table 3 Re-implementation of state-of-the art models on the validation set of SceneParse150

Networks	Pixel Acc. (%)	Mean IoU
DilatedResNet-18	77.41	0.3534
DilatedResNet-50	77.53	0.3549
DilatedResNet-18 + PPM (Zhao et al. 2017b)	78.64	0.3800
DilatedResNet-50 + PPM (Zhao et al. 2017b)	80.23	0.4204
DilatedResNet-101 + PPM (Zhao et al. 2017b)	80.91	0.4253
UPerNet-50 (Xiao et al. 2018)	80.23	0.4155
UPerNet-101 (Xiao et al. 2018)	81.01	0.4266

PPM Pyramid Pooling Module

Table 4 Comparisons of models trained with various batch normalization settings

BN status	Batch size	BN size	Pixel Acc. (%)	Mean IoU
Synchronized	16	16	79.73	0.4126
	8	8	80.05	0.4158
	4	4	79.71	0.4119
	2	2	75.26	0.3355
Unsynchronized	16	2	75.28	0.3403
Frozen	16	N/A	78.32	0.3809
	8	N/A	78.29	0.3793
	4	N/A	78.34	0.3833
	2	N/A	78.81	0.3856

The framework used is a Dilated ResNet-50 with Pyramid Pooling Module

4.3 Effect of Batch Normalization for Scene Parsing

An overwhelming majority of semantic segmentation models are fine-tuned from a network trained on ImageNet (Russakovsky et al. 2015), the same as most of the object detection models (Ren et al. 2015; Lin et al. 2017; He et al. 2017). There has been work (Peng et al. 2018) exploring the effects of the size of batch normalization (BN) (Ioffe and Szegedy 2015). The authors discovered that, if a network is trained with BN, only by a sufficiently large batch size of BN can the network achieve state-of-the-art performance. We conduct control experiments on ADE20K to explore the issue in terms of semantic segmentation. Our experiments show that a reasonably large batch size is essential for matching the highest score of the-state-or-the-art models, while a small batch size such as 2, shown in Table 4, lower the score of the model significantly by 5%. Thus training with a single GPU with limited RAM or with multiple GPUs under unsynchronized BN is unable to reproduce the best reported numbers. The possible reason is that the BN statics, i.e., mean and standard variance of activations may not be accurate when the batch size is not sufficient.

Our baseline framework is the PSPNet with a dilated ResNet-50 as the backbone. Besides those BN layers in the ResNet, BN is also used in the PPM. The baseline framework is trained with 8 GPUs and 2 images on each GPU. We adopt synchronized BN for the baseline network, i.e., the BN size should be the same as the batch size. Besides the synchronized BN setting, we also report the unsynchronized BN setting and frozen BN setting. The former one means that the BN size is the number of images on each GPU; the latter one means that the BN layers are frozen in the backbone network, and removed from the PPM. The training iterations and learning rate are set to 100k and 0.02 for the baseline, respectively. For networks trained under the frozen BN setting, the learning rate for the network with 16 batch size is set to 0.004 to prevent gradient explosion. And for networks with batch size smaller than 16, we both linearly decrease the learning rate and increase the training iterations according to previous works (Goyal et al. 2017). Differently from Table 3, the results are obtained w/o multi-scale testing.

We report the results in Table 4. In general, we empirically find that using BN layers with a sufficient BN size leads to better performance. The model with batch size and BN size as 16 (line 2) outperforms the one with batch size 16 and frozen BN (line 7) by 1.41% and 3.17% in terms of Pixel Acc. and Mean IoU respectively. We witness negligible changes of performance when batch (and BN) size changes
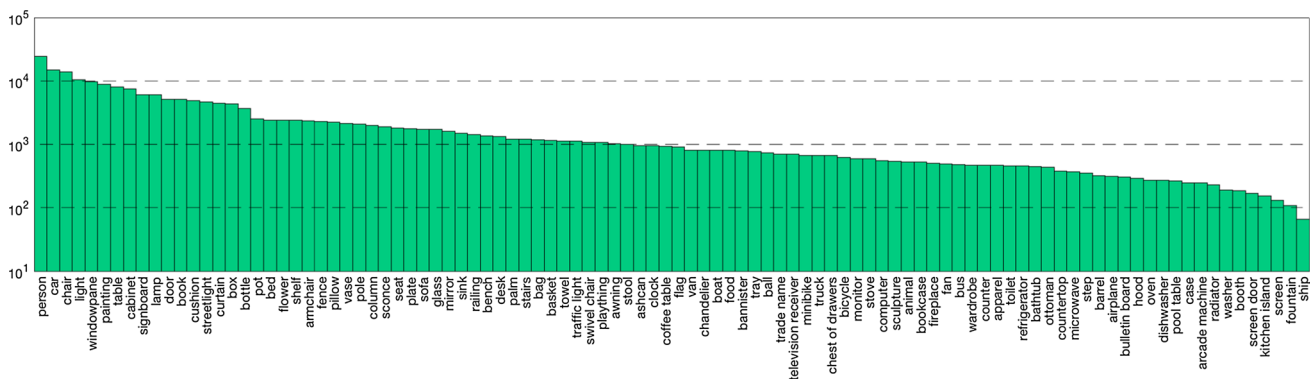
Fig. 10 Instance number per object in instance segmentation benchmark. All the objects except *ship* have more than 100 instances

in the range from 4 to 16 under synchronized BN setting (line 2–4). However, when the BN size drops to 2, the performance downgrades significantly (line 5). Thus a BN size of 4 is the inflection point in our experiments. This finding is different from the finding for object detection Peng et al. (2018), in which the inflection point is at a BN size of 16. We conjecture that it is due to images for semantic segmentation are densely annotated, different from those for object detection with bounding-box annotations. Therefore it is easier for semantic segmentation networks to obtain more accurate BN statistics with fewer images.

When we experiment with unsynchronized BN setting, i.e., we increase the batch size but do not change the BN size (line 6), the model yields almost identical result compared with the one with the same BN size but smaller batch size (line 5). Also, when we freeze the BN layers during the fine-tuning, the models are not sensitive to the batch size (line 7-10). These two set of experiments indicate that, for semantic segmentation models, the BN size is the one that matters instead of the batch size. However, we do note that smaller batch size leads to longer training time because we need to increase the number of training iterations for models with small batch size.

4.4 Instance Segmentation Benchmark

To benchmark the performance of instance segmentation, we select 100 foreground object categories from the full dataset, termed as InstSeg100. The plot of the instance number per object in InstSeg100 is shown in Fig. 10. The total number of object instances is 218K, on average there are 2.2K instances per object category and 10 instances per image; all the objects except *ship* have more than 100 instances.

We use Mask R-CNN He et al. (2017) models as baselines for InstSeg100. The models use FPN-50 as backbone network, initialized from ImageNet, other hyper-parameters follow those used in He et al. (2017). Two variants are presented, one with single scale training and the other with

Table 5 Baseline performance on the validation set of InstSeg100

Networks	mAP$_S$	mAP$_M$	mAP$_L$	mAP
Mask R-CNN single-scale	0.0542	0.1737	0.2883	0.1832
Mask R-CNN multi-scale	0.0733	0.2256	0.3584	0.2241

multi-scale training. Their performance on the validation set is shown in Table 5. We report the overall mean Average Precision mAP, along with metrics on different object scales, denoted by mAP$_S$ (objects smaller than 32×32 pixels), mAP$_M$ (between 32×32 and 96×96 pixels) and mAP$_L$ (larger than 96×96 pixels). Numbers suggest that: (1) multi-scale training could greatly improve the average performance ($\sim$ 0.04 in mAP); (2) instance segmentation of small objects on our dataset is extremely challenging and it does not improve ($\sim$ 0.02) as much as large objects ($\sim$ 0.07) when using multi-scale training. Qualitative results of the Mask R-CNN model are presented in Fig. 11. We can see that it is a strong baseline, giving correct detections and accurate object boundaries. Some typical errors are object reflections in the mirror, as shown in the bottom right example.

4.5 How Does Scene Parsing Performance Improve with Instance Information?

In the previous sections, we train and test semantic and instance segmentation tasks separately. Given that instance segmentation is trained with additional instance information compared to scene parsing, we further analyze how instance information can assist scene parsing.

Instead of re-modeling, we study this problem by fusing results from our trained state-of-the-art models, PSPNet for scene parsing and Mask R-CNN for instance segmentation. Concretely, we first take Mask R-CNN outputs and threshold predicted instances by confidence ($\geq$ 0.95); then we overlay the instance masks on to the PSPNet predictions; if one pixel belongs to multiple instances, it takes the semantic

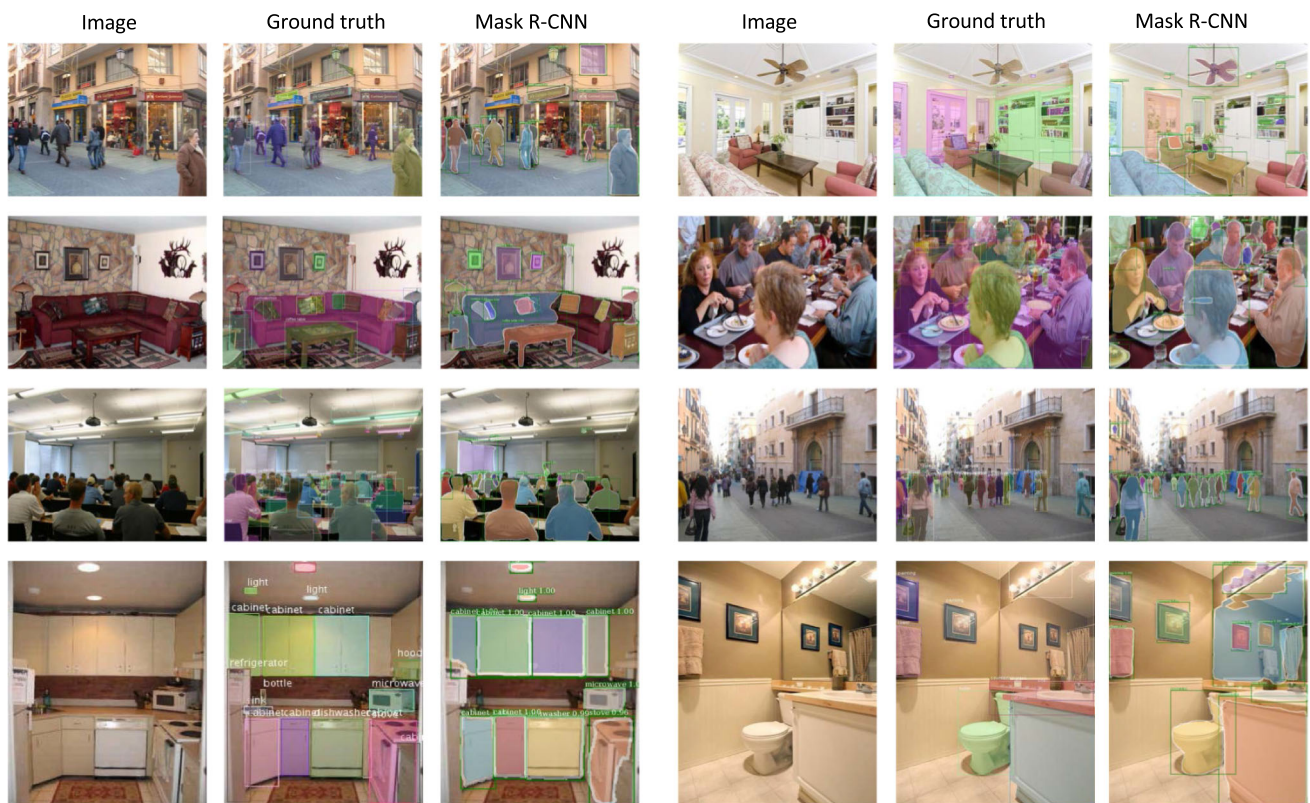| Image | Ground truth | Mask R-CNN | Image | Ground truth | Mask R-CNN |

Fig. 11 Images, ground-truths, and instance segmentation results given by multi-scale Mask R-CNN model

Table 6 Scene parsing performance before and after fusing outputs from instance segmentation model Mask R-CNN

Networks	Pixel Acc.		Mean IoU	
	Before (%)	After (%)	Before	After
DilatedResNet-50 + PPM (Zhao et al. 2017b)	80.23	80.21	0.4204	0.4256
DilatedResNet-101 + PPM (Zhao et al. 2017b)	80.91	80.91	0.4253	0.4290

label with the highest confidence. Note that instance segmentation only works for 100 foreground object categories as opposed to 150 categories, so stuff predictions come from the scene parsing model. Quantitative results are shown in Table 6. Overall the fusion improves scene parsing performance, while pixel accuracy stays around the same, the mean IoU improves around 0.4–0.5%. This experiment demonstrates that instance level information is useful for helping the non-instance-aware scene parsing task.
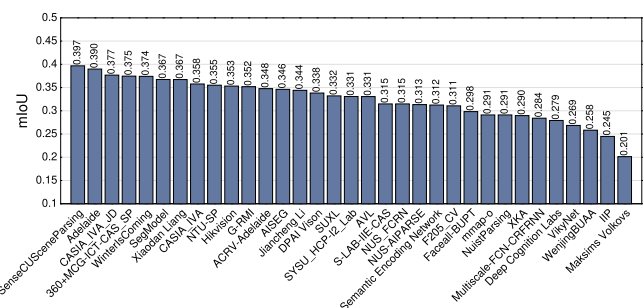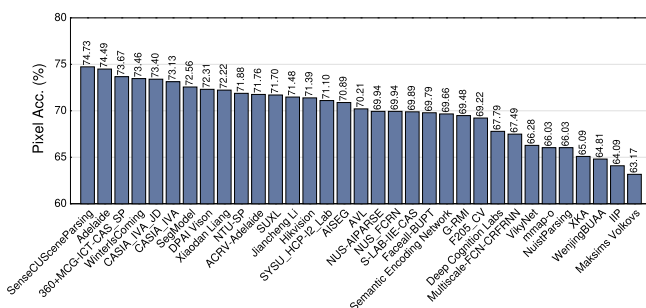

Fig. 12 Scene Parsing Track Results, ranked by pixel accuracy and mean IoU

Table 7 Top performing models in Scene Parsing for Places Challenge 2016

Team	Pixel Acc. (%)	Mean IoU	Score
SenseCUSceneParsing (Zhao et al. 2017b)	74.73	0.3968	0.5720
Adelaide (Wu et al. 2016)	74.49	0.3898	0.5673
360-MCG-CT-CAS_SP	73.67	0.3746	0.5556

Table 8 Top performing models in Scene Parsing for Places Challenge 2017

Team	Pixel Acc. (%)	Mean IoU	Score
CASIA_IVA_JD	73.40	0.3754	0.5547
WinterIsComing	73.46	0.3741	0.5543
Xiaodan Liang	72.22	0.3672	0.5447

5 Places Challenges

In order to foster new models for pixel-wise scene understanding, we organized in 2016 and 2017 the Places Challenge including the scene parsing track and instance segmentation track.

5.1 Scene Parsing Track

Scene parsing submissions were ranked based on the average score of the mean IoU and pixel-wise accuracy in the benchmark test set, as shown in Fig. 12.

The Scene Parsing Track totally received 75 submissions from 22 teams in 2016 and 27 submissions from 11 teams in 2017. The top performing teams for both years are shown in Tables 7 and 8. The winning team in 2016, proposing PSPNet (Zhao et al. 2017b) still holds the highest score. Figure 13 shows some qualitative results from the top performing models on each year.
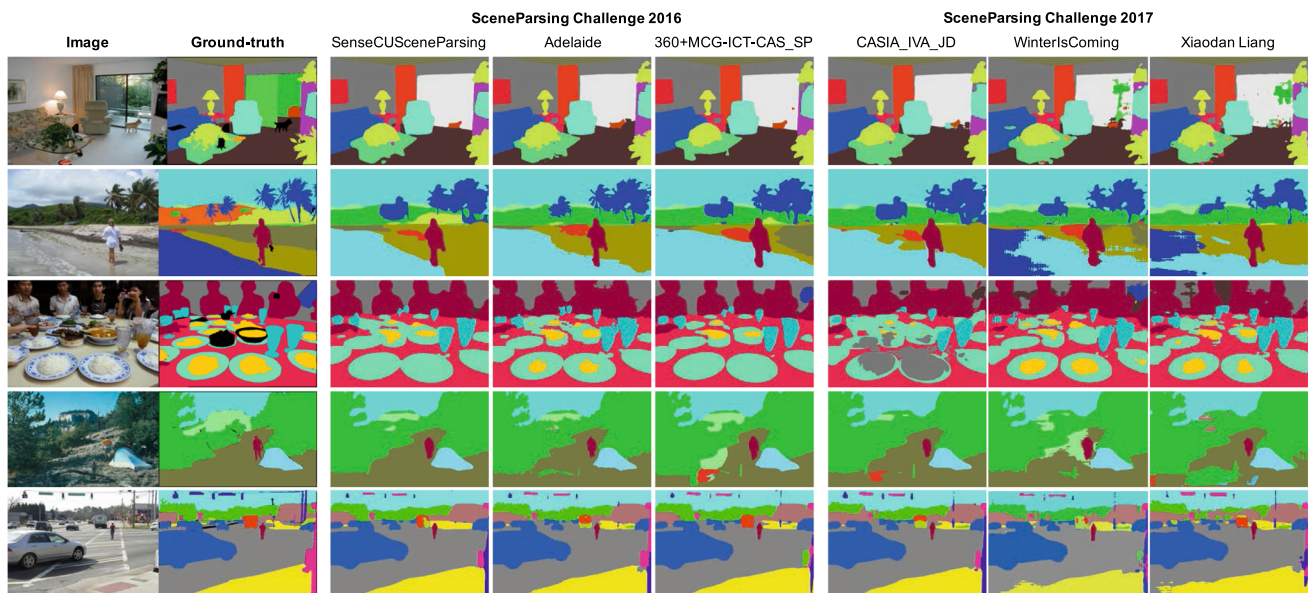
In Fig. 14 we compare the top models against the proposed baselines and human performance (approximately measured as the annotation consistency in Sect. 3.3), which could be the upper bound performance. As an interesting comparison, if we use the image mode generated in Fig. 6 as prediction on the testing set, it achieves 20.30% pixel accuracy, which could be the lower bound performance for all the models.

Some error cases are shown in Fig. 15. We can see that models usually fail to detect the concepts in some images that have occlusions or require high-level context reasoning. For example, the boat in the first image is not a typical view of a boat, making the models fail; For the last image, the muddy car is missed by all the top performer networks because of its muddy camouflage.

5.2 Instance Segmentation Track

For instance segmentation, we used the mean Average Precision (mAP), following the evaluation metrics of COCO.

The Instance Segmentation Track, introduced in Places Challenge 2017, received 12 submissions from 5 teams. Two teams beat the strong Mask R-CNN baseline by a good margin, their best model performances are shown in Table 9 together with the Mask R-CNN baseline trained by ourselves. The performances for small, medium and large objects are also reported, following Sect. 4.4. Figure 16 shows qualitative results from the best model of the teams.

Fig. 13 Scene Parsing results given by top methods for Places Challenge 2016 and 2017
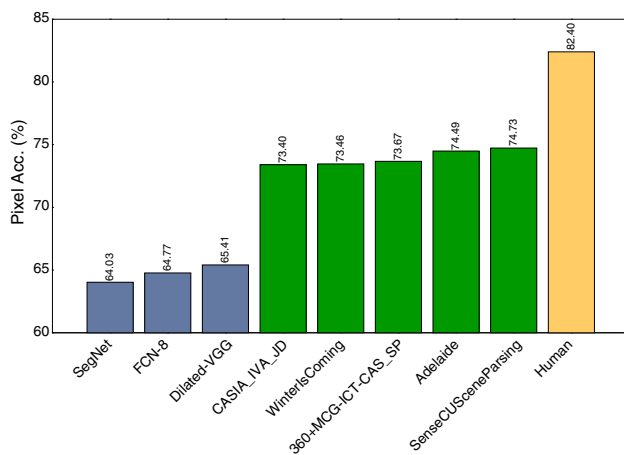
Fig. 14 Top scene parsing models compared with human performance and baselines in terms of pixel accuracy. Scene parsing based on the image mode has a 20.30% pixel accuracy

As can be seen in Table 9, both methods outperform the Mask R-CNN at any of the object scales, even though they still struggle with medium and small objects. The submission from Megvii (Face++) seems to particularly advantage G-RMI in terms of small objects, probably due to the use of contextual information. Their mAP on small objects shows a relative improvement over G-RMI of 41%, compared to the 19% and 6% of medium and large objects.

This effect can be qualitatively seen in Fig. 16. While both methods perform similarly well in finding large object classes such as people or tables, Megvii (Face++) is able to detect small paintings (rows 1 and 3) or lights (row 5) occupying small regions.

5.3 Take-Aways from the Challenge

Looking at the challenge results, there are several peculiarities that make ADE20K challenging for instance segmentation. First, ADE20K contains plenty of small objects. It is hard for most of instance segmentation frameworks to distinguish small objects from background, and even harder to recognize and classify them into correct categories. Second, ADE20K is highly diverse in terms of scenes and objects, requiring models of strong capability to achieve better performance in various scenes. Third, scenes in ADE20K are generally crowded. The inter-class occlusion and intro-class occlusion create problems for object detection as well as instance segmentation. This is can be seen in Fig. 16, where the models struggle to detect some of the boxes in the cluttered areas (row 2, left) or the counter inf row 4, covered by multiple people.
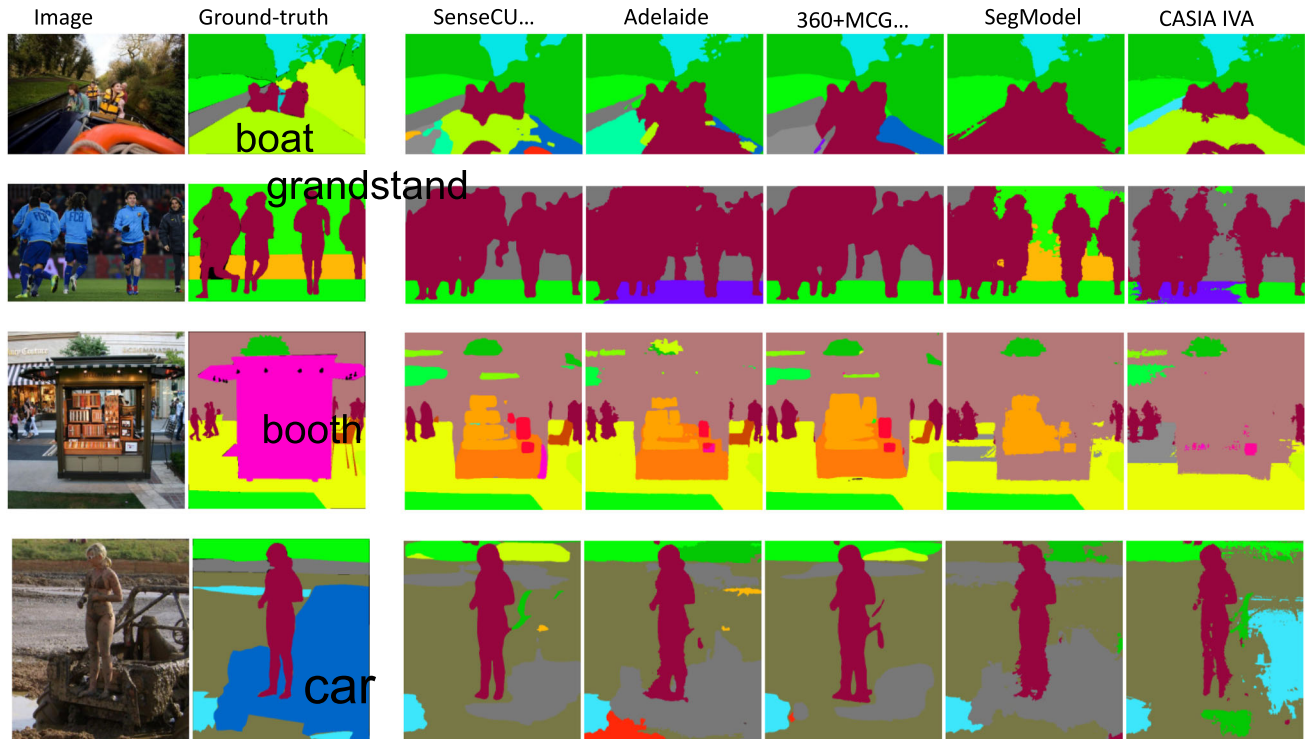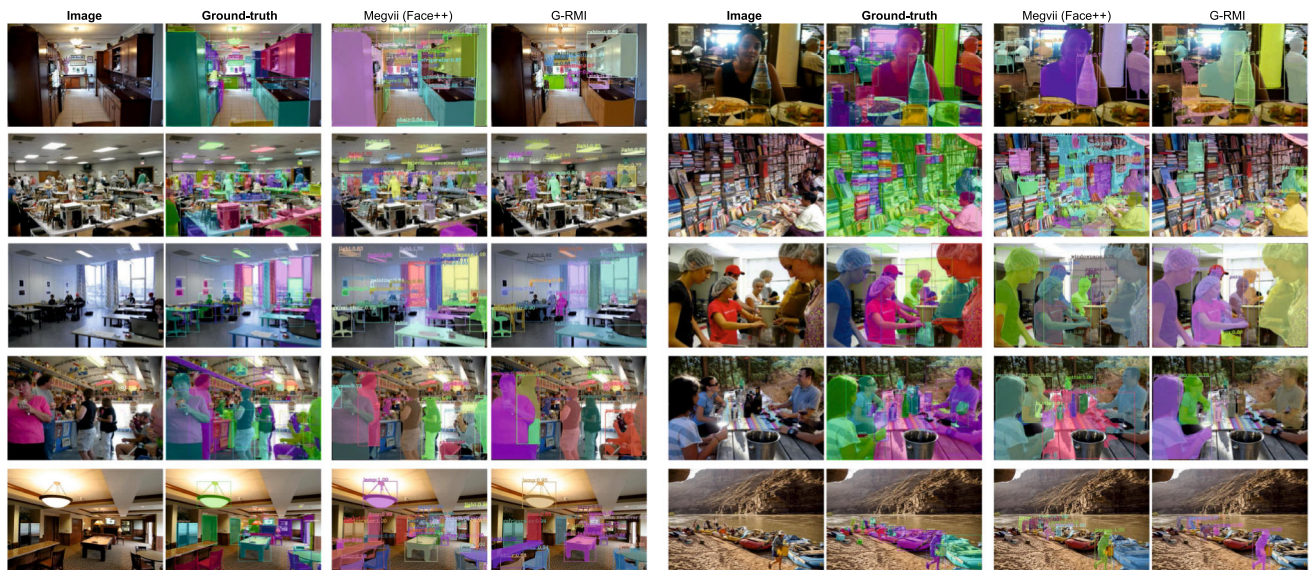

Fig. 15 Ground-truths and predictions given by top methods for scene parsing. The mistaken regions are labeled. We can see that models make mistakes on objects in non-canonical views such as the boat in first example, and on objects which require high-level reasoning such as the muddy car in the last example

Table 9 Top performing models in Instance Segmentation for Places Challenge 2017

Team	mAP$_S$	mAP$_M$	mAP$_L$	mAP
Megvii (Face++) (Peng et al. 2018)	0.1386	0.3015	0.4119	0.2977
G-RMI	0.0980	0.2523	0.3858	0.2415
Baseline Mask R-CNN	0.0733	0.2256	0.3584	0.2241

Fig. 16 Instance Segmentation results given by top methods for Places Challenge 2017

To further gain insight from the insiders, we invite the leading author of the winning method for the instance segmentation track in Places Challenge, Tete Xiao, to give a summary of their method as follows (the method itself is not open-sourced due to the company policy):

Following a top-down instance segmentation framework, we starts with a module to generate object proposals first then classify each pixel within the proposal. But unlike RoI Align used in Mask-RCNN (He et al. 2017), we use Precise RoI Pooling (Jiang et al. 2018) to extract features for each proposal. Precise RoI Pooling avoids sampling the pivot points used in RoI Align by regarding a discrete feature map as a continuous interpolated feature map and directly computing a two-order integral. The good alignment of features provide good improvement for object detection, while even higher gain for instance segmentation. To improve the recognition of small objects, we make use of contextual information by combining, for each proposal, the features of the previous and following layers. Given that top-down instance segmentation relies heavily on object detection, the model ensembles multiple object bounding-boxes before fed into a mask generator. We also find that the models cannot avoid predicting objects in the mirror, which indicates that current models are still incapable of high-level reasoning in parallel with low-level visual cues.

6 Object-Part Joint Segmentation

Since ADE20K contains part annotations for various object classes, we further train a network to jointly segment objects and parts. There are 59 out of total 150 objects that contain parts, some examples can be found in Fig. 3. In total there are 153 part classes included. We use UPerNet (Xiao et al. 2018) to jointly train object and part segmentation. During training, we include the non-part classes and only calculate softmax loss within the set of part classes via ground-truth object class. During inference, we first pick out a predicted object class and get the predicted part classes from its corresponding part set. This is organized in a cascaded way. We show the qualitative results of UPerNet in Fig. 17, and the quantitative performance of part segmentation for several selected objects in Fig. 18.

Fig. 17 Object and part joint segmentation results predicted by UPerNet (Xiao et al. 2018). Object parts are segmented based on the top of the corresponding object segmentation mask
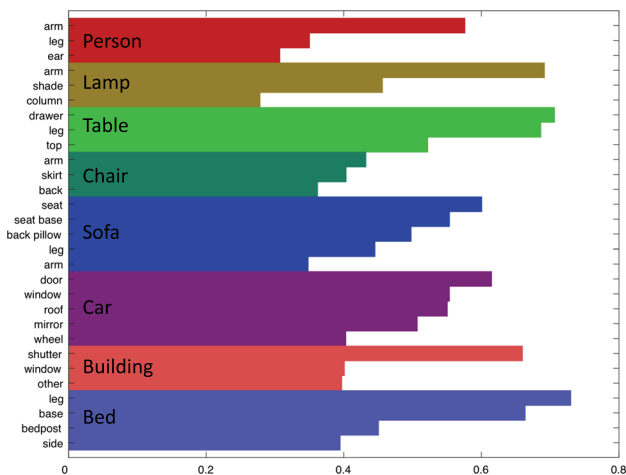

Fig. 18 Part segmentation performance (in mean IoU) grouped by several selected objects predicted by UPerNet

7 Applications

Accurate scene parsing leads to wider applications. Here we take the hierarchical semantic segmentation, automatic scene content removal, and scene synthesis as exemplar applications of the scene parsing models.

Hierarchical Semantic Segmentation Given the wordnet tree constructed on the object annotations shown in Fig. 7, the 150 categories are hierarchically connected and have hyponyms relations. Thus we can gradually merge the objects into their hyponyms so that classes with similar semantics are merged at the early levels. In this way, we generate a hierarchical semantic segmentation of the image shown in Fig. 19. The tree also provides a principled way to segment more general visual concepts. For example, to detect all furniture in a scene, we can simply merge the hyponyms associated with that synset, such as the *chair*, *table*, *bench*, and *bookcase*.

Automatic Image Content Removal Image content removal methods typically require the users to annotate the pre-
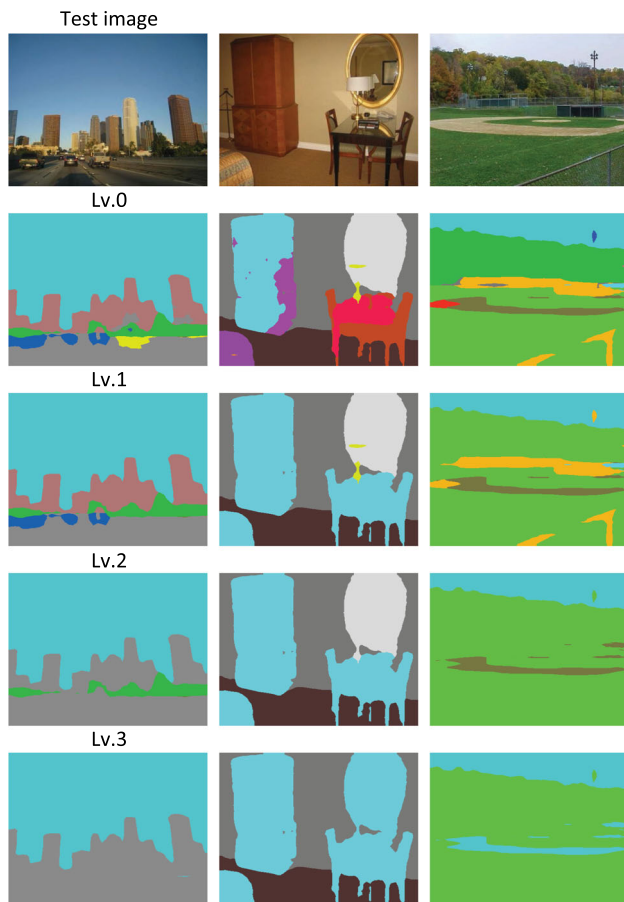
Test image

Lv.0

Lv.1

Lv.2

Lv.3

Fig. 19 The examples of the hierarchical semantic segmentation. Objects with similar semantics like furnitures and vegetations are merged at early levels following the wordnet tree

person

tree

car

all the objects

Fig. 20 Automatic image content removal using the predicted object score maps given by the scene parsing network. We are able to remove not only individual objects such as person, tree, car, but also groups of them or even all the discrete objects. For each row, the first image is the original image, the second is the object score map, and the third one is the filled-in image

Fig. 21 Scene synthesis. Given annotation masks, images are synthesized by coupling the scene parsing network and the image synthesis method proposed in Nguyen et al. (2016)

cise boundary of the target objects to be removed. Here, based on the predicted object probability map from scene parsing networks, we automatically identify the image regions of the target objects. After cropping out the target objects using the predicted object score maps, we simply use image completion/inpainting methods to fill the holes in the image. Figure 20 shows some examples of the automatic image content removal. It can be seen that with the object score maps, we are able to crop out the objects from an image precisely. The image completion technique used is described in (Huang et al. 2014).

Scene Synthesis Given a scene image, the scene parsing network can predict a semantic label mask. Furthermore, by coupling the scene parsing network with the recent image synthesis technique proposed in Nguyen et al. (2016), we can also synthesize a scene image given the semantic label mask. The general idea is to optimize the input code of a deep image generator network to produce an image that highly activates the pixel-wise output of the scene parsing network. Figure 21 shows three synthesized image samples
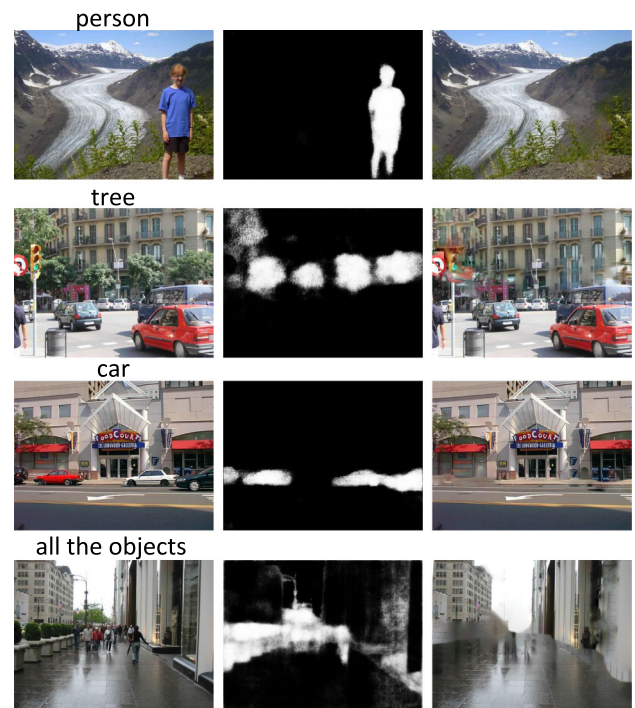
given the semantic label mask in each row. As comparison, we also show the original image associated with the semantic label mask. Conditioned on a semantic mask, the deep image generator network is able to synthesize an image with similar spatial configuration of visual concepts.

8 Conclusion

In this work we introduced the ADE20K dataset, a densely annotated dataset with the instances of stuff, objects, and parts, covering a diverse set of visual concepts in scenes. The dataset was carefully annotated by a single annotator to ensure precise object boundaries within the image and the consistency of object naming across the images. Benchmarks for scene parsing and instance segmentation are constructed based on the ADE20K dataset. We further organized challenges and evaluated the state-of-the-art models on our benchmarks. All the data and pre-trained models are released to the public.

References

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*, 2481–2495.

Bell, S., Upchurch, P., Snavely, N., & Bala, K. (2013). OpenSurfaces: A richly annotated catalog of surface appearance. *ACM Transactions on Graphics (TOG)*, *32*, 111.

Bell, S., Upchurch, P., Snavely, N., & Bala, K. (2015). Material recognition in the wild with the materials in context database. In *Proceedings of CVPR*.

Caesar, H., Uijlings, J., & Ferrari, V. (2017). Coco-stuff: Thing and stuff classes in context.

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2016). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. arXiv:1606.00915.

Chen, X., Mottaghi, R., Liu, X., Cho, N. G., Fidler, S., Urtasun, R., & Yuille, A. (2014). Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proceedings of CVPR*.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., et al. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of CVPR*.

Dai, J., He, K., & Sun, J. (2015). Convolutional feature masking for joint object and stuff segmentation. In *Proceedings of CVPR*.

Dai, J., He, K., & Sun, J. (2016). Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of CVPR*.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, *88*, 303–338.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The kitti vision benchmark suite. In *Proceedings of CVPR*.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., et al. (2017). Accurate, large minibatch SGD: Training imagenet in 1 hour. ArXiv preprint arXiv:1706.02677.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In *Proceedings of ICCV*.

Huang, J. B., Kang, S. B., Ahuja, N., & Kopf, J. (2014). Image completion using planar structure guidance. *ACM Transactions on Graphics (TOG)*, *33*, 129.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. ArXiv preprint arXiv:1502.03167.

Jiang, B., Luo, R., Mao, J., Xiao, T., & Jiang, Y. (2018). Acquisition of localization confidence for accurate object detection. In *Proceedings of ECCV*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*.

Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of CVPR*.

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., et al. (2014). Microsoft coco: Common objects in context. In *Proceedings of ECCV*.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of CVPR*.

Martin, D., Fowlkes, C., Tal, D., & Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of ICCV*.

Mottaghi, R., Chen, X., Liu, X., Cho, N. G., Lee, S. W., Fidler, S., et al. (2014). The role of context for object detection and semantic segmentation in the wild. In *Proceedings of CVPR*.

Nathan Silberman, P. K., Derek, H., & Fergus, R. (2012). Indoor segmentation and support inference from RGBD images. In *Proceedings of ECCV*.

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., & Clune, J. (2016). Synthesizing the preferred inputs for neurons in neural networks via deep generator networks.

Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of ICCV*.

Patterson, G., & Hays, J. (2016). Coco attributes: Attributes for people, animals, and objects. In *Proceedings of ECCV*.

Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., et al. (2018). Megdet: A large mini-batch object detector. In *Proceedings of CVPR*, pp. 6181–6189.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252.

Russell, B. C., Torralba, A., Murphy, K. P., & Freeman, W. T. (2008). Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, *77*, 157–173.

Song, S., Lichtenberg, S. P., & Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of CVPR*.

Spain, M., & Perona, P. (2010). Measuring and predicting object importance. *International Journal of Computer Vision*, *91*, 59–76.

Wu, Z., Shen, C., van den Hengel, A. (2016). Wider or deeper: Revisiting the resnet model for visual recognition. CoRR arXiv:1611.10080.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., & Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In *Proceedings of CVPR*.

Xiao, T., Liu, Y., Zhou, B., Jiang, Y., & Sun, J. (2018). Unified perceptual parsing for scene understanding. In *Proceedings of ECCV*.

Yu, F., & Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions.

Zhao, H., Puig, X., Zhou, B., Fidler, S., Torralba, A. (2017a). Open vocabulary scene parsing. In *International Conference on Computer Vision (ICCV)*.

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017b). Pyramid scene parsing network. In *Proceedings of CVPR*.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., & Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., & Torralba, A. (2017). Scene parsing through ade20k dataset. In *Proceedings of CVPR*.