Scene Parsing through ADE20K Dataset

Bolei Zhou[1], Hang Zhao[1], Xavier Puig[1], Sanja Fidler[2], Adela Barriuso[1], Antonio Torralba[1]
[1]Massachusetts Institute of Technology, USA
[2]University of Toronto, Canada.

Abstract

Scene parsing, or recognizing and segmenting objects and stuff in an image, is one of the key problems in computer vision. Despite the community's efforts in data collection, there are still few image datasets covering a wide range of scenes and object categories with dense and detailed annotations for scene parsing. In this paper, we introduce and analyze the ADE20K dataset, spanning diverse annotations of scenes, objects, parts of objects, and in some cases even parts of parts. A scene parsing benchmark is built upon the ADE20K with 150 object and stuff classes included. Several segmentation baseline models are evaluated on the benchmark. A novel network design called Cascade Segmentation Module is proposed to parse a scene into stuff, objects, and object parts in a cascade and improve over the baselines. We further show that the trained scene parsing networks can lead to applications such as image content removal and scene synthesis[1].

Figure 1. Images in the ADE20K dataset are densely annotated in details with objects and parts. The first row shows the sample images, the second row shows the annotation of objects and stuff, and the third row shows the annotation of object parts.

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 [26], COCO [17] and Places [35], along with the rapid development of the deep convolutional neural network (ConvNet) approaches, have brought great advancements to visual scene understanding. Nowadays, given a visual scene of a living room, a robot equipped with a trained ConvNet 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. It needs to recognize and localize not only the objects like sofa, table, and TV, but also their parts, e.g., a seat of a chair or a handle of a cup, to allow proper interaction, as well as to segment the stuff like floor, wall and ceiling for spatial navigation.

Scene parsing, or recognizing and segmenting objects and stuff in an image, remains one of the key problems in scene understanding. Going beyond image-level recognition, scene parsing requires a much denser annotation of scenes with a large set of objects. However, the current datasets have limited number of objects (e.g., COCO [17], Pascal [10]) and in many cases those objects are not the most common objects one encounters in the world (like frisbees or baseball bats), or the datasets only cover a limited set of scenes (e.g., Cityscapes [7]). Some notable exceptions are Pascal-Context [21] and the SUN database [32]. However, Pascal-Context still contains scenes primarily focused on 20 object classes, while SUN has noisy labels at the object level.

Our goal is to collect a dataset that has densely annotated images (every pixel has a semantic label) with a large and an unrestricted open vocabulary. The images in our dataset are manually segmented in great detail, covering a diverse set of scenes, object and object part categories. The challenges for collecting such annotations include 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 inconsis-

[1]Dataset and pretrained models are available at http://groups.csail.mit.edu/vision/datasets/ADE20K/

tencies 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 the 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. Fig. 1 shows examples from our dataset.

The paper is organized as follows. Firstly we describe the ADE20K dataset, the collection process and statistics. We then introduce a generic network design called Cascade Segmentation Module, which enables neural networks to segment stuff, objects, and object parts in cascade. Several semantic segmentation networks are evaluated on the scene parsing benchmark of ADE20K as baselines. The proposed Cascade Segmentation Module is shown to improve over those baselines. We further apply the scene parsing networks to the tasks of automatic scene content removal and scene synthesis.

1.1. 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 [26], Pascal [10], and KITTI [12]. 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 as well as 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 [10] and 91 in Microsoft COCO [17]). Collecting dense annotations where all pixels are labeled is much more challenging. Such efforts include SIFT Flow dataset [18], Pascal-Context [21], NYU Depth V2 [22], SUN database [32], SUN RGB-D dataset [28], CityScapes dataset [7], and OpenSurfaces [2, 3].

Datasets with objects, parts and attributes. Core dataset [6] is one of the earliest work that explores the object part annotation across categories. Recently, 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 [5], or material classes, i.e. OpenSurfaces [2, 3]. We advance this effort by collecting very high-resolution imagery 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 Sec. 2.5.

Semantic segmentation/parsing models.. There are a lot of models proposed for image parsing. For example, MRF frameworks are proposed to parse images in different levels [30] or segment rare object classes [33]; detection is combined with segmentation to improve the performance [31]; stuff classes are leveraged to localize objects [13]. With the success of convolutional neural networks (CNN) for image classification [16], there is growing interest for semantic image parsing using CNNs with dense output, such as the multiscale CNN [11], recurrent CNN [25], fully CNN [19], deconvolutional neural networks [24], encoder-decoder SegNet [1], multi-task network cascades [9], and DilatedNet [4, 34]. They are benchmarked on Pascal dataset with impressive performance on segmenting the 20 object classes. Some of them [19, 1] are evaluated on Pascal-Context [21] or SUN RGB-D dataset [28] to show the capability to segment more object classes in scenes. Joint stuff and object segmentation is explored in [8] which uses pre-computed superpixels and feature masking to represent stuff. Cascade of instance segmentation and categorization has been explored in [9]. In this paper we introduce a generic network module to segment stuff, objects, and object parts jointly in a cascade, which could be integrated in existing networks.

2. ADE20K: Fully Annotated Image Dataset

In this section, we describe our ADE20K dataset and analyze it through a variety of informative statistics.

2.1. Dataset summary

There are 20,210 images in the training set, 2,000 images in the validation set, and 3,000 images in the testing set. 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. The annotations in the dataset are still growing. Sample images and annotations from the ADE20K dataset are shown in Fig. 1.

2.2. Image annotation

For our dataset, we are interested in having a diverse set of scenes with dense annotations of all the objects present. Images come from the LabelMe [27], SUN datasets [32], and Places [35] and were selected to cover the 900 scene

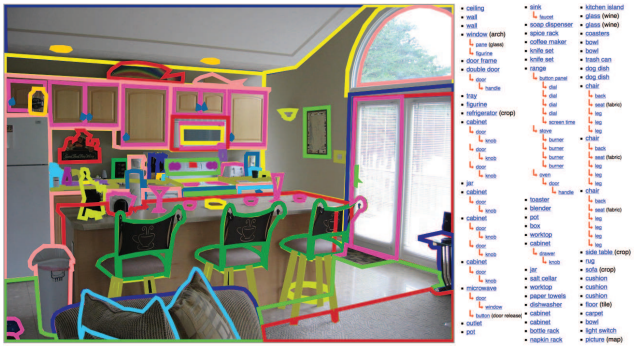

Figure 2. Annotation interface, the list of the objects and their associated parts in the image.

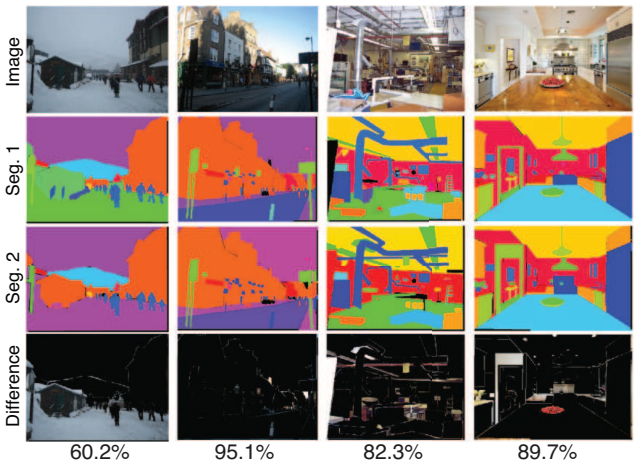

Figure 3. Analysis of annotation consistency. Each column shows an image and two segmentations done by the same annotator at different times. Bottom row shows the pixel discrepancy when the two segmentations are subtracted, and the percentage of pixels with the same label. On average across all re-annotated 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.

categories defined in the SUN database. Images were annotated by a single expert worker using the LabelMe interface [27]. Fig. 2 shows a snapshot of the annotation interface and one fully segmented image. The worker provided three types of annotations: object 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. Datasets such as COCO [17], Pascal [10] or Cityscape [7] 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. 5.d). 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 we label these associations 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'. The total part hierarchy has a depth of 3. The object and part hierarchy is in the supplementary materials.

2.3. 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 annotator's best effort, the process is not free of 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 3. 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. 5.7% of the pixels had this type of error.

- **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'. 6.0% of the pixels had naming issues. 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 of Fig 3 the annotator missed some

small objects in different annotations. 5.9% of the pixels are due to missing labels. A similar issue existed in segmentation datasets such as the Berkeley Image segmentation dataset [20].

The median error values for the three error types are: 4.8%, 0.3% and 2.6% showing 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 [17]). 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.

2.4. Dataset statistics

Fig. 4.a 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 [29, 32]. They differ from the ones from datasets such as COCO or ImageNet where the distribution is more uniform resulting from manual balancing.

Fig. 4.b shows the distributions of annotated parts grouped by the objects they belong to and sorted by frequency within each object class. Most object classes also have a non-uniform distribution of part counts. Fig. 4.c and Fig. 4.d show how objects are shared across scenes and how parts are shared by objects. Fig. 4.e shows the variability in the appearances of the part 'door'.

The mode of the object segmentations is shown in Fig. 5.a 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. Fig. 5.b 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). Fig. 5.c shows the distribution of parts.

Table 1. Comparison of semantic segmentation datasets.

	Images	Obj. Inst.	Obj. Cls.	Part Inst.	Part Cls.	Obj. Cls. per Img.
COCO	123,287	886,284	91	0	0	3.5
ImageNet*	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	4,479	0	0	9.8
OpenSurfaces	22,214	71,460	160	0	0	N/A
PascalContext	10,103	$\sim$104,398**	540	181,770	40	5.1
ADE20K	22,210	434,826	2,693	175,961	476	9.9

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

As the list of object classes is not predefined, there are new classes appearing over time of annotation. Fig. 5.d shows the number of object (and part) classes as the number of annotated instances increases. Fig. 5.e 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.

2.5. Comparison with other datasets

We compare ADE20K with existing datasets in Tab. 1. Compared to the largest annotated datasets, COCO [17] and Imagenet [26], 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 [32], 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 [21, 5] 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.

3. Cascade Segmentation Module

While the frequency of objects appearing in scenes follows a long-tail distribution, the pixel ratios of objects also follow such a distribution. For example, the stuff classes like 'wall', 'building', 'floor', and 'sky' occupy more than 40% of all the annotated pixels, while the discrete objects, such as 'vase' and 'microwave' at the tail of the distribution (see Fig. 4b), occupy only 0.03% of the annotated pixels. Because of the long-tail distribution, a semantic segmentation network can be easily dominated by the most frequent

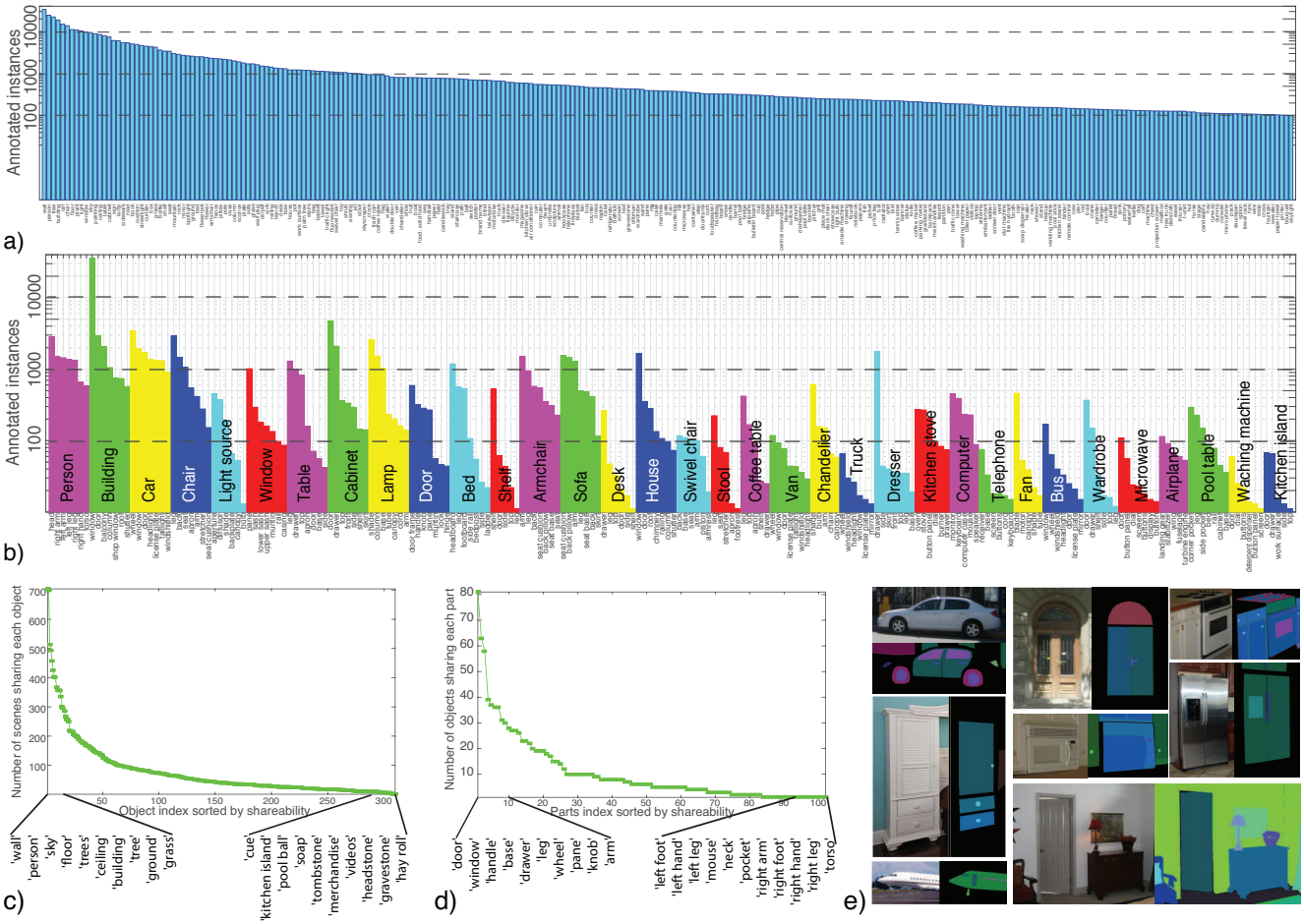

Figure 4. a) Object classes sorted by frequency. Only the top 270 classes with more than 100 annotated instances are shown. 68 classes have more than a 1000 segmented instances. b) Frequency of parts grouped by objects. There are more than 200 object classes with annotated parts. Only objects with 5 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 they are part of. d) Object parts ranked by the number of objects they are part of. e) Examples of objects with doors. The bottom-right image is an example where the door does not behave as a part.

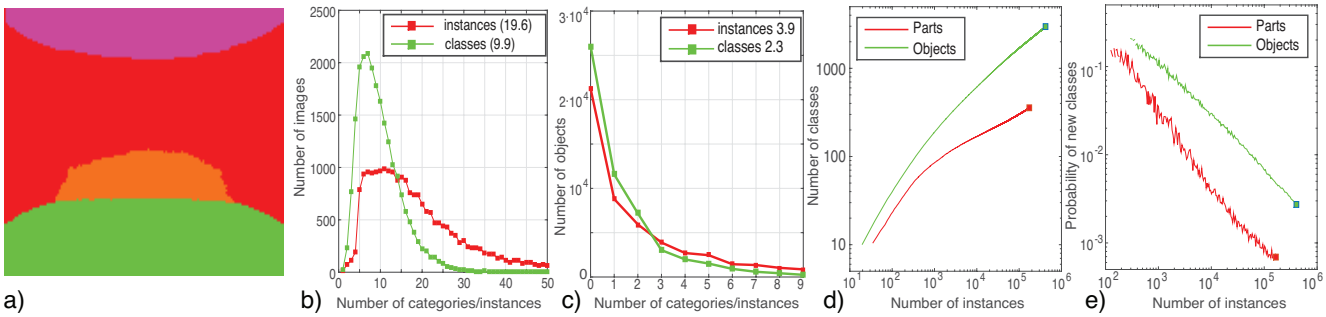

Figure 5. a) Mode of the object segmentations contains 'sky', 'wall', 'building' and 'floor'. b) Histogram of the number of segmented object instances and classes per image. c) Histogram of the number of segmented part instances and classes per object. d) Number of classes as a function of segmented 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.

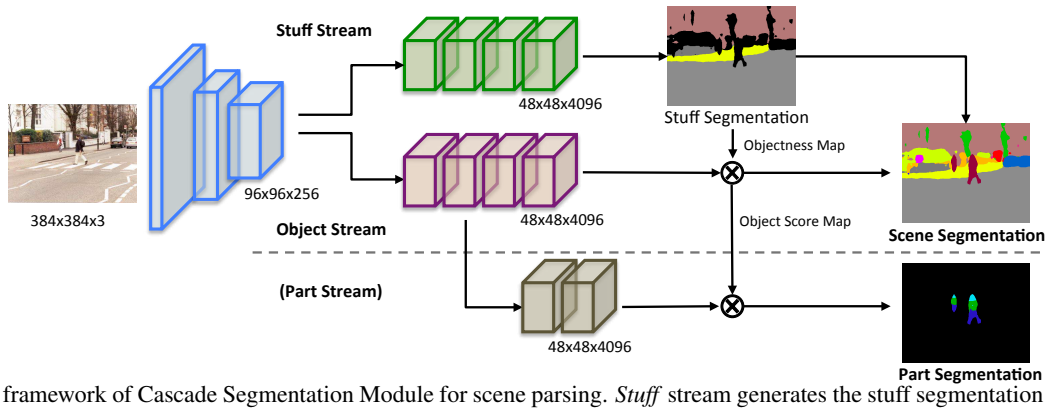

Figure 6. The framework of Cascade Segmentation Module for scene parsing. *Stuff* stream generates the stuff segmentation and objectness map from the shared feature activation. The *object* stream then generates object segmentation by integrating the objectness map from the stuff stream. Finally the full scene segmentation is generated by merging the object segmentation and the stuff segmentation. Similarly, *part* stream takes object score map from object stream to further generate object-part segmentation. Since not all objects have part annotation, the part stream is optional. Feature sizes are based on the Cascade-dilatedNet, the Cascade-SegNet has different but similar structures.

stuff classes. On the other hand, there are spatial layout relations among stuffs and objects, and the objects and the object parts, which are ignored by the design of the previous semantic segmentation networks. For example, a drawing on a wall is a part of the wall (the drawing occludes the wall), and the wheels on a car are also parts of the car.

To handle the long-tail distribution of objects in scenes and the spatial layout relations of scenes, objects, and object parts, we propose a network design called Cascade Segmentation Module. This module is a generic network design which can potentially be integrated in any previous semantic segmentation networks. We first categorize semantic classes of the scenes into three macro classes: *stuff* (sky, road, building, etc), foreground *objects* (car, tree, sofa, etc), and object *parts* (car wheels and door, people head and torso, etc). Note that in some scenarios there are some object classes like 'building' or 'door' that could belong to either of two macro classes, here we assign the object classes to their most likely macro class.

In the network for scene parsing, different streams of high-level layers are used to represent different macro classes and recognize the assigned classes. The segmentation results from each stream are then fused to generate the segmentation. The proposed module is illustrated in Fig. 6.

More specifically, the stuff stream is trained to classify all the stuff classes plus one foreground object class (which includes all the non-stuff classes). After training, the stuff stream generates stuff segmentation and a dense objectness map indicating the probability that a pixel belongs to the foreground object class. The object stream is trained to classify the discrete objects. All the non-discrete objects are ignored in the training loss function of the object stream. After training, the object stream further segments each discrete object on the predicted objectness map from the stuff stream. The result is merged with the stuff segmentation to generate the scene segmentation. For those discrete objects annotated with parts, the part stream can be jointly trained to segment object parts. Thus the part stream further segments parts on each object score map predicted from the object stream.

The network with the two streams (stuff+objects) or three streams (stuff+objects+parts) could be trained end-to-end. The streams share the weights of the lower layers. Each stream has a training loss at the end. For the stuff stream we use the per-pixel cross-entropy loss, where the output classes are all the stuff classes plus the foreground class (all the discrete object classes are included in it). We use the per-pixel cross-entropy loss for the object stream, where the output classes are all the discrete object classes. The objectness map is given as a ground-truth binary mask that indicates whether a pixel belongs to any of the stuff classes or the foreground object class. This mask is used to exclude the penalty for pixels which belong to any of the stuff classes in the training loss for the object stream. Similarly, we use cross-entropy loss for the part stream. All part classes are trained together, while non-part pixels are ignored in training. In testing, parts are segmented on their associated object score map given by the object stream. The training losses for the two streams and for the three streams are $\mathcal{L} = \mathcal{L}_{stuff} + \mathcal{L}_{object}$ and $\mathcal{L} = \mathcal{L}_{stuff} + \mathcal{L}_{object} + \mathcal{L}_{part}$ respectively.

The configurations of each layer are based on the baseline network being used. We integrate the proposed module on two baseline networks Segnet [1] and DilatedNet [4, 34]. In the following experiments, we evaluate that the proposed module brings great improvements for scene parsing.

4. Experiments

To train the networks for scene parsing, we select the top 150 objects ranked by their total pixel ratios from the ADE20K dataset and build a scene parsing benchmark of

ADE20K, termed as MIT SceneParse150[2]. As the original images in the ADE20K dataset have various sizes, for simplicity we rescale those large-sized images to make their minimum heights or widths as 512 in the benchmark. Among the 150 objects, there are 35 stuff classes (i.e., wall, sky, road) and 115 discrete objects (i.e., car, person, table). The annotated pixels of the 150 objects occupy 92.75% of all the pixels in the dataset, where the stuff classes occupy 60.92%, and discrete objects occupy 31.83%.

4.1. Scene parsing

As for baselines of scene parsing on SceneParse150 benchmark, we train three semantic segmentation networks: SegNet [1], FCN-8s [19], and DilatedNet [4, 34]. SegNet has encoder and decoder architecture for image segmentation; FCN upsamples the activations of multiple layers in the CNN for pixelwise segmentation; DilatedNet drops `pool4` and `pool5` from fully convolutional VGG-16 network, and replaces the following convolutions with dilated convolutions (or atrous convolutions), a bilinear upsampling layer is added at the end.

We integrate the proposed cascade segmentation module on the two baseline networks: SegNet and DilatedNet. We did not integrate it with FCN because the original FCN requires a large amount of GPU memory and has skip connections across layers. For the Cascade-SegNet, two streams share a single encoder, from `conv1_1` to `conv5_3`, while each stream has its own decoder, from `deconv5_3` to `loss`. For the Cascade-DilatedNet, the two streams split after `pool3`, and keep spatial dimensions of their feature maps afterwards. For a fair comparison and benchmark purposes, the cascade networks only have stuff stream and object stream. We train these network models using the Caffe library [15] on NVIDIA Titan X GPUs. Stochastic gradient descent with 0.001 learning rate and 0.9 momentum is used as optimizer, and we drop learning rate every 10k iterations.

Results are reported in four metrics commonly used for semantic segmentation [19]: **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.

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 will be made available to the public. We take the average of the pixel accuracy and mean IoU as the evaluation criteria in the challenge.

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

Table 2. 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
DilatedNet	73.55%	44.59%	0.3231	0.6014
Cascade-SegNet	71.83%	37.90%	0.2751	0.5805
Cascade-DilatedNet	**74.52%**	**45.38%**	**0.3490**	**0.6108**

Table 3. Performance of stuff and discrete object segmentation.

	35 stuff		115 discrete objects	
Networks	Mean Acc.	Mean IoU	Mean Acc.	Mean IoU
FCN-8s	46.74%	0.3344	38.36%	0.2816
SegNet	43.17%	0.3051	27.48%	0.1894
DilatedNet	49.03%	0.3729	43.24%	0.3080
Cascade-SegNet	40.46%	0.3245	37.12%	0.2600
Cascade-DilatedNet	**49.80%**	**0.3779**	**44.04%**	**0.3401**

The segmentation results of the baselines and the cascade networks are listed in Table 2. Among the baselines, the DilatedNet achieves the best performance on the SceneParse150. The cascade networks, Cascade-SegNet and Cascade-DilatedNet both outperform the original baselines. In terms of mean IoU, the improvement brought by the proposed cascade segmentation module for SegNet is 6%, and for DilatedNet is 2.5%. We further decompose the performance of networks on 35 stuff and 115 discrete object classes respectively, in Table 3. We observe that the two cascade networks perform much better on the 115 discrete objects compared to the baselines. This validates that the design of cascade module helps improve scene parsing for the discrete objects as they have less training data but more visual complexity compared to those stuff classes.

Segmentation examples from the validation set are shown in Fig. 7. Compared to the baseline SegNet and DilatedNet, the segmentation results from the Cascade-SegNet and Cascade-DilatedNet are more detailed. Furthermore, the objectness maps from the stuff stream highlight the possible discrete objects in the scenes.

4.2. Part segmentation

For part segmentation, we select the eight object classes frequently annotated with parts: 'person', 'building', 'car', 'chair','table', 'sofa', 'bed', 'lamp'. After we filter out the part classes of those objects with instance number smaller than 300, there are 36 part classes included in the training and testing. We train the part stream on the Cascade-DilatedNet used in the scene parsing.

The results of joint segmentation for stuff, objects, and object parts are shown in Fig. 8. In a single forward pass the network with the proposed cascade module is able to parse scenes at different levels. We use the accuracy instead of the IoU as the metric to measure the part segmentation results, as the parts of object instances in the dataset are not fully annotated. The accuracy for all the parts of the eight objects is plotted in Fig.8.a The average accuracy is 55.47%.

4.3. Further applications

We show two applications of the scene parsing below:

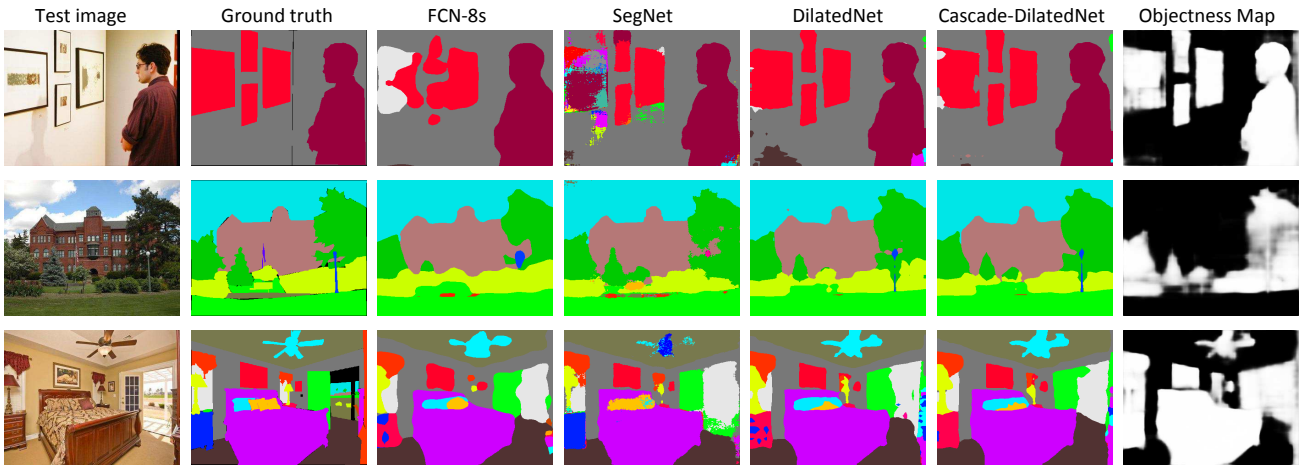

Figure 7. Ground-truths, segmentation results given by the networks, and objectness maps given by the Cascade-DilatedNet.

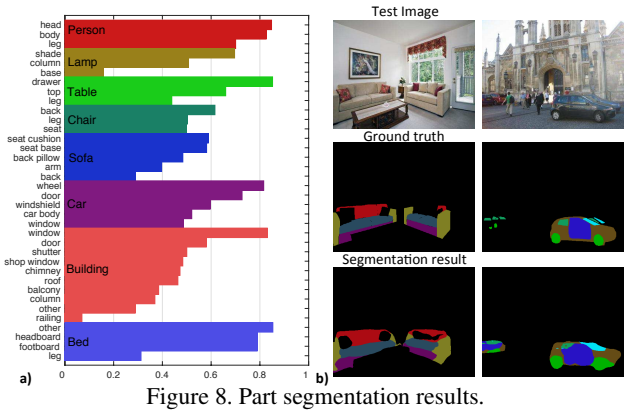

Figure 8. Part segmentation results.

Automatic image content removal. Image content removal methods typically require the users to annotate the precise boundary of the target objects to be removed. Here, based on the predicted object probability map from Cascade-DilatedNet, we automatically identify the image region of the target objects. After cropping out the target objects using the predicted object probability map, we simply use image completion/inpainting methods to fill the holes in the image. Fig. 9.a shows some examples of the automatic image content removal. It can be seen that with the predicted object score maps, we are able to crop out the objects from the image in a precise way. We used the image completion technique described in [14].

Scene synthesis. Given an scene image, the scene parsing network could predict a semantic label mask. Furthermore, by coupling the scene parsing network with the recent image synthesis technique proposed in [23], we could also synthesize a scene image given the semantic label mask. The general idea is to optimize the code input of a deep image generator network to produce an image that highly activates the pixel-wise output of the scene parsing network. Fig. 9.b shows two 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.

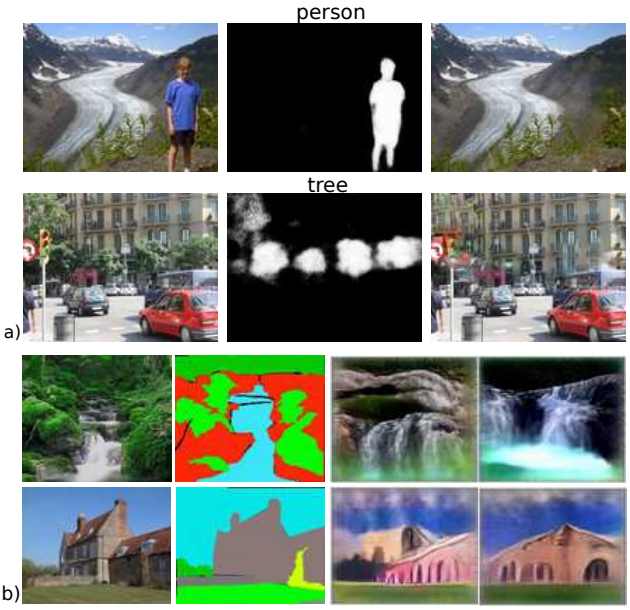

Figure 9. Applications of scene parsing: a) Automatic image content removal using the object score maps predicted by the scene parsing network. The first column shows the original images, the second column contains the object score maps, the third column shows the filled-in images. b) Scene synthesis. Given annotation masks, images are synthesized by coupling the scene parsing network and the image synthesis method proposed in [23].

5. Conclusion

In this paper, we introduced a new densely annotated dataset with the instances of stuff, objects, and parts, covering a diverse set of visual concepts in scenes. A generic network design was proposed to parse scenes into stuff, objects, and object parts in a cascade.

Acknowledgement: This work was supported by Samsung and NSF grant No.1524817 to AT. SF acknowledges the support from NSERC. BZ is supported by Facebook Fellowship.

References

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

[2] S. Bell, P. Upchurch, N. Snavely, and K. Bala. OpenSurfaces: A richly annotated catalog of surface appearance. *ACM Trans. on Graphics (SIGGRAPH)*, 32(4), 2013.

[3] S. Bell, P. Upchurch, N. Snavely, and K. Bala. Material recognition in the wild with the materials in context database. In *Proc. CVPR*, 2015.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *arXiv:1606.00915*, 2016.

[5] X. Chen, R. Mottaghi, X. Liu, N.-G. Cho, S. Fidler, R. Urtasun, and A. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proc. CVPR*, 2014.

[6] U. o. I. a. U.-C. Computer Vision Group. Cross-category object recognition. In *http://vision.cs.uiuc.edu/CORE/*, 2009.

[7] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset. In *CVPR Workshop on The Future of Datasets in Vision*, 2015.

[8] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *Proc. CVPR*, 2015.

[9] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. *Proc. CVPR*, 2016.

[10] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int'l Journal of Computer Vision*, 2010.

[11] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2013.

[12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. CVPR*, 2012.

[13] G. Heitz and D. Koller. Learning spatial context: Using stuff to find things. In *Proc. ECCV*, 2008.

[14] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf. Image completion using planar structure guidance. *ACM Transactions on Graphics (TOG)*, 2014.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Proc. ECCV*. 2014.

[18] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *Proc. CVPR*, 2009.

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, 2015.

[20] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. ICCV*, 2001.

[21] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proc. CVPR*, 2014.

[22] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *Proc. ECCV*, 2012.

[23] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *NIPS 2016*, 2016.

[24] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proc. ICCV*, 2015.

[25] P. H. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *ICML*, pages 82–90, 2014.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int'l Journal of Computer Vision*, 115(3):211–252, 2015.

[27] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *Int'l Journal of Computer Vision*, 2008.

[28] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proc. CVPR*, 2015.

[29] M. Spain and P. Perona. Measuring and predicting object importance. *International Journal of Computer Vision*, 2010.

[30] J. Tighe and S. Lazebnik. Understanding scenes on many levels. In *Proc. ICCV*, 2011.

[31] J. Tighe and S. Lazebnik. Finding things: Image parsing with regions and per-exemplar detectors. In *Proc. CVPR*, 2013.

[32] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*, 2010.

[33] J. Yang, B. Price, S. Cohen, and M.-H. Yang. Context driven scene parsing with attention to rare classes. In *Proc. CVPR*, 2014.

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

[35] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *In Advances in Neural Information Processing Systems*, 2014.