Not All Pixels Are Equal: Difficulty-Aware Semantic Segmentation via Deep Layer Cascade

Xiaoxiao Li[1] Ziwei Liu[1] Ping Luo[2,1] Chen Change Loy[1,2] Xiaoou Tang[1,2]
[1]Department of Information Engineering, The Chinese University of Hong Kong
[2]Shenzhen Key Lab of Comp. Vis. & Pat. Rec., Shenzhen Institutes of Advanced Technology, CAS, China
{lx015,lz013,pluo,ccloy,xtang}@ie.cuhk.edu.hk

Abstract

We propose a novel deep layer cascade (LC) method to improve the accuracy and speed of semantic segmentation. Unlike the conventional model cascade (MC) that is composed of multiple independent models, LC treats a single deep model as a cascade of several sub-models. Earlier sub-models are trained to handle easy and confident regions, and they progressively feed-forward harder regions to the next sub-model for processing. Convolutions are only calculated on these regions to reduce computations. The proposed method possesses several advantages. First, LC classifies most of the easy regions in the shallow stage and makes deeper stage focuses on a few hard regions. Such an adaptive and 'difficulty-aware' learning improves segmentation performance. Second, LC accelerates both training and testing of deep network thanks to early decisions in the shallow stage. Third, in comparison to MC, LC is an end-to-end trainable framework, allowing joint learning of all sub-models. We evaluate our method on PASCAL VOC and Cityscapes datasets, achieving state-of-the-art performance and fast speed.

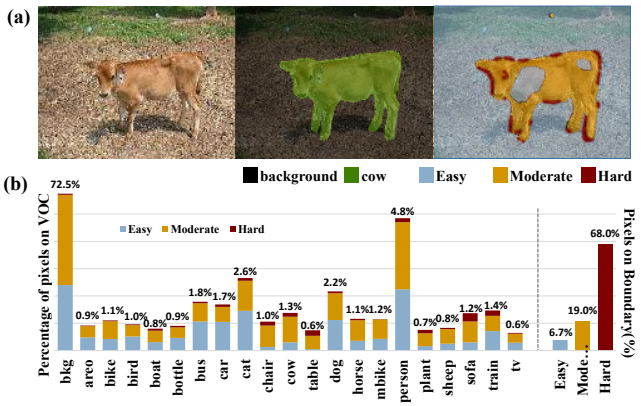


Figure 1: (a) shows an image of 'cow' and 'background' (left) and its ground truth label map (middle) from the Pascal VOC 2012 dataset. The difficulty level (*e.g.* recognizability) of pixels are visualized in the right image, where pixels are partitioned into three sets, including 'easy' (ES), 'moderate' (MS), and 'extremely hard' (HS) sets. (b) depicts two histograms. The left one plots the percentages of pixels in VOC validation set with respect to each object category. It can be observed that ES occupies at least 30% pixels of most objects. The right one reveals that 70% pixels in HS are located at object boundaries, which have large ambiguity. **Best viewed in color with 300% zoom**.

1. Introduction

Semantic image segmentation enjoys wide applications, such as video surveillance [9, 36] and autonomous driving [10, 5]. Recent advanced deep architectures, such as the residual network (ResNet) [13] and Inception [32], significantly improve the accuracy of image segmentation by increasing the depth and number of parameters in deep models. For example, ResNet-101 is six times deeper than VGG-16 [29] network, with the former outperforms the latter by 4 percent on the challenging PASCAL VOC 2012 image segmentation benchmark [8].

Although promising results can be achieved through the increase of model capacity, they come with a price of runtime complexity, which impedes the deployments of existing deep models in many applications that demand real-time performance. For instance, the segmentation speeds of VGG, ResNet-101, and Inception-ResNet on a 300×500 image are 5.7, 7.1 and 9.0 frame per second (FPS), which are far away from real time. To address this issue, this work presents *Deep Layer Cascade* (LC), which not only substantially reduces the runtime of deep models, but also improves their segmentation accuracies. Many deep architectures, including VGG, ResNet, and Inception, can benefit from the above appealing properties by adapting their structures into LC.

Layer Cascade inherits the advantage of the conventional model cascade (MC) [18, 35], which has multiple stages and usually trains one classifier in each stage. MC is capable of increasing both speed and accuracy for object

detection, because the earlier stages (classifiers) reject most of the easy samples (detection windows) and the later stages can pay attention on a small number of difficult samples, thus reducing false alarms. Different from MC, LC is carefully devised for deep models in the task of image segmentation. It considers different layers in a deep network as different stages. In particular, most of the pixels in an image are recognizable by the lower stages and the higher stages, which typically possess far more parameters than the bottom layers, are learned to recognize a small set of challenging pixels. In this case, the runtime of deep models can be significantly reduced by LC. Moreover, unlike MC that learns the current stage by keeping all previous stages fixed, LC trains all stages jointly to boost performance.

Another important difference between LC and MC is the cascade strategy. In MC, the current stage propagates a sample to the next stage, if its classification score or probability (*i.e*. the response after softmax) is higher than a large threshold, such as 0.95, indicating that this sample is classified as positive by the current stage with 95% confidence. In other words, later stages refine the labels of samples that are considered highly positive in the previous stages, so as to reduce false alarms.

In contrast, LC 'rejects' samples with high scores in earlier stages, but those samples with low and moderate confidences are propagated forward. Figure 1 takes the segmentation results of LC as an example to illustrate this cascade strategy. In (a), an image of 'cow' and 'background' and its ground truth label map from the VOC validation set (VOC val) are shown on the left and middle respectively. We partition all pixels in the validation set into three different sets, namely "easy", "moderate", and "extremely hard" sets. The easy set (ES) contains pixels that are correctly classified with larger than 95% confidence, while the extremely hard set (HS) comprises pixels that are misclassified with larger than 95% confidence. The moderate set (MS) covers pixels that have classification scores smaller than 0.95.

In a certain stage of LC, ES and HS are discarded and MS is propagated to the next stage, because of the following two reasons. First, as shown in the right histogram of Fig. 1(b), we observe that almost 70 percent[1] pixels in HS are located on the boundaries between objects, demonstrating that these pixels are extremely hard to be recognized because of large ambiguity. An example is given by the right image of Fig. 1(a). Fitting HS during training may lead to over-fitting in the test stage. Second, the left histogram of Fig. 1(b) plots the percentages of pixels with respect to each object category in VOC val. For most of the categories, we found that at least 30 percent pixels belong to ES. As the background pixels are dominated (72.5%), rejecting ES and HS reduces more than 40 present pixels in earlier stages and thus significantly reduces computations of deep networks, while improves accuracy, by enabling deeper layers to focus on foreground objects.

This study makes three main **contributions**. (1) This is the first attempt to identify the segmentation difficulty of pixels for deep models. With this observation, a novel *Deep Layer Cascade* (LC) approach is proposed to significantly reduce computations of deep networks while improving their segmentation accuracies. (2) LC's properties can be easily applied to many recent advanced network structures. After applying LC on Inception-ResNet-v2 (IRNet) [32], its speed and accuracy are improved by 42.8% and 1.7%, respectively. (3) Connections between LC and previous models such as model cascade, deeply supervised network [17], and dropout [30] are clearly presented. Extensive studies are conducted to demonstrate the superiority of LC.

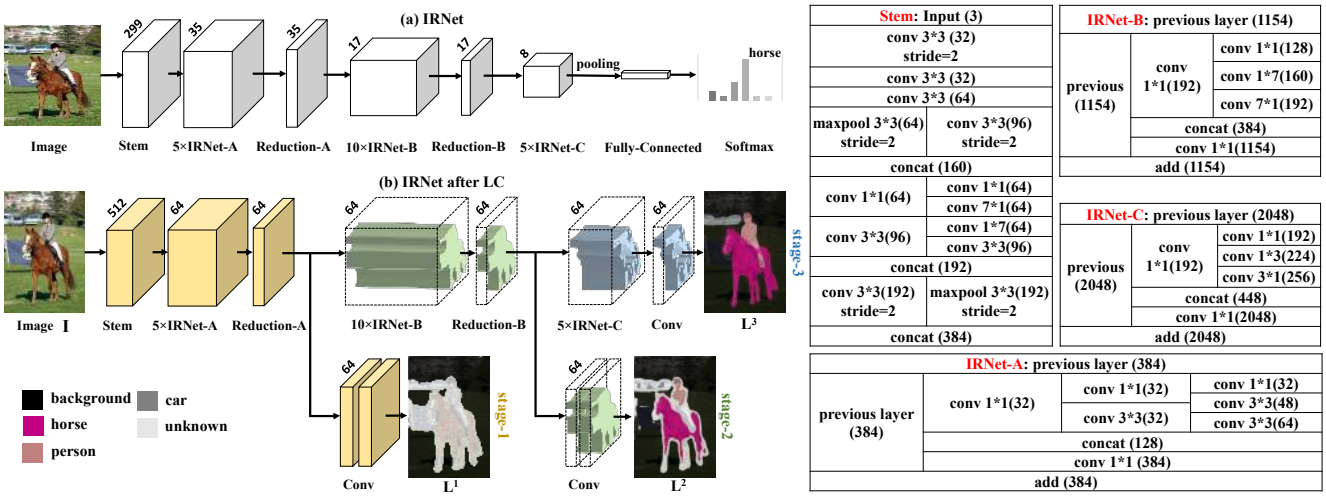
2. Related Work

Semantic Image Segmentation. While early efforts focused on structural models with handcrafted features [15, 16, 34, 38], recent studies employ deep convolutional neural network (CNN) to learning strong representation, which improves segmentation accuracy significantly [3, 22, 23, 25, 40]. For instance, Long *et al*. [25] transformed fully-connected layers of CNN into convolutional layers, making accurate per-pixel classification possible using the contemporary CNN architectures that were pre-trained on ImageNet [7]. Chen *et al*. [3], Zheng *et al*. [40], and Liu *et al*. [22, 23] further showed that back-propagation and inference of Markov Random Field (MRF) can be incorporated into CNN. Though attaining high accuracy, these models generally have high computational costs, preventing them from deploying in real-time.

Another line of research [1, 21, 27] alleviates this problem by using lightweight network architectures. For example, SegNet [1] adopted a convolutional encoder-decoder and removed unnecessary layers to reduce the number of parameters. ENet [27] utilized a bottleneck module to reduce computation of convolutions. Although these networks are speeded up, they sacrificed high performances as presented in previous deep models. This work proposes Deep Layer Cascade (LC), which improves both speed and accuracy of existing deep networks. It achieves state-of-the-art performances on both Pascal VOC and Cityscape datasets, and runs in real time.

Deep Learning Cascade. Network cascades [2, 18, 26, 33, 24] have been studied to improve the performance in classification [26], detection [18], and pose estimation [33]. For example, Deep Decision Network [26] improved the image classification performance by dividing easy data from the

[1]We found that the other 30 percent pixels in HS have wrong annotations. Since our purpose is to improve speed and accuracy of deep models, we do not correct those wrong annotations to enable a fair comparison with previous works.



Stem: Input (3)
conv 3*3 (32) stride=2
conv 3*3 (32)
conv 3*3 (64)
maxpool 3*3(64) stride=2 / conv 3*3(96) stride=2
concat (160)
conv 1*1(64) / conv 1*1(64)
conv 7*1(64)
conv 3*3(96) / conv 1*7(64)
conv 3*3(96)
concat (192)
conv 3*3(192) stride=2 / maxpool 3*3(192) stride=2
concat (384)

IRNet-B: previous layer (1154)		
previous (1154)	conv 1*1(192)	conv 1*1(128)
		conv 1*7(160)
		conv 7*1(192)
	concat (384)	
	conv 1*1(1154)	
add (1154)		

IRNet-C: previous layer (2048)		
previous (2048)	conv 1*1(192)	conv 1*1(192)
		conv 1*3(224)
		conv 3*1(256)
	concat (448)	
	conv 1*1(2048)	
add (2048)		

IRNet-A: previous layer (384)			
previous layer (384)	conv 1*1(32)	conv 1*1(32)	conv 1*1(32)
			conv 3*3(48)
		conv 3*3(32)	conv 3*3(64)
	concat (128)		
	conv 1*1 (384)		
add (384)			

Figure 2: (a) depicts the Inception-ResNet-v2 (IRNet) for classification task. (b) is the architecture of Layer Cascade IRNet (IRNet-LC). The tables at the right show the structure of IRNet.

hard ones. The hard cases with high confusion will be propagated and handled by the subsequent expert networks. Li *et al*. [18] used CNN cascade for face detection, which rejects false detections quickly in early stages and carefully refines detections in later stages. DeepPose [33] employed a divide-and-conquer strategy and designed a cascaded deep regression framework for human pose estimation. Different from previous network cascades that train each network separately, LC is jointly optimized to boost the segmentation accuracy.

3. Deep Layer Cascade (LC)

Sec. 3.1 takes Inception-ResNet-v2 [32] as an example to illustrate how one could turn a deep model into LC. The approach can be easily generalized to the other deep networks. Sec. 3.3 introduces the training algorithm of LC.

3.1. Turning a Deep Model into LC

Network Overview. To illustrate the effectiveness of LC, we choose Inception-ResNet-v2 pre-trained on ImageNet dataset as a strong baseline, denoted as IRNet, which outperforms ResNet-101 by 1.2% on the Pascal VOC2012 validation set. Experiments demonstrate that LC is able to achieve 1.7% improvement on this competitive baseline.

Figure 2 (a) visualizes the architecture of IRNet, which has six different components, including 'Stem', 'IRNet-A/B/C', and 'Reduction-A/B'. Different components have different configurations of layers, such as convolution, pooling, and concatenation layer. The right column of Fig. 2 shows the structures of 'Stem' and 'IRNet-A/B/C' respectively, including layer types, kernel sizes, and the number of channels (in bracket). The stride typical equals one unless otherwise stated. For example, 'Stem' employs an RGB image as an input and produces features of 384 channels. More specifically, the input image is forwarded to three convolutional layers with 3×3 kernels, and then the learned features are split into two streams, which have 3 and 5 convolutional layers respectively.

Similar network structure as IRNet has achieved great success in image recognition [32]. However, two important modifications are necessary to adapt it to image segmentation. Firstly, to increase the resolution of prediction, we remove the pooling layer at the end of IRNet and enlarge the size of feature maps by decreasing the convolutional strides in 'Reduction-A/B' (from 2 to 1). In this case, we expand the size of network outputs (label maps) by $4\times$. We also replace convolutions in 'IRNet-B/C' by the dilated convolutions similar to [3]. Secondly, as feature maps with high resolution consume a large amount of GPU memory in the learning process, they limit the size of mini-batch (*e.g*. 8), making the batch normalization (BN) layers [14] unstable (as which need to estimate sample mean and variance from the data in a mini-batch). We cope with this issue by simply fixing the values of all parameters in BNs. This strategy works well in practice.

From IRNet to LC (IRNet-LC). IRNet is turned into LC by dividing its different components as different stages. The number of stages is three, which is a common setting in previous cascade methods [18, 31, 33]. As shown in Fig. 2 (b), components before 'Reduction-A' are considered as the first stage, components between 'Reduction-A' and '-B' are the second stage, and the remaining layers become the third stage. In Fig. 2 (b), these three stages are distinguished in yellow, green, and blue respectively. For instance, stage-1 contains one 'Stem', five 'IRNet-A', and one 'Reduction-A'. In addition, we append two convolutional layers and a softmax loss at the end of each stage. In this case, the original IRNet with one loss function develops into multiple

stages, where each stage has its own loss function.

Now we introduce the information flows for three stages in IRNet-LC. In the first stage as shown in Fig. 2 (b), given a $3\times512\times512$ image I, stage-1 predicts a $21\times64\times64$ segmentation label map L^1, where each 21×1 column vector, denoted as $L^1_i \in \mathbb{R}^{21\times1}$, indicates the probabilities (confidence scores) of the i-th pixel belonging to 21 object categories in VOC respectively. We have $\sum_{j=1}^{21} L^1_{ij} = 1$, which can be satisfied by using the softmax function. If the maximum score of the i-th pixel, $\ell^1_i = \max(L^1_i)$ and $\ell^1_i \in \{L^1_{ij}|j = 1...21\}$, is larger than a threshold ρ ($\ell^1_i \geq \rho$), we accept its prediction and do not propagate it forward to stage-2. The value of ρ is usually larger than 0.95. As introduced in Sec. 1, those pixels in stage-1 that fulfil $\ell^1 \geq 0.95$ occupy nearly 40% region of an image, containing a lot of easy pixels and a small number of extremely hard pixels that have high confidence to be misclassified. Removing them from the network significantly reduces computations and improves accuracy, by enabling deeper layers to focus on foreground objects.

Stage-2 strictly follows the same procedure as above to determine which pixel is forwarded to stage-3. In other words, LC only introduces one hyper-parameter ρ to IRNet. In our implementation, the value of ρ is the same for both stage-1 and -2. Specifically, ρ represents how many easy and extremely hard pixels are rejected (discarded) in each stage. A larger value of ρ rejects a smaller number of pixels, whilst smaller ρ discards more pixels. To the extreme, when $\rho = 1.0$, no pixels are rejected. IRNet-LC becomes the original IRNet. When $\rho = 0.9$, 52% and 35% pixels are discarded in stage-1 and -2 respectively.

However, if ρ becomes smaller, *i.e.* $\rho < 0.9$, more 'moderate' pixels that locate on the important parts of objects are discarded, hindering the performance of the deep model. Experiments show that IRNet-LC is robust when $\rho \in [0.9, 1.0]$. For example, when $\rho = 0.95$, IRNet-LC obtains nearly realtime of 18 FPS compared to 9 FPS of IRNet, while outperforms it by 0.8% accuracy on VOC val. When $\rho = 0.985$, IRNet-LC improves IRNet by 1.7% with a speed of 15 FPS.

After propagating an image through all three stages, we directly combine the predicted label maps of these stages as the final prediction, because different stages predict different regions. For example, as shown in Fig. 2 (b), stage-1 trusts the predictions in most of the 'background' (pixels with $\ell^1_i \geq \rho$) and propagates the other region forward. Pixels in this region are marked as 'unknown' because $\ell^1_i < \rho$. In stage-2, 'IRNet-B' and 'Reduction-B' only compute convolutions with respect to the forwarded region. It is learned to predict 'harder' region, such as 'person' and 'horse'. This process is repeated in stage-3.

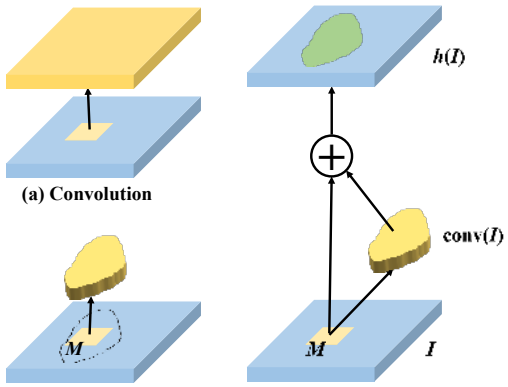


Figure 3: (a) shows the conventional convolution that operates on an entire image. (b) is region convolution (RC) where filters only convolve irregular region of interest denoted as M. Values of the other region are set as zeros. (c) illustrates RC in a residual module. **Best viewed in color.**

3.2. Region Convolution

As presented above, stage-2 and -3 only calculate convolutions on those pixels that have been propagated forward. Fig. 3(b) illustrates this region convolution (RC) compared to the traditional convolution in (a), which is applied on an entire feature map. The filters in RC only convolves a region of interest, denoted as M, and ignores the other region, reducing computations a lot. The values of the other region are directly set as zeros. M can be implemented as a binary mask, where the pixels inside M equal one, otherwise zero.

Specifically, (c) shows how to apply RC on a residual module, which can be represented as $h(I) = I + \text{conv}(I)$, where feature h is attained by an identity mapping [13] of I and a convolution over I. We replace the conventional convolution with a RC as introduced above, and the feature $h'(I)$ is the elementwise sum between I and the output of RC. This is equivalent to learn a masked residual representation, where values inside M are the outputs of RC and those outside M are copied from I. It works well because different stages in LC cope with different non-overlapping regions, and each stage only needs to learn features of regions it concerns.

3.3. Training IRNet-LC

The parameters of IRNet are initialized by pre-training in ImageNet. Since IRNet-LC has additional convolutional layers stacked before each loss function, their parameters are initialized by sampling from a normal distribution. Given a set of images and their per-pixel label maps, IRNet-LC is learned in two steps, where the first one aims at initial training and the second one employs cascade training.

Initial Training. This step is similar to deeply supervised network (DSN) [17], which has multiple identical loss functions in different layers of the network. Its objective

is to adapt IRNet pre-trained by classifying one thousand image categories in ImageNet to the task of image segmentation. It learns discriminative and robust features. In IRNet-LC, every stage is trained to minimize a pixel-wise softmax loss function, measuring the discrepancies between the predicted label map and the ground truth label map of the entire image. These loss functions are jointly optimized by using back-propagation (BP) and stochastic gradient descent (SGD).

Cascade Training. Once we finish the initial training, we fine-tune each stage of IRNet-LC by leveraging the cascade strategy of ρ as introduced in Sec. 3.1. Similar to the previous step, all stages are trained jointly, but different stages minimize their pixel-wise softmax losses with respect to different regions. More specific, the gradients in BP are only propagated to the region of interest in each stage, which is able to learn discriminative features corresponding to regions (pixels) in a specific difficulty-level. Intuitively, the current stage is fine-tuned on pixels that have low confidences in the previous stage, enabling 'harder' pixels to be captured by deeper layers to improve segmentation accuracy and reduce computation.

3.4. Relations with Previous Models

The relationships and differences between LC and MC have been discussed in Sec. 1. LC also relates to deeply supervised nets (DSN) [17] and dropout [30].

DSN. Similar to DSN, LC adds supervision to each stage. However, to enable adaptive processing of hard/easy regions, LC employs different supervisions for different stages. In contrast, the supervision used in each stage of DSN are kept the same. Specifically, the stage-wise supervision in LC is determined by the estimated difficulty of each pixel. In this way, each stage in LC is able to focus on regions with a similar difficulty level.

Dropout. LC connects to dropout in the sense that both methods discard some regions in the feature maps, but they are essentially different. LC drops those pixels with high confidences and only propagates difficult pixels forward to succeeding stages. The easy and ambiguous regions are perpetually dropped in upper layers so as to reduce computations and the deeper layers focus more on 'hard' regions such as foreground objects. Dropout randomly zeros out pixels in each layer independently. It prevents over-fitting but slightly increases computations. In the experiment, LC is compared with dropout to identify that the performance gain mainly comes from the proposed cascade strategy.

4. Experiments

Settings. We evaluate our method on the PASCAL VOC 2012 (VOC12) [8] and Cityscapes [5] datasets. VOC12 dataset is a generic object segmentation benchmark with

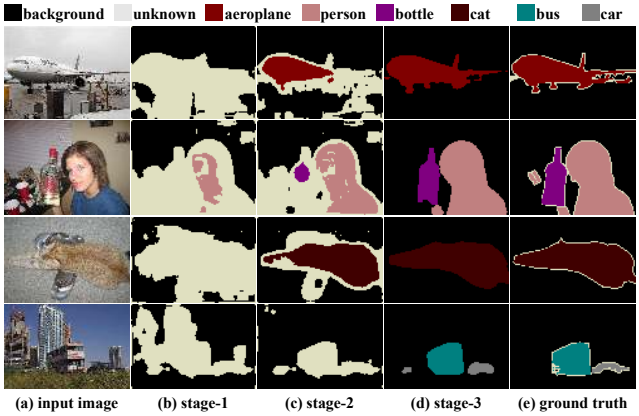


Figure 4: Visualization of different stages' outputs in VOC12 dataset. **Best viewed in color.**

21 classes. Following previous works, we also use the extra annotations provided by [12], which contains $10,582$ images for training, $1,449$ images for validation, and $1,456$ images for testing. Cityscapes dataset, on the other hand, focuses on street scenes segmentation and contains 19 categories. In our experiments, we only employ images with fine pixel-level annotations. There are 2975 training, 500 validation and 1525 testing images. This is consistent with existing studies [19, 4]. We adopt mean intersection over union (mIoU) to evaluate the performance of different methods.

4.1. Ablation Study

In this section, we investigate the effects of adjusting probability threshold in LC and demonstrate the merits of LC by comparing it to other counterparts. All performance are reported on the *validation set* of VOC12.

Probability Thresholds. In each stage of LC, we employ a pixel-wise probability from softmax layer to represent the confidence of prediction. By choosing appropriate probability threshold ρ, LC can separate easy regions, moderate regions and extremely hard regions for adaptive processing. As discussed in Sec. 3.1, ρ controls how many easy and extremely hard pixels are discarded in each stage.

Table 1 lists the processed pixel percentage in stage-1 & -2 and the overall performance as ρ varies. If $\rho = 1$, LC will degenerate to DSN, which is slightly better than fully convolutional IRNet. When ρ decreases, more easy regions are classified in early stages while hard regions are progressively handled by later stages. It can be understood as hard negative mining [11, 28] which improves the performance. On the other hand, if the value of ρ is too small, the algorithm might become too optimistic, *i.e.* many hard regions are processed in early stages and early decisions are made. The performance will be harmed by overly early decisions when hard regions do not receive sufficient inference using deeper layers. As shown in Table 1, when

Table 1: Ablation study on probability thresholds ρ.

ρ	1	0.995	0.985	0.970	0.950	0.930	0.900	0.800
stage-1 (%)	0	15	23	30	35	35	44	56
stage-2 (%)	0	14	29	31	30	41	31	29
mIoU (%)	72.70	73.56	**73.91**	73.63	73.03	72.53	71.20	66.95

Table 2: Comparisons with related methods.

	mIoU(%)
IRNet [32]	72.22
DSN [17]	72.70
DSN [17] + Dropout [30]	72.63
Model Cascade (MC)	44.20
Layer Cascade (LC)	**73.91**

$\rho = 0.985$, *i.e.*, LC processes around 52% regions in early stages and achieves the best performance. This value is used in all the following experiments. In practice, the value of ρ can be chosen empirically using a validation set.

Effectiveness of Layer Cascade. To show the merits of LC, we compare it to some important counterparts as discussed in Sec. 3.4, including:

- **IRNet [32]:** We use the model describe in Sec. 3.1 as baseline. To conduct a fair comparison, all the following methods are based on this backbone network.
- **DSN [17]:** By setting ρ = 1, we make LC degenerate to a DSN, where each stage process all regions and has full supervision as the final target.
- **DSN [17] + Dropout [30]:** To distinguish our method from dropout, LC is compared against DSN equipped with random label dropout in each stage. We keep the dropout ratio identical as that in LC.
- **Model Cascade:** MC has a similar network architecture to LC, but with different training strategy as discussed in Sec. 1. Specifically, MC divides the IRNet into three stages, and each stage is trained separately. When we train a certain stage, we fix the parameters of all previous stages. The same threshold as in LC is employed here, *i.e.*, $\rho = 0.985$.

The results are summarized in Table 2. We have three observations here. Firstly, the improvement from deep supervision (DSN) is relatively limited, which only leads to 0.48 mIoU gain in comparison to the baseline IRNet. Since pre-training on ImageNet has been a common practice in semantic segmentation [25], which effectively prevents gradients exploding or vanishing, it renders the advantages of deeply supervision marginal. Secondly, random label dropout does not bring significant effect to the result. The result is expected because the dropout technique is designed to alleviate the hazard of overfitting given small training data size. However, semantic segmentation is a per-pixel labeling task and we have abundant training data to support the learning task. Thirdly, Model Cascade (MC) performs even worse than the baseline IRNet. It is because MC divides the IRNet into several independent sub-models. But each sub-model is shallow and therefore weaken the overall modeling capacity. On the contrary, LC has the appealing properties of cascading and also keeping the *intrinsic depth* for the whole model. The capability of maintaining the model depth adaptively for hard regions makes our approach outstanding in the comparison.

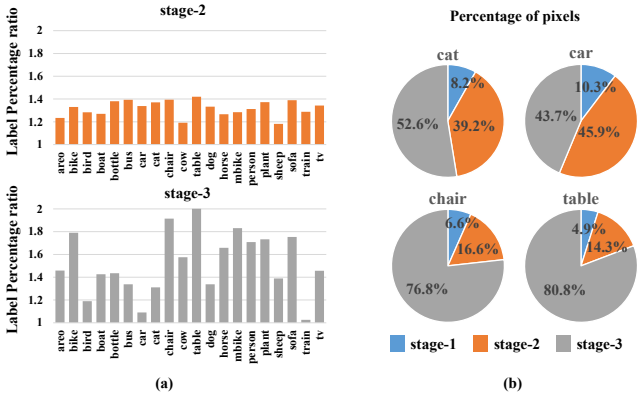


Figure 5: (a) is the change of label distribution in stage-2 and -3. (b) shows the percentage of pixels that are classified in different stages.

4.2. Stage-wise Analysis

In this section, we demonstrate how LC enables adaptive processing for different classes and visualize the regions handled by different regions.

Stage-wise Label Distribution. First, we provide a label distribution analysis across different stages. Here we take the 20 classes (excluding "background") in VOC12 as an example. Fig. 5 (a) shows how the number of pixels changes with respect to each class in stage-2 and -3. For example, the upper histogram shows a ratio for each class, obtained by dividing its number of pixels in stage-2 by those in stage-1. Ratios larger than one indicates stage-2 focus more on the corresponding classes than stage-1 does. We find that all ratios have increased and belong to the range of 1 to 1.4. It is because stage-1 already handles the easy regions (*i.e.* "background") and leaves the hard regions (*i.e.* "foreground") to stage-2. Ratios of stage-3 can be obtained similarly in the bottom histogram. When comparing stage-3 to -2, we can see that stage-3 further focus on harder classes (*e.g.* "bicycle", "chair" and "dining table"). LC learns to process samples in a "difficulty-aware" manner. We also conduct a per-class analysis as illustrated in Fig. 5 (b). Harder classes like "chair" and "table" have more pixels handled by deeper layers (stage-3).

Stage-wise Visualization. Here we visualize the output label maps of different stages for both VOC12 and

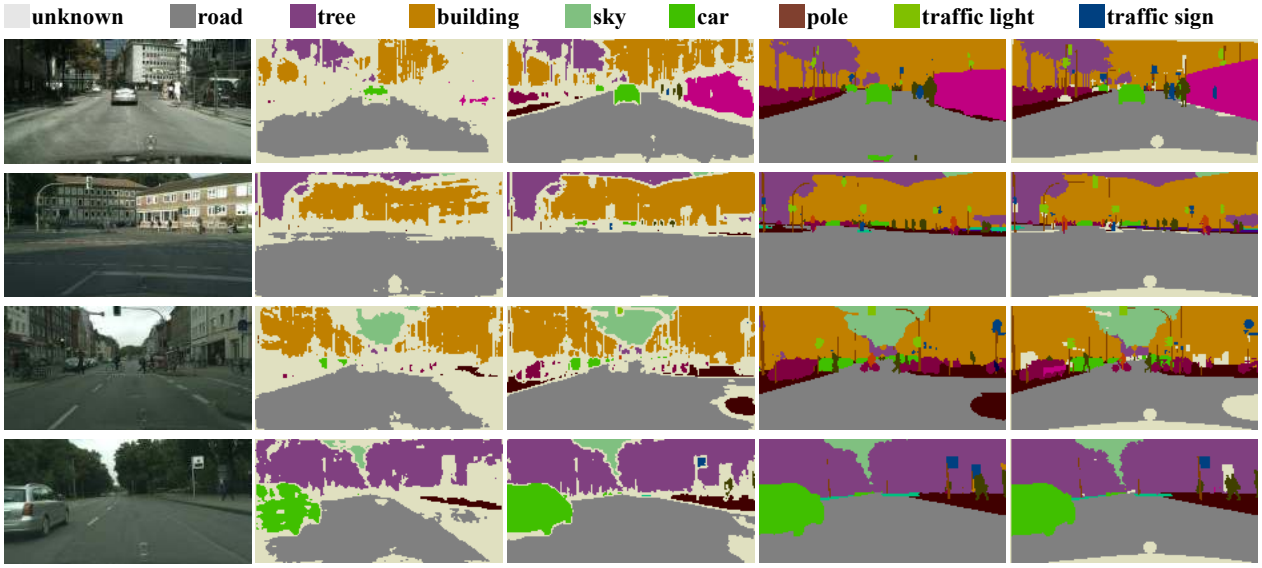


Figure 6: Visualization of different stages' outputs in Cityscapes dataset. **Best viewed in color.**

Cityscapes, as shown in Fig. 4 and 6. The uncertain regions in different stages are also marked out. In VOC12, the easy regions like "background" and "human faces" are first labeled by stage-1 in LC. The remaining foreground and boundary regions are then progressively labeled by stage-2 and stage-3 in LC. Similarly, in Cityscapes, the easy regions like "road" and "building" are first labeled by stage-1. Other small objects and fine details like "pole" and "pedestrian" are handled by stage-2 and -3.

4.3. Performance and Speed Analysis

Comparisons with DeepLab and SegNet. To highlight the trade-off between performance and speed, we compare the proposed LC model with two representative state-of-the-art methods, DeepLab-v2 [4] and SegNet [1]. The performance are reported on VOC12 and summarized in Table 3. The runtime speed is measured on a single Titan X GPU. To ensure a fair comparison, we evaluate DeepLab-v2 and SegNet without any pre- and post-processing, *e.g.*, training with extra data, multi-scale fusion, or smoothing with conditional random fields (CRF).

DeepLab-v2 achieves an acceptable mIoU of 70.42. Nonetheless, it uses an ultra-deep ResNet-101 model as the backbone network, its speed of inference is thus slow (7.1 FPS). On the contrary, SegNet is faster due to a smaller model size, however, its accuracy is greatly compromised. In particular, it increases its speed to 14.6 FPS through sacrificing of over 10 mIoU. The proposed LC alleviates the need of trading-off speed with a large drop in performance. The cascaded end-to-end trainable framework with region convolution allows it to achieve the best performance (73.91 mIoU) with an acceptable speed (14.7 FPS).

Table 3: A comparison of performance and speed of Layer Cascade (LC) against existing methods.

	mIoU	ms	FPS
DeepLab-v2 [4]	70.42	140.0	7.1
SegNet [1]	59.90	69.0	14.6
LC	**73.91**	65.1	14.7
LC (fast)	66.95	42.5	**23.6**

Further Performance and Speed Trade-off. It is worth pointing out that the runtime of LC can be further reduced by decreasing ρ to allow more regions to be handled by early stages. The performance and speed trade-off is depicted in Fig. 7 (a) with the corresponding ρ values. It is observed that decreasing ρ slightly affects the accuracy, but it greatly reduces the computation time. Notably, when LC attains real-time inference at 23.6 FPS, it still exhibits competitive mIoU of 66.95, in comparison to mIoU of 70.42 yielded by at 7.1 FPS. We also include the per-stage runtime in Fig. 7 (b). The increasing computation for higher performance mainly comes from later stages.

4.4. Benchmark

In this section, we show that LC can achieve state-of-the-art performance on standard benchmarks like VOC12 [8] and Cityscapes [5] datasets. Following [4], atrous spatial pyramid pooling [4], three-scale testing and dense CRF [16] are employed.

VOC12. Table 4 lists the per-class and overall mean IoU

Table 4: Per-class results on VOC12 *test set*. Approaches pre-trained on COCO [20] are marked with †.

	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mIoU
FCN [25]	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1	62.2
DeepLab [3]	84.4	54.5	81.5	63.6	65.9	85.1	79.1	83.4	30.7	74.1	59.8	79.0	76.1	83.2	80.8	59.7	82.2	50.4	73.1	63.7	71.6
RNN [40]	87.5	39.0	79.7	64.2	68.3	87.6	80.8	84.4	30.4	78.2	60.4	80.5	77.8	83.1	80.6	59.5	82.8	47.8	78.3	67.1	72.0
Adelaide [37]	91.9	48.1	93.4	**69.3**	75.5	94.2	87.5	92.8	36.7	86.9	65.2	89.1	90.2	86.5	87.2	64.6	**90.1**	59.7	85.5	72.7	79.1
RNN† [40]	90.4	55.3	88.7	68.4	69.8	88.3	82.4	85.1	32.6	78.5	64.4	79.6	81.9	86.4	81.8	58.6	82.4	53.5	77.4	70.1	74.7
BoxSup† [6]	89.8	38.0	89.2	68.9	68.0	89.6	83.0	87.7	34.4	83.6	67.1	81.5	83.7	85.2	83.5	58.6	84.9	55.8	81.2	70.7	75.2
DPN† [22]	89.0	61.6	87.7	66.8	74.7	91.2	84.3	87.6	36.5	86.3	66.1	84.4	87.8	85.6	85.4	63.6	87.3	61.3	79.4	66.4	77.5
DeepLab-v2† [4]	92.6	60.4	91.6	63.4	76.3	95.0	88.4	92.6	32.7	88.5	67.6	89.6	**92.1**	87.0	87.4	63.3	88.3	60.0	**86.8**	74.5	79.7
LC	**94.1**	63.0	91.2	67.9	79.5	93.4	**90.0**	**93.8**	37.4	83.7	65.9	**90.7**	86.1	88.8	87.5	68.5	86.9	64.3	85.6	72.2	**80.3**
LC†	85.5	**66.7**	**94.5**	67.2	**84.0**	**96.1**	89.8	93.5	**47.2**	**90.4**	**71.5**	88.9	91.7	**89.2**	**89.1**	**70.4**	89.4	**70.7**	84.2	**79.6**	**82.7**

Table 5: Per-class results on Cityscapes *test set*. "sub" denotes whether the method used subsampling images for training.

	sub	road	swalk	build.	wall	fence	pole	tlight	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIoU
RNN [40]	2	96.3	73.9	88.2	47.6	41.3	35.2	49.5	59.7	90.6	66.1	93.5	70.4	34.7	90.1	39.2	57.5	55.4	43.9	54.6	62.5
DeepLab [3]	2	97.3	77.7	87.7	43.6	40.5	29.7	44.5	55.4	89.4	67.0	92.7	71.2	49.4	91.4	48.7	56.7	49.1	47.9	58.6	63.1
FCN [25]	no	97.4	78.4	89.2	34.9	44.2	47.4	60.1	65	91.4	69.3	93.9	77.1	51.4	92.6	35.3	48.6	46.5	51.6	66.8	65.3
DPN [22]	no	97.5	78.5	89.5	40.4	45.9	51.1	56.8	65.3	91.5	69.4	**94.5**	77.5	54.2	92.5	44.5	53.4	49.9	52.1	64.8	66.8
Dilation10 [39]	no	97.6	79.2	89.9	37.3	47.6	53.2	58.6	65.2	91.8	69.4	93.7	78.9	55	93.3	45.5	53.4	47.7	52.2	66	67.1
DeepLab-v2 [4]	no	97.8	81.3	90.3	48.7	47.3	49.5	57.8	67.2	91.8	69.4	94.1	79.8	59.8	93.7	56.5	67.4	**57.4**	57.6	68.8	70.4
Adelaide [19]	no	**98.0**	82.6	90.6	44.0	50.7	51.1	**65.0**	71.7	**92.0**	**72.0**	94.1	**81.5**	**61.1**	**94.3**	61.1	65.1	53.8	**61.6**	70.6	**71.6**
LC	no	97.9	**83.1**	**91.6**	**53.7**	**57.4**	**58.4**	62.0	**73.3**	91.9	61.3	93.8	78.8	53.1	93.4	**62.2**	**76.9**	53.5	57.0	**74.7**	71.1

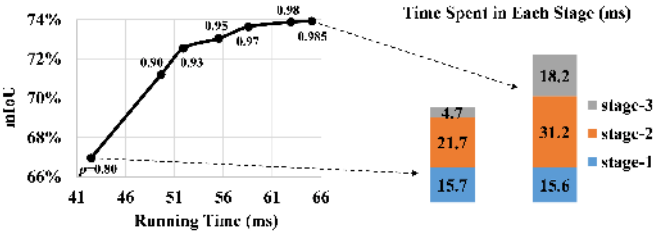


Figure 7: (a) shows the performance and speed trade-off in Layer Cascade (LC) by adjusting ρ. (b) is the time spent in each stage.

on VOC12 *test set*. The approaches pre-trained on COCO [20] are marked with †. LC achieves a mIoU of 80.3 and further improves the mIoU to 82.7 with pre-training on COCO, which is the best-performing method on VOC12 benchmark. By inspecting closer, we observe that LC wins 16 out of 20 foreground classes. For other 4 classes, LC also achieves competitive performance. Large gain is observed in some particular classes such as "bike", "chair", "plant", and "sofa". Based on our statistics in Fig. 5, we found that these few classes, in general, require a deeper stage to make decisions on hard regions.

Cityscapes. Next, we evaluate LC on Cityscapes benchmark, with results summarized in Table 5. "sub" denotes whether the method used subsampling images for training. LC also achieves promising performance with a mIoU of 71.1, which shows its great generalization ability to diverse objects and scenes. Lin *et al*. [19]'s performance is slightly better than ours, however, LC still wins on 9 out of 19 classes. It is noticed that [19] used a deeper backbone-network and explored richer contextual information. We believe that further performance gain can be achieved if LC is incorporated with these techniques. LC gains outstanding performance on the classes that are 'traditionally regarded' as hard classes, *e.g*., "fence", "pole", "sign", "truck", "bus" and "bike", which usually exhibit flexible shapes and fine-grained details. The results suggest that the end-to-end cascading mechanism in LC is meaningful, especially in alleviating the burden of deeper layers on analyzing easy regions but focusing themselves on hard regions adaptively.

5. Conclusion

Deep layer cascade (LC) is proposed in this work to simultaneously improve the accuracy and speed of semantic image segmentation. It has three advantages over previous approaches. First, LC adopts a "difficulty-aware" learning paradigm, where earlier stages are trained to handle easy and confident regions and hard regions are progressively forwarded to later stages. Secondly, since each stage only processes part of the input, LC can accelerate both training and testing by the usage of region convolution. Thirdly, LC is an end-to-end trainable framework that jointly optimizes the feature learning for different regions, thus achieving state-of-the-art performance on both PASCAL VOC and Cityscapes datasets. LC is capable of running in real-time yet still yielding competitive accuracies.

Acknowledgement. This work is supported by SenseTime Group Limited, the Hong Kong Innovation and Technology Support Programme, the General Research Fund sponsored by the Research Grants Council of the Hong Kong SAR (CUHK 416713, 14241716, 14224316), and the National Natural Science Foundation of China (61503366, 91320101, 61472410; Corresponding author is Ping Luo).

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015. 2, 7

[2] Z. Cai, M. Saberian, and N. Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *ICCV*, pages 3361–3369, 2015. 2

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 2, 3, 8

[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, 7, 8

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 1, 5, 7

[6] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. *arXiv:1503.01640v2*, 2015. 8

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 2

[8] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 1, 5, 7

[9] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *PAMI*, 35(8):1915–1929, 2013. 1

[10] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361, 2012. 1

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014. 5

[12] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, pages 991–998, 2011. 5

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 1, 4

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[15] A. Kae, K. Sohn, H. Lee, and E. Learned-Miller. Augmenting crfs with boltzmann machine shape priors for image labeling. In *CVPR*, pages 2019–2026, 2013. 2

[16] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *NIPS*, 2011. 2, 7

[17] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, volume 2, page 6, 2015. 2, 4, 5, 6

[18] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *CVPR*, pages 5325–5334, 2015. 1, 2, 3

[19] G. Lin, C. Shen, I. Reid, and A. Hengel. Efficient piecewise training of deep structured models for semantic segmentation. *arXiv:1504.01013v2*, 23 Apr 2015. 5, 8

[20] 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 *ECCV*, pages 740–755. 2014. 8

[21] B. Liu and X. He. Learning dynamic hierarchical models for anytime scene labeling. In *ECCV*, pages 650–666. Springer, 2016. 2

[22] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *ICCV*, pages 1377–1385, 2015. 2, 8

[23] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Deep learning markov random field for semantic segmentation. *arXiv preprint arXiv:1606.07230*, 2016. 2

[24] Z. Liu, S. Yan, P. Luo, X. Wang, and X. Tang. Fashion landmark detection in the wild. In *ECCV*, pages 229–245, 2016. 2

[25] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 2, 6, 8

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

[27] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016. 2

[28] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, pages 761–769, 2016. 5

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

[30] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014. 2, 5, 6

[31] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *CVPR*, pages 3476–3483, 2013. 3

[32] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016. 1, 2, 3, 6

[33] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, pages 1653–1660, 2014. 2, 3

[34] V. Vineet, J. Warrell, and P. H. Torr. Filter-based mean-field inference for random fields with higher-order terms and product label-spaces. In *ECCV*, pages 31–44. 2012. 2

[35] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages I–511, 2001. 1

[36] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Val Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 1

[37] Z. Wu, C. Shen, and A. v. d. Hengel. High-performance semantic segmentation using very deep fully convolutional networks. *arXiv preprint arXiv:1604.04339*, 2016. 8

[38] J. Yang, B. Price, S. Cohen, and M.-H. Yang. Context driven scene parsing with attention to rare classes. In *CVPR*, pages 3294–3301, 2014. 2

[39] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 8

[40] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. *arXiv:1502.03240v2*, 30 Apr 2015. 2, 8