Pattern Recognition 90 (2019) 119–133

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Wider or Deeper: Revisiting the ResNet Model for Visual Recognition

Zifeng Wu*, Chunhua Shen, Anton van den Hengel

University of Adelaide, Adelaide, SA, 5005, Australia

ARTICLE INFO

Article history:
Received 24 June 2018
Revised 4 November 2018
Accepted 4 January 2019
Available online 6 January 2019

Keywords:
Image classification
Semantic segmentation
Residual network

ABSTRACT

The community has been going deeper and deeper in designing one cutting edge network after another, yet some works are there suggesting that we may have gone too far in this dimension. Some researchers unravelled a residual network into an exponentially wider one, and assorted the success of residual networks to fusing a large amount of relatively shallow models. Since some of their early claims are still not settled, we in this paper dig more on this topic, i.e., the unravelled view of residual networks. Based on that, we try to find a good compromise between the depth and width. Afterwards, we walk through a typical pipeline of developing a deep-learning-based algorithm. We start from a group of relatively shallow networks, which perform as well or even better than the current (much deeper) state-of-the-art models on the ImageNet classification dataset. Then, we initialize fully convolutional networks (FCNs) using our pre-trained models, and tune them for semantic image segmentation. Results show that the proposed networks, as pre-trained features, can boost existing methods a lot. Even without exhausting the sophistical techniques to improve the classic FCN model, we achieve comparable results with the best performers on four widely-used datasets, i.e., Cityscapes, PASCAL VOC, ADE20k and PASCAL-Context. The code and pre-trained models are released for public access[1].

1. Introduction

The convolutional network has recently become the *de facto* standard model in various visual-based algorithms [1–3]. Convolutional networks used by the computer vision community have been growing deeper and deeper each year since Krizhevsky et al. [4] proposed AlexNet in 2012. Here, depth refers to the number of layers in a network, while width refers to the number of kernels per layer. The deepest network [5] in the literature is a residual network (ResNet) with 1,202 trainable layers. This 1,202-layer ResNet was trained using the tiny images in the CIFAR-10 dataset [6]. The image size here is important, because it means that the size of corresponding feature maps is relatively small, which is critical in practice to train extremely deep models. Most networks operating on more practically interesting image sizes tend to have the order of one, to two, hundred layers, e.g., the 200-layer ResNet [7] and 96-layer Inception-ResNet [8]. The progression to deeper networks continues, with Zhao et al. [9] having trained a 269-layer network for semantic image segmentation. These networks were trained using the ImageNet classification dataset [10], where the images are of much higher resolution. Each additional layer requires extra GPU memory, as the number of feature maps increases. Nevertheless, the marginal gains achieved by each additional layer diminish with depth, to the point where Zhao et al. [9] achieved only an improvement of 1.1% (from 42.2% to 43.3% by mean intersection-over-union scores) after almost doubling the number of layers (from 152 to 269). On the other hand, Zagoruyko and Komodakis showed that it is possible to train much shallower but wider networks on CIFAR-10, which outperform a ResNet[5] with its more than one thousand layers. The question thus naturally arises as to whether deep, or wide, is the right strategy.

In order to examine the issue we first need to understand the mechanism behind ResNets. Veit et al. [11] once claimed that ResNets actually behave as exponential ensembles of relatively shallow networks. However, there is a gap between their proposed unravelled view of a ResNet, and a real exponential ensemble of sub-networks, as illustrated in the top row of Fig. 1. Since the residual units are non-linear, we cannot further split the bottom path, i.e., M_e, into two sub-networks, i.e., M_c and M_d. It turns out that ResNets are only assembling linearly growing numbers of sub-networks. Besides, the key difference of our introduced views is that it depends on the effective depth l of a network. This l amounts to the number of residual units through which backward gradients can go during training. When $l \geq 2$, the two-unit ResNet in Fig. 1 can be seen as an ensemble of three sub-networks, i.e.,

* Corresponding author.
E-mail addresses: Zifeng.Wu@adelaide.edu.au, zifeng.wu@adelaide.edu.au (Z. Wu), Chunhua.Shen@adelaide.edu.au (C. Shen), anton.vandenhengel@adelaide.edu.au (A. van den Hengel).

[1] https://github.com/itijyou/ademxapp

https://doi.org/10.1016/j.patcog.2019.01.006
0031-3203/

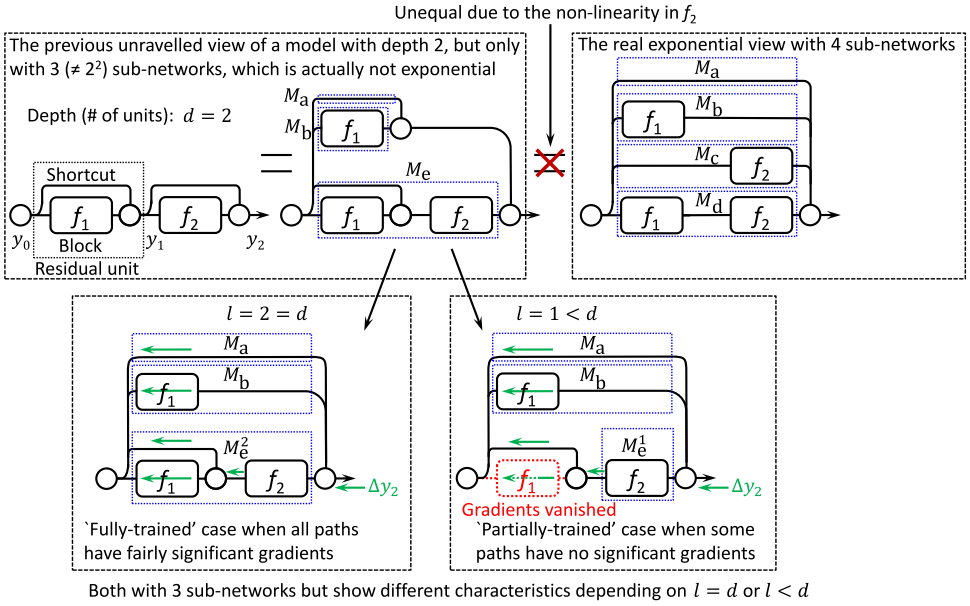

Fig. 1. Various unravelled views of a simple ResNet with only two units. The fact that $f_2(\cdot)$ is non-linear gives rise to the inequality in the top row, as $f_2(a+b) \neq f_(a) + f_2(b)$. This shows that $f_2(\cdot)$ never operates independently of the result of $f_1(\cdot)$, and thus that the number of independent sub-networks increases linearly (instead of exponentially) with the number of residual units. Here, there are three (instead of four) sub-networks, i.e., M_a, M_b and M_e^l, where l is the effective depth of the ResNet. Different values of l lead to the two different unravelled views in the bottom row.

M_a, M_b, and M_e^2, as shown in the bottom-left part. When $l = 1$, nothing changes except that we replace the third sub-network with a shallower one M_e^1, as shown in the bottom-right example. The superscripts in M_e^1 and M_e^2 denote their actual depths. About the unravelled view, the effective depth of a ResNet, and the actual depth of a sub-network, more details will be provided in the sequence. It is also worth noting that Veit et al. [11] empirically found that most gradients in a 110-layer ResNet can only go through up to seventeen residual units, which supports our above hypothesis that the effective depth l exists for a specific network.

In this paper, our contributions include:

1. We further develop the unravelled view of ResNets, which helps us better understand their behaviours. We demonstrate this in the context of a training process, which is the key difference from the original version [11].
2. We propose a group of relatively shallow convolutional networks based on our new understanding. Some of them perform comparably with the state-of-the-art approaches on the ImageNet classification dataset [10].
3. We evaluate the impact of using different networks on the performance of semantic image segmentation, and show these networks, as pre-trained features, can boost existing algorithms a lot.

2. Related work

Our work is closely related to residual network (ResNet) based image classification and fully convolutional network (FCN) based semantic image segmentation.

He et al. [5,7] recently proposed the ResNet to combat the vanishing gradient problem during training very deep convolutional networks. ResNets have outperformed previous models at a variety of tasks, such as object detection [12] and semantic image segmentation [13]. They have gradually replaced VGGNets [14] in the computer vision community, as the standard feature extractors. Nevertheless, the real mechanism underpinning the effectiveness of ResNets is not yet clear. Veit et al. [11] once claimed that they behave like exponential ensembles of relatively shallow networks, but the 'exponential' nature of the ensembles has yet to be theoretically verified. Residual units are usually non-linear, which prevents a ResNet from exponentially expanding into separated sub-networks, as illustrated in Fig. 1. It is also unclear as to whether a residual structure is required to train very deep networks. For example, Szegedy et al. [8] showed that it is 'not very difficult' to train competitively deep networks, even without residual shortcuts. Currently, the most clear advantage of ResNets is in their fast convergence [5]. Szegedy et al. [8] observed similar empirical results to support that. On the other hand, Zagoruyko and Komodakis [15] found that a wide sixteen-layer ResNet outperformed the original thin thousand-layer ResNet [7] on datasets composed of tiny images such as CIFAR-10 [6]. The analysis we present here is motivated by their empirical testing and the observation that a grid search of configuration space is sometimes intractable on large scale datasets such as the ImageNet classification dataset [10].

Semantic image segmentation amounts to predicting the categories for each pixel in an image. Long et al. [16] proposed FCNs for this purpose. And they soon became the mainstream approach to dense prediction based tasks, especially due to its efficiency. Besides, empirical results in the literature [13] showed that stronger pre-trained features can yet further improve their performances. We thus here base our semantic image segmentation approach on FCNs, and will show the impact of different pre-trained features (as backbone networks) on final segmentation results.

3. Residual networks reviewed

3.1. Definition of residual networks

We are concerned here with the full pre-activation version of residual networks (ResNets) [7]. For shortcut connections, we consider identity mappings [7] only. We omit the raw input and the top-most linear classifier for clarity. Usually, there may be a stem block [8] or several traditional convolution layers [5,7] following the raw input. We omit these too, again, for the purpose of clarity.

For the residual Unit i, let y_{i-1} be the input, and let $f_i(\cdot)$ be its trainable non-linear mappings, also named Block i. The output of

Unit i is recursively defined as:

$$y_i = f_i(y_{i-1}, \boldsymbol{w}_i) + y_{i-1}, \tag{1}$$

where $\boldsymbol{w}_i$ denotes the trainable parameters, and $f_i(\cdot)$ is often two or three stacked convolution steps. In the full pre-activation version, the components of a convolution step are in turn a batch normalization [17], a rectified linear unit [18] (ReLU) non-linearity, and a convolution layer.

3.2. Residual networks unravelled

Applying Eq. (1) in one substitution step, we expand the forward pass into:

$$y_2 = y_1 + f_2(y_1, \boldsymbol{w}_2) \tag{2}$$

$$= y_0 + f_1(y_0, \boldsymbol{w}_1) + f_2(y_0 + f_1(y_0, \boldsymbol{w}_1), \boldsymbol{w}_2) \tag{3}$$

$$\neq y_0 + f_1(y_0, \boldsymbol{w}_1) + f_2(y_0, \boldsymbol{w}_2) + f_2(f_1(y_0, \boldsymbol{w}_1), \boldsymbol{w}_2), \tag{4}$$

which describes the unravelled view proposed by Veit et al. [11], also as shown in the top row of Fig. 1. They further empirically showed that the paths which gradients take through a ResNet are typically far shorter than the total depth of that network, and thus introduced the idea of 'effective depth' as a measure for the true lengths of these paths. Based on these two observations, they once claimed that ResNets are exponential ensembles of relatively shallow networks, although the 'exponential' nature of the ensembles has yet to be theoretically verified. And aside from that, they have neither covered how the flows of gradients look like during training in these ResNets, nor the consequences of these behaviours.

We show that the exponential nature is not theoretically supported with Eqs. (3) and (4) and the top-row of Fig. 1. Expanding ResNets exponentially requires the mappings in all residual blocks to be linear, which is not true in convolutional neural networks (CNNs), and so for ResNets. Since $f_2(\cdot)$ is non-linear, we cannot derive Eq. (4) from Eq. (3). So the whole network is not equivalent to an exponentially growing ensemble of sub-networks. It is rather, more accurately, described as a linearly growing ensemble of sub-networks. For the two-unit ResNet as illustrated in the top-left part of Fig. 1, there are three sub-networks, i.e.,M_a, M_b, and M_e, respectively corresponding to the three terms in Eq. (3), i.e., y_0, $f_1(y_0, \boldsymbol{w}_1)$, and $f_2(y_0 + f_1(y_0, \boldsymbol{w}_1), \boldsymbol{w}_2)$. However, as shown in the top-right part of Fig. 1, M_e cannot further be unravelled into M_c and M_d, respectively corresponding to $f_2(y_0, \boldsymbol{w}_2)$ and $f_2(f_1(y_0, \boldsymbol{w}_1), \boldsymbol{w}_2)$ in Eq. (4).

3.3. Residual networks unravelled during training

Gradients only take relative short paths in ResNets [11], which is the key to analyse the behaviour of ResNets. Therefore, it is also important to show this effect in the unravelled view. By characterising the units of a two-unit ResNet given its effective depth l, we illuminate the impact of various paths in the network, as shown in the bottom row of Fig. 1. Let d be the number of residual units. When $l = d$, the gradients can go through all the paths, and when $l < d$, the gradients only take the relatively short paths. We here illustrate this impact in terms of small effective depths, because to do so for larger ones would require diagrams of enormous networks. Nevertheless, the impact is as the same. For example, we further show a three-unit ResNet in Fig. 2, which can be seen as composed of four sub-networks.

Take the ResNet in Fig. 1 for example again. In an SGD iteration, the backward gradients are:

$$\Delta \boldsymbol{w}_2 = \frac{\mathrm{d}f_2}{\mathrm{d}\boldsymbol{w}_2} \cdot \Delta y_2 \tag{5}$$

$$\Delta y_1 = \Delta y_2 + f_2' \cdot \Delta y_2 \tag{6}$$

$$\Delta \boldsymbol{w}_1 = \frac{\mathrm{d}f_1}{\mathrm{d}\boldsymbol{w}_1} \cdot \Delta y_2 + \frac{\mathrm{d}f_1}{\mathrm{d}\boldsymbol{w}_1} \cdot f_2' \cdot \Delta y_2, \tag{7}$$

where f_2' denotes the derivative of $f_2(\cdot)$ to its input y_1. When effective depth $l \geq 2$, both terms in Eq. (7) are non-zeros, which corresponds to the bottom-left case in Fig. 1. Namely, Block 1 receives gradients from both M_b and M_e^2. However, when effective depth $l = 1$, the gradient Δy_2 vanishes after passing through Block 2. Namely, $f_2' \cdot \Delta y_2 \to 0$. So, the second term in Eq. (7) also goes to zeros, which is illustrated by the bottom-right case in Fig. 1. The weights in Block 1 indeed vary across different SGD iterations. However, they are updated only in M_b, but not in M_e^1, since no gradients can go through Block 2 and reach Block 1. As a result, for M_e^1, Block 1 is not trainable and more like an extra preprocessing operation to enhance the input features. In this case, we say that the ResNet is only 'partially-trained', even with those shortcut connections. Otherwise, when all paths contribute gradients in the training process, we say that the ResNet is 'fully-trained', as shown by the bottom-left case in Fig. 1.

Generally, given a ResNet, letting d be the total number of residual units, we can see this ResNet as an ensemble of sub-networks, i.e., $M_i, i = \{0, 1, \cdots, d\}$. The effective depth of M_i is $\min(i, l)$, where l is the effective depth of the original ResNet. We further show an unravelled three-unit ResNet with different effective depths in Fig. 2. By way of example, M_0 has no residual block inside, which directly passes its input to the output, M_1 contains only Block 1, whereas M_2 contains both Block 1 and Block 2. Among the three cases illustrated, the top-right one is the 'fully-trained' case. Beside that, the bottom-right one shows an extreme example, where $l = 1$. To simulate this situation, consider the residual units are relatively deep themselves, e.g., each with ten convolution layers instead of two. As a results, it is not possible for the gradients to go through more than a single unit. And the whole 30-layer model can be seen as three 10-layer models (M_1, M_2 and M_3) fused with their outputs added into the original input (M_0). The difference between M_1 and M_2 is that the latter's input is preprocessed by the parameters borrowed from M_1. Similarly, the input of M_3 is preprocessed by the parameters borrowed from M_1 and M_2. On the other hand, the bottom-left one is more complicated, where $l = 2$. For the sub-network M_3^2, the gradient of Block 1 is $f_3' \cdot \Delta y_3 + f_2' \cdot f_3' \cdot \Delta y_3$, where the first term is non-zero. M_3^2 will thus update Block 1 at each iteration. Considering the non-linearity in Block 3, we cannot trivially split M_3^2 into even smaller sub-networks. Thus, it becomes even harder to tell if this is as good as the 'fully-trained' case (M_3 in the top right of Fig. 2). An investigation of this issue remains future work.

3.4. Wider or deeper?

To summarize the previous subsections, shortcut connections enable us to train deeper networks. As the depth growing to some point, we will face the dilemma between width and depth. From that point, going deeper, we will get an actually wider (but not deeper) network, with additional preprocessed features at its bottom which are not 'fully-trained'; going wider, we will get an indeed wider network. We have learned the strength of depth from the previous vanilla deep networks without any shortcuts, e.g., the AlexNet [4] and VGGNets [14]. However, it is not clear whether those additional features in very deep ResNets can perform as well as conventional 'fully-trained' features.

In practice, algorithms are often limited by their spatial costs (GPU memory usages). One possible solution is to use more

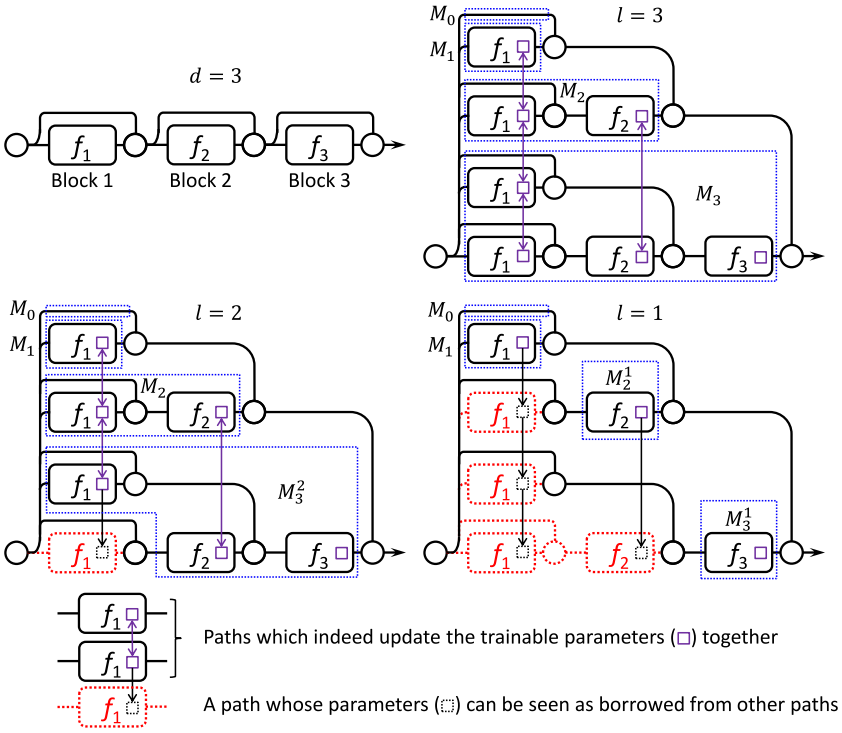

Fig. 2. Various unravelled views of a three-unit ResNet given different effective depth *l*. Top-right: The 'fully-trained' case, where parameters in all blocks are updated by gradients from all related paths. Bottom-right: The extreme 'partially-trained' case, where parameters in each block are only updated by gradients from a single path. Bottom-left: The intermediate case, where parameters in Block 1 are not updated by gradients from all of its related paths, but those in Blocks 2 and 3 are.

devices, which will however increase data transmission costs. For example, the winning algorithm (MegDet [19]) in the COCO 2017 detection competition [20] used a mini-batch size of 256, instead of the 2 used in a classic Faster R-CNN [21]. To this end, the authors had to implement a synchronized version of batch normalization [17], and largely increased the data transmission cost by using multiple (up to 128) GPU devices. Thus, spatial compactness may be preferable in practice, especially when temporal and spatial limits come up. On the other hand, with similar memory costs, shallower but wider networks can have times more numbers of trainable parameters. And increasing the depth (number of layers) and size (number of parameters) are together among the most reliable ways to improve the capacity of a model.

Therefore, given the following observations in the literature,

- Zagoruyko and Komodakis [15] found that the performance of a ResNet was related to the number of trainable parameters. Szegedy et al. [8] came to a similar conclusion, according to the comparison between their proposed various Inception networks.
- Veit et al. [11] found that there was a relatively small effective depth for a very deep ResNet, e.g., seventeen residual units for a 110-layer ResNet.

many of the current state-of-the-art models on the ImageNet classification dataset [10] seem over-deepened, e.g., the 200-layer ResNet [7] and 96-layer Inception-ResNet [8]. The reason is that we should make a model relatively shallow to effectively utilize GPU memory. As demonstrated previously, paths longer than the effective depth in ResNets are not 'fully-trained'. It is doubtful for these paths to contribute much for the whole models. Thus, we choose to remove most of these paths by directly reducing the number of residual units. For example, in our best performing network, there are exactly seventeen residual units.

With empirical results, we will show the advantages of our designed relatively shallow networks against the previous much deeper ResNets. They perform better even with less GPU memory usages, especially as feature extractors. However, even if a rather shallow network (eight-unit, or twenty-layer) can outperform ResNet-152 on the ImageNet classification dataset, we will not go that shallow, because an appropriate depth is still vital to train good features.

4. Approach to image classification

We show the proposed networks in Fig. 3. There are three architectures, with different input sizes. Dashed blue rectangles to denote convolution steps, which are respectively composed of a batch normalization, an ReLU non-linearity and a convolution layer, following the second version of ResNets [7]. The closely stacked two or three convolution steps denote different kinds of residual units (B1–B7), with inner shortcut connections [7]. Each kind corresponds to a level, where all units share the same kernel sizes and numbers of channels, as given in the dashed black rectangles in the left-most column of Fig. 3. As mentioned before, there are two 3×3 convolution layers in most residual units (B1–B5). However, in B6 and B7, we use bottleneck structures as in ResNets [5], except that we adjust the numbers of channels to avoid drastic changes in width. Each of our networks usually consists of one B6, one B7, and different numbers of B1–B5. For those with a 224×224 input, we do not use B1 due to limited GPU memory. Each of the green triangles denotes a down-sampling operation with a rate of two, which is clear given the feature map sizes of different convolution steps (in dashed blue rectangles). To this end, we can let the first convolution layer at according levels have a stride of two. Or, we can use an extra spatial pooling layer, whose kernel size is three and stride is two. In a network whose classification results are reported in this paper, we always use pooling layers for down-sampling. We average the top-most feature maps into 4,096-dimensional final features, which matches the cases of AlexNet [4] and VGGNets [14]. We will show more details about network structures in Section 6.1.

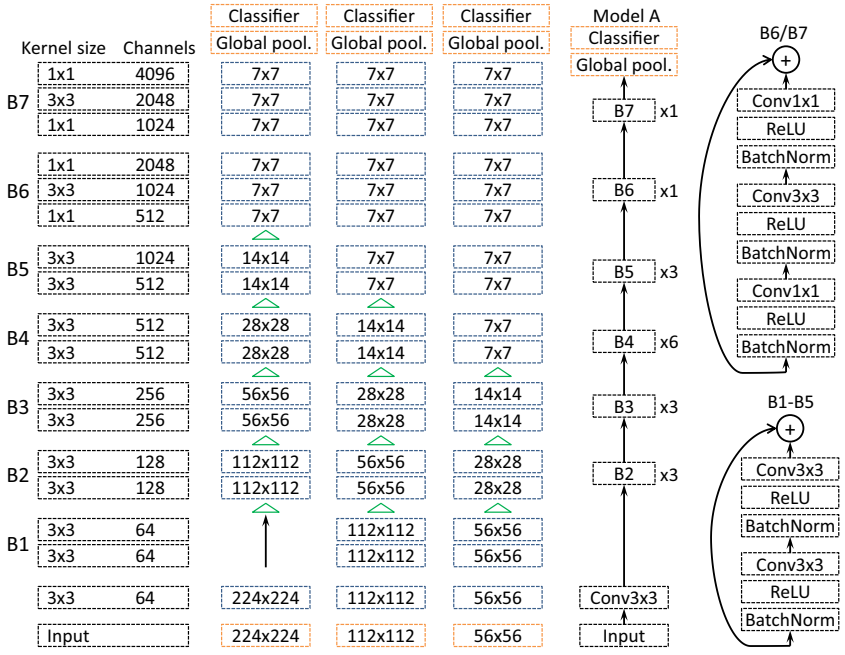

Fig. 3. Overview of our proposed networks with different input sizes. B1–B7 are respectively a residual unit, and thus have skip connections, as shown in the right-most column. Networks consist of different numbers of these blocks, e.g., Model A as shown in the middle.

Table 1
Comparison of networks by top-1 and top-5 errors (%) on the ILSVRC 2012 val set [10] with 50k images, using a single crop. Inference speeds (images/second) are evaluated with 10 images/mini-batch using CUDA 8.0 and cuDNN 5.1 on a GTX 980 card. Input sizes during training are also listed, and a smaller size often leads to faster training.

method	depth	tr. input	top-1	top-5	speed
VGG16 [14], 10 crops	16	224	28.1	9.3	9.21
ResNet-50 [5], our tested	50	224	23.5	6.8	89.0
ResNet-101 [5], our tested	101	224	22.1	6.1	73.2
ResNet-152 [5], our tested	152	224	21.8	5.8	53.8
ResNet-152 [5]	152	224	21.3	5.5	–
ResNet-152 [7], pre-act.	152	224	21.1	5.5	–
ResNet-200 [7], pre-act.	200	224	20.7	5.3	–
Inception-v4 [8]	76	299	20.0	5.0	80.0
Inception-ResNet-v2 [8]	96	299	19.9	4.9	56.0
56-1-1-1-1-9-1-1, Model F	34	56	25.2	7.8	83.5
112-1-1-1-1-5-1-1, Model E	26	112	22.3	6.2	92.3
112-1-1-1-1-9-1-1, Model D	34	112	22.1	6.0	70.6
112-1-1-1-1-13-1-1, Model C	42	112	21.8	5.9	57.1
224-0-1-1-1-1-1-1	16	224	22.0	5.8	73.4
224-0-1-1-1-3-1-1, Model B	20	224	21.0	5.5	62.1
224-0-3-3-6-3-1-1, Model A	38	224	**19.2**	**4.7**	21.7

Implementation details. We run all experiments using the MXNet framework [22], with four devices (two K80 or four Maxwell Titan X cards) on a single node. We follow settings in the re-implementation of ResNets by Gross and Wilber [23] as possible. But, we use a linear learning rate schedule, as suggested by Mishkin et al. [24]. Take Model A in Table 1 for example. We start from 0.1, and linearly reduce the learning rate to 10^{-6} within 450k iterations.

5. Approach to semantic image segmentation

Our approach is similar to the fully convolutional networks (FCN) [16] implemented in the first version of DeepLab [25]. However, without getting too many factors entangled, we generally do not introduce any multi-scale structures [13], deep supervision signals [9,16], or global context features [9]. But note that there are exceptions for part of the results reported on test sets.

Given a pre-trained network, there are three steps to reshape it into a network suitable for semantic image segmentation, as stated below.

1) **Resolution**. To generate score maps at 1/8 resolution, we remove down-sampling operations and increase dilation rates accordingly in some convolution layers. For clarity, first suppose that we always down-sample features maps using a convolution layer with a stride of two. Take networks with 224×224 inputs for example. We set stride of the first convolution layer in B5 to one, and increase the dilation rate from one to two for the following layers; We do the same thing to the first convolution layer in B6 too, and increase the dilation rate from two to four for the following layers. In the case of down-sampling using a pooling layer, everything is the same except that we set stride of that pooling layer to one. Sometimes, we will have to apply a pooling layer with dilation [26]. However, we do not make any change for networks with 56×56 inputs, since there are only three down-sampling operations in each of them.

It is notable that all down-sampling operations are implemented using spatial max pooling layers in our pre-trained networks. We find it harmful for FCNs in preliminary experiments, probably due to too strong spatial invariance. We thus remove some of the pooling layers in these networks, and tune them further. Take Model A in Table 1 for example. We remove all the pooling layers (before B2, B3, B4, B5 and B6), increase the strides of according convolution layers up to two, and tune it for 45k iterations using the ImageNet dataset [10], starting from a learning rate of 0.01. This model achieves an intersection-over-union score of 43.09% on the ADE20K val set, while the one always uses pooling layers for down-sampling only achieves 41.24%.

2) **Classifier**. We remove the top-most linear classifier and the global pooling layer from the pre-trained networks, to reshape them into pre-trained features. For semantic segmentation, we should at least add back one convolution layer (as the new linear classifier), whose channel number is the same as the number of pixel categories, e.g., 21 in the case of PASCAL VOC 2012 [27].

Table 2
Comparison of semantic segmentation scores, i.e., pixel accuracies (%), class-wise mean accuracies (%), class-wise mean intersection-over-union (IoU) scores (%), and GPU memory usages (GB/device) during fine-tuning on the PASCAL VOC val set [27]. Memory usages are obtained with four images/device using MXNet [22], whose version may lead to variations in these usage numbers.

method	pixel acc.	mean acc.	mean IoU	memory usage
VGG16, 1 conv.	92.84	77.52	68.33	3.41
ResNet-101, 1 conv.	94.52	82.17	75.35	12.3
ResNet-152, 1 conv.	94.56	82.12	75.37	16.7
Model F, 1 conv.	92.91	76.26	68.36	11.7
Model E, 1 conv.	94.13	80.84	73.64	10.1
Model D, 1 conv.	94.59	82.50	75.66	11.7
Model C, 1 conv.	94.56	82.01	75.45	13.4
Model B, 1 conv.	93.94	80.96	73.37	7.6
Model A, 1 conv.	**95.28**	**85.23**	**78.76**	11.0
Model A2, 1 conv.	95.16	85.72	78.14	11.0
Model A, 2 conv.	**95.70**	**86.26**	**80.84**	11.8

We name this kind of models as '1 convolution' in Tables 2 and 4. Moreover, we find that an additional 512-channel convolution layer (between the features and the new classifier) with non-linearity can improve the performance much. We name this kind of models as '2 convolution' in Tables 2 and 4. All the newly added layers have 3×3 kernels and a dilation rate of 12, following the large field-of-view setting in DeepLab-v2 [13]. In a '2 convolution' model, the top-most two new layers as a whole thus have a receptive field of 392×392, with respect to the pre-trained features.

3) **Dropout**. To alleviate over-fitting, we also apply the traditional dropout [28] to very wide residual units. The dropout rate is 0.3 for those with 2,048 channels, e.g., the last three units in ResNets and the second last units (B6) in our networks; while 0.5 for those with 4,096 channels, e.g., the top-most units (B7) in our networks.

Implementation details. We fix the moving means and variations in batch normalization layers during fine-tuning [5]. We use four GPU devices on a single node. The batch size is 16, so there are four examples per device. We first tune each network for a number of iterations, keeping the learning rate unchanged at 0.0016. And then, we reduce the learning rate gradually during another number of iterations, following a linear schedule [24]. For datasets with available testing sets, we evaluate these numbers of iterations on validation sets. During training, we first resize an image by a ratio randomly sampled from [0.7, 1.3], and then generate a sample by cropping one 500×500 sub-window at a randomly selected location. Besides, we also apply the online bootstrapping [29] technique in all experiments.

6. Experimental results

6.1. Image classification results

We evaluate our proposed networks on the ILSVRC 2012 classification dataset [10], with 1.28 million images for training, respectively belonging to 1,000 categories. We report top-1 and top-5 error rates on the validation set. We compare various networks in Table 1, where we obtain all the results by testing on a single crop. However, we list the ten-crop result for VGG16 [14] since it is not inherently a fully convolutional network. For networks trained with 224×224 inputs, the testing crop size is 320×320, following the setting used by He et al. [7]. For those with 112×112 and 56×56 inputs, we use 160×160 and 80×80 crops respectively. For Inception networks [8], the testing crop size is 299×299 [7]. The names of our proposed networks are composed of training crop sizes and the numbers of residual units on different levels. Take 56-1-1-1-1-9-1-1 for example. Its input size is 56, and there are only one unit on all levels except for Level 5 (B5 in Fig. 3).

Table 3
Comparison with previous results by mean IoU scores (%) on the PASCAL VOC test set [27]. Asterisked methods use ResNet-101 [5] as features, while others use VGG16 [14].

method	aero.	bicy.	bird	boat	bott.	bus	car	cat	chai.	cow	dini.	dog	hors.	moto.	pers.	pott.	shee.	sofa	trai.	tvmo.	mean
using augmented PASCAL VOC data only																					
FCN-8s [16]	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
CRFasRNN [33]	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
DeconvNet [34]	89.9	39.3	79.7	63.9	68.2	87.4	81.2	86.1	28.5	77.0	62.0	79.0	80.3	83.6	80.2	58.8	83.4	54.3	80.7	65.0	72.5
DPN [35]	87.7	59.4	78.4	64.9	70.3	89.3	83.5	86.1	31.7	79.9	62.6	81.9	80.0	83.5	82.3	60.5	83.2	53.4	77.9	65.0	74.1
Context [36]	90.6	37.6	80.0	67.8	74.4	92.0	85.2	86.2	39.1	81.2	58.9	83.8	83.9	84.3	84.8	62.1	83.2	58.2	80.8	72.3	75.3
VeryDeep* [29]	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
Ours	**94.4**	**72.9**	**94.9**	68.8	**78.4**	90.6	**90.0**	92.1	**40.1**	**90.4**	**71.7**	**89.9**	**93.7**	**91.0**	**89.1**	**71.3**	**90.7**	**61.3**	**87.7**	**78.1**	**82.5**
using augmented PASCAL VOC & COCO data																					
Context [36]	94.1	40.4	83.6	67.3	75.6	93.4	84.4	88.7	41.6	86.4	63.3	85.5	89.3	85.6	86.0	67.4	90.1	62.6	80.9	72.5	77.8
DeepLab-v2* [13]	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
PSPNet* [9]	95.8	72.7	95.0	**78.9**	84.4	94.7	**92.0**	**95.7**	43.1	91.0	**80.3**	91.3	96.3	**92.3**	90.1	71.5	**94.4**	66.9	88.8	**82.0**	85.4
DeepLab-v3* [32]	**96.4**	**76.6**	92.7	77.8	**87.6**	**96.7**	90.2	95.4	47.5	**93.4**	76.3	**91.4**	**97.2**	91.0	**92.1**	71.3	90.9	**68.9**	**90.8**	79.3	**85.7**
Ours	96.2	75.2	**95.4**	74.4	81.7	93.7	89.9	92.5	**48.2**	92.0	79.9	90.1	95.5	91.8	91.2	**73.0**	90.5	65.4	88.7	80.6	84.9

Table 4
Comparison of semantic segmentation scores (%) on the Cityscapes val set [37], and the ADE20K val set [38].

method	pixel acc.	mean acc.	mean IoU
results on the Cityscapes val set			
ResNet-101, 1 conv.	95.49	81.76	73.63
ResNet-152, 1 conv.	95.53	81.61	73.50
Model A, 1 conv.	95.80	83.81	76.57
Model A2, 1 conv.	**95.91**	**84.48**	**77.18**
Model A2, 2 conv.	**96.05**	**84.96**	**77.86**
results on the ADE20K val set			
ResNet-101, 2 conv.	79.07	48.73	39.40
ResNet-152, 2 conv.	79.33	49.55	39.77
Model E, 2 conv.	79.61	50.46	41.00
Model D, 2 conv.	79.87	51.34	41.91
Model C, 2 conv.	80.53	52.32	43.06
Model A, 2 conv.	80.41	52.86	42.71
Model A2, 2 conv.	**81.17**	**53.84**	**43.73**

Notable points about the results are as follows.

1) Relatively shallow networks can outperform very deep ones, which is probably due to large model capacity, coinciding with the results reported by Zagoruyko and Komodakis [15]. For example, the much shallower Model B achieves similar error rates as ResNet-152, and even runs slightly faster. And particularly, Model A performs the best among all the networks.

2) We can trade performance for efficiency by using a small input size. For example, Model D performs slightly worse than ResNet-152, but is almost two times faster. This may be useful when efficiency is strictly required. Mishkin et al. [24] also reduced the input size for efficiency. However, they did not remove downsampling operations accordingly to preserve the size of final feature maps, which resulted in much degraded performance.

3) Models C, D and E perform comparably, even though Model C has larger depth and more parameters. This comparison shows the importance of designing a network properly. In these models, we put too many layers on low resolution levels (7×7, B5 in Fig. 3).

6.2. Semantic image segmentation results

We evaluate our proposed networks on four widely used datasets. When available, we report, 1) the pixel accuracy, which is the percentage of correctly labelled pixels on a whole test set, 2) the mean pixel accuracy, which is the mean of class-wise pixel accuracies, and 3) the mean IoU score, which is the mean of class-wise intersection-over-union scores.

PASCAL VOC 2012 [27]. This dataset consists of daily life photos. There are 1,464 labelled images for training and another 1,449 for validation. Pixels either belong to the background or twenty object categories, including *bus, car, cat, sofa, monitor*, etc. Following the common criteria in the literature [13,16], we augment the dataset with extra labelled images from the semantic boundaries dataset [30]. So in total, there are 10,582 images for training.

We first compare different networks in Table 2. Notable points about the results are as follows.

1) We cannot make statistically significant improvements using ResNet-152 instead of ResNet-101. However, Model A performs better than ResNet-152 by 3.4%. Using one hidden layer leads to a further improvement of 2.1%.

2) The very deep ResNet-152 uses too much device memory due to intentionally enlarged depth. With our settings, it even cannot be tuned using many mainstream GPUs with up to 12GB memory.

3) Model B performs worse than ResNet-101, even if it is better on the classification task as shown in Table 1. This shows that it is not reliable to tell a good feature extractor only depending on its classification performance on ImageNet. And it again shows why we should still favour models that are deep enough.

4) Model A2 performs worse than Model A on this dataset. We initialize it using weights of Model A, and tune it with the Places 365 data [31] for 45k iterations. This is reasonable since there are only object categories in this dataset, while Places 365 is for scene classification tasks.

We then compare our method with previous ones on the test set in Table 3. Only using the augmented PASCAL VOC data for training, we achieve a mean IoU score of 82.5%, which is better than the previous best one by 3.4%. This is a significant margin, considering that the gap between ResNet-based and VGGNet-based methods is 3.8%. In some works [13], models were further pre-trained using the Microsoft COCO [20] data, which consists of 120k labelled images. In this case, the current best mean IoU score is 85.7% reported by Chen et al. [32]. On top of the classic dilated FCNs [25], they also introduced multi-scale structure, atrous spatial pyramid pooling, image-level representations, CRF-based post-processing and several other techniques to improve the performance, which we do not consider here. Nevertheless, our method performs almost comparably, which shows the effectiveness of our pre-trained features.

Cityscapes [37]. This dataset consists of street scene photos taken by car-carried cameras. There are 2975 labelled images for training and another 500 for validation. Besides, there is also an extended set with 19,998 coarsely labelled images. Pixels belong to nineteen semantic classes, including *road, car, pedestrian, bicycle*, etc. These further belong to seven categories, i.e., *flat, nature, object, sky, construction, human*, and *vehicle*.

We first compare different networks in Table 4. On this dataset, ResNet-152 again shows no advantage against ResNet-101. However, Model A1 outperforms ResNet-101 by 4.2% in terms of mean IoU scores, which again is a significant margin. Because there are many scene classes, models pre-trained using Places 365 [31] are supposed to perform better, which coincides with our results.

We then compare our method with previous ones on the test set in Table 5. The official criteria on this dataset includes two levels, i.e., *class* and *category*. Our method achieves a class-level IoU score of 81.3%, which is comparable with the current state-of-the-art in the literature. Again note that we achieve this score without exhaustively exploring the sophistical techniques in the previous approaches [32] except for combining the features extracted from two scales (at full and half resolutions), which shows the strength of our pre-trained features.

ADE20K [38]. This dataset consists of both indoor and outdoor images with large variations. There are 20,210 labelled images for training and another 2k for validation. Pixels belong to 150 semantic categories, including *sky, house, bottle, food, toy*, etc.

We first compare different networks in Table 4. On this dataset, ResNet-152 performs slightly better than ResNet-101. However, Model A2 outperforms ResNet-152 by 4.0% in terms of mean IoU scores. Being similar with Cityscapes, this dataset has many scene

Table 5
Comparison of semantic segmentation scores (%) on the Cityscapes test set [37].

method	feature	IoU class	IoU category
DPN [35]	VGG16	66.8	86.0
Dilation10 [39]	VGG16	67.1	86.5
Context [36]	VGG16	71.6	87.3
LRR [40]	ResNet-101	76.8	–
DeepLab-v2 [7]	ResNet-101	79.7	–
PSPNet [9]	ResNet-101	81.2	91.2
DeepLab-v3 [32]	ResNet-101	**81.3**	**91.6**
Ours		**81.3**	91.5

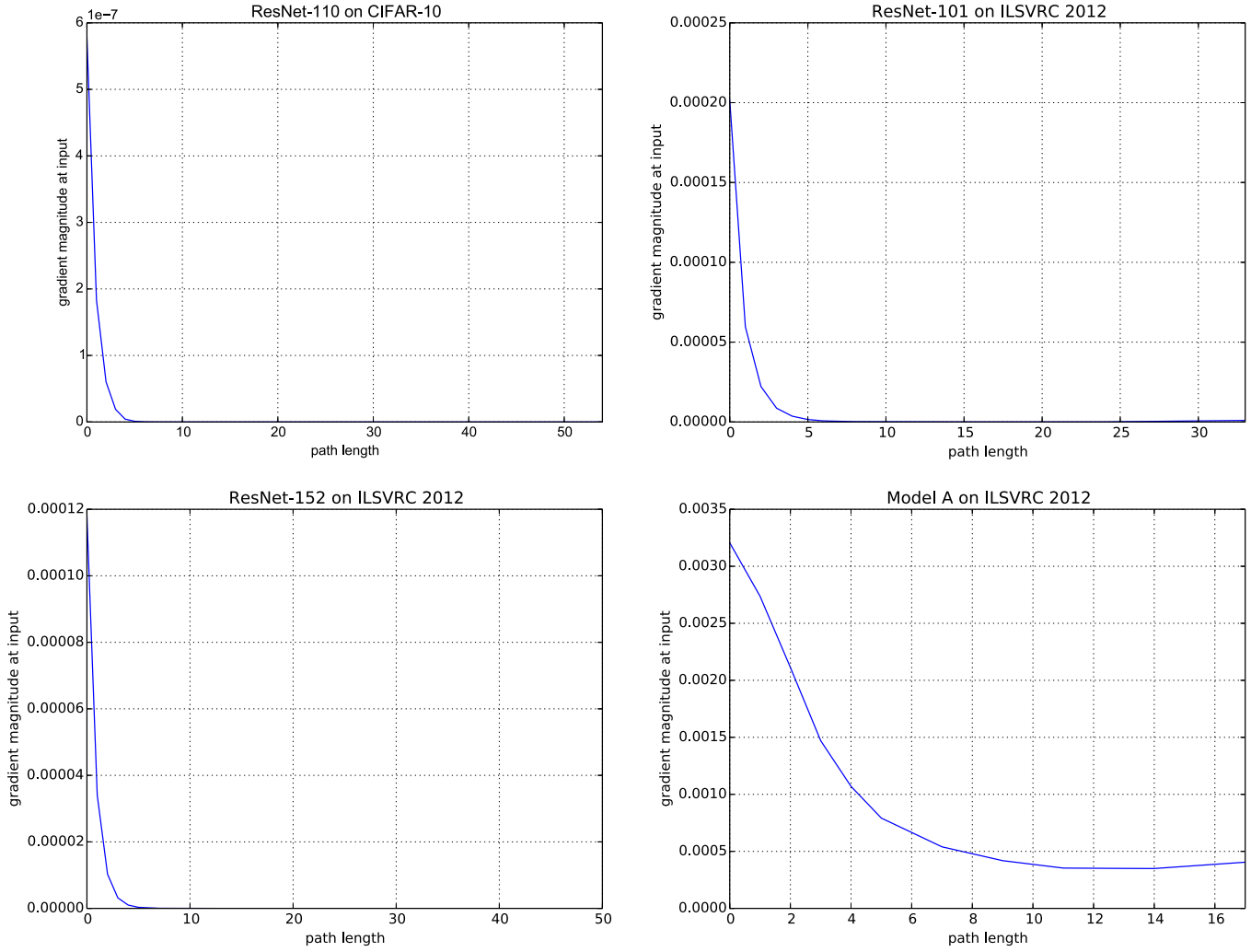

Fig. 4. Gradient magnitude at input given a path length k in various residual networks. See the text for details.

Table 6
Comparison of semantic segmentation scores (%) on the ADE20K test set [38].

method		ave. of pixel acc. & mean IoU
SegModel		53.23
CASIA_IVA		54.33
360+MCG-ICT-CAS_SP		54.68
SenseCUSceneParsing		55.38
Ours		**56.41**
method	models	ave. of pixel acc. & mean IoU
NTU-SP	2	53.57
SegModel	5	54.65
360+MCG-ICT-CAS_SP	–	55.57
SenseCUSceneParsing	–	**57.21**
Ours	2	56.74

categories. So, Model A2 performs slightly better than Model A. Another notable point is that, Model C takes the second place on this dataset, even if it performs worse than Model A in the image classification task on the ImageNet dataset. This shows that large model capacity may become more critical in complicated tasks, since there are more parameters in Model C.

We then compare our method with others on the test set in Table 6. The official criteria on this dataset is the average of pixel accuracies and mean IoU scores. For better performance, we apply multi-scale testing, model averaging and post-processing with CRFs. Our Model A2 performs the best among all methods using only a single pre-trained model. However, in this submission, we only managed to include two kinds of pre-trained features, i.e., Models A and C. Nevertheless, our method only performs slightly worse than the winner by a margin of 0.47%.

PASCAL-Context [41]. This dataset consists of images from PASCAL VOC 2010 [27] with extra object and stuff labels. There are 4,998 images for training and another 5,105 for validation. Pixels either belong to the background category or 59 semantic categories, including *bag, food, sign, ceiling, ground* and *snow*. All images in this dataset are no larger than 500×500. Since the test set is not available, here we directly apply the hyper-parameters which are used on the PASCAL VOC dataset. Our method again performs the best with a clear margin by all the three kinds of scores, as shown in Table 7. In particular, we improve the IoU score by 2.4% compared to the previous best method [13].

6.3. Gradients in residual networks

We show results of the experiment on gradients proposed by Veit et al. [11], with various residual networks. Namely, for a trained network with n units, we sample individual paths of a certain length k, and measure the norm of gradients that arrive at

Fig. 5. Qualitative results on the PASCAL VOC 2012 [27] val set. The model was trained using the train set augmented using SBD [30]. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

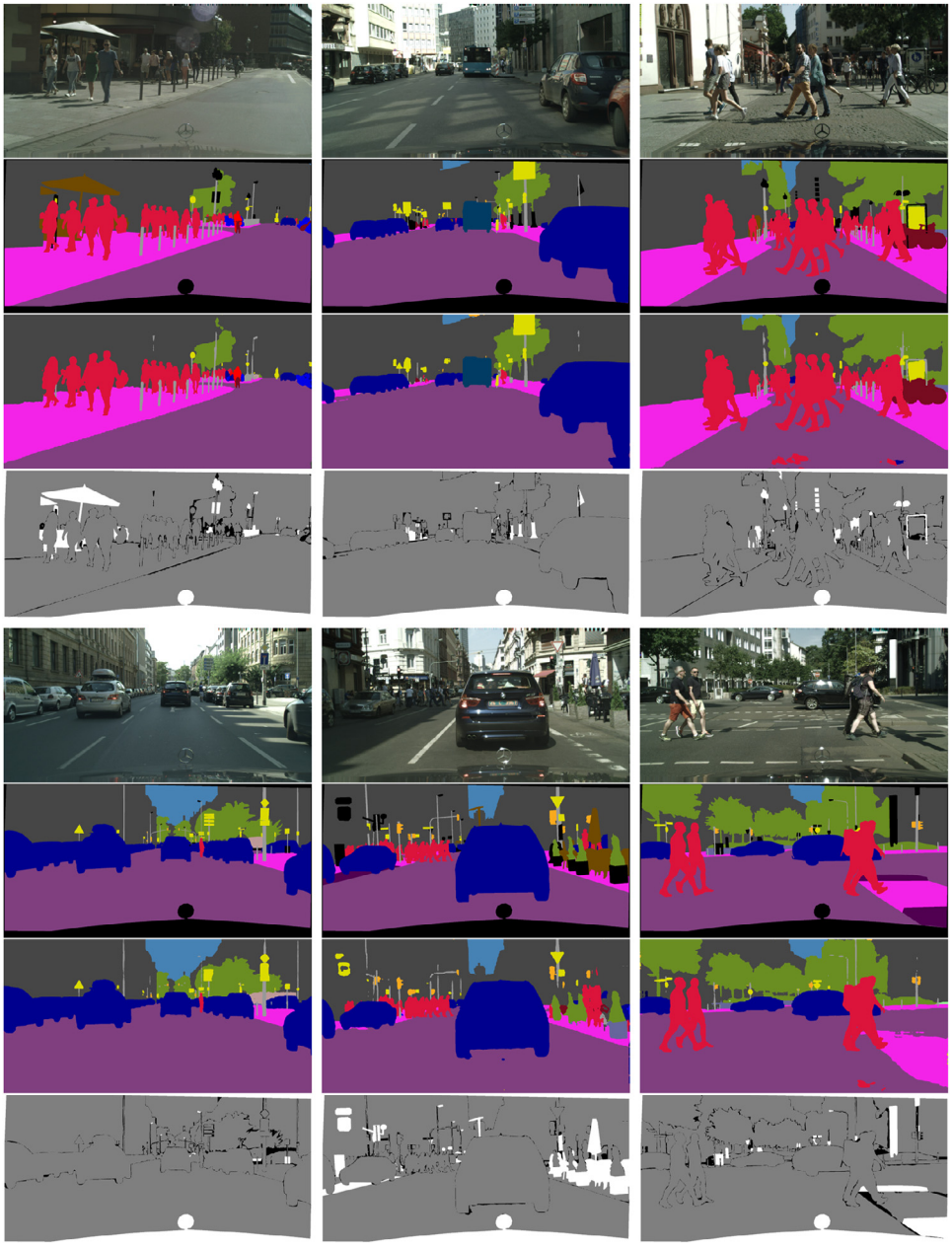

Fig. 6. Qualitative results on the Cityscapes [37] val set. The model was trained using the train set. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

Table 7
Semantic segmentation scores (%) on the PASCAL-Context val set [41].

method	feature	pixel acc.	mean acc.	mean IoU
FCN-8s [16]	VGG16	65.9	46.5	35.1
BoxSup [42]	VGG16	–	–	40.5
Context [36]	VGG16	71.5	53.9	43.3
VeryDeep [29]	ResNet-101	72.9	54.8	44.5
DeepLab-v2 [13]	ResNet-101	–	–	45.7
Ours		**75.0**	**58.1**	**48.1**

the input. Each time, we first feed a batch forward through the whole network; then during the backward pass, we randomly sample k units. For them, we only propagate gradients through their trainable mapping functions, but without their shortcut connections. For the remaining $n - k$ units, we do the opposite, namely, only propagating gradients through their shortcut connections. We record the norm of those gradients that reach the input for varying path length k, and show the results in Fig. 4. Note the varying magnitude and maximum path length in individual figures. These are compared to the middle part of Fig. 6 in [11]. However, differently, we further divide the computed norm of a batch by its number of examples. According to the results in Fig. 4, ResNet-110 trained on CIFAR-10, as well as ResNet-101 and ResNet-152 trained on ILSVRC 2012, generate much smaller gradients from their long paths than from their short paths. In contrast, our Model A trained on ILSVRC 2012, generates more comparable gradients from its paths with different lengths.

6.4. Qualitative results

We show qualitative results of semantic image segmentation on PASCAL VOC [27], Cityscapes [37], ADE20K [38], and PASCAL Context [41], respectively in Figs. 5, 6, 9, 10 and 11, and show

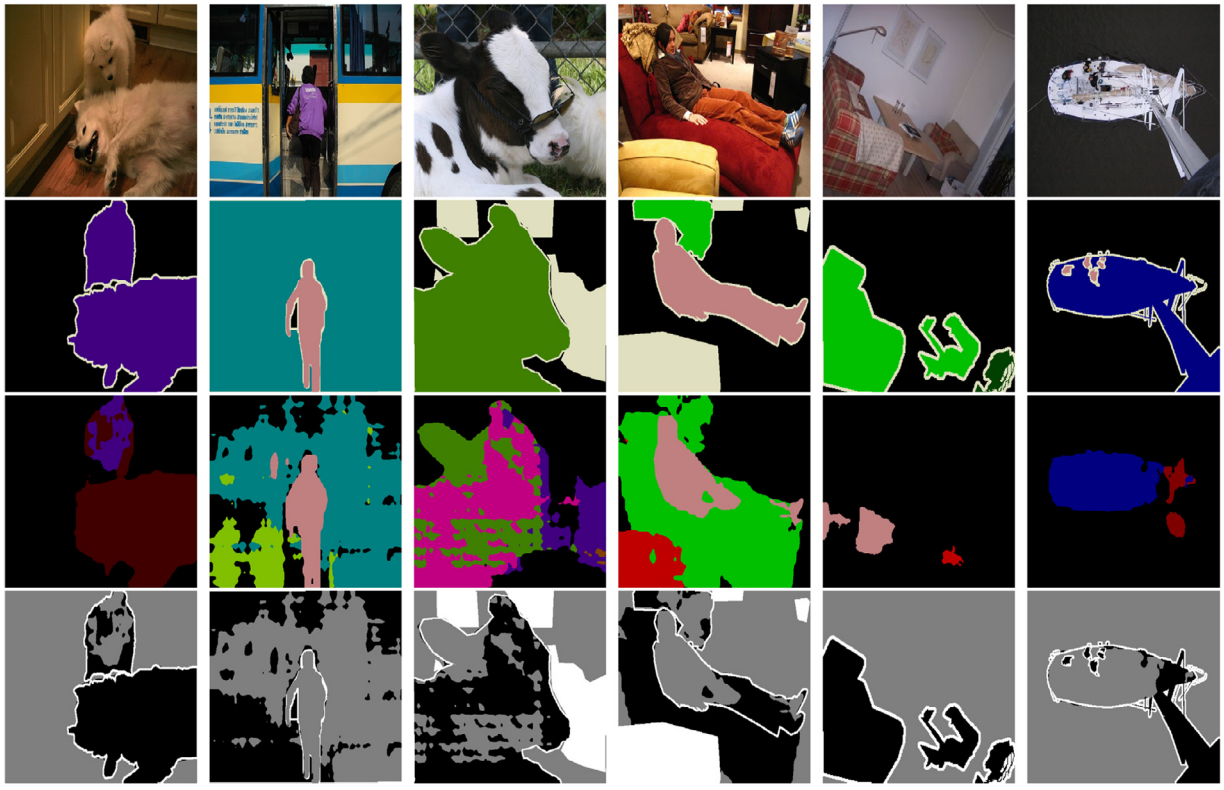

Fig. 7. Failure cases on the PASCAL VOC 2012 [27] val set. The model was trained using the train set augmented using SBD [30]. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

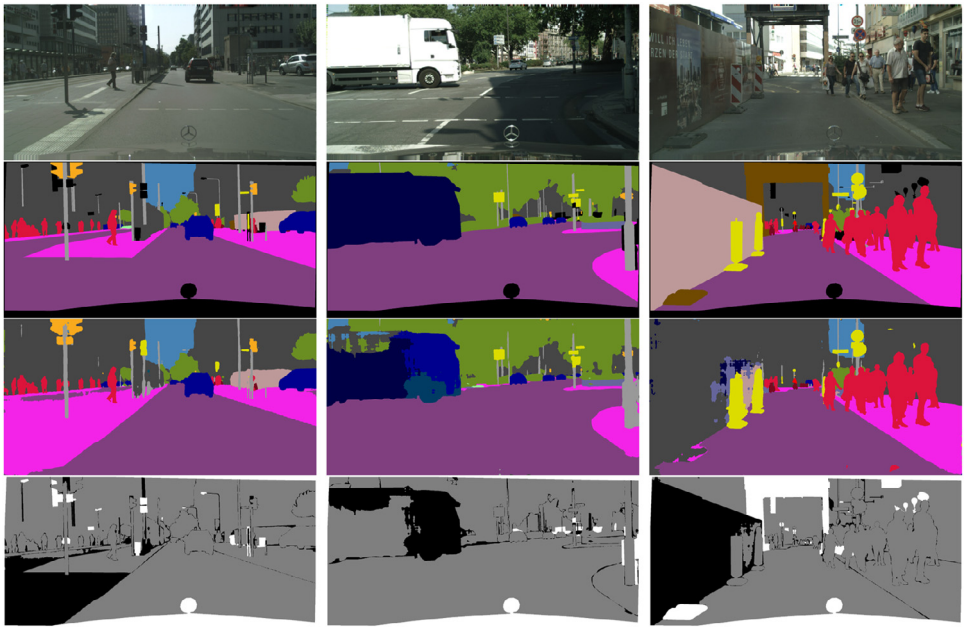

Fig. 8. Failure cases on the Cityscapes [37] val set. The model was trained using the train set. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

some failure cases in Figs 7 and 8. In a difference map, grey and black respectively denotes correctly and wrongly labelled pixels, while white denotes the officially ignored pixels during evaluation. Note that we do not apply post-processing with CRFs, which can smooth the output but is too slow in practice, especially for large images.

7. Conclusion and discussion

We have analysed the ResNet architecture, in terms of the ensemble classifiers therein and the effective depths of the residual units. On the basis of that analysis we have calculated a more spatially efficient and better performing architecture for large net-

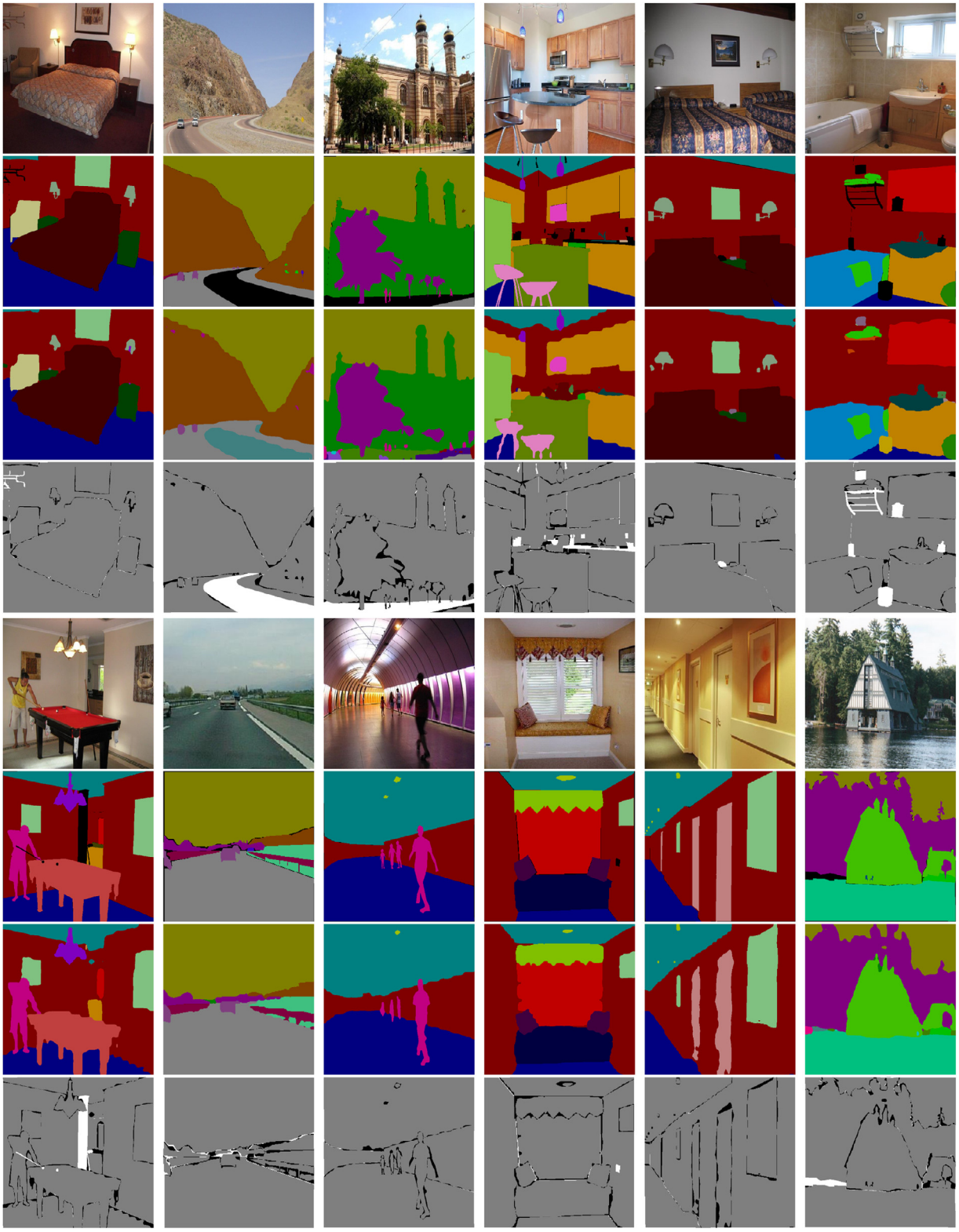

Fig. 9. Qualitative results on the ADE20K [38] val set. The model was trained using the train set. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

works. Using this we have designed a group of correspondingly shallow networks, and showed that they outperform the previous much deeper residual networks not only on the ImageNet classification dataset, but also when applied to semantic segmentation as pre-trained features.

Either based on the literature [11,15] or our work, it is clear that increasing the number of layers may become much less efficient as a model growing too deep. The need thus arises as to identify the turning point. As we have done in Section 6.3 and Fig. 4, one can use the gradient experiment to somehow check if a network has

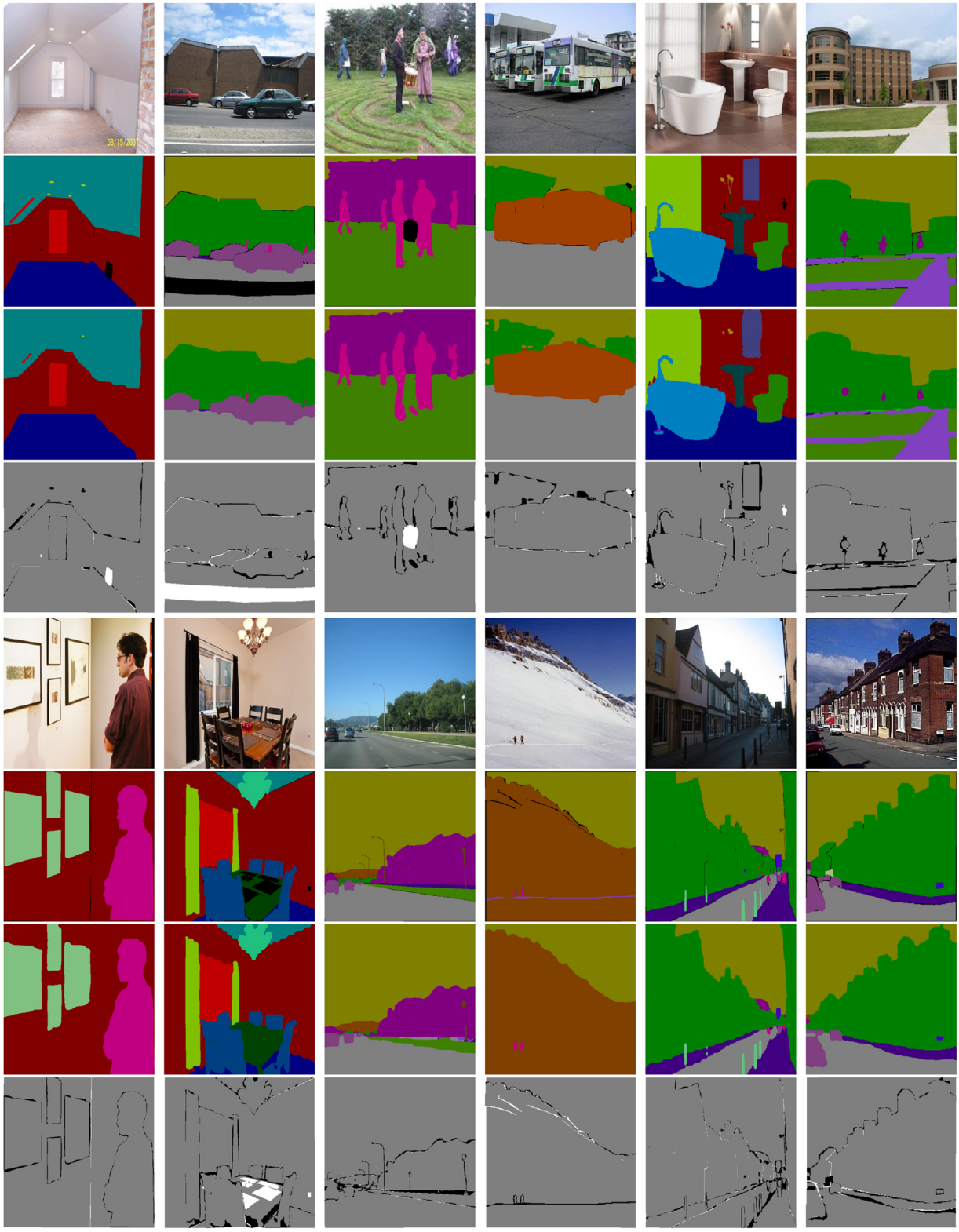

Fig. 10. More qualitative results on the ADE20K [38] val set. The model was trained using the train set. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

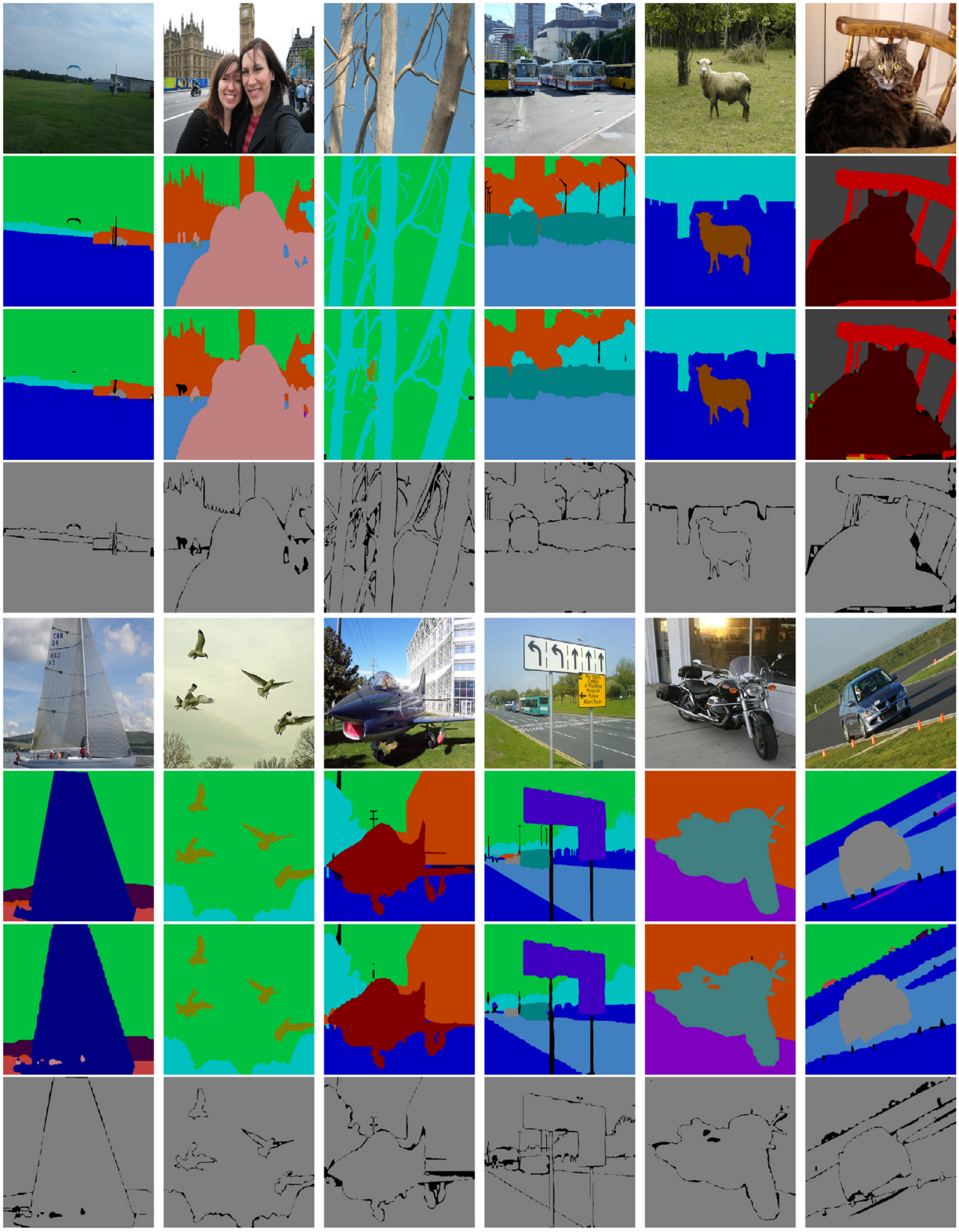

Fig. 11. Qualitative results on the PASCAL Context [41] val set. The model was trained using the train set. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

been over-deepened. Besides, the spatial efficiency of our models may sometimes become important. There are many works claiming the importance of a relatively large batch size for good performance, e.g., PSPNet [9] and MegDet [19]. Particularly, in semantic segmentation and object detection tasks, we can easily hit the GPU memory limit due to the requirement for high-resolution feature maps. For the future, there are at least two directions to improve our work in this paper. First, we should further strengthen the theoretical support for the empirical results. Second, we could search for better network structures, either by reducing the number of

stacked layers in an over-deepened network, or by finding a new way to enable the effective training of long paths in a very deep network, instead of using the skip connections.

References

[1] X. Bai, B. Shi, C. Zhang, X. Cai, L. Qi, Text/non-text image classification in the wild with convolutional neural networks, Pattern Recogn. 66 (2017) 437–446.
[2] X. Shi, M. Sapkota, F. Xing, F. Liu, L. Cui, L. Yang, Pairwise based deep ranking hashing for histopathology image classification and retrieval, Pattern Recogn. 81 (2018) 14–22.
[3] Y. Zhou, Q. Hu, Y. Wang, Deep super-class learning for long-tail distributed image classification, Pattern Recogn. 80 (2018) 118–128.
[4] A. Krizhevsky, I. Sutskever, G. Hinton, ImageNet classification with deep convolutional neural networks, in: Proc. Advances in Neural Inf. Process. Syst., 2012, pp. 1097–1105.
[5] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2016, pp. 770–778.
[6] A. Krizhevsky, Learning multiple layers of features from tiny images, Technical Report, University of Toronto, 2009.
[7] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: Proc. Eur. Conf. Comp. Vis., 2016, pp. 630–645.
[8] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, Inception-Resnet and the impact of residual connections on learning, in: Proc. AAAI Conf. on Artificial Intell., 2017, pp. 4278–4284.
[9] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2017, pp. 6230–6239.
[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, Int. J. Comput. Vision 115 (3) (2015) 211–252.
[11] A. Veit, M. Wilber, S. Belongie, Residual networks behave like ensembles of relatively shallow networks, in: Proc. Advances in Neural Inf. Process. Syst., 2016, pp. 550–558.
[12] J. Dai, K. He, J. Sun, Instance-aware semantic segmentation via multi–task network cascades, in: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2016, pp. 3150–3158.
[13] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2017) 834–848.
[14] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proc. Int. Conf. Learn. Representations, 2015.
[15] S. Zagoruyko, N. Komodakis, Wide residual networks, in: Proc. British Machine Vis. Conf., 2016, pp. 87.1–87.12.
[16] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (4) (2017) 640–651.
[17] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proc. Int. Conf. Mach. Learn., 37, 2015, pp. 448–456.
[18] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proc. Int. Conf. Mach. Learn., 2010, pp. 807–814.
[19] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, J. Sun, MegDet: A large mini-batch object detector, in: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2018.
[20] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D.R.P. Dollár, C. Zitnick, Microsoft COCO: Common objects in context, in: Proc. Eur. Conf. Comp. Vis., 2014, pp. 740–755.
[21] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, IEEE Trans. Pattern Anal. Mach. Intell. 39 (6) (2015) 1137–1149.
[22] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, Z. Zhang, MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems, in: Proc. Advances in Neural Inf. Process. Syst., Workshop on Mach. Learn. Syst., 2016.
[23] S. Gross, M. Wilber, Training and investigating residual nets, 2016, (http://torch.ch/blog/2016/02/04/resnets.html).
[24] D. Mishkin, N. Sergievskiy, J. Matas, Systematic evaluation of CNN advances on the ImageNet, Comp. Vis. Image Understanding 161 (C) (2017) 11–19.
[25] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. Yuille, Semantic image segmentation with deep convolutional nets and fully connected CRFs, in: Proc. Int. Conf. Learn. Representations, 2015.
[26] J. Wu, C.-W. Xie, J.-H. Luo, Dense CNN learning with equivalent mappings, 2016, (CoRR abs/1605.07251).
[27] M. Everingham, S. Eslami, L. van Gool, C. Williams, J. Winn, A. Zisserman, The PASCAL visual object classes challenge: A retrospective, Int. J. Comput. Vision 111 (1) (2015) 98–136.
[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[29] Z. Wu, C. Shen, A. van den Hengel, Bridging category-level and instance-level semantic image segmentation, 2016, (CoRR abs/1605.06885).
[30] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, J. Malik, Semantic contours from inverse detectors, in: Proc. IEEE Int. Conf. Comp. Vis., 2011, pp. 991–998.
[31] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, A. Oliva, Places: A 10 million image database for scene understanding, IEEE Trans. Pattern Anal. Mach. Intell. 40 (6) (2018) 1452–1464.
[32] L.-C. Chen, G. Papandreou, F. Schroff, H. Adam, Rethinking atrous convolution for semantic image segmentation, 2017, (CoRR abs/1706.05587).
[33] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, P. Torr, Conditional random fields as recurrent neural networks, in: Proc. IEEE Int. Conf. Comp. Vis., 2015, pp. 1529–1537.
[34] H. Noh, S. Hong, B. Han, Learning deconvolution network for semantic segmentation, in: Proc. IEEE Int. Conf. Comp. Vis., 2015, pp. 1520–1528.
[35] Z. Liu, X. Li, P. Luo, C. Loy, X. Tang, Semantic image segmentation via deep parsing network, in: Proc. IEEE Int. Conf. Comp. Vis., 2015, pp. 1377–1385.
[36] G. Lin, C. Shen, A. van den Hengel, I. Reid, Exploring context with deep structured models for semantic segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 40 (6) (2017) 1352–1366.
[37] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The Cityscapes dataset for semantic urban scene understanding, in: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2016, pp. 3213–3223.
[38] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, A. Torralba, Scene parsing through ADE20K dataset, in: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2017, pp. 5122–5130.
[39] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, in: Proc. Int. Conf. Learn. Representations, 2016.
[40] G. Ghiasi, C.C. Fowlkes, Laplacian pyramid reconstruction and refinement for semantic segmentation, in: Proc. Eur. Conf. Comp. Vis., 2016, pp. 519–534.
[41] R. Mottaghi, X. Chen, X. Liu, N. Cho, S. Lee, S. Fidler, R. Urtasun, A. Yuille, The role of context for object detection and semantic segmentation in the wild, in: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2014, pp. 891–898.
[42] J. Dai, K. He, J. Sun, BoxSup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation, in: Proc. IEEE Int. Conf. Comp. Vis., 2015, pp. 1635–1643.

Zifeng Wu studied at Beihang University, Beijing, China, and received his PhD degrees in the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2015, and he is now a post-doctoral researcher working at the Australian Centre for Visual Technologies at the University of Adelaide, Adelaide, South Australia, Australia. His research interests include computer vision and deep learning.

Chunhua Shen studied at Nanjing University, at Australian National University, and received the PhD degree from the University of Adelaide. He is a professor at School of Computer Science, University of Adelaide. His research interests are in the intersection of computer vision and statistical machine learning. In 2012, he was awarded the Australian Research Council Future Fellowship.

Anton van den Hengel received the bachelor of mathematical science degree, bachelor of laws degree, master's degree in computer science, and the PhD degree in computer vision from the University of Adelaide in 1991, 1993, 1994, and 2000, respectively. He is a professor and the founding director of the Australian Centre for Visual Technologies, at the University of Adelaide, focusing on innovation in the production and analysis of visual digital media.