Wider or Deeper: Revisiting the ResNet Model for Visual Recognition*

Zifeng Wu, Chunhua Shen, and Anton van den Hengel
School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia
e-mail: firstname.lastname@adelaide.edu.au

Abstract

The trend towards increasingly deep neural networks has been driven by a general observation that increasing depth increases the performance of a network. Recently, however, evidence has been amassing that simply increasing depth may not be the best way to increase performance, particularly given other limitations. Investigations into deep residual networks have also suggested that they may not in fact be operating as a single deep network, but rather as an ensemble of many relatively shallow networks. We examine these issues, and in doing so arrive at a new interpretation of the unravelled view of deep residual networks which explains some of the behaviours that have been observed experimentally. As a result, we are able to derive a new, shallower, architecture of residual networks which significantly outperforms much deeper models such as ResNet-200 on the ImageNet classification dataset. We also show that this performance is transferable to other problem domains by developing a semantic segmentation approach which outperforms the state-of-the-art by a remarkable margin on datasets including PASCAL VOC, PASCAL Context, and Cityscapes. The architecture that we propose thus outperforms its comparators, including very deep ResNets, and yet is more efficient in memory use and sometimes also in training time. The code and models are available at https://github.com/itijyou/ademxapp.

Contents

*Correspondence should be addressed to C. Shen.

1. Introduction

The convolutional networks used by the computer vision community have been growing deeper and deeper each year since Krizhevsky et al. [16] proposed AlexNet in 2012. The deepest network [12] in the literature is a residual network (ResNet) with 1,202 trainable layers, which was trained using the tiny images in the CIFAR-10 dataset [15]. The image size here is important, because it means that the size of corresponding feature maps is relatively small, which is critical in training 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 [13] and 96-layer Inception-ResNet [30]. The progression to deeper networks continues, however, with Zhao et al. [37] having trained a 269-layer network for semantic image segmentation. These networks were trained using the ImageNet classification dataset [27], where the images are of much higher resolution. Each additional layer requires not only additional memory, but also additional training. The marginal gains achieved by each additional layer diminish with depth, however, to the point where Zhao et al. [37] 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[12] 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. [31] have claimed that they 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 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 characteristic of our introduced view is that it depends on the effective depth l of a network. This l amounts to the number of residual units which backward gradients during training can go through. When $l \geq 2$, the two-unit ResNet in Fig. 1 can be seen as an ensemble of three sub-networks, i.e., M_{a}, M_{b}, and M_{e}^2, as shown in the bottom left. 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. [31] 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:

- We introduce a further developed intuitive view of ResNets, which helps us understand their behaviours, and find possible directions to further improvements.
- We propose a group of relatively shallow convolutional networks based on our new understanding. Some of them achieve the state-of-the-art results on the ImageNet classification dataset [27].
- 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. We achieve the best results on PASCAL VOC [8], PASCAL Context [24], and Cityscapes [5].

2. Related work

Our work here is closely related to two topics, residual network (ResNet) based image classification and semantic image segmentation using fully convolutional networks .

As we have noted above, He et al. [12, 13] recently proposed the ResNets 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 [7] and semantic image segmentation [3]. They are gradually replacing VGGNets [28] 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. [31] claimed that they behave like exponential ensembles of relatively shallow networks, yet 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. [30] 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 [12]. Szegedy et al. [30] observed similar empirically results to support that. On the other hand, Zagoruyko and Komodakis [36] found that a wide sixteen-layer ResNet outperformed the original thin thousand-layer ResNet [13] on datasets composed of tiny images such as CIFAR-10 [15]. The analysis we present here is motivated by their empirical testing, but aims at a more theoretical approach, and the observation that a grid

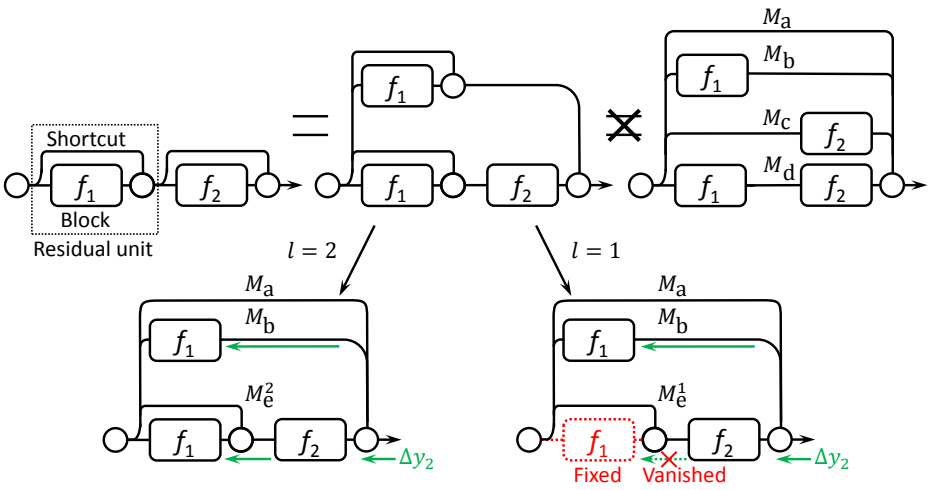

Figure 1. The unravelled view of a simple ResNet. The fact that $f_2(\cdot)$ is non-linear gives rise to the inequality in the top row, as $f_2(a+b) \neq f_2(a) + f_2(b)$. This shows that $f_2(\cdot)$ never operates independently of the result of $f_1(\cdot)$, and thus that the number of independent classifiers increases linearly with the number of residual units. Analysing the interactions between residual units at a given effective depth, here labelled l, illuminates the paths taken by gradients during training.

search of configuration space is impractical on large scale datasets such as the ImageNet classification dataset [27].

Semantic image segmentation amounts to predicting the categories for each pixel in an image. Long et al. [22] proposed the fully convolutional networks (FCN) to this end. FCNs soon became the mainstream approach to dense prediction based tasks, especially due to its efficiency. Besides, empirical results in the literature [3] showed that stronger pre-trained features can yet further improve their performance. We thus here base our semantic image segmentation approach on fully convolutional networks, and will show the impact of different pre-trained features on final segmentation results.

3. Residual networks revisited

We are concerned here with the full pre-activation version of residual networks (ResNet) [13]. For shortcut connections, we consider identity mappings [13] only. We omit the raw input and the top-most linear classifier for clarity. Usually, there may be a stem block [30] or several traditional convolution layers [12, 13] directly after the raw input. We omit these also, for the purpose of simplicity.

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:

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

where $\boldsymbol{w}_i$ denotes the trainable parameters, and $f_i(\cdot)$ is often two or three stacked convolution stages. In the full pre-activation version, the components of a stage are in turn a batch normalization [14], a rectified linear unit [25] (ReLU) non-linearity, and a convolution layer.

3.1. Residual networks unravelled online

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

$$\begin{aligned} y_2 &= y_1 + f_2(y_1, \boldsymbol{w}_2) && (2)\\ &= y_0 + f_1(y_0, \boldsymbol{w}_1) + f_2(y_0 + f_1(y_0, \boldsymbol{w}_1), \boldsymbol{w}_2) && (3)\\ &\neq y_0 + f_1(y_0, \boldsymbol{w}_1) + f_2(y_0, \boldsymbol{w}_2) + f_2(f_1(y_0, \boldsymbol{w}_1), \boldsymbol{w}_2), && (4) \end{aligned}$$

which describes the unravelled view by Veit et al. [31], as shown in the top row of Fig. 1. Since $f_2(\cdot)$ is non-linear, we cannot derive Eqn.(4) from Eqn.(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 Fig. 1, there are three, e.g., M_a, M_b, and M_e^2, sub-networks respectively corresponding to the three terms in Eqn.(3), i.e., y_0, $f_1(y_0, \boldsymbol{w}_1)$, and $f_2(y_0 + f_1(y_0, \boldsymbol{w}_1), \boldsymbol{w}_2)$.

Veit et al. in [31] showed that the paths which gradients take through a ResNet are typically far shorter than the total depth of that network. They thus introduced the idea of *effective depth* as a measure for the true length of these paths. By characterising the units of a ResNet given its effective depth, we illuminate the impact of varying paths that gradients actually take, as in Fig. 1. We here illustrate this impact in terms of small effective depths, because to do so for larger ones would require diagrams of enormous networks. The impact is the same, however.

Take the ResNet in Fig. 1 for example again. In an SGD

iteration, the backward gradients are:

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

$$\Delta y_1 = \Delta y_2 + f_2' \cdot \Delta y_2 \quad (6)$$

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

where f_2' denotes the derivative of $f_2(\cdot)$ to its input y_1. When effective depth $l \geq 2$, both terms in Eqn.(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 Eqn.(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 iterations, but they are updated only by M_b. To M_e^1, Block 1 is no more than an additional input providing preprocessed representations, because Block 1 is not end-to-end trained, from the point of view of M_e^1. In this case, we name M_e^1 to have an *actual depth* of one. We say that the ResNet is over-deepened, and that it cannot be trained in a fully end-to-end manner, even with those shortcut connections.

Let d be the total number of residual units. We can see a ResNet as an ensemble of different sub-networks, i.e., $M_i, i = \{0, 1, \cdots, d\}$. The actual depth of M_i is $\min(i, l)$. We show an unravelled three-unit ResNet with different effective depths in Fig. 2. By way of example, note that M_1 in Fig. 2 contains only Block 1, whereas M_2 contains both Block 1 and Block 2. Among the cases illustrated, the bottom left example is more complicated, where $d = 3$ and $l = 2$. From the point of view of 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, it is non-trivial to tell if this is as good as the fully end-to-end training case, as illustrated by M_3 in the top right example. An investigation of this issue remains future work.

3.2. Residual networks behaviours revisited

Very deep ResNets. Conventionally, it is not easy to train very deep networks due to the vanishing gradient problem [1]. To understand how a very deep ResNet is trained, the observation by Veit et al. [31] is important, i.e., gradients vanish exponentially as the length of paths increases. Now refer to the top-right example in Fig. 2. This is somewhat similar to the case of a shallow or reasonably deep ResNet, when $d \leq l$. At the beginning, the shallowest sub-network, i.e., M_1, converges fast, because it gives Block 1 the largest gradients. From the point of view of M_2, Block 2 may also receive large gradients due to the path with a length of one. However, the input of Block 2 partly depends

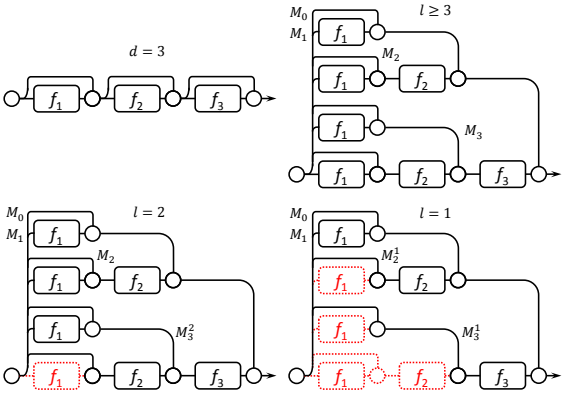

Figure 2. The impact of inserting an extra residual unit into a two-unit ResNet, which depends on the effective depth l.

on Block 1. It would not be easy for Block 2 to converge before the output of Block 1 stabilises. Similarly, Block 3 will need to wait for Blocks 1 and 2, and so forth. In this way, a ResNet seems like an ensemble with a growing number of sub-networks. Besides, each newly added sub-network will have a larger actual depth than all the previous ones. Note that Littwin and Wolf [20], in a concurrent work, have theoretically showed that ResNets are virtual ensembles whose depth grows as training progresses. Their result to some extent coincides with the above described process.

The story will however be different when the actual depth becomes as large as the effective depth. Refer to the bottom-right example in Fig. 2. This is somewhat similar to the case of an over-deepened ResNet, when d is much larger than l. Again, Block 1 in M_1 gets trained and stabilises first. However, this time M_2^1 is not fully end-to-end trained any more. Since M_2^1 gives no gradients to Block 1, it becomes a one-block sub-network trained on top of some preprocessed representations, which are obtained by adding the output of Block 1 up to the original input. In this way, the newly added sub-network M_2^1 still has an actual depth of one, which is no deeper than the previous one, i.e., M_1, and so forth for M_3^1. ResNets thus avoid the vanishing gradient problem by reshaping themselves into multiple shallower sub-networks. This is just another view of delivering gradients to bottom layers through shortcut connections. Researchers [17, 30] have claimed that the residual shortcut connections are not necessary even in very deep networks. However, there are usually short paths in their proposed networks as well. For example, the 76-layer Inception-v4 network [30] has a much shorter twenty-layer route from its input to the output. There might be differences in the details [32] between fusion by concatenation (Inception-v4) and fusion by summation (ResNets). However, the manner of avoiding the vanishing gradient problem is probably similar, i.e., using shortcuts, either with trainable weights or not. We are thus not yet in a position to be able to claim that

the vanishing gradient problem has been solved.

Wide ResNets. Conventionally, wide layers are more prone to over-fitting, and sometimes require extra regularization such as dropout [29]. However, Zagoruyko and Komodakis [36] showed the possibility to effectively train times wider ResNets, even without any extra regularization. To understand how a wide ResNet is trained, refer to the top right example in Fig. 2. This simulates the case of a rather shallow network, when d is smaller than l. We reuse the weights of Block 1 for four times. Among these, Block 1 is located in three different kinds of circumstances. In the bottom-most path of the sub-network M_3, it is supposed to learn some low-level features; in M_2, it should learn both low-level and mid-level features; and in M_1, it has to learn everything. This format of weight sharing may suppress over-fitting, especially for those units far from the top-most linear classifier. Hence ResNets inherently introduce regularization by weight sharing among multiple very different sub-networks.

Residual unit choices. For better performance, we hope that a ResNet should expand into a sufficiently large number of sub-networks, some of which should have large model capacity. So, given our previous observations, the requirements for an ideal mapping function in a residual unit are, 1) being strong enough to converge even if it is reused in many sub-networks, and 2) being shallow enough to enable an large effective depth. Since it is very hard to build a model with large capacity using a single trainable layer [13], the most natural choice would be a residual unit with two wide convolution stages. This coincides with empirical results reported by Zagoruyko and Komodakis [36]. They found that, among the most trivial structure choices, the best one is to stack two 3×3 convolution stages.

3.3. Wider or deeper?

To summarize the previous subsections, shortcut connections enable us to train wider and deeper networks. As they growing to some point, we will face the dilemma between width and depth. From that point, going deep, we will actually get a wider network, with extra features which are not completely end-to-end trained; going wider, we will literally get a wider network, without changing its end-to-end characteristic. We have learned the strength of depth from the previous plain deep networks without any shortcuts, e.g., the AlexNet [16] and VGGNets [28]. However, it is not clear whether those extra features in very deep residual networks can perform as well as conventional fully end-to-end trained features. So in this paper, we only favour a deeper model, when it can be completely end-to-end trained.

In practice, algorithms are often limited by their spatial costs. One way is to use more devices, which will however increase communication costs among them. With similar memory costs, a shallower but wider network can have times more number of trainable parameters. Therefore, given the following observations in the literature,

- Zagoruyko and Komodakis [36] found that the performance of a ResNet was related to the number of trainable parameters. Szegedy et al. [30] came to a similar conclusion, according to the comparison between their proposed Inception networks.
- Veit et al. [31] found that there is a relatively small effective depth for a very deep ResNet, e.g., seventeen residual units for a 110-layer ResNet.

most of the current state-of-the-art models on the ImageNet classification dataset [27] seem over-deepened, e.g., the 200-layer ResNet [13] and 96-layer Inception-ResNet [30]. The reason is that, to effectively utilize GPU memories, we should make a model shallow. According to our previous analysis, paths longer than the effective depth in ResNets are not trained in a fully end-to-end manner. Thus, we can 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 that our fully end-to-end networks can perform much better than the previous much deeper ResNets, 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 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 stages, which are respectively composed of a batch normalization, an ReLU nonlinearity and a convolution layer, following the second version of ResNets [13]. The closely stacked two or three convolution stages denote different kinds of residual units (B1–B7), with inner shortcut connections [13]. 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 [12], 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 memories. 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 stages (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.

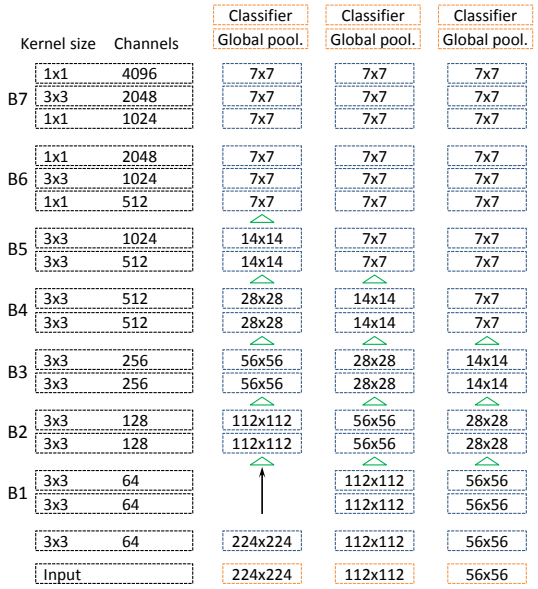

Figure 3. Overview of our proposed networks with different input sizes. Note that B1–B7 are respectively a residual unit.

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 [16] and VGGNets [28]. We will show more details about network structures in Subsection 6.1.

Implementation details. We run all experiments using the MXNet framework [4], 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 [10] as possible. But, we use a linear learning rate schedule, which was reported as a better choice by Mishkin et al. [23]. 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) [22] implemented in the first version of DeepLab [2]. However, without getting too many factors entangled, we in this paper do not introduce any multi-scale structures [18, 3], deep supervision signals [22, 37], or global context features [37]. Besides, we do not apply any multi-scale testing, model averaging or CRF based post-processing, except for the test set of ADE20K [40].

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 [33]. On the other hand, 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 pooling layers in our originally pre-trained networks. We find it harmful for FCNs in our preliminary experiments, probably due to too strong spatial invariance. To this end, we replace several top-most down-sampling operations in a network, and then tune it for some additional iterations. Take Model A in Table 1 for example again. We remove the top-most three pooling layers (before B4, B5 and B6), increase the strides of according convolution layers up to two, and tune it for 45k iterations using the ImageNet dataset [27], starting from a learning rate of 0.01.

2) **Classifier**. We remove the top-most linear classifier and the global pooling layer, and then consider two cases. For one thing, we follow a basic large field-of-view setting in DeepLab-v2 [3], called '1 convolution'. Namely, we just add back a single linear layer as the new classifier. For anther, we insert an additional non-linear convolution stage (without batch normalization) below the linear classifier. This case is called '2 convolutions'. Both of the added layers have 3×3 kernels, with a dilation rate of twelve. The top-most two-layer classifier thus has a receptive field of 392×392 on the final feature maps. By default, we let the number of channels in the hidden layer be 512.

3) **Dropout**. To alleviate over-fitting, we also apply the traditional dropout [29] 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 [12]. We use four devices on a single node. The batch size is sixteen, 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 [23]. For datasets with available testing sets, we evaluate these numbers of iterations on validation sets. During training, we first re-

size 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.

6. Experimental results

6.1. Image classification results

We evaluate our proposed networks[1] on the ILSVRC 2012 classification dataset [27], 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 [28] 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. [13]. For those with 112×112 and 56×56 inputs, we use 160×160 and 80×80 crops respectively. For Inception networks [30], the testing crop size is 299×299 [13]. 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).

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 [36]. 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. [23] also reduced the input size for efficiency. However, they did not remove down-sampling 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.

[1] We will release these networks soon.

method	depth	tr. input	top-1	top-5	speed
VGG16 [28], 10 crops	16	224	28.1	9.3	–
ResNet-50 [12], our tested	50	224	23.5	6.8	75.2
ResNet-101 [12], our tested	101	224	22.1	6.1	56.8
ResNet-152 [12], our tested	152	224	21.8	5.8	41.8
ResNet-152 [13]	152	224	21.3	5.5	–
ResNet-152 [13], pre-act.	152	224	21.1	5.5	–
ResNet-200 [13], pre-act.	200	224	20.7	5.3	–
Inception-v4 [30]	76	299	20.0	5.0	–
Inception-ResNet-v2 [30]	96	299	19.9	4.9	–
56-1-1-1-1-9-1-1, Model F	34	56	25.2	7.8	113.5
112-1-1-1-1-5-1-1, Model E	26	112	22.3	6.2	97.3
112-1-1-1-1-9-1-1, Model D	34	112	22.1	6.0	81.2
112-1-1-1-1-13-1-1, Model C	42	112	21.8	5.9	69.2
224-0-1-1-1-1-1-1	16	224	22.0	5.8	55.3
224-0-1-1-1-3-1-1, Model B	20	224	21.0	5.5	43.3
224-0-3-3-6-3-1-1, Model A	38	224	**19.2**	**4.7**	15.7

Table 1. Comparison of networks by top-1 (%) and top-5 (%) errors on the ILSVRC 2012 validation set [27] with 50k images, obtained using a single crop. Testing speeds (images/second) are evaluated with ten images/mini-batch using cuDNN 4 on a GTX 980 card. Input sizes during training are also listed. Note that a smaller size often leads to faster training speed.

PASCAL VOC 2012 [8]. 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 [22, 2], we augment the dataset with extra labelled images from the semantic boundaries dataset [11]. 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 improvement by 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 by 2.1%.

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

3) Model B performs worse than ResNet-101, even if it performs 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. And it again shows why we should favour deeper models.

4) Model A2 performs worse than Model A on this dataset. We initialize it using weights from Model A, and tune it with the Places 365 data [39] 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

method	pixel acc.	mean acc.	mean IoU	mem.
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

Table 2. Comparison by semantic image segmentation scores (%) and GPU memory usages (GB/device) during tuning on the PASCAL VOC val set [8] with 1,449 images. Memory usages are obtained with four images/device using MXNet [4].

test set in Table 3. Only using the augmented PASCAL VOC data for training, we achieve a mean IoU score of 82.5%[2], 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%. Our method wins for seventeen out of the twenty object categories, which was the official criteria used in the PASCAL VOC challenges [8]. In some works [18, 3], models were further pre-trained using the Microsoft COCO [19] data, which consists of 120k labelled images. In this case, the current best mean IoU is 79.7% reported by Chen et al. [3]. They also used multi-scale structure and CRF-based post-processing in their submission, which we do not consider here. Nevertheless, our method outperforms theirs by 2.8%, which further shows the effectiveness of our features pre-trained only using the ImageNet classification data [27].

Cityscapes [5]. 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 classes 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 [39] 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*. Besides, there is also an instance-weighted IoU score for each of the two, which assigns high scores to those pixels of small instances.

[2] http://host.robots.ox.ac.uk:8080/anonymous/H0KLZK.html

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**

Table 4. Comparison by semantic image segmentation scores (%) on the Cityscapes val set [5] with 500 images, and the ADE20K val set [40] with 2k images.

method	cla. IoU	cla. iIoU	cat. IoU	cat. iIoU
Dilation10 [35]	67.1	42.0	86.5	71.1
DeepLab-v2* [13]	70.4	42.6	86.4	67.7
Context [18]	71.6	51.7	87.3	74.1
LRR+ [9]	71.8	47.9	88.4	73.9
Model A2, 2 conv.	**78.4**	**59.1**	**90.9**	**81.1**

Table 5. Comparison by semantic image segmentation scores (%) on the Cityscapes test set [5] with 1,525 images. DeepLab-v2 [13] uses ResNet-101 [12], while others use VGG16 [28]. LRR [9] also uses the coarse set for training.

Namely, this score penalizes methods ignoring small instances, which may cause fatal problems in vehicle-centric scenarios. Our method achieves a class-level IoU score of 78.4%[3], and outperforms the previous best one by 6.6%. Furthermore, in the case of instance-weighted IoU score, our method also performs better than the previous best one by 6.4%. It is notable that these significant improvements show the strength of our pre-trained features, considering that DeepLab-v2 [3] uses ResNet-101, and LRR [9] uses much more data for training.

ADE20K [40]. 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 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 classifica-

[3] https://www.cityscapes-dataset.com/benchmarks

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 [22]	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 [38]	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 [26]	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 [21]	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 [18]	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* [34]	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
Model A, 2 conv.	**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 [18]	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* [3]	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

Table 3. Comparison with previous results by mean intersection-over-union scores (%) on the PASCAL VOC test set [8] with 1,456 images. Asterisked methods use ResNet-101 [12], while others use VGG16 [28].

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

Table 6. Comparison by semantic image segmentation scores (%) on the ADE20K test set [40] with 3,352 images.

tion 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 [24]. This dataset consists of images from PASCAL VOC 2010 [8] 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 [3].

method	feature	pixel acc.	mean acc.	mean IoU
FCN-8s [22]	VGG16	65.9	46.5	35.1
BoxSup [6]	VGG16	–	–	40.5
Context [18]	VGG16	71.5	53.9	43.3
VeryDeep [34]	ResNet-101	72.9	54.8	44.5
DeepLab-v2 [3]	ResNet-101	–	–	45.7
Model A2, 2 conv.		**75.0**	**58.1**	**48.1**

Table 7. Comparison by semantic image segmentation scores (%) on the PASCAL Context val set [24] with 5,105 images.

7. Conclusion

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 calculated a new, more spatially efficient, and better performing architecture which actually achieves fully end-to-end training for large networks. Using this new architecture we designed a group of correspondingly shallow networks, and showed that they outperform the previous very deep residual networks not only on the ImageNet classification dataset, but also when applied to semantic image segmentation. These results show that the proposed architecture delivers better feature extraction performance than the current state-of-the-art.

A. Appendix

A.1. Network structures

The graph structures of Model A for the ImageNet (ILSVRC 2012) [27] classification can be accessed at:
https://cdn.rawgit.com/itijyou/ademxapp/master/misc/ilsvrc_model_a.pdf
Model A2 for the PASCAL VOC 2012 [8] segmentation can be accessed at:
https://cdn.rawgit.com/itijyou/ademxapp/master/misc/voc_model_a2.pdf.

A.2. Gradients in residual networks

We show results of the experiment on gradients proposed by Veit et al. [31], 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 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 [31]. 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.

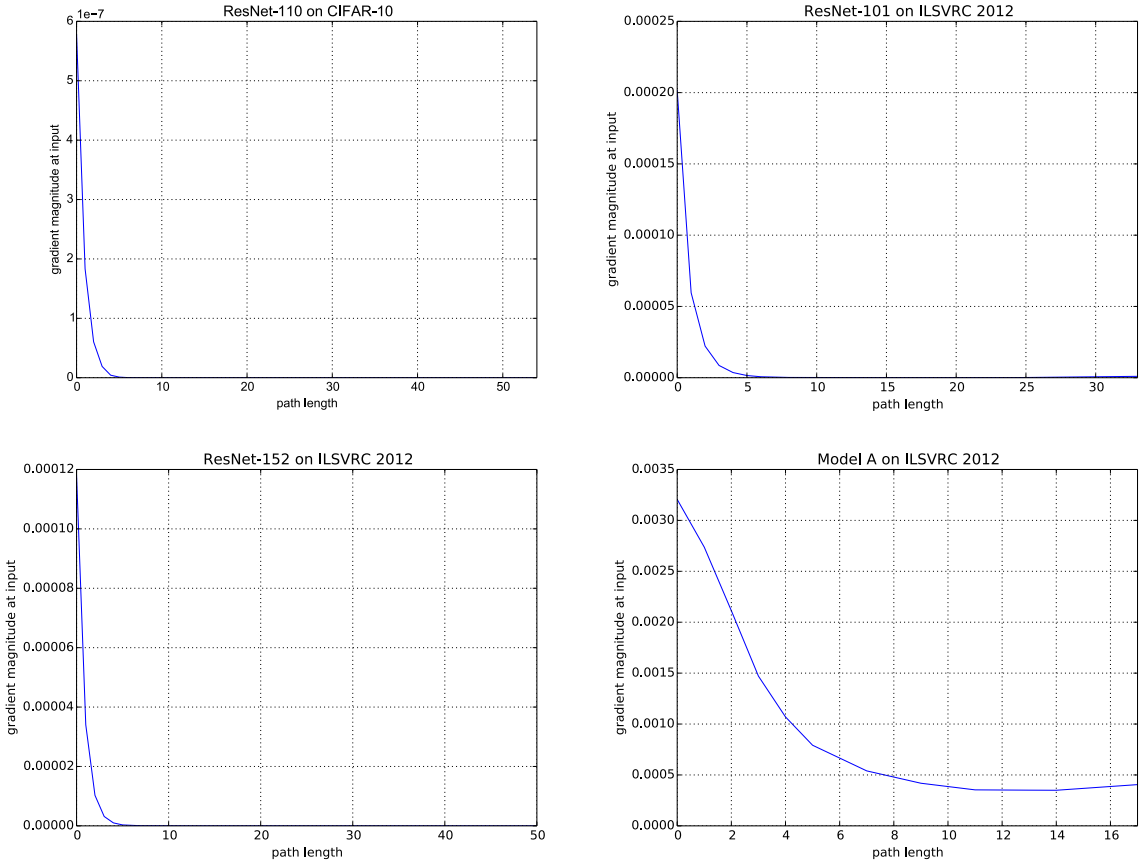

Figure 4. Gradient magnitude at input given a path length k in various residual networks. See the text for details.

A.3. Qualitative results

We show qualitative results of semantic image segmentation on PASCAL VOC [8], Cityscapes [5], ADE20K [40], and PASCAL Context [24], respectively in Figs. 5, 6, 9, 10 and 11, and show 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.

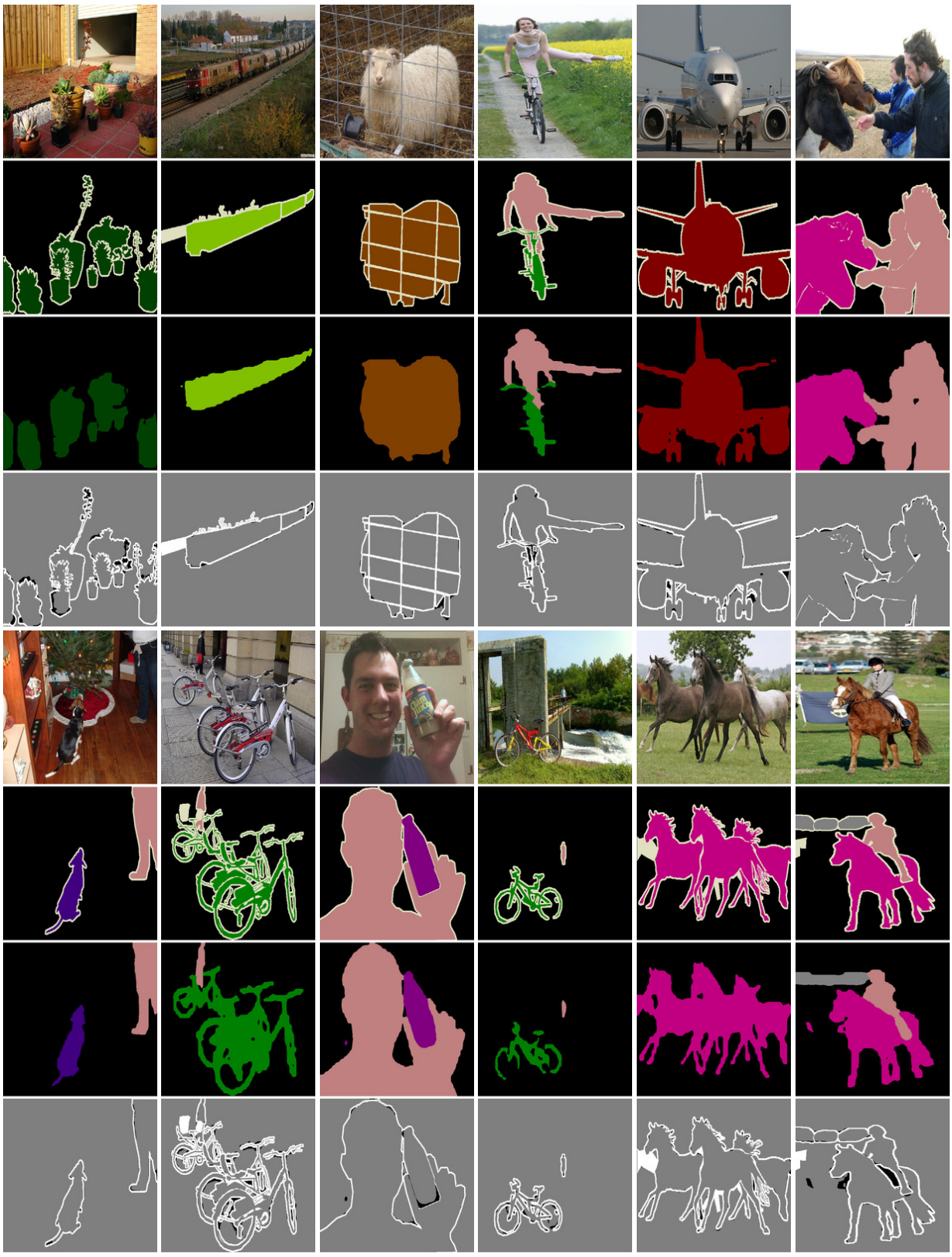

Figure 5. Qualitative results on the PASCAL VOC 2012 [8] val set. The model was trained using the train set augmented using SBD [11]. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

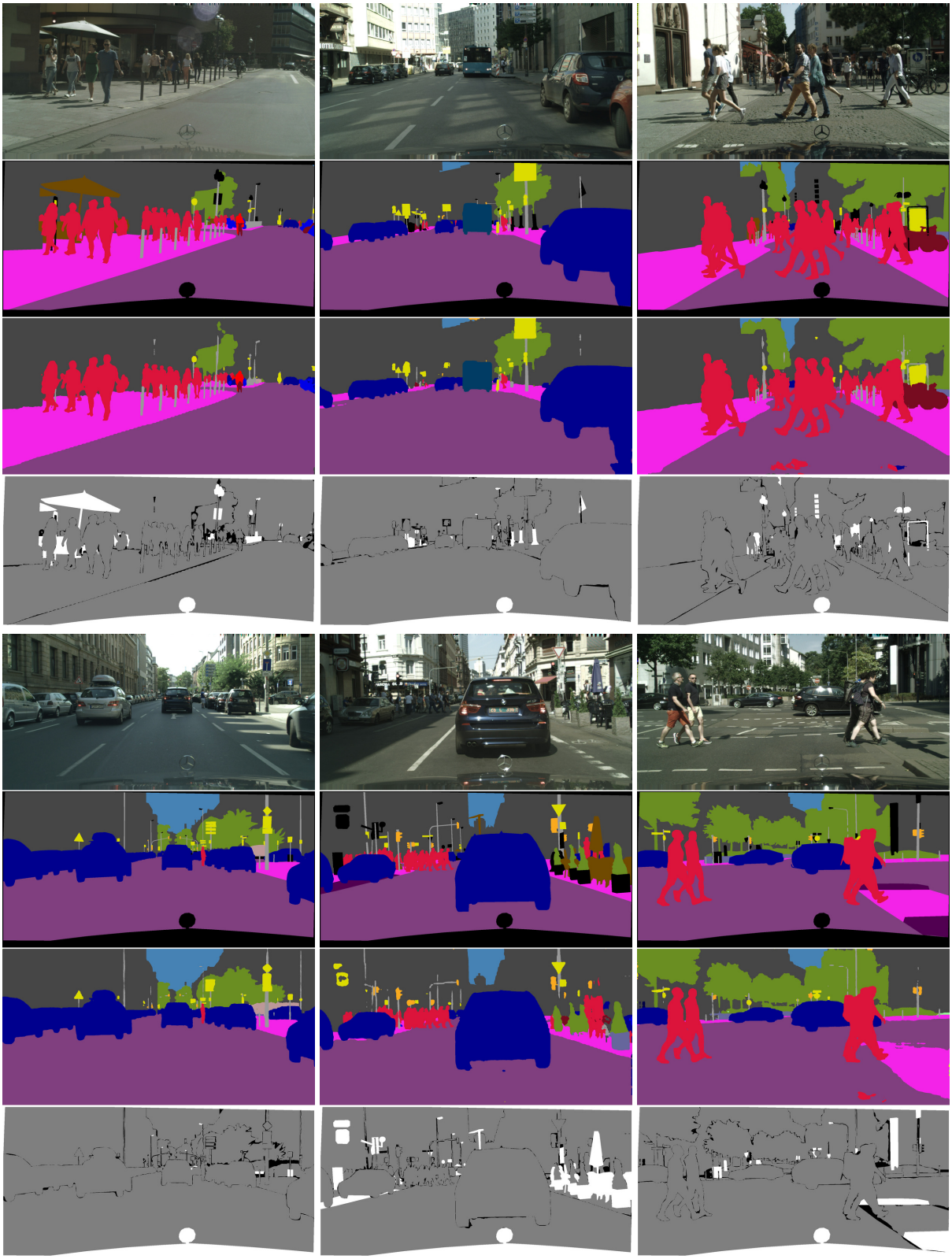

Figure 6. Qualitative results on the Cityscapes [5] val set. The model was trained using the train set. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

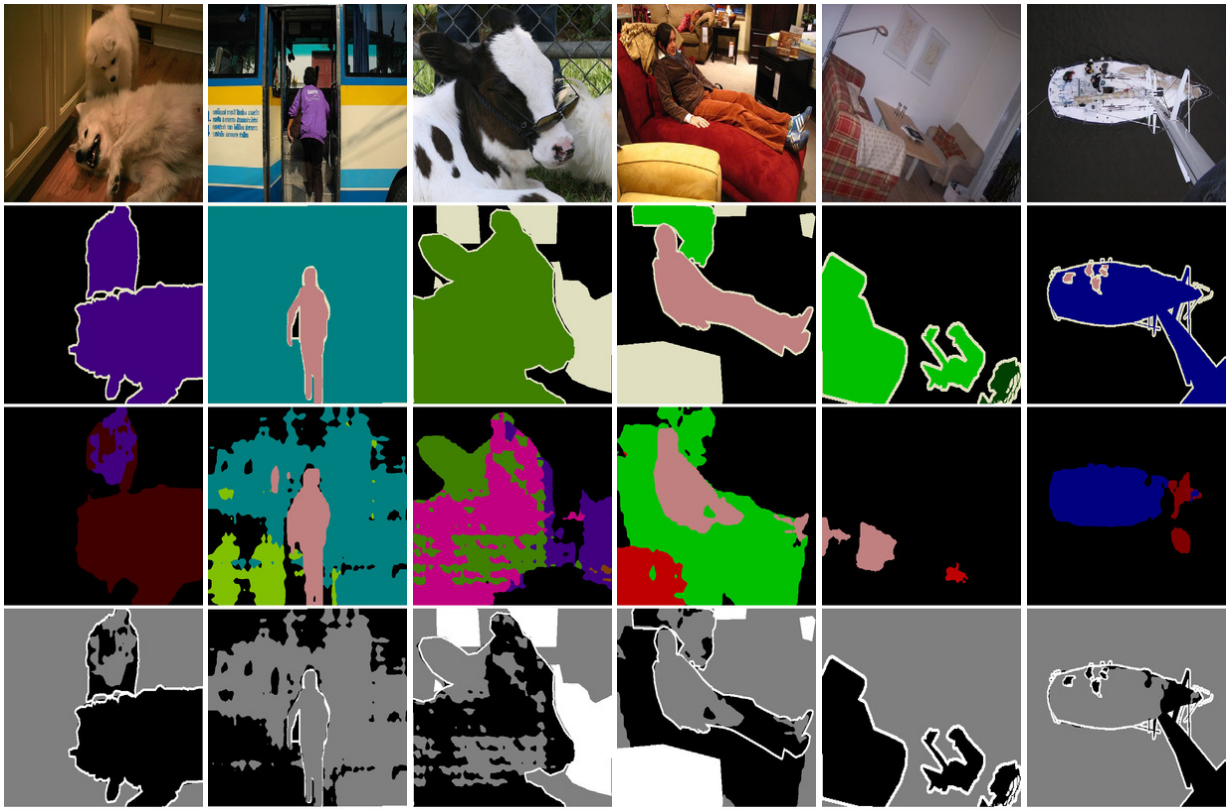

Figure 7. Failure cases on the PASCAL VOC 2012 [8] val set. The model was trained using the train set augmented using SBD [11]. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

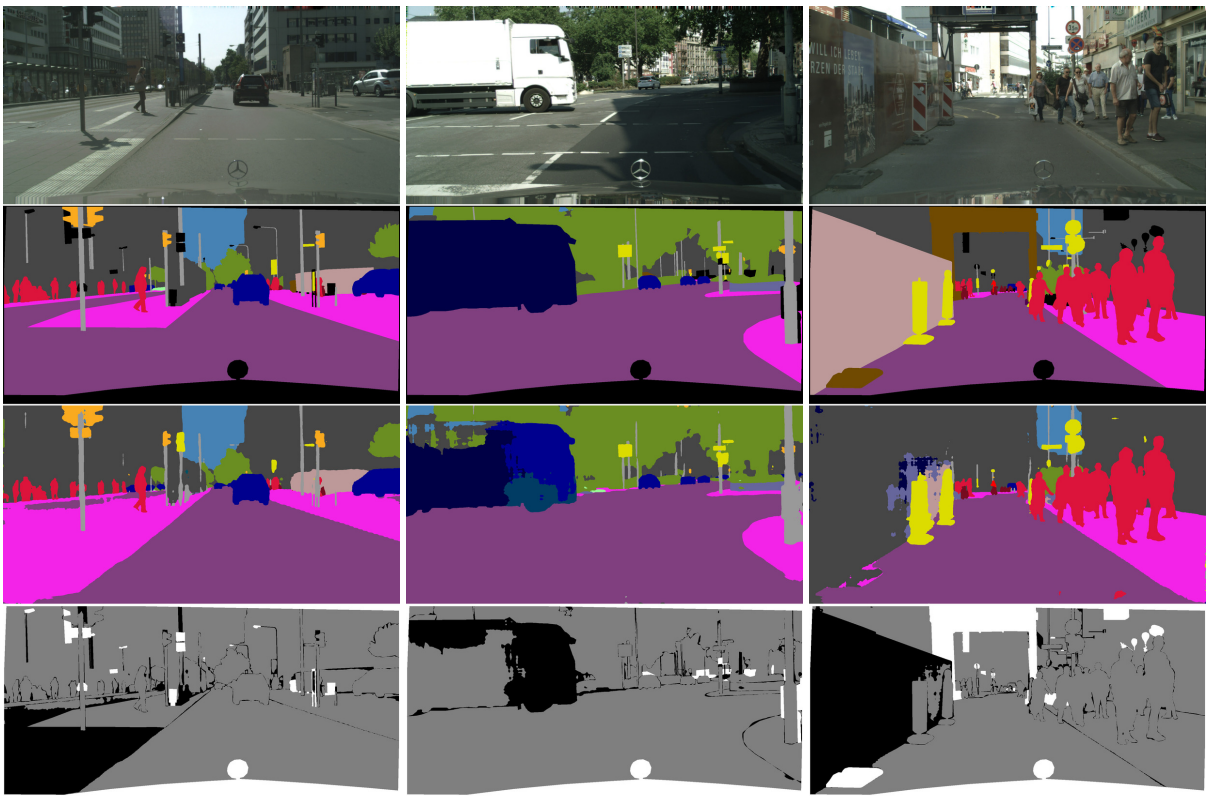

Figure 8. Failure cases on the Cityscapes [5] val set. The model was trained using the train set. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

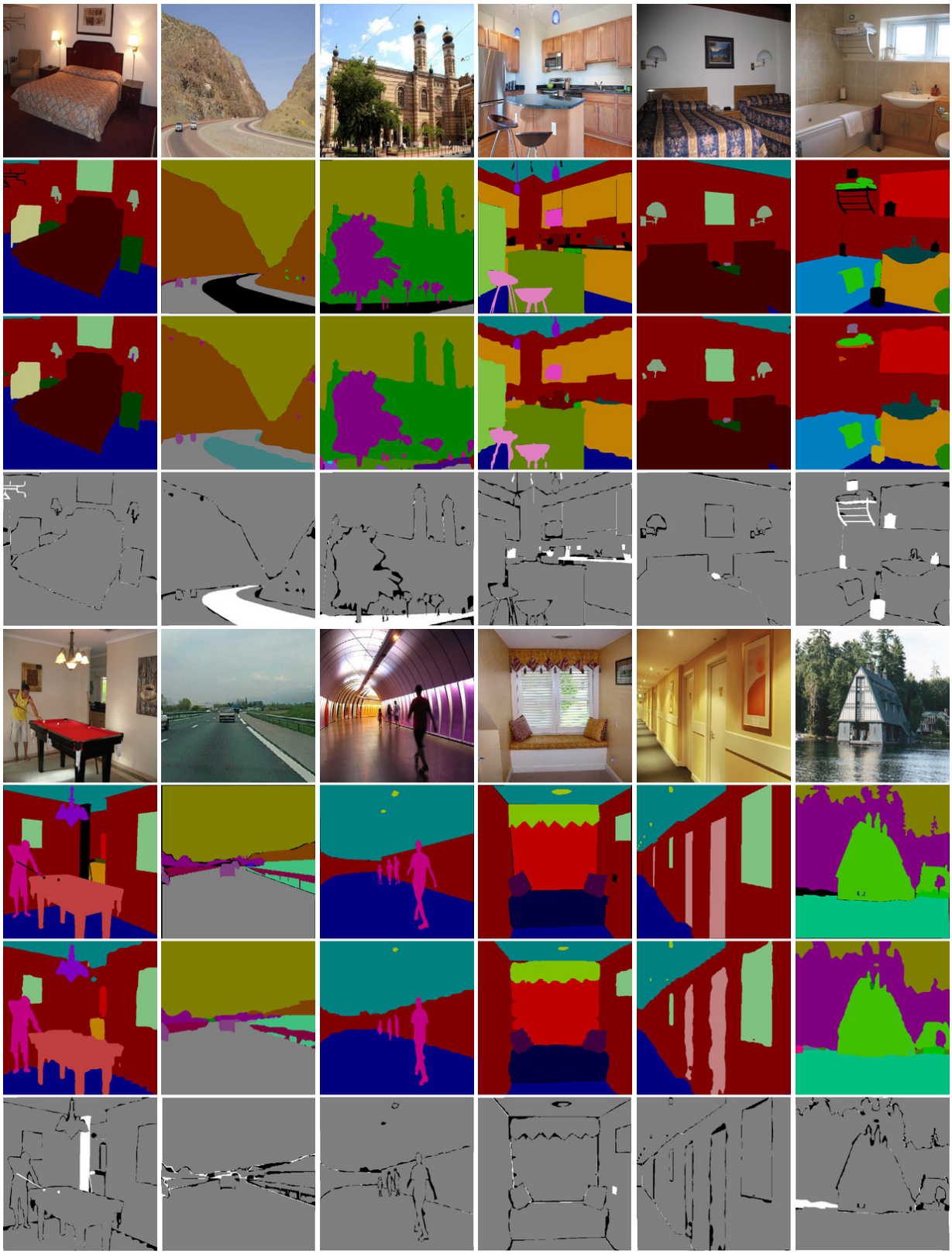

Figure 9. Qualitative results on the ADE20K [40] val set. The model was trained using the train set. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

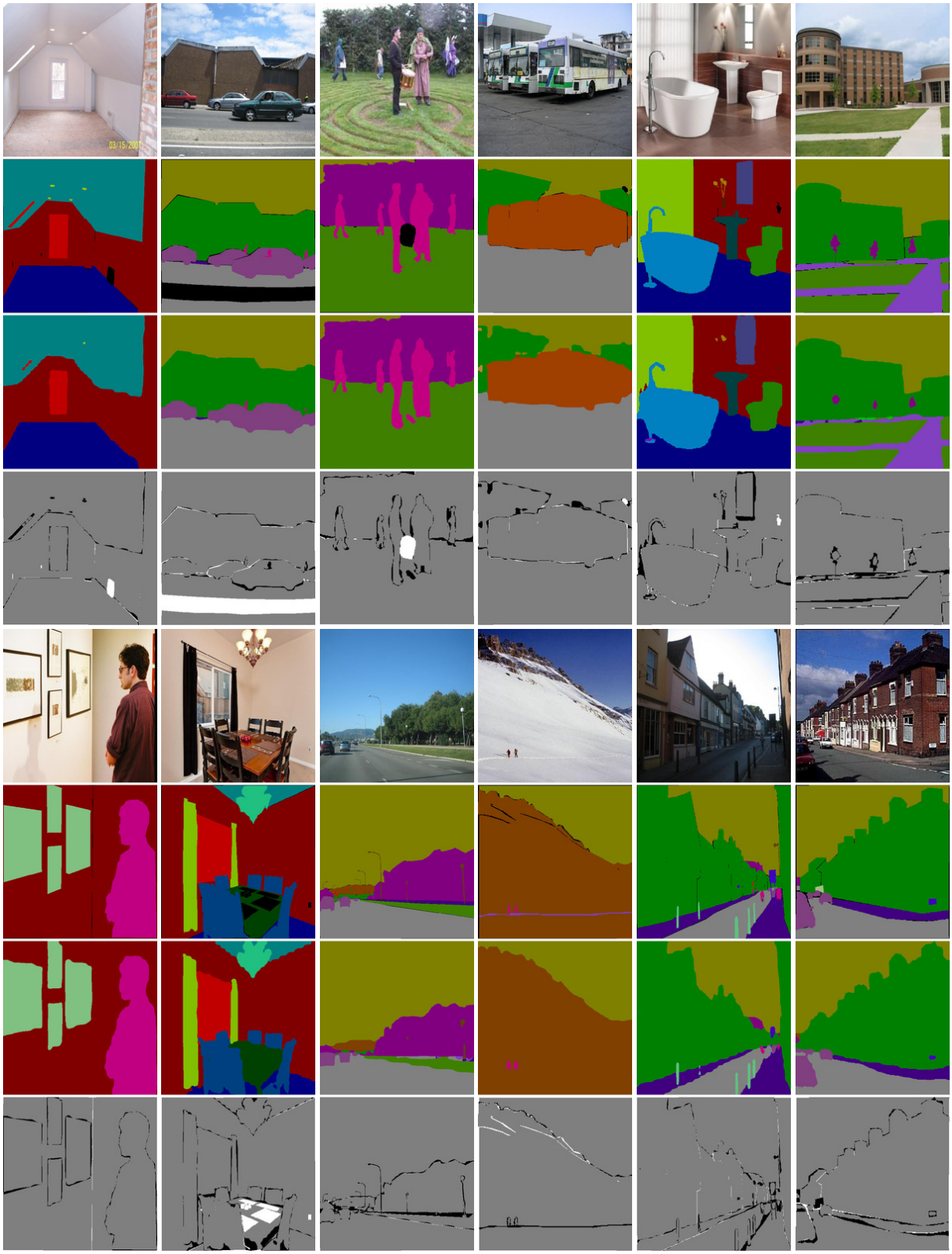

Figure 10. More qualitative results on the ADE20K [40] val set. The model was trained using the train set. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

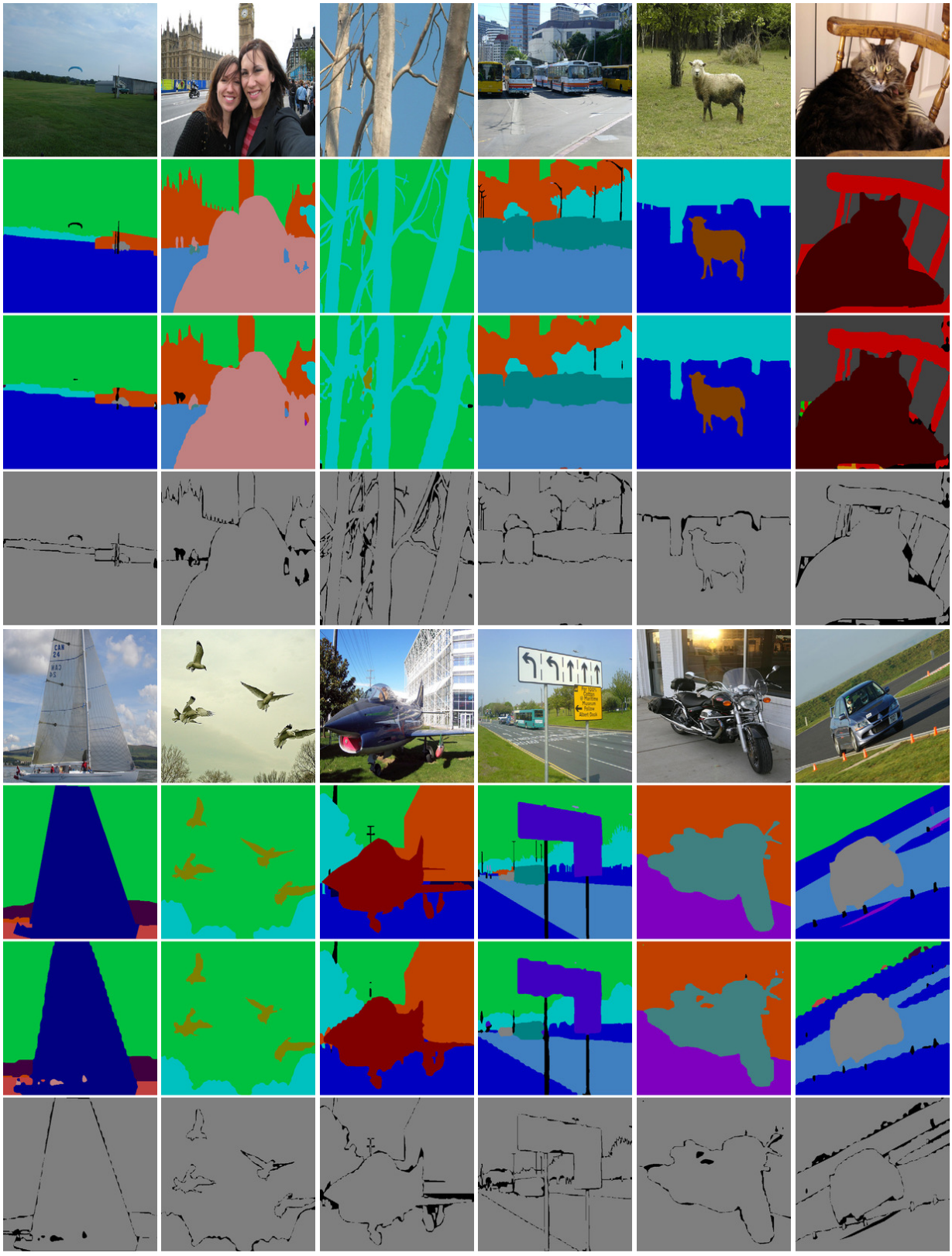

Figure 11. Qualitative results on the PASCAL Context [24] val set. The model was trained using the train set. In each example, from top to bottom, there are in turn the original image, the ground-truth, the predicted label, and the difference map between the ground-truth and the predicted label.

References

[1] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.*, 5(2):157–166, Mar. 1994. 4

[2] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *Proc. Int. Conf. Learn. Representations*, 2015. 6, 7

[3] 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. 2, 3, 6, 8, 9

[4] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv:1512.01274, 2015. 6, 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 *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016. 2, 8, 11, 13, 14

[6] J. Dai, K. He, and J. Sun. BoxSup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015. 9

[7] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016. 2

[8] M. Everingham, S. Eslami, L. van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes challenge: A retrospective. *Int. J. Comput. Vision*, 2014. 2, 7, 8, 9, 10, 11, 12, 14

[9] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. arXiv:1605.02264, 2016. 8

[10] S. Gross and M. Wilber. Training and investigating residual nets. http://torch.ch/blog/2016/02/04/resnets.html, 2016. 6

[11] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2011. 7, 12, 14

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016. 2, 3, 5, 6, 7, 8, 9

[13] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. arXiv:1603.05027, 2016. 2, 3, 5, 7, 8

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

[15] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 2

[16] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Proc. Advances in Neural Inf. Process. Syst.*, 2012. 2, 5, 6

[17] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. arXiv:1605.07648, 2016. 4

[18] G. Lin, C. Shen, A. van den Hengel, and I. Reid. Exploring context with deep structured models for semantic segmentation. arXiv:1603.03183, 2016. 6, 8, 9

[19] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. R. P. Dollár, and C. Zitnick. Microsoft COCO: Common objects in context. In *Proc. Eur. Conf. Comp. Vis.*, 2014. 8

[20] E. Littwin and L. Wolf. The loss surface of residual networks: Ensembles and the role of batch normalization. arXiv:1611.02525, 2016. 4

[21] Z. Liu, X. Li, P. Luo, C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015. 9

[22] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015. 3, 6, 7, 9

[23] D. Mishkin, N. Sergievskiy, and J. Matas. Systematic evaluation of CNN advances on the ImageNet. arXiv:1606.02228, 2016. 6, 7

[24] R. Mottaghi, X. Chen, X. Liu, N. Cho, S. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014. 2, 9, 11, 17

[25] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. Int. Conf. Mach. Learn.*, 2010. 3

[26] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015. 9

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vision*, 115(3):211–252, 2015. 2, 3, 5, 6, 7, 8, 10

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2014. 2, 5, 6, 7, 8, 9

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):19291958, Jan. 2014. 5, 6

[30] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, Inception-Resnet and the impact of residual connections on learning. arXiv:1602.07261, 2016. 2, 3, 4, 5, 7

[31] A. Veit, M. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow networks. arXiv:1605.06431, 2016. 2, 3, 4, 5, 10

[32] J. Wang, Z. Wei, T. Zhang, and W. Zeng. Deeply-fused nets. arXiv:1605.07716, 2016. 4

[33] J. Wu, C.-W. Xie, and J.-H. Luo. Dense CNN learning with equivalent mappings. arXiv:1605.07251, 2016. 6

[34] Z. Wu, C. Shen, and A. van den Hengel. Bridging category-level and instance-level semantic image segmentation. arXiv:1605.06885, 2016. 9

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

[36] S. Zagoruyko and N. Komodakis. Wide residual networks. arXiv:1605.07146, 2016. 2, 5, 7

[37] H. Zhao, J. Shi, X. Qi, X. Wang, T. Xiao, and J. Jia. Understanding scene in the wild. http://image-net.org/challenges/talks/2016/SenseCUSceneParsing.pdf, 2016. 2, 6

[38] 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. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015. 9

[39] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. arXiv:1610.02055, 2016. 7, 8

[40] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through ADE20K dataset. arXiv:1608.05442, 2016. 6, 8, 9, 11, 15, 16