RefineNet: Multi-Path Refinement Networks for Dense Prediction

Guosheng Lin, Fayao Liu, Anton Milan, Chunhua Shen, Ian Reid

Abstract—Recently, very deep convolutional neural networks (CNNs) have shown outstanding performance in object recognition and have also been the first choice for dense prediction problems such as semantic segmentation and depth estimation. However, repeated subsampling operations like pooling or convolution striding in deep CNNs lead to a significant decrease in the initial image resolution. Here, we present *RefineNet*, a generic multi-path refinement network that explicitly exploits all the information available along the down-sampling process to enable high-resolution prediction using long-range residual connections. In this way, the deeper layers that capture high-level semantic features can be directly refined using fine-grained features from earlier convolutions. The individual components of RefineNet employ residual connections following the identity mapping mindset, which allows for effective end-to-end training. Further, we introduce chained residual pooling, which captures rich background context in an efficient manner. We carry out comprehensive experiments on semantic segmentation which is a dense classification problem and achieve good performance on seven public datasets. We further apply our method for depth estimation and demonstrate the effectiveness of our method on dense regression problems.

Index Terms—Convolutional Neural Network, Semantic Segmentation, Object Parsing, Human Parsing, Scene Parsing, Depth Estimation, Dense Prediction.

1 Introduction

Dense prediction, also known as pixel-wise prediction problems, is a fundamental category of computer vision topics. The dense prediction task here is to assign a class label or continuous value to every single pixel in the given image. A variety of vision tasks can be formulated as dense prediction problems, either discrete or continuous value prediction. For instance, semantic segmentation and object parsing are typical dense classification problems, while depth estimation from monocular images is a representative dense regression problem.

Semantic segmentation is a crucial component in image understanding. The task here is to assign a unique label (or category) to every single pixel in the image, which is considered as a dense classification problem. The related problem of so-called object parsing, which aims to segment and recognize the parts of an object, can usually be cast as a semantic segmentation task. Depth estimation from single monocular images is to predict the pixel-wise depth values (continuous real values) of a single image, which can be formulated as a dense regression problem. It has found wide applications in 3D reconstruction, visual recognition, scene understanding, autonomous driving etc..

Recently, deep learning methods, and in particular convolutional neural networks (CNNs), e.g., VGG [52], Residual Net [25], have shown remarkable results in recognition tasks.

- *G. Lin is with School of Computer Science and Engineering, Nanyang Technological University, Singapore.*
- *A. Milan is with Amazon Core ML Berlin, Germany.*
- *C.Shen, I.Reid are with the ARC Centre of Excellence for Robotic Vision, School of Computer Science, The University of Adelaide, Australia.*
- *Corresponding author: Fayao Liu (fayaoliu@gmail.com).*

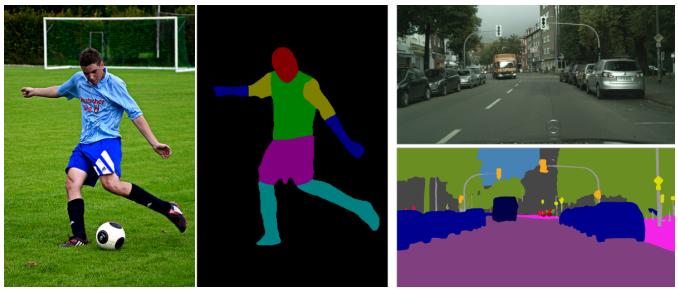


Fig. 1 – Example results of our method on the task of object parsing *(left)* and semantic segmentation *(right)*.

However, these approaches exhibit clear limitations when it comes to dense prediction in tasks like dense depth or normal estimation [13, 40, 41] and semantic segmentation [43, 5]. Multiple stages of spatial pooling and convolution strides reduce the final output prediction typically by a factor of 32 in each dimension, thereby losing much of the finer image structure. How to apply the most advanced CNNs methods, e.g., VGG [52], Residual Net [25], for high-resolution dense prediction has become a hot topic in the community.

One way to address this limitation is to learn deconvolutional filters as an up-sampling operation [45, 43] to generate high-resolution feature maps. The deconvolution operations are not able to recover the low-level visual features which are lost after the down-sampling operation in the convolution forward stage. Therefore, they are unable to output accurate high-resolution prediction. Low-level visual information is essential for accurate prediction on the boundaries or details. The method DeepLab recently proposed by Chen et al. [6] employs atrous (or dilated) convolutions to account for larger receptive fields without downscaling the image. DeepLab is

widely applied and represents state-of-the-art performance on semantic segmentation. This strategy, although successful, has at least two limitations. First, it needs to perform convolutions on a large number of detailed (high-resolution) feature maps that usually have high-dimensional features, which are computationally expensive. Moreover, a large number of high-dimensional and high-resolution feature maps also require huge GPU memory resources, especially in the training stage. This hampers the computation of high-resolution predictions and usually limits the output size to $1/8$ of the original input. Second, dilated convolutions introduce a coarse sub-sampling of features, which potentially leads to a loss of important details.

Another type of methods exploits features from intermediate layers for generating high-resolution prediction, e.g., the FCN method in [43] and Hypercolumns in [22]. The intuition behind these works is that features from middle layers are expected to describe mid-level representations for object parts, while retaining spatial information. This information is though to be complementary to the features from early convolution layers which encode low-level spatial visual information like edges, corners, circles, etc., and also complementary to high-level features from deeper layers which encode high-level semantic information, including object- or category-level evidence, but which lack strong spatial information.

We argue that features from all levels are helpful for semantic segmentation. High-level semantic features helps the category recognition of image regions, while low-level visual features help to generate sharp, detailed boundaries for high-resolution prediction. How to effectively exploit middle layer features remains an open question and deserves more attentions. To this end we propose a novel network architecture which effectively exploits multi-level features for generating high-resolution predictions.

Our main *contributions* are as follows:

1) We propose a multi-path refinement network (RefineNet) which exploits features at multiple levels of abstraction for high-resolution dense prediction. RefineNet refines low-resolution (coarse) features with fine-grained low-level features in a recursive manner to generate high-resolution feature maps. Our model is flexible in that it can be cascaded and modified in various ways.
2) Our cascaded RefineNets can be effectively trained end-to-end, which is crucial for good prediction performance. More specifically, all components in RefineNet employ residual connections [25] with identity mappings [26], such that gradients can be directly propagated through short-range *and* long-range residual connections allowing for both effective and efficient end-to-end training.
3) We propose a new network component we call "chained residual pooling" which is able to capture background context from a large image region. It does so by efficiently pooling features with multiple window sizes and fusing them together with residual connections and learnable weights.
4) The proposed RefineNet achieves excellent performance on several semantic segmentation datasets, including PASCAL VOC 2012, PASCAL-Context, NYUDv2, SUN-RGBD, Cityscapes, ADE20K, and the object parsing Person-Parts dataset. We further apply our method for depth estimation from a single image and achieve competitive performance.

Our preliminary result is published in [36]. To facilitate future research, we release both source code and trained models for our RefineNet.[1]

1.1 Related Work

We focus on semantic segmentation as a typical dense classification problem and depth estimation as a representative dense continuous value prediction problem. We next review the works most related to these two tasks.

Semantic segmentation

CNNs become the most successful methods for semantic segmentation in recent years. The early methods in [18, 23] are region-proposal-based methods which classify region proposals to generate segmentation results. Recently fully convolution network (FCNNs) based based methods [43, 5, 10] show effective feature generation and end-to-end training, and thus become the most popular choice for semantic segmentation. FCNNs have also been widely applied in other dense-prediction tasks, e.g., depth estimation [15, 13, 40], image restoration [14], image super-resolution [12]. The proposed method here is also based on fully convolution-style networks.

FCNN based methods usually have the limitation of down-resolution prediction. There are a number of proposed techniques which addressed this limitation and aim to generate high-resolution predictions. The atrous convolution based approach DeepLab-CRF in [5] directly output a middle-resolution score map then applies the dense CRF method [30] to refine boundaries by leveraging color contrast information. CRF-RNN [59] extends this approach by implementing recurrent layers for end-to-end learning of the dense CRF and FCNN. Deconvolution methods [45, 2] learn deconvolution layers to up-sample the low-resolution predictions. The depth estimation method [41] employs super-pixel pooling to output high-resolution continuous predictions.

There are several existing methods which exploit middle layer features for segmentation. The FCN method in [43] adds prediction layers to middle layers to generate prediction scores at multiple resolutions. They average the multi-resolution scores to generate the final prediction mask. Their system is trained in a stage-wise manner rather than end-to-end training. The method Hypercolumn [22] merges features from middle layers and learns dense classification layers. Their method employs stage-wise training instead of end-to-end training. The method Seg-Net [2] and U-Net [49] apply skip-connections in the deconvolution architecture to exploit the features from middle layers. Exploring middle level features and performing multi-path fusion are also applied for object detection tasks [58].

Although there are a few existing work, how to effectively exploit middle layer features remains an open question. We

1. Our source code is available at https://github.com/guosheng/refinenet

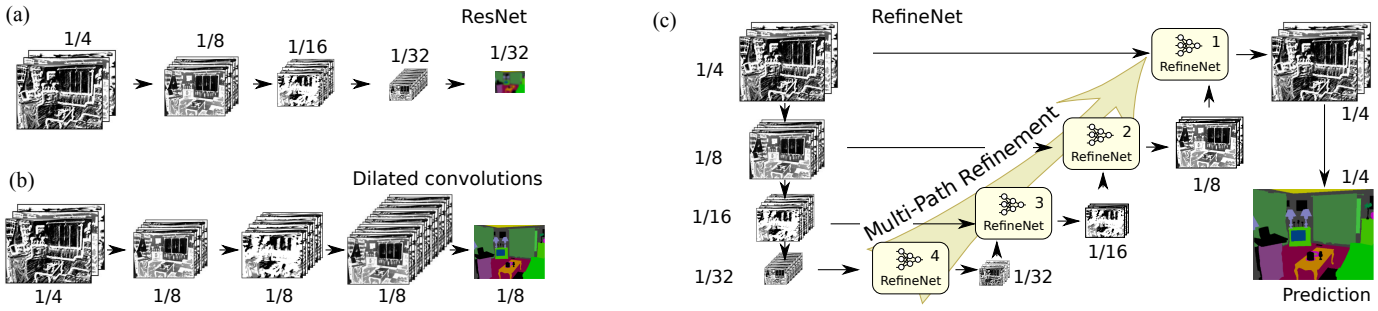


Fig. 2 – Comparison of fully convolutional approaches for dense classification. Standard multi-layer CNNs, such as ResNet *(a)* suffer from downscaling of the feature maps, thereby losing fine structures along the way. Dilated convolutions *(b)* remedy this shortcoming by introducing atrous filters, but are computationally expensive to train and quickly reach memory limits even on modern GPUs. Our proposed architecture that we call RefineNet *(c)* exploits various levels of detail at different stages of convolutions and fuses them to obtain a high-resolution prediction without the need to maintain large intermediate feature maps. The details of the RefineNet block are outlined in Sec. 3 and illustrated in Fig 3.

propose a novel network architecture, RefineNet, to address this question. The network architecture of RefineNet is clearly different from existing methods. RefineNet consists of a number of specially designed components which are able to refine the coarse high-level semantic features by exploiting low-level visual features. In particular, RefineNet employs short-range and long-range residual connections with identity mappings which enable effective end-to-end training of the whole system, and thus help to archive good performance. Comprehensive empirical results clearly verify the effectiveness of our novel network architecture for exploiting middle layer features. Our work can be further applied for instance segmentation and object detection [24, 46, 47, 31, 58] to generate high-resolution semantic feature maps and thus to improve the instance segmentation performance.

Depth estimation

Recently, deep learning methods have been widely applied to depth estimations and have now become state-of-the-art methods for this task. The pioneering work can be attributed to [15, 40]. In [15], Eigen et al. proposed a multi-scale CNN approach for depth estimation. They directly regress the depth values of the convolution maps to the ground-truth depth maps using basic CNN networks. Later on, they extend their work to [13] by proposing a unified multi-scale CNN architecture for depth, surface normal and semantic label prediction. Liu et al. [40] propose an end-to-end deep structured learning model that jointly exploits continuous CRFs and CNNs. Their method relies on graphical models and performs inference by network forward at the original image size, therefore does not have the down-resolution issue. However, it requires a super-pixel over-segmentation procedure beforehand, which limits its flexibility. In [50], Roy et al. propose a deep architecture, i.e., neural regression forest, by combining deep CNNs with regression forests for estimating depths from single images. Laina et al. [33] proposes a fully convolutional architecture exploiting the recent CNN architecture, i.e., residual net [26], and introduce the reverse Huber loss function for depth estimation.

We directly apply our method for dense depth regression without considering any domain knowledge or special design for depth estimation. Our network architecture for this task is completely the same as that for semantic segmentation. We demonstrate our general framework for high-resolution dense prediction is able to achieve very competitive results for depth estimation.

2 Background

Before presenting our approach, we first review the structure of fully convolutional networks for semantic segmentation [43] in more detail and also discuss the recent dilated convolution technique [6] which is specifically designed to generate high-resolution predictions.

Very deep CNNs have shown outstanding performance on object recognition problems. Specifically, the recently proposed Residual Net (ResNet) [25] has shown step-change improvements over earlier architectures, and ResNet models pre-trained for ImageNet recognition tasks are publicly available. Because of this, in the following we adopt ResNet as our fundamental building block for semantic segmentation. Note, however, that replacing it with any other deep network is straightforward.

Since semantic segmentation can be cast as a dense classification problem, the ResNet model can be easily modified for this task. This is achieved by replacing the single label prediction layer with a dense prediction layer that outputs the classification confidence for each class at every pixel. This approach is illustrated in Fig. 2(a). As can be seen, during the forward pass in ResNet, the resolution of the feature maps (layer outputs) is decreased, while the feature depth, i.e. the number of feature maps per layer (or *channels*) is increased. The former is caused by striding during convolutional and pooling operations.

The ResNet layers can be naturally divided into 4 blocks according to the resolution of the output feature maps, as shown in Fig. 2(a). Typically, the stride is set to 2, thus reducing the feature map resolution to one half when passing from one block to the next. This sequential sub-sampling has two effects: first it increases the receptive field of convolutions at deeper levels, enabling the filters to capture more global and contextual information which is essential for high quality classification; second it is necessary to keep the training efficient and tractable because each layer comprises a large number of filters and therefore produces an output which has

a corresponding number of channels, thus there is a trade-off between the number of channels and resolution of the feature maps. Typically the final feature map output ends up being 32 times smaller in each spatial dimension than the original image (but with 1000s of channels). This low-resolution feature map loses important visual details captured by early low-level filters, resulting in a rather coarse segmentation map. This issue is a well-known limitation of deep CNN-based segmentation methods.

An alternative approach to avoid lowering the resolution while retaining a large receptive field is to use dilated (atrous) convolution. This method introduced in [6], has the state-of-the-art performance on semantic segmentation. The sub-sampling operations are removed (the stride is changed from 2 to 1), and all convolution layers after the first block use dilated convolution. Such a dilated convolution (effectively a sub-sampled convolution kernel) has the effect of increasing the receptive field size of the filters without increasing the number of weights that must be learned (see illustration in Fig. 2(b)). Even so, there is a significant cost in memory, because unlike the image sub-sampling methods, one must retain very large numbers of feature maps at higher resolution. For example, if we retain all channels in all layers to be at least 1/4 of the original image resolution, and consider a typical number of filter channels to be 1024, then we can see that the memory capacity of even high-end GPUs is quickly swamped by very deep networks. In practice, therefore, dilation convolution methods usually have a resolution prediction of no more than 1/8 size of the original rather than 1/4, when using a deep network.

In contrast to dilated convolution methods, in this paper we propose a means to enjoy both the memory and computational benefits, while still able to produce effective and efficient high-resolution segmentation prediction, as described in the following section.

3 Proposed Method

We propose a new framework that provides multiple paths over which information from different resolutions and via potentially long-range connections, is assimilated using a generic building block, the RefineNet. Fig. 2(c) shows one possible arrangement of the building blocks to achieve our goal of high resolution semantic segmentation. We begin by describing the multi-path refinement arrangement in Sec. 3.1 followed by a detailed description of each RefineNet block in Sec. 3.2.

3.1 Multi-Path Refinement

As noted previously, we aim to exploit multi-level features for high-resolution prediction with long-range residual connections. RefineNet provides a generic means to fuse coarse high-level semantic features with finer-grained low-level features to generate high-resolution semantic feature maps. A crucial aspect of the design ensures that the gradient can be effortlessly propagated backwards through the network all the way to early low-level layers over long-range residual connections, ensuring that the entire network can be trained end-to-end.

For our standard multi-path architecture, we divide the pre-trained ResNet (trained with ImageNet) into 4 blocks according to the resolutions of the feature maps, and employ a 4-cascaded architecture with 4 RefineNet units, each of which directly connects to the output of one ResNet block as well as to the preceding RefineNet block in the cascade. Note, however, that such a design is not unique. In fact, our flexible architecture allows for a simple exploration of different variants. For example, a RefineNet block can accept input from multiple ResNet blocks. We will analyse a 2-cascaded version, a single-block approach as well as a 2-scale 7-path architecture later in Sec. 5.1.2.

We denote RefineNet-m as the RefineNet block that connects to the output of block-m in ResNet. In practice, each ResNet output is passed through one convolutional layer to adapt the dimensionality. Although all RefineNets share the same internal architecture, their parameters are not tied, allowing for a more flexible adaptation for individual levels of detail. Following the illustration in Fig. 2(c) bottom up, we start from the last block in ResNet, and connect the output of ResNet block-4 to RefineNet-4. Here, there is only one input for RefineNet-4, and RefineNet-4 serves as an extra set of convolutions which adapt the pre-trained ResNet weights to the task at hand, in our case, semantic segmentation. In the next stage, the output of RefineNet-4 and the ResNet block-3 are fed to RefineNet-3 as 2-path inputs. The goal of RefineNet-3 is to use the high-resolution features from ResNet block-3 to refine the low-resolution feature map output by RefineNet-4 in the previous stage. Similarly, RefineNet-2 and RefineNet-1 repeat this stage-wise refinement by fusing high-level information from the later layers and high-resolution but low-level features from the earlier ones. As the last step, the final high-resolution feature maps are fed to a dense soft-max layer to make the final prediction in the form of a dense score map. This score map is then up-sampled to match the original image using bilinear interpolation.

The entire network can be efficiently trained end-to-end. It is important to note that we introduce long-range residual connections between the blocks in ResNet and the RefineNet modules. During the forward pass, these long-range residual connections convey the low-level features that encode visual details for refining the coarse high-level feature maps. In the training step, the long-range residual connections allow direct gradient propagation to early convolution layers, which helps effective end-to-end training.

3.2 RefineNet

The architecture of one RefineNet block is illustrated in Fig. 3(a). In the multi-path overview shown in Fig 2(c), RefineNet-1 has one input path, while all other RefineNet blocks have two inputs. Note, however, that our architecture is generic and each Refine block can be easily modified to accept an arbitrary number of feature maps with arbitrary resolutions and depths.

Residual convolution unit. The first part of each RefineNet block consists of an adaptive convolution set that mainly fine-tunes the pretrained ResNet weights for our task. To that end,

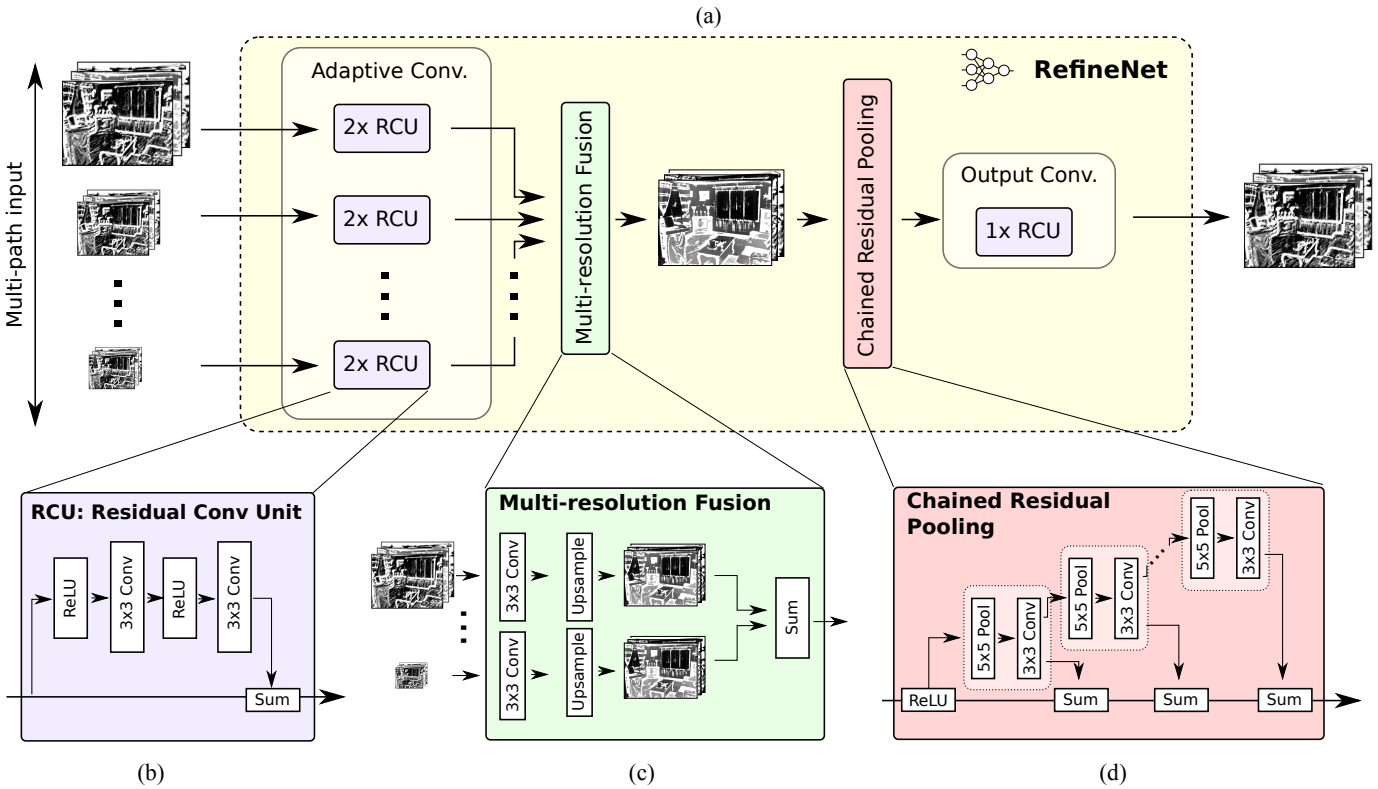


Fig. 3 – The individual components of our multi-path refinement network architecture RefineNet. Components in RefineNet employ residual connections with identity mappings. In this way, gradients can be directly propagated within RefineNet via local residual connections, and also directly propagate to the input paths via long-range residual connections, and thus we achieve effective end-to-end training of the whole system.

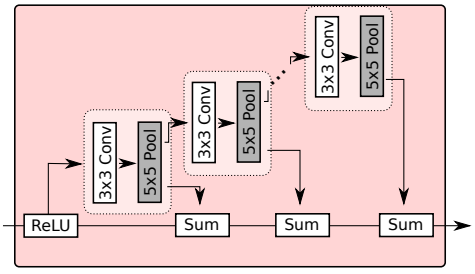


Fig. 4 – An alternative architecture for chained residual pooling (CRP). Compared to the architecture of CRP in Fig. 3 (d), the position of the pooling layer (marked in gray) and that of the convolution layer is exchanged.

each input path is passed sequentially through two residual convolution units (RCU), which is a simplified version of the convolution unit in the original ResNet [25], where the batch-normalization layers are removed (c.f. Fig. 3(b)). The filter number for each input path is set to 512 for RefineNet-4 and 256 for the remaining ones in our experiments.

Multi-resolution fusion. All path inputs are then fused into a high-resolution feature map by the multi-resolution fusion block, depicted in Fig. 3(c). This block first applies convolutions for input adaptation, which generate feature maps of the same feature dimension (the smallest one among the inputs), and then up-samples all (smaller) feature maps to the largest resolution of the inputs. Finally, all features maps are fused by summation. The input adaptation in this block also helps to re-scale the feature values appropriately along different paths, which is important for the subsequent sum-fusion. If there is only one input path (e.g., the case of RefineNet-4 in Fig. 2(c)), the input path will directly go through this block without changes.

Chained residual pooling. The output feature map then goes through the chained residual pooling block, schematically depicted in Fig. 3(d). The proposed chained residual pooling aims to capture background context from a large image region. It is able to efficiently pool features with multiple window sizes and fuse them together using learnable weights. In particular, this component is built as a chain of multiple pooling blocks, each consisting of one max-pooling layer and one convolution layer. One pooling block takes the output of the previous pooling block as input. Therefore, the current pooling block is able to re-use the result from the previous pooling operation and thus access the features from a large region without using a large pooling window. Using more pooling blocks usually gains better performance. In our experiment, our best reported results are based on a setting of 4 pooling blocks in one chained residual pooling module.

The output feature maps of all pooling blocks are fused together with the input feature map through summation of residual connections. Note that, our choice to employ residual connections also persists in this building block, which once

again facilitates gradient propagation during training. In one pooling block, each pooling operation is followed by convolutions which serve as a weighting layer for the summation fusion. It is expected that this convolution layer will learn to accommodate the importance of the pooling block during the training process.

We can also consider alternative architectures for our chained residual pooling block. Fig. 4 shows another architecture of chained residual pooling. This alternative architecture is modified from the architecture shown in Fig. 3(d) by exchanging the position of the convolution layer and the pooling layer in one pooling. This convolution layer will learn to adapt the input features and accommodate the importances of the input features before going through the pooling layer. In our observation, this alternative architecture may sometimes perform slightly better in some datasets compared to the original architecture.

Output convolutions. The final step of each RefineNet block is another residual convolution unit (RCU). This results in a sequence of three RCUs between each block. To reflect this behavior in the last RefineNet-1 block, we place two additional RCUs before the final softmax prediction step. The goal here is to employ non-linearity operations on the multi-path fused feature maps to generate features for further processing or for final prediction. The feature dimension remains the same after going through this block.

3.3 Identity Mappings in RefineNet

Note that all convolutional components of the RefineNet have been carefully constructed inspired by the idea behind residual connections and follow the rule of identity mapping [26]. This enables effective backward propagation of the gradient through RefineNet and facilitates end-to-end learning of cascaded multi-path refinement networks.

Employing residual connections with identity mappings allows the gradient to be directly propagated from one block to any other blocks, as was recently shown by [26]. This concept encourages to maintain a clean information path for shortcut connections, so that these connections are not "blocked" by any non-linear layers or components. Instead, non-linear operations are placed on branches of the main information path. We follow this guideline for developing the individual components in RefineNet, including all convolution units. It is this particular strategy that allows the multi-cascaded RefineNet to be trained effectively. Note that we include one non-linear activation layer (ReLU) in the chained residual pooling block. We observed that this ReLU is important for the effectiveness of subsequent pooling operations and it also makes the model less sensitive to changes in the learning rate. We observed that one single ReLU in each RefineNet block does not noticeably reduce the effectiveness of gradient flow.

We have both short-range and long-range residual connections in RefineNet. Short-range residual connections refer to local shot-cut connections in one RCU or the residual pooling component, while long-range residual connections refer to the connection between RefineNet modules and the ResNet blocks. With long-range residual connections, the gradient can be directly propagated to early convolution layers in ResNet and thus enables end-to-end training of all network components.

The fusion block fuses the information of multiple shortcut paths, which can be considered as performing summation fusion of multiple residual connections with necessary dimension or resolution adaptation. In this aspect, the role of the multi-resolution fusion block here is analogous to the role of the "summation" fusion in a conventional residual convolution unit in ResNet. There are certain layers in RefineNet, and in particular within the fusion block, that perform linear feature transformation operations, like linear feature dimension reduction or bilinear up-sampling. These layers are placed on the shortcut paths, which is similar to the case in ResNet [25]. As in in ResNet, when a shortcut connection crosses two blocks, it will include a convolution layer in the shortcut path for linear feature dimension adaptation, which ensures that the feature dimension matches the subsequent summation in the next block. Since only linear transformation are employed in these layers, gradients still can be propagated through these layers effectively.

4 RefineNet for Dense Prediction

We in this section present the training objectives for different dense prediction tasks, i.e., dense classification and dense regression.

4.1 Dense classification

We demonstrate our method for semantic segmentation which is a representative task of dense classification. We denote $\hat{\mathbf{Y}}_{ij}$ as the ground-truth labeling of the pixel indexed by (i, j), and $\mathbf{Z}_{ijk}$ as the output of the k-th channel and pixel indexed by (i, j) before the softmax layer. C is the total number of semantic categories. The softmax loss is then used for dense classification tasks:

$$\mathcal{L}(\mathbf{Z}, \hat{\mathbf{Y}}) = -\sum_{ij}\sum_{k=1}^{C} \mathbb{1}(\hat{\mathbf{Y}}_{ij} = k) \log \frac{\exp(\mathbf{Z}_{ijk})}{\sum_{l=1}^{C} \exp(\mathbf{Z}_{ijl})}. \quad (1)$$

Here, $\mathbb{1}(\cdot)$ is an indicator function which output 1 when the input statement is true and 0 otherwise.

4.2 Dense regression

Depth estimation from monocular images is to infer the pixel-wise depth values from a single RGB image. We here apply RefineNet for depth estimation to demonstrate the capacity of our method for dense regression problems predicting continuous values.

The proposed RefineNet can be easily adapted for continuous dense prediction tasks, with minimum modifications, i.e., change the softmax layer to a regression layer. We here apply the most commonly used least square loss, which is to minimize the squared Euclidean distance between the prediction map $\mathbf{Y}$ and the ground-truth $\hat{\mathbf{Y}}$.

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = \sum_{ij} (\mathbf{Y}_{ij} - \hat{\mathbf{Y}}_{ij})^2. \quad (2)$$

Here the output for one pixel is one-dimensional depth value instead of C-dimensional probabilities in the case of semantic segmentation.

5 Experiments

To show the effectiveness of our approach, we carry out comprehensive experiments on seven public datasets, which include six popular datasets for semantic segmentation on indoors and outdoors scenes (NYUDv2, PASCAL VOC 2012, SUN-RGBD, PASCAL-Context, Cityscapes, ADE20K MIT), and one dataset for object parsing called Person-Part. We also demonstrate depth estimation on the NYUDv2 dataset. The segmentation quality is measured by the intersection-over-union (IoU) score [16], the pixel accuracy and the mean accuracy [43] over all classes. As commonly done in the literature, we apply simple data augmentation during training. Specifically, we perform random scaling (ranging from 0.7 to 1.3), random cropping and horizontal flipping of the images. If not further specified, we apply test-time multi-scale evaluation, which is a common practice in segmentation methods [10, 6]. For multi-scale evaluation, we average the predictions on the same image across different scales for the final prediction. Following the work in [25] for transferring pre-trained ResNets, the mean and variance parameters in the batch normalization layers of the pre-trained ResNet are frozen and thus are not updated in the training process of our RefineNet. We also present an ablation experiment to inspect the impact of various components and alternative architectures of our model. Our system is built on MatConvNet [54].

5.1 Ablation study

5.1.1 Analysis of components in RefineNet

In Table 1, we present an ablation experiment to quantify the influence of the following components: Network depth, chained residual pooling and multi-scale evaluation (Msc Eva), as described earlier. This experiment shows that each of these three factors consistently improve the performance as measured by IoU. Our chained pooling significantly improves the performance, and basically using more pooling blocks helps to achieve better result. The best result is achieved by using 4 pooling blocks in our chained residual pooling module. In the subsequent experiments, we indicate this setting as "Pool4". For the setting of using 4 pooling blocks, we use the alternative architecture of chained pooling in Fig. 4. Generally deeper initialization networks, e.g., ResNet-101 and ResNet-152, lead to better performance.

5.1.2 Variants of cascaded RefineNet

As discussed earlier, our RefineNet is flexible in that it can be cascaded in various manners for generating various architectures. Here, we discuss several variants of our RefineNet. Specifically, we present the architectures of using a single RefineNet, a 2-cascaded RefineNet and a 4-cascaded RefineNet with 2-scale ResNet. The architectures of all three variants are illustrated in Fig. 5. The architecture of 4-cascaded RefineNet is already presented in Fig. 2(c). Please note that this 4-cascaded RefineNet model is the one used in all other experiments. Chained pooling with 2 pooling blocks are applied in this experiment.

The single RefineNet model is the simplest variant of our network. It consists of only one single RefineNet block, which takes all four inputs from the four blocks of ResNet and fuses all-resolution feature maps in a single process. The 2-cascaded version is similar our main model (4-cascaded) from Fig. 2(c), but employs only two RefineNet modules instead of four. The bottom one, RefineNet-2, has two inputs from ResNet blocks 3 and 4, and the other one has three inputs, two coming from the remaining ResNet blocks and one from RefineNet-2. For the 2-scale model in Fig. 5(c), we use 2 scales of the image as input and respectively 2 ResNets to generate feature maps; the input image is scaled to a factor of 1.2 and 0.6 and fed into 2 independent ResNets.

The evaluation results of these variants on the NYUD dataset are shown in Table 2. This experiment demonstrates that the 4-cascaded version yields better performance than the 2-cascaded and 1-cascaded version, and using 2-scale image input with 2 ResNet is better than using 1-scale input. This is expected due to the larger capacity of the network. However, it also results in longer training times. Hence, we resort to using the single-scale 4-cascaded version as the standard architecture in all our experiments.

5.1.3 Memory and computation analysis

In this section we discuss the feature map memory consumption and computational cost in terms of FLOPs of our method and compare with dilated convolution based approaches. Here we use 4-cascased RefineNets with the setting of using 4 chaining blocks in our residual chained pooling module for discussion. Results[2] are shown in Table 3. We compare our RefineNet with dilated convolution approaches. The comparing method denoted by "Dilated ResNet" is a Residual Net model with the dilated (atrous) convolution setting [6] for generating high-resolution feature maps. The column "Output resolution ratio indicates the ratio of output feature map size to the input image size. It shows that our method is both memory and computation efficient for generating high-resolution prediction (e.g., 1/4 of input image size) compared to dilated convolution based approaches. Particularly, our memory consumption is much lower than comparing methods.The efficiency benefits of our method is especially significant when using very deep base networks (e.g., ResNet-152).

A detailed breakdown analysis of memory usage and computational cost for our RefineNet is shown in Table 4. The input image size is 512 x 512. It shows that the memory consumption and the computational cost of our RefineNet are constant and they are irrelevant to the choice of base networks. Hence our RefineNet are much more efficient than the dilated (atrous) convolution based approaches when incorporating with very deep base networks for high-resolution prediction.

5.2 Object Parsing

In this section we present our results on the task of object parsing, which consists of recognizing and segmenting object

2. The results of DeepLabV2 are generated from https://github.com/albanie/convnet-burden

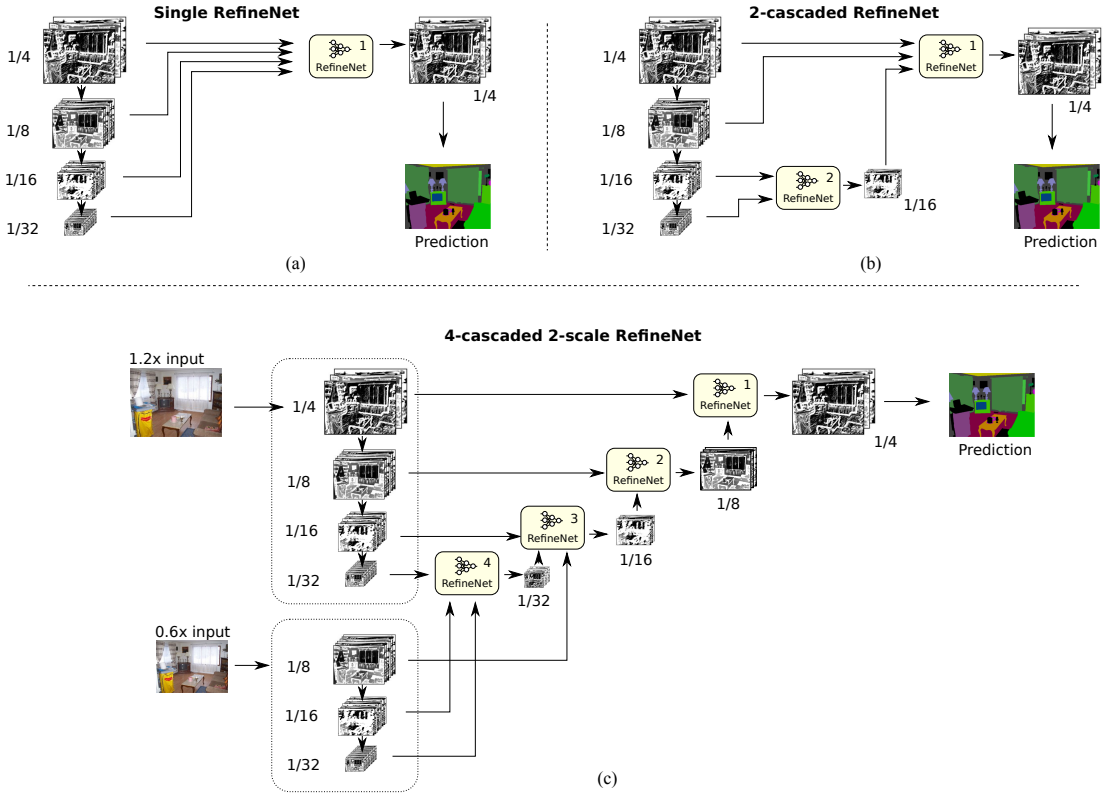


Fig. 5 – Illustration of 3 variants of our network architecture: (a) single RefineNet, (b) 2-cascaded RefineNet and (c) 4-cascaded RefineNet with 2-scale ResNet. Note that our proposed RefineNet block can seamlessly handle different numbers of inputs of arbitrary resolutions and dimensions without any modification.

TABLE 1 – Ablation experiments of our RefineNet on NYUDv2. We analyze the contribution of different components in our method. Our chained pooling significantly improves the performance, and using more pooling blocks helps to achieve better result. Generally deeper initialization networks lead to better performance. "Msc Eva" indicates test time multi-scale evaluation.

Initialization	Chained pool.	# Pooling blocks	Msc Eva	IoU
ResNet-50	no	0	no	40.4
ResNet-50	yes	2	no	42.5
ResNet-50	yes	2	yes	43.8
ResNet-101	yes	2	no	43.6
ResNet-101	yes	2	yes	44.7
ResNet-152	yes	2	yes	46.5
ResNet-101	yes	4	yes	46.4
ResNet-152	yes	4	yes	**47.6**

TABLE 2 – Evaluations of 4 variants of cascaded RefineNet: single RefineNet, 2-cascaded RefineNet, 4-cascaded RefineNet, 4-cascaded RefineNet with 2-scale ResNet on the NYUDv2 dataset. We use chained pooling with 2 pooling blocks in this experiment. The 4-cascaded version are used as our main architecture throughout all experiments in the paper because this turns out to be the best compromise between accuracy and efficiency.

Variant	Initialization	Msc Eva	IoU
single RefineNet	ResNet-50	no	40.3
2-cascaded RefineNet	ResNet-50	no	40.9
4-cascaded RefineNet	ResNet-50	no	42.5
4-cascaded 2-scale RefineNet	ResNet-50	no	43.1

parts. We carry out experiments on the Person-Part dataset [8, 7] which provides pixel-level labels for six person parts including Head, Torso, Upper/Lower Arms and Upper/Lower Legs. The rest of each image is considered background. There are training 1717 images and 1818 test images. We use four pooling blocks in our chained residual pooling module, which is indicated by "Pool4" in the result entry.

We compare our results to a number of state-of-the-art methods, listed in Table 5. The results clearly demonstrate the improvement over previous works. In particular, we significantly outperform the the recent DeepLab-v2 approach [6] which is based on dilated convolutions for high-resolution segmentation, using the same ResNet as initialization. Qualitative examples of our object parsing on this dataset are shown in Fig.6.

TABLE 3 – Memory and computation analysis. We compare our RefineNet with dilated convolution approaches for generating high-resolution feature maps. The column "Output resolution ratio indicates the ratio of output feature map size to the input image size. It shows that our method is both memory and computation efficient for generating high-resolution prediction (e.g., 1/4 of input image size) compared to dilated convolution based approaches. Particularly, our memory consumption is much lower than comparing methods. The efficiency benefits of our method is especially significant when using very deep base networks (e.g., ResNet-152).

Methods	Base network	Input image size	Output size ratio	Memory	FLOPs
Dilated ResNet	ResNet-101	512 x 512	1/8	3.3GB	177G
Dilated ResNet	ResNet-152	512 x 512	1/8	4.4GB	241G
Dilated ResNet	ResNet-101	512 x 512	1/4	11.6GB	695G
Dilated ResNet	ResNet-152	512 x 512	1/4	16.7GB	951G
DeepLabv2 (dilated) [6]	ResNet-101	513 x 513	1/8	4.0GB	346G
RefineNets+ResNet (ours)	ResNet-101	512 x 512	1/4	1.9GB	261G
RefineNets+ResNet (ours)	ResNet-152	512 x 512	1/4	2.3GB	280G

TABLE 4 – Memory and computation breakdown details of RefineNet. The input image size is 512 x 512. It shows that the memory consumption and the computational cost of our RefineNet are constant and they are irrelevant to the choice of base networks. Hence our RefineNet are much more efficient than the dilated (atrous) convolution based approaches when incorporating with very deep base networks for high-resolution prediction.

Methods	Base network	Feature memory breakdown	FLOPs breakdown
RefineNets+ResNet	ResNet-101	1.1GB (ResNet)+ 0.8GB (RefineNets) = 1.9GB	40G (ResNet) + 221G (RefineNets) = 261G
RefineNets+ResNet	ResNet-152	1.5GB (ResNet)+ 0.8GB (RefineNets) = 2.3GB	59G (ResNet) + 221G (RefineNets) = 280G

TABLE 5 – Object parsing results on the Person-Part dataset. Our method achieves the best performance (bold).

method	IoU
Attention [7]	56.4
HAZN [56]	57.5
LG-LSTM [35]	58.0
Graph-LSTM [34]	60.2
DeepLab [5]	62.8
DeepLab-v2 (Res101) [6]	64.9
RefineNet-Res101-Pool4	68.9
RefineNet-Res152-Pool4	**69.4**

TABLE 6 – Segmentation results on NYUDv2 (40 classes). Our RefineNet performs the best.

method	training data	pixel acc.	mean acc.	IoU
Gupta et al. [20]	RGB-D	60.3	-	28.6
FCN-32s [43]	RGB	60.0	42.2	29.2
FCN-HHA [43]	RGB-D	65.4	46.1	34.0
Context [37]	RGB	70.0	53.6	40.6
RefineNet-Res50-Pool2	RGB	72.2	55.8	43.8
RefineNet-Res101-Pool4	RGB	73.8	58.8	46.4
RefineNet-Res152-Pool4	RGB	74.4	59.6	**47.6**

5.3 Semantic Segmentation

We now describe our experiments on dense semantic labeling on six public benchmarks and show that our RefineNet outperforms previous methods on all datasets.

5.3.1 NYUDv2

The NYUDv2 dataset [51] consists of 1449 RGB-D images showing interior scenes. We use the segmentation labels provided in [19], in which all labels are mapped to 40 classes. We use the standard training/test split with 795 and 654 images, respectively. We train our models only on RGB images without using the depth information. Quantitative results are shown in Table 6. Our RefineNet achieves new state-of-the-art result on the NYUDv2 dataset.

Ablation experiments on the NYUDv2 dataset are shown in Table 1 to evaluate the effect of different settings. Once again, this study demonstrates the benefits of adding the proposed chained residual pooling component and deeper networks, both of which consistently improve the performance as measured by IoU.

5.3.2 PASCAL VOC 2012

PASCAL VOC 2012 [16] is a well-known segmentation dataset which includes 20 object categories and one background class. This dataset is split into a training set, a validation set and a test set, with 1464, 1449 and 1456 images each. Since the test set labels are not publicly available, all reported results have been obtained from the VOC evaluation server. Following the common convention [5, 6, 59, 42], the training set is augmented by additional annotated VOC images provided in [21] as well as with the training data from the MS COCO dataset [38]. We compare our RefineNet on the PASCAL VOC 2012 test set with a number of competitive methods, showing superior performance. Here we use 4 pooling blocks in our chained residual pooling module (indicated by "P4" in the result table). Kindly noted that there is *no* post CRF refinement for our method.

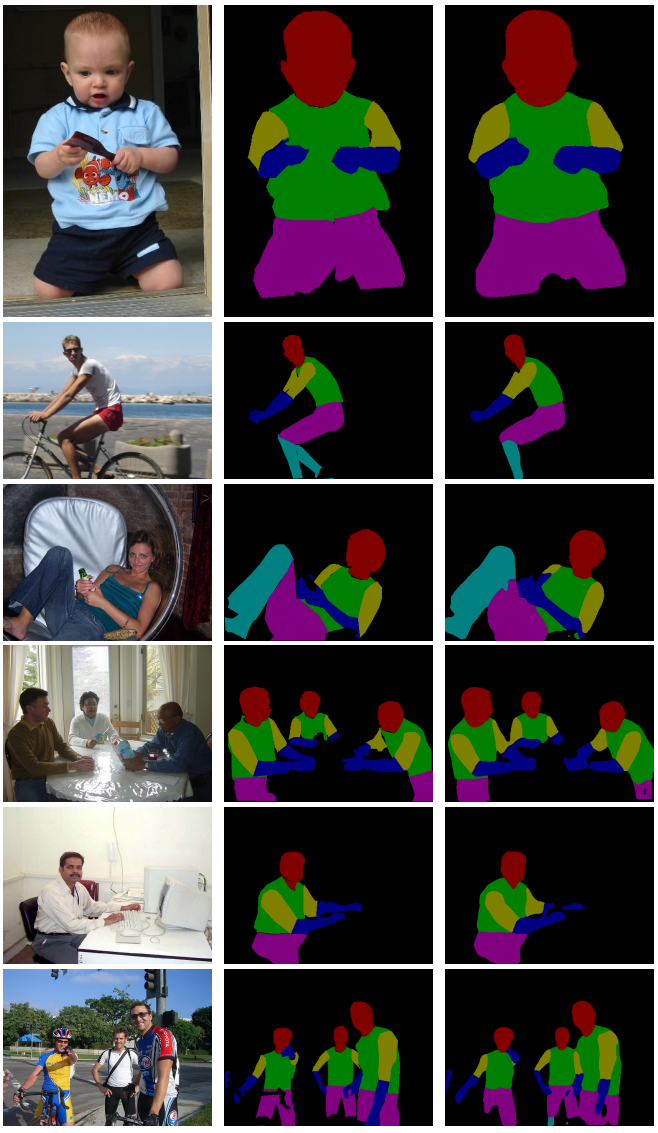
The detailed results for each category and the mean IoU scores are shown in Table 7. When using Res101 as initialization network We achieve an IoU score of 83.8, which sets a new state-of-the-art result on this challenging dataset. The result link to the VOC evaluation server can be found here [3] for RefineNet-Res101-P4 and here [4] for RefineNet-Res152-P4.

We outperform competing methods in almost all categories. In particular, we significantly outperform the method DeepLab-v2 [6] which is the currently best known dilation convolution method and uses the same ResNet-101 network

3. http://host.robots.ox.ac.uk:8080/anonymous/TGDPAW.html
4. http://host.robots.ox.ac.uk:8080/anonymous/ZGXPB5.html

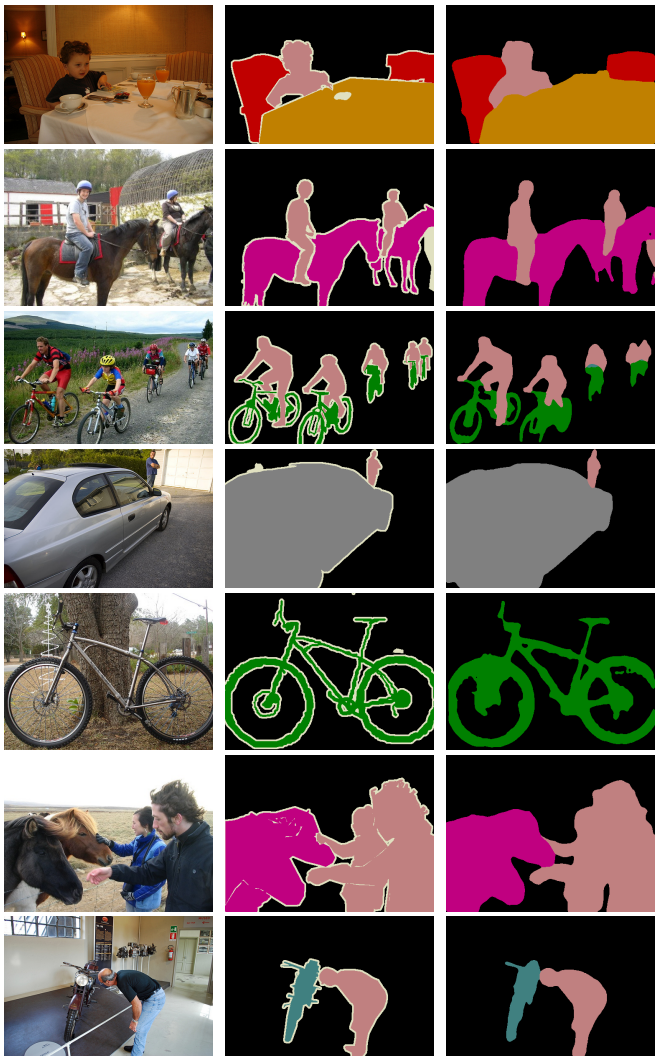
TABLE 7 – Results on the PASCAL VOC 2012 test set (IoU scores). Our RefineNet archives the best performance (IoU 83.8).

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	potted	sheep	sofa	train	tv	**mean**
FCN-8s [43]	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
DeconvNet [45]	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
CRF-RNN [59]	90.4	55.3	88.7	68.4	69.8	88.3	82.4	85.1	32.6	78.5	64.4	79.6	81.9	86.4	81.8	58.6	82.4	53.5	77.4	70.1	74.7
BoxSup [10]	89.8	38.0	89.2	68.9	68.0	89.6	83.0	87.7	34.4	83.6	67.1	81.5	83.7	85.2	83.5	58.6	84.9	55.8	81.2	70.7	75.2
DPN [42]	89.0	61.6	87.7	66.8	74.7	91.2	84.3	87.6	36.5	86.3	66.1	84.4	87.8	85.6	85.4	63.6	87.3	61.3	79.4	66.4	77.5
Context [37]	94.1	40.7	84.1	67.8	75.9	93.4	84.3	88.4	42.5	86.4	64.7	85.4	89.0	85.8	86.0	67.5	90.2	63.8	80.9	73.0	78.0
DeepLab [5]	89.1	38.3	88.1	63.3	69.7	87.1	83.1	85.0	29.3	76.5	56.5	79.8	77.9	85.8	82.4	57.4	84.3	54.9	80.5	64.1	72.7
DeepLab2-Res101 [6]	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
CSupelec-Res101 [4]	92.9	61.2	91.0	66.3	77.7	95.3	88.9	92.4	33.8	88.4	69.1	89.8	92.9	87.7	87.5	62.6	89.9	59.2	87.1	74.2	80.2
RefineNet-Res101-P4	94.8	65.1	93.6	77.3	80.9	96.5	89.7	91.6	39.3	92.9	73.9	89.8	93.8	87.7	87.5	69.1	90.5	65.0	86.3	78.6	82.9
RefineNet-Res152-P4	95.5	63.2	95.0	78.3	83.5	95.7	88.5	92.9	41.5	92.2	78.2	89.7	93.3	90.4	88.5	69.2	92.5	67.3	88.7	78.6	**83.8**



(a) Test Image *(b)* Ground Truth *(c)* Prediction

Fig. 6 – Our prediction examples on Person-Parts dataset.



(a) Test Image *(b)* Ground Truth *(c)* Prediction

Fig. 7 – Our prediction examples on VOC 2012 dataset.

as initialization. Selected prediction examples are shown in Fig. 7.

5.3.3 PASCAL-Context

The PASCAL-Context [44] dataset provides the segmentation labels of the whole scene for the PASCAL VOC images. We use the segmentation labels which contain 60 classes (59 object categories plus background) for evaluation as well as the provided training/test splits. The training set contains 4998 images and the test set has 5105 images. Results are shown in Table 8. Even without additional training data and with the same underlying ResNet architecture with 101 layers, we outperform the previous state-of-the-art achieved by DeepLab. For our method, using more pooling blocks, i.e. using 4 blocks

TABLE 8 – Segmentation results on PASCAL-Context dataset (60 classes). Our method performs the best. We only use the VOC training images.

Method	Extra train data	IoU
O2P [3]	-	18.1
CFM [11]	-	34.4
FCN-8s [43]	-	35.1
BoxSup [10]	-	40.5
HO-CRF [1]	-	41.3
Context [37]	-	43.3
DeepLab-v2(Res101) [6]	COCO (∼100K)	45.7
RefineNet-Res101-Pool2	-	47.1
RefineNet-Res152-Pool2	-	47.3
RefineNet-Res101-Pool4	-	48.0
RefineNet-Res152-Pool4	-	**48.4**

TABLE 9 – Segmentation results on SUN-RGBD dataset (37 classes). We compare to a number of recent methods. Our RefineNet significantly outperforms the existing methods.

Method	Train data	Pixel acc.	Mean acc.	IoU
Liu et al. [39]	RGB-D	–	10.0	–
Ren et al. [48]	RGB-D	–	36.3	–
Kendall et al. [28]	RGB	71.2	45.9	30.7
Context [37]	RGB	78.4	53.4	42.3
RefineNet-Res101-Pool2	RGB	80.4	57.8	45.7
RefineNet-Res152-Pool2	RGB	80.6	58.5	45.9
RefineNet-Res101-Pool4	RGB	80.8	57.3	46.3
RefineNet-Res152-Pool4	RGB	81.1	57.7	**47.0**

("Pool4"), achieves best performance.

5.3.4 SUN-RGBD

SUN-RGBD [53] is a segmentation dataset that contains around 10,000 RGB-D indoor images and provides pixel labeling masks for 37 classes. Results are shown in Table 9. Our method outperforms all existing methods by a large margin across all evaluation metrics, even though we do not make use of the depth information for training. Our best result is achieved by using 4 pooling blocks in our chained pooling.

5.3.5 ADE20K MIT

ADE20K [60] is a newly released dataset for scene parsing which provides dense labels of 150 classes on more than 20K scene images. The categories include a large variety of objects (e.g., person, car, etc.) and stuff (e.g., sky, road, etc.). The provided validation set consisting of 2000 images is used for quantitative evaluation. Results are shown in Table 10. Our method clearly outperforms the baseline methods described in [60]. For our method, using 4 pooling blocks ("Pool4" in the result table) in our chained pooling performs the best.

5.3.6 Cityscapes

Cityscapes [9] is a very recent dataset on street scene images from 50 different European cities. This dataset provides fine-grained pixel-level annotations of roads, cars, pedestrians, bicycles, sky, etc.. The provided training set has 2975 images and the validation set has 500 images. In total, 19 classes are considered for training and evaluation. The test set ground-truth is withheld by the organizers, and we evaluate our method on the their evaluation server. The test results are shown in Table 11. In this challenging setting, our architecture again outperforms previous methods. A few test images along with ground truth and our predicted semantic maps are shown in Fig. 8.

TABLE 10 – Segmentation results on the ADE20K dataset (150 classes) *val* set. our method achieves the best performance.

Method	IoU
FCN-8s [43]	29.4
SegNet [2]	21.6
DilatedNet [5, 57]	32.3
Cascaded-SegNet [60]	27.5
Cascaded-DilatedNet [60]	34.9
RefineNet-Res101-Pool2	40.2
RefineNet-Res152-Pool2	40.7
RefineNet-Res101-Pool4	41.6
RefineNet-Res152-Pool4	**42.4**

TABLE 11 – Segmentation results on the Cityscapes *test* set. our method achieves the best performance.

Method	IoU
FCN-8s [43]	65.3
DPN [42]	66.8
Dilation10 [57]	67.1
Context [37]	71.6
LRR-4x [17]	71.8
DeepLab [5]	63.1
DeepLab-v2(Res101) [6]	70.4
RefineNet-Res101 (ours)	**73.6**

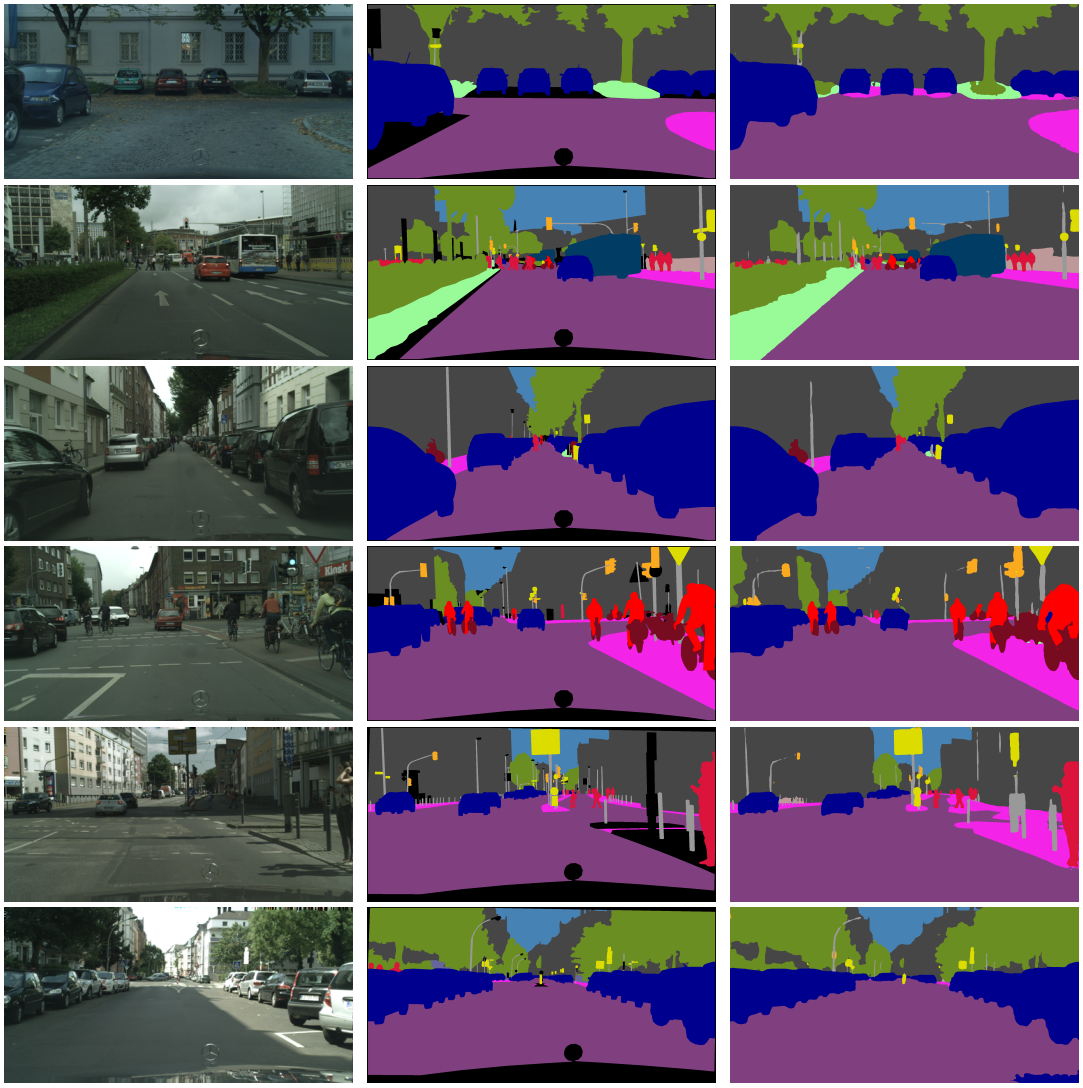
5.4 Depth Estimation

We here demonstrate the efficacy of our method for monocular image depth estimation. We perform experiments on the widely evaluated NYUDv2 dataset [51], which contains RGBD indoor scene sequences. The raw dataset consists of 464 scenes, with 249 for training and 215 for test. Following a similar setting in previous work [13, 33], we equally sample frames out of each training scene, leading to 12K images in total for training. The standard test set of 654 images with filled depth values are used for test. We use several measures commonly used in prior works for quantitative evaluations:

- average relative error (rel): $\frac{1}{T}\sum_p \frac{|d_p^{gt}-d_p|}{d_p^{gt}}$;
- root mean squared error (rms): $\sqrt{\frac{1}{T}\sum_p (d_p^{gt}-d_p)^2}$;
- average $\log_{10}$ error (log10): $\frac{1}{T}\sum_p |\log_{10} d_p^{gt} - \log_{10} d_p|$;
- accuracy with threshold thr: percentage (%) of d_p s.t. $\max(\frac{d_p^{gt}}{d_p}, \frac{d_p}{d_p^{gt}}) = \delta < thr$;

where d_p^{gt} and d_p are the ground-truth and predicted depths respectively at pixel indexed by p, and T is the total number of pixels in all the evaluated images.

The results are reported in Table 12. We use cascaded RefineNets with 4 pooling blocks in the chained residual pooling module. Our method achieves good performance for depth estimation. Kindly note that we directly apply the



(a) Test Image *(b)* Ground Truth *(c)* Prediction

Fig. 8 – Our prediction examples on Cityscapes dataset.

same architecture of RefineNet for dense regression without considering domain knowledge, special layer design or special loss function for depth estimation. We show some prediction examples in Fig. 9. As we can see, our method accurately infers the depth values of a scene even in cluttered backgrounds.

6 Conclusion

We have presented RefineNet, a novel multi-path refinement network for high-resolution dense prediction. The cascaded architecture is able to effectively combine high-level abstracts with low-level features to produce high-resolution pixel-wise prediction maps. Our design choices are inspired by the idea of identity mapping which facilitates gradient propagation across long-range connections and thus enables effective end-to-end learning. We focus on the semantic segmentation and monocular image depth estimation tasks for representative discrete and continuous dense prediction problems. Extensive experiments show that our RefineNet outperforms most previous works on multiple public benchmarks, setting a new mark for the state of the art in semantic labeling. Note that our method can also be applied to other dense prediction tasks like low-level vision tasks including image denoising, super-resolution, edge detection. This can be explored in the future work.

Acknowledgements

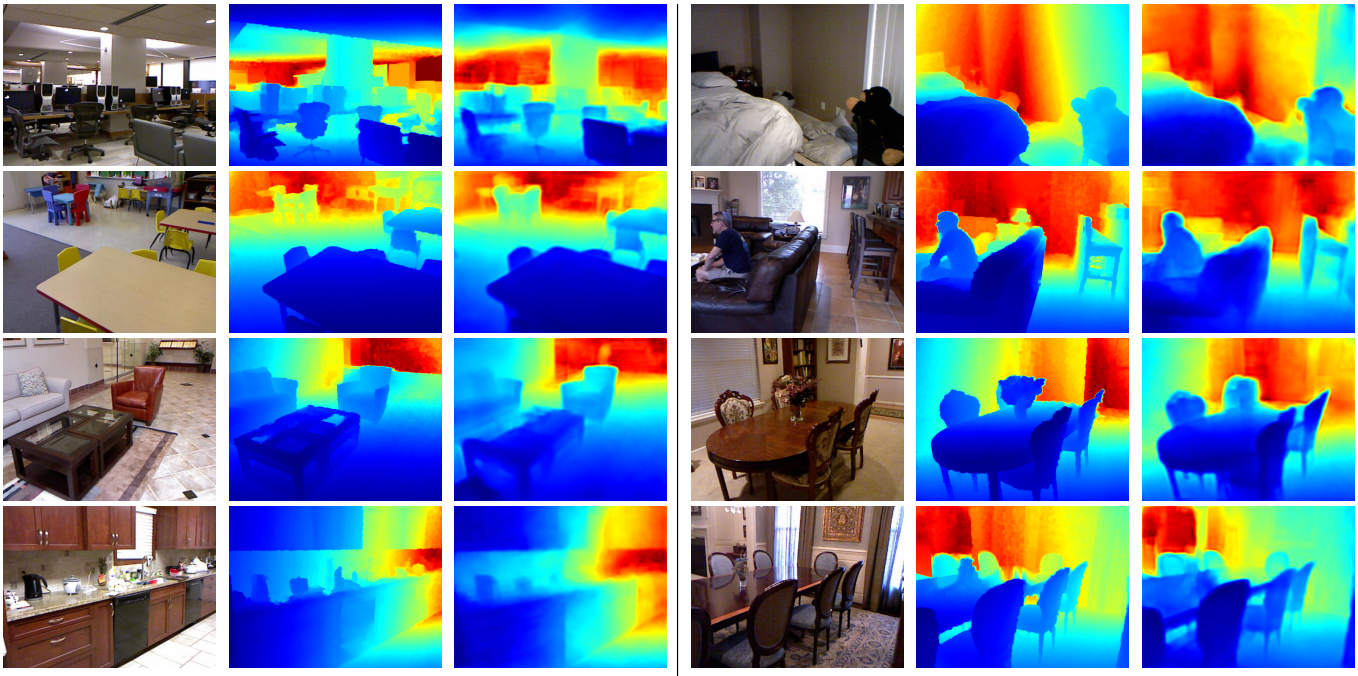
This research was supported by the Australian Research Council through the Australian Centre for Robotic Vision (CE140100016). C. Shen's participation was supported by an ARC Future Fellowship (FT120100969). I. Reid's participation was supported by an ARC Laureate Fellowship (FL130100102). G. Lin's participation was partly supported by a NTU start-up grant and a MOE Tier-1 grant (2017-T1-002-091-02).

References

[1] A. Arnab, S. Jayasumana, S. Zheng, and P. H. Torr. Higher order conditional random fields in deep neural

TABLE 12 – State-of-the-art comparisons for depth estimation on the NYUDv2 dataset. Our method achieves comparable performance.

Method	Error (lower is better)			Accuracy (higher is better)		
	rel	log10	rms	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
DepthTransfer [27]	0.35	0.131	1.2	-	-	-
Ladicky et al. [32]	-	-	-	0.542	0.829	0.941
Eigen et al. [15]	0.215	-	0.907	0.611	0.887	0.971
Eigen et al. [13]	0.158	-	0.641	0.769	0.950	0.988
Wang et al. [55]	0.220	0.094	0.745	0.605	0.890	0.970
DCNF-FCSP [41]	0.213	0.087	0.759	0.650	0.906	0.976
JCNF [29]	0.201	0.077	0.711	0.690	0.910	0.979
Roy et al. [50]	0.187	0.078	0.744	-	-	-
Laina et al. [33]	0.127	0.055	**0.573**	0.811	0.953	0.988
RefineNet (Ours)	0.148	0.062	0.574	0.805	0.960	0.989



(a) Test Image *(b)* Ground Truth *(c)* Prediction *(d)* Test Image *(e)* Ground Truth *(f)* Prediction

Fig. 9 – Our prediction examples on the NYUv2 dataset.

networks. In *European Conference on Computer Vision*. Springer, 2016. 11

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, 2015. 2, 11

[3] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012. 11

[4] S. Chandra and I. Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. In *ECCV*, 2016. 10

[5] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015. 1, 2, 9, 10, 11

[6] L. 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. *CoRR*, abs/1606.00915, 2016. 1, 3, 4, 7, 8, 9, 10, 11

[7] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. *arXiv preprint arXiv:1511.03339*, 2015. 8, 9

[8] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1971–1978, 2014. 8

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 11

[10] J. Dai, K. He, and J. Sun. BoxSup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *ICCV*, 2015. 2, 7, 10, 11

[11] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015. 11

[12] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014. 2

[13] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 1, 2, 3, 11, 13

[14] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *ICCV*, 2013. 2

[15] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014. 2, 3, 13

[16] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. In *IJCV*, 2010. 7, 9

[17] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *ECCV*, 2016. 11

[18] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2

[19] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *CVPR*, 2013. 9

[20] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, 2014. 9

[21] B. Hariharan, P. Arbelaez, L. D. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 9

[22] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2014. 2

[23] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014. 2

[24] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017. 3

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR 2016*, 2016. 1, 2, 3, 5, 6, 7

[26] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016. 2, 3, 6

[27] K. Karsch, C. Liu, and S. B. Kang. Depthtransfer: Depth extraction from video using non-parametric sampling. *PAMI*, 2014. 13

[28] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *CoRR*, abs/1511.02680, 2015. 11

[29] S. Kim, K. Park, K. Sohn, and S. Lin. Unified depth prediction and intrinsic image decomposition from a single image via joint convolutional neural fields. In *ECCV*, pages 143–159, 2016. 13

[30] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *NIPS*, 2012. 2

[31] P. Krahenbuhl and V. Koltun. Learning to propose objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1574–1582, 2015. 3

[32] L. Ladick, J. Shi, and M. Pollefeys. Pulling things out of perspective. In *CVPR*, 2014. 13

[33] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248, 2016. 3, 11, 13

[34] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan. Semantic object parsing with graph lstm. *arXiv preprint arXiv:1603.07063*, 2016. 9

[35] X. Liang, X. Shen, D. Xiang, J. Feng, L. Lin, and S. Yan. Semantic object parsing with local-global long short-term memory. *arXiv preprint arXiv:1511.04510*, 2015. 9

[36] G. Lin, A. Milan, C. Shen, and I. Reid. RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017. 2

[37] G. Lin, C. Shen, A. van den Hengel, and I. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*, 2016. 9, 10, 11

[38] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 9

[39] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE T. Pattern Analysis & Machine Intelligence*, 2011. 11

[40] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *CVPR*, 2015. 1, 2, 3

[41] F. Liu, C. Shen, G. Lin, and I. D. Reid. Learning depth from single monocular images using deep convolutional neural fields. *TPAMI*, 2016. 1, 2, 13

[42] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *ICCV*, 2015. 9, 10, 11

[43] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2, 3, 7, 9, 10, 11

[44] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, et al. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014. 10

[45] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 1, 2, 10

[46] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015. 3

[47] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016. 3

[48] X. Ren, L. Bo, and D. Fox. Rgb-(d) scene labeling: Features and algorithms. In *CVPR*, 2012. 11

[49] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015. 2

[50] A. Roy and S. Todorovic. Monocular depth estimation using neural regression forest. In *CVPR*, pages 5506–5514, 2016. 3, 13

[51] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 9, 11

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

[53] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015. 11

[54] A. Vedaldi and K. Lenc. MatConvNet – convolutional neural networks for matlab, 2014. 7

[55] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille. Towards unified depth and semantic prediction from a single image. In *CVPR*, pages 2800–2809, 2015. 13

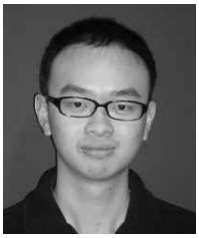
[56] F. Xia, P. Wang, L.-C. Chen, and A. L. Yuille. Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net. *arXiv preprint arXiv:1511.06881*, 2015. 9

[57] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, 2015. 11

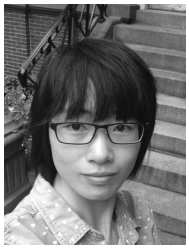
[58] S. Zagoruyko, A. Lerer, T.-Y. Lin, P. H. Pinheiro, S. Gross, S. Chintala, and P. Dollar. A multipath network for object detection. In *Proceedings of the British Machine Vision Conference*, 2016. 2, 3

[59] 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 *ICCV*, 2015. 2, 9, 10

[60] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ADE20K dataset. *CoRR*, abs/1608.05442, 2016. 11



Guosheng Lin is an Assistant Professor at School of Computer Science and Engineering, Nanyang Technological University, Singapore. He received his PhD degree at The University of Adelaide in 2014. His research interests are in computer vision and machine learning. He received a Bachelor degree and a Master degree from the South China University of Technology in computer science in 2007 and 2010, respectively.



Fayao Liu received her PhD degree at the University of Adelaide, Australia in 2015. She reveived the B.Eng. and M.Eng. degrees from School of Computer Science, National University of Defense Technology, Hunan, China, in 2008 and 2010, respectively. Her current research interests include machine learning and computer vision.

Anton Milan is an Applied Scientist at Amazon Core ML Berlin, Germany. He studied Computer Science and Philosophy at the University of Bonn and received his PhD in May 2013. In 2014 he joined the Australian Centre for Visual Technologies (ACVT) at the University of Adelaide as a post-doctoral researcher. He was at ETH in 2015 and at the University of Bonn in 2016 as a visiting researcher.

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

Ian Reid is an Australian Laureate Fellow and Professor of Computer Science at the University of Adelaide, where he has been since September 2012. Prior to that he was a Professor of Engineering Science at the University of Oxford. He received the BSc degree in computer science and mathematics with first class honors from the University of Western Australia in 1987 and was awarded a Rhodes Scholarship in 1988 in order to study at the University of Oxford, where he received the DPhil degree in 1991.