Image Segmentation

Wei-Ta Chu

Fully Convolutional Networks

Wei-Ta Chu

Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully Convolutional Networks for Semantic Segmentation," CVPR, 2015.

Introduction

- Our model transfers recent success in classification to dense prediction by reinterpreting classification nets as fully convolutional and fine-tuning from their learned representations.
- Semantic segmentation faces an inherent tension between semantics and location: global information resolves *what* while local information resolves *where*.

Fully convolutional networks

- Each layer of data in a convnet is a three-dimensional array of size $h \times w \times d$, where h and w are spatial dimensions, and d is the feature or channel dimension.
- Writing $\mathbf{x}_{ij}$ for the data vector at location (i, j) in a particular layer, and $\mathbf{y}_{ij}$ for the following layer, these functions compute outputs $\mathbf{y}_{ij}$ by

$$\mathbf{y}_{ij} = f_{ks}\left(\{\mathbf{x}_{si+\delta i, sj+\delta j}\}_{0 \le \delta i, \delta j \le k}\right)$$

where k is called the kernel size, s is the stride, and f_{ks} determines the layer type.

Fully convolutional networks

- Layer type: a matrix multiplication for convolution or average pooling, a spatial max for max pooling, or an elementwise nonlinearity for an activation function, and so on for other types of layers.
- An FCN naturally operates on an input of any size, and produces an output of corresponding spatial dimensions.

Fully convolutional networks

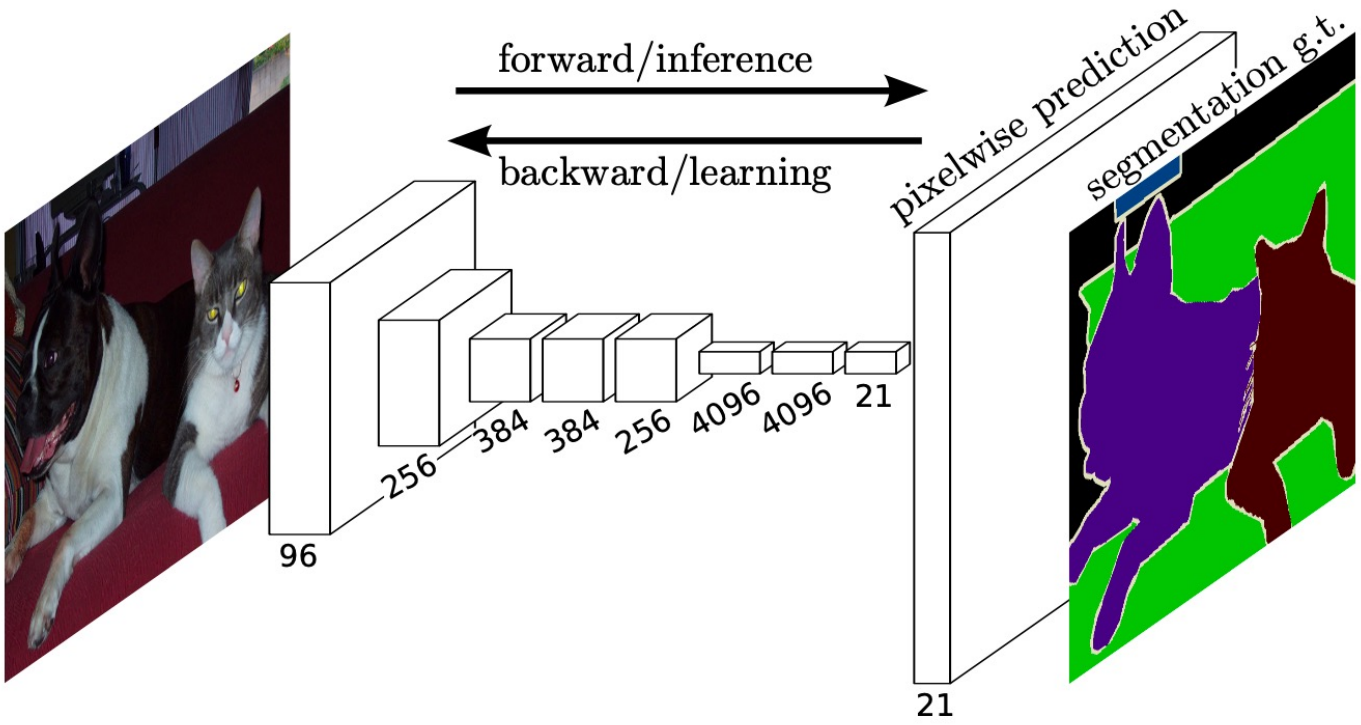


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Adapting classifiers for dense prediction

- Typical CNNs take fixed-sized inputs and produce non-spatial outputs. The fully connected layers of these nets have fixed dimensions and throw away spatial coordinates.
- However, these *fully connected layers can also be viewed as convolutions with kernels that cover their entire input regions*. Doing so casts them into fully convolutional networks that take input of any size and output classification maps.

Adapting classifiers for dense prediction

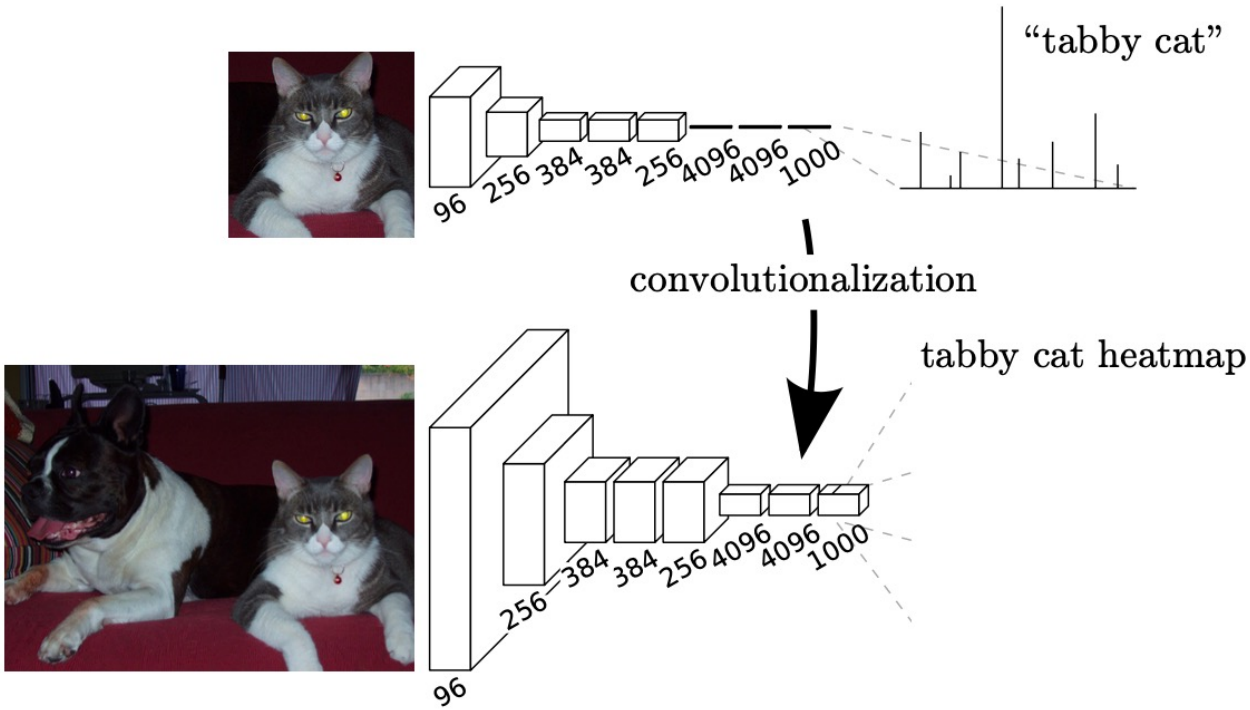


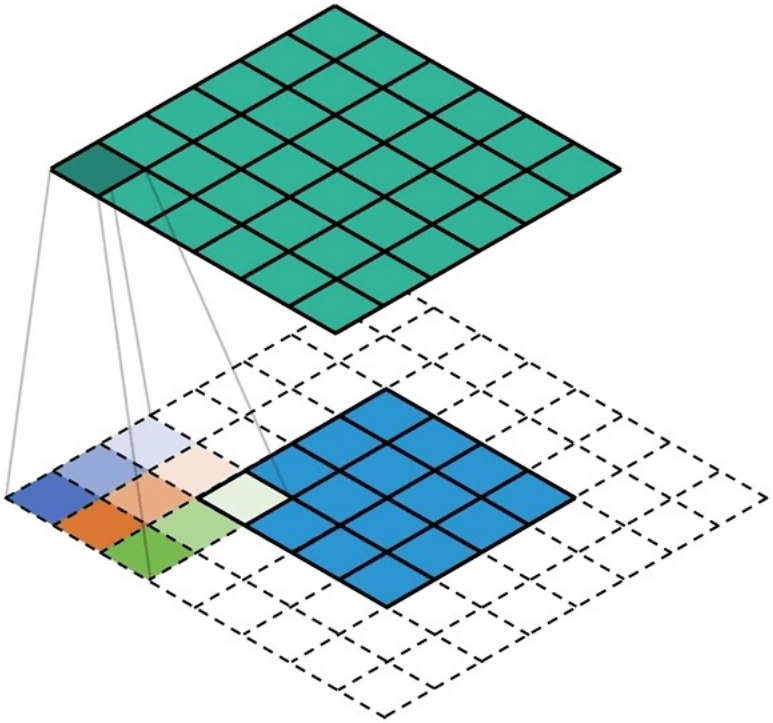
Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

Upsampling is backwards strided convolution

- Upsampling with factor *f* is convolution with a fractional input stride of 1/*f*. As *f* is integral, a natural way to upsample is therefore backwards convolution (sometimes called deconvolution) with an output stride of *f*.

Upsampling is backwards strided convolution

- Deconvolution



Segmentation Architecture

- We cast ILSVRC classifiers into FCNs and augment them for dense prediction with in-network upsampling and a pixelwise loss.
- We train for segmentation by fine-tuning. Next, we add skips between layers to fuse coarse, semantic and local, appearance information.
- We train with a per-pixel multinomial logistic loss and validate with the standard metric of mean pixel intersection over union.

Segmentation Architecture

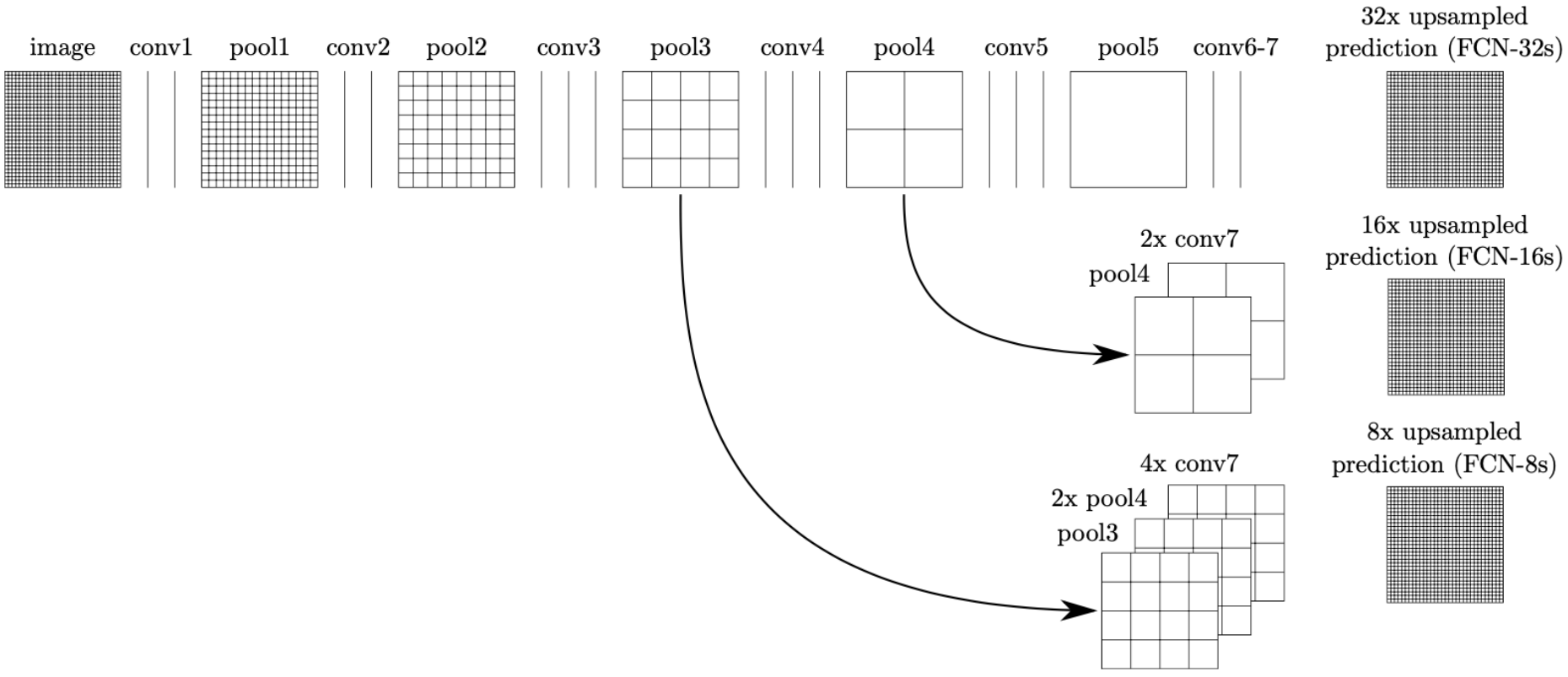


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

Segmentation Architecture

- From classifier to dense FCN
  - We consider AlexNet, VGG nets and GoogLeNet
  - We append a 1 × 1 convolution with channel dimension 21 to predict scores for each of the PASCAL classes (including background) at each of the coarse output locations, followed by a deconvolution layer to bi-linearly upsample the coarse outputs to pixel-dense outputs.

Segmentation Architecture

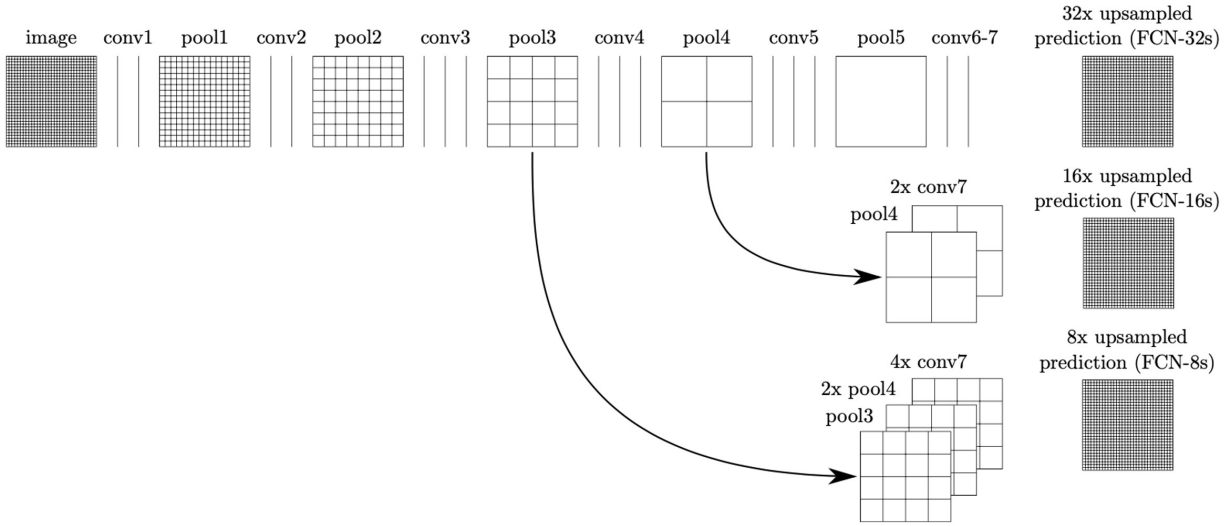
- From classifier to dense FCN
  - We consider AlexNet, VGG nets and GoogLeNet

Table 1. We adapt and extend three classification convnets. We compare performance by mean intersection over union on the validation set of PASCAL VOC 2011 and by inference time (averaged over 20 trials for a 500×500 input on an NVIDIA Tesla K40c). We detail the architecture of the adapted nets with regard to dense prediction: number of parameter layers, receptive field size of output units, and the coarsest stride within the net. (These numbers give the best performance obtained at a fixed learning rate, not best performance possible.)

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

Segmentation Architecture

- Combining what and where
  - While fully convolutionalized classifiers can be fine-tuned to segmentation, their output is dissatisfyingly coarse.
  - We address this by adding skips that combine the final prediction layer with lower layers with finer strides.



Segmentation Architecture

- Combining what and where

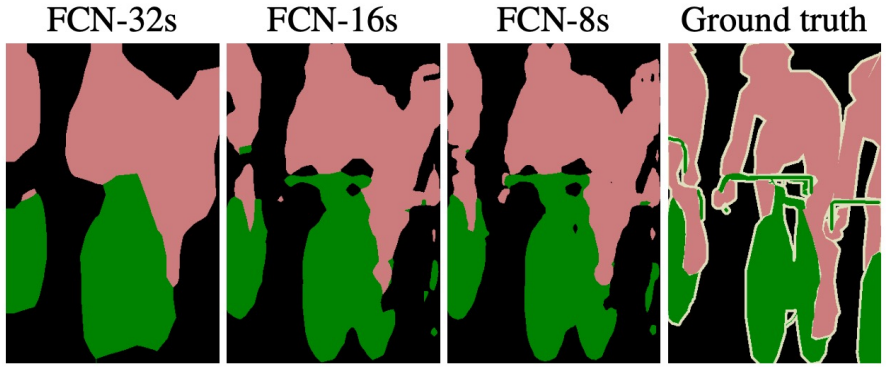


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

Table 2. Comparison of skip FCNs on a subset[7] of PASCAL VOC 2011 segval. Learning is end-to-end, except for FCN-32s-fixed, where only the last layer is fine-tuned. Note that FCN-32s is FCN-VGG16, renamed to highlight stride.

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

Results

- We test our FCN on semantic segmentation and scene parsing, exploring PASCAL VOC, NYUDv2, and SIFT Flow.
- Let n_{ij} be the number of pixels of class i predicted to belong to class j, where there are n_{cl} different classes, and let $t_i = \sum_j n_{ij}$ be the total number of pixels of class i.
  - **pixel accuracy:** $\sum_i n_{ii} / \sum_i t_i$
  - **mean accuraccy:** $(1/n_{\text{cl}}) \sum_i n_{ii}/t_i$
  - **mean IU:** $(1/n_{\text{cl}}) \sum_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$
  - **frequency weighted IU:** $(\sum_k t_k)^{-1} \sum_i t_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$

Results

- PASCAL VOC

Table 3. Our fully convolutional net gives a 20% relative improvement over the state-of-the-art on the PASCAL VOC 2011 and 2012 test sets and reduces inference time.

	mean IU VOC2011 test	mean IU VOC2012 test	inference time
R-CNN [10]	47.9	-	-
SDS [15]	52.6	51.6	~ 50 s
FCN-8s	**62.7**	**62.2**	~ **175 ms**

Results

- NYUDv2: an RGB-D dataset collected using the Microsoft Kinect. It has 1449 RGB-D images, with pixel- wise labels that have been coalesced into a 40 class.

Table 4. Results on NYUDv2. *RGBD* is early-fusion of the RGB and depth channels at the input. *HHA* is the depth embedding of [13] as horizontal disparity, height above ground, and the angle of the local surface normal with the inferred gravity direction. *RGB-HHA* is the jointly trained late fusion model that sums RGB and HHA predictions.

	pixel acc.	mean acc.	mean IU	f.w. IU
Gupta *et al.* [13]	60.3	-	28.6	47.0
FCN-32s RGB	60.0	42.2	29.2	43.9
FCN-32s RGBD	61.5	42.4	30.5	45.5
FCN-32s HHA	57.1	35.2	24.2	40.4
FCN-32s RGB-HHA	64.3	44.9	32.8	48.0
FCN-16s RGB-HHA	**65.4**	**46.1**	**34.0**	**49.5**

Results

- SIFT Flow is a dataset of 2,688 images with pixel labels for 33 semantic categories.

Table 5. Results on SIFT Flow[9] with class segmentation (center) and geometric segmentation (right). Tighe [33] is a non-parametric transfer method. Tighe 1 is an exemplar SVM while 2 is SVM + MRF. Farabet is a multi-scale convnet trained on class-balanced samples (1) or natural frequency samples (2). Pinheiro is a multi-scale, recurrent convnet, denoted RCNN$_3$ (o^3). The metric for geometry is pixel accuracy.

	pixel acc.	mean acc.	mean IU	f.w. IU	geom. acc.
Liu *et al.* [23]	76.7	-	-	-	-
Tighe *et al.* [33]	-	-	-	-	90.8
Tighe *et al.* [34] 1	75.6	41.1	-	-	-
Tighe *et al.* [34] 2	78.6	39.2	-	-	-
Farabet *et al.* [7] 1	72.3	50.8	-	-	-
Farabet *et al.* [7] 2	78.5	29.6	-	-	-
Pinheiro *et al.* [28]	77.7	29.8	-	-	-
FCN-16s	**85.2**	**51.7**	39.5	76.1	**94.3**

U-Net

Wei-Ta Chu

Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2015.

Introduction

- We modify FCN and extend this architecture such that it works with very few training images and yields more precise segmentations.
- In the upsampling part we have also a large number of feature channels, which allow the network to propagate context information to higher resolution layers.
- The expansive path is more or less symmetric to the contracting path, and yields a U-shaped architecture.

Introduction

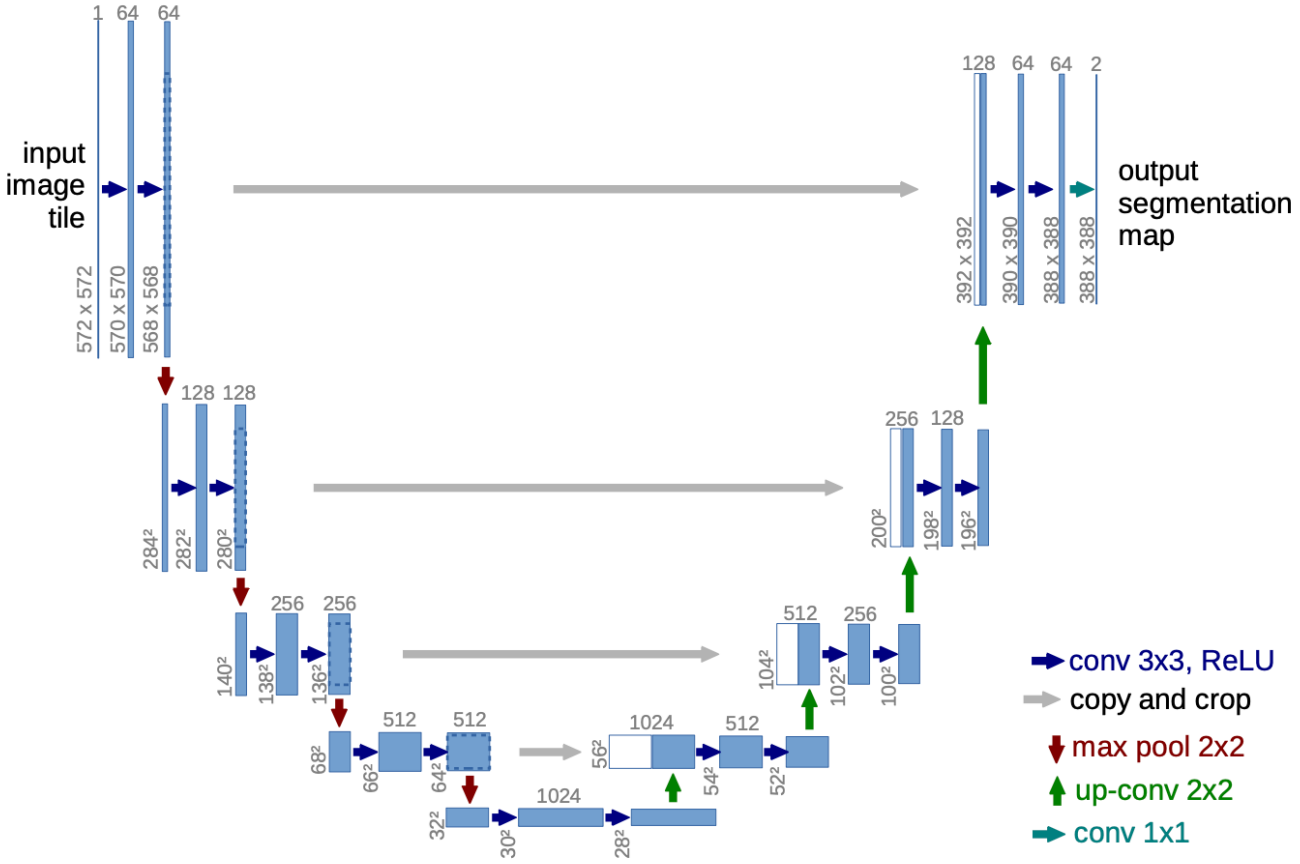
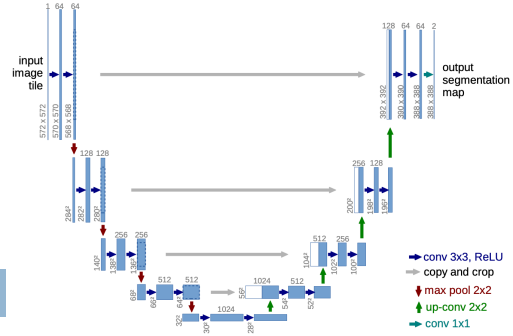


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Introduction

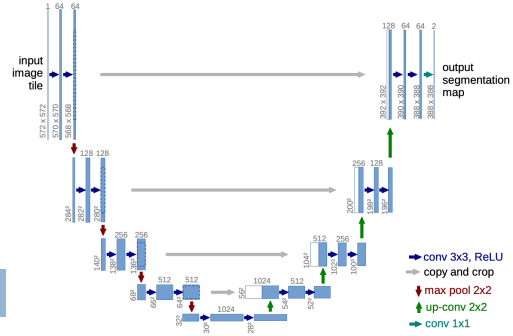
- We use excessive data augmentation by applying elastic deformations to the available training images.
- The resulting network is applicable to various biomedical segmentation problems.

Network Architecture



- U-Net consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels.

Network Architecture



- Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution (“up-convolution”) that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU.
- At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.

Training

- The input images and their corresponding segmentation maps are used to train the network with the stochastic gradient descent.
- The energy function is computed by a pixel-wise soft-max over the final feature map combined with the cross entropy loss function.
- The soft-max is defined as

$$p_k(\mathbf{x}) = \exp(a_k(\mathbf{x})) / \left(\sum_{k'=1}^{K} \exp(a_{k'}(\mathbf{x}))\right)$$

where $a_k(\mathbf{x})$ denotes the activation in feature channel k at the pixel position $\mathbf{x}$.

Training

- The cross entropy then penalizes at each position the deviation of $p_{l(\mathbf{x})}(\mathbf{x})$ from 1 using

$$E = \sum_{\mathbf{x} \in \Omega} w(\mathbf{x}) \log(p_{\ell(\mathbf{x})}(\mathbf{x}))$$

where l: $\Omega \rightarrow \{1,...,K\}$ is the true label of each pixel and w: $\Omega \rightarrow \mathrm{R}$ is a weight map that we introduced to give some pixels more importance in the training.

Training

- We pre-compute the weight map for each ground truth segmentation to compensate the different frequency of pixels from a certain class in the training data set. The weight map is computed as

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right)$$

where $w_c : \Omega \rightarrow \mathrm{R}$ is the weight map to balance the class frequencies, $d_1 : \Omega \rightarrow \mathrm{R}$ denotes the distance to the border of the nearest cell and $d_2 : \Omega \rightarrow \mathrm{R}$ the distance to the border of the second nearest cell.

Training

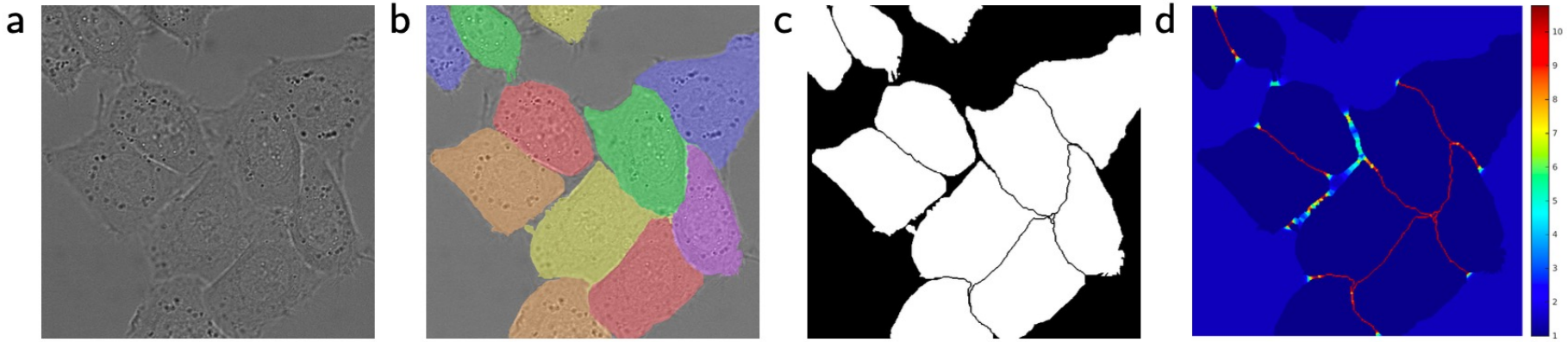


Fig. 3. HeLa cells on glass recorded with DIC (differential interference contrast) microscopy. (**a**) raw image. (**b**) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (**c**) generated segmentation mask (white: foreground, black: background). (**d**) map with a pixel-wise loss weight to force the network to learn the border pixels.

Experiments

- Segmentation of neuronal structures in electron microscopic recordings
- The training data is a set of 30 images (512x512 pixels). Each image comes with a corresponding fully annotated ground truth segmentation map for cells (white) and membranes (black).
- The evaluation is done by thresholding the map at 10 different levels and computation of the “warping error”, the “Rand error” and the “pixel error”

Experiments

Table 1. Ranking on the EM segmentation challenge [14] (march 6th, 2015), sorted by warping error.

Rank	Group name	Warping Error	Rand Error	Pixel Error
	** human values **	0.000005	0.0021	0.0010
1.	u-net	**0.000353**	0.0382	0.0611
2.	DIVE-SCI	0.000355	0.0305	0.0584
3.	IDSIA [1]	0.000420	0.0504	0.0613
4.	DIVE	0.000430	0.0545	**0.0582**
⋮				
10.	IDSIA-SCI	0.000653	**0.0189**	0.1027

Experiments

- Cell segmentation task in light microscopic images. Part of the ISBI cell tracking challenge 2014 and 2015.
- It contains 35 partially annotated training images.
- We achieve an average IOU ("intersection over union") of 92%, which is significantly better than the second best algorithm with 83%.

Table 2. Segmentation results (IOU) on the ISBI cell tracking challenge 2015.

Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	**0.9203**	**0.7756**

Mask R-CNN

Wei-Ta Chu

Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick, “Mask R-CNN,” ICCV, 2017.

Introduction

- Instance segmentation requires *object detection* and *semantic segmentation* -- the goal is to classify each pixel into a fixed set of categories without differentiating object instances.

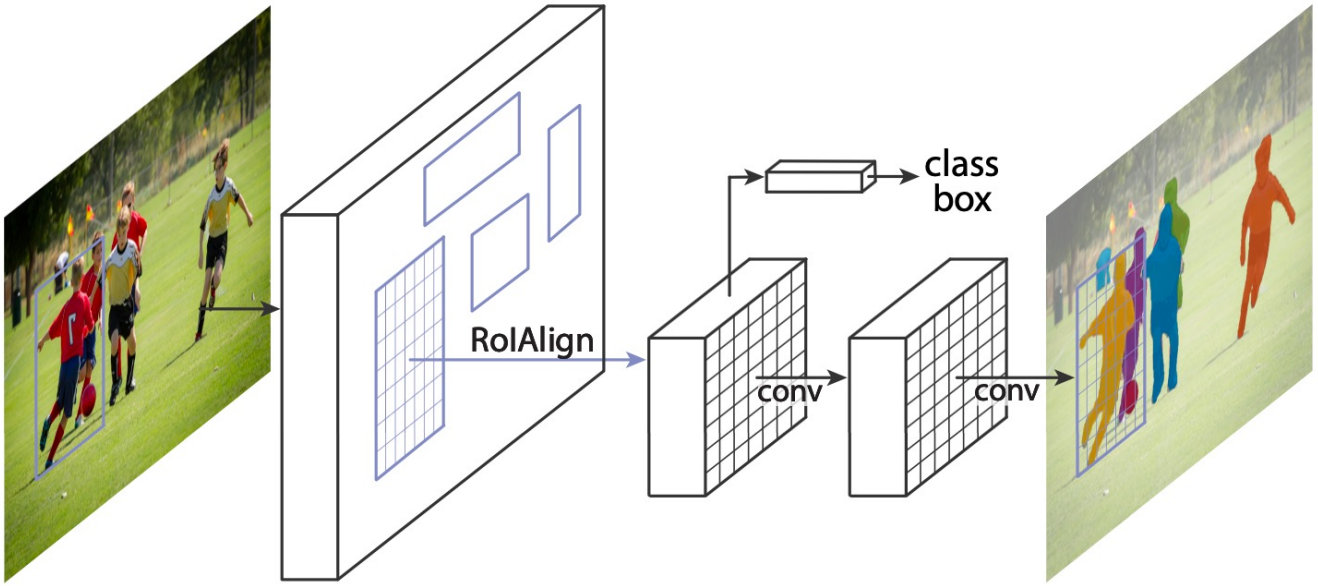


Figure 1. The **Mask R-CNN** framework for instance segmentation.

Introduction

- Mask R-CNN constructs the mask branch properly
  - A simple, quantization-free layer, called RoIAlign, faithfully preserves exact spatial locations
- It is essential to decouple mask and class prediction
- Mask R-CNN surpasses all previous state-of-the-art single-model results on the COCO instance segmentation task
- This method also excels on the COCO object detection task
- This model can run at about 200ms per frame on a GPU, and training on COCO takes one to two days on a single 8-GPU machine.

Mask R-CNN

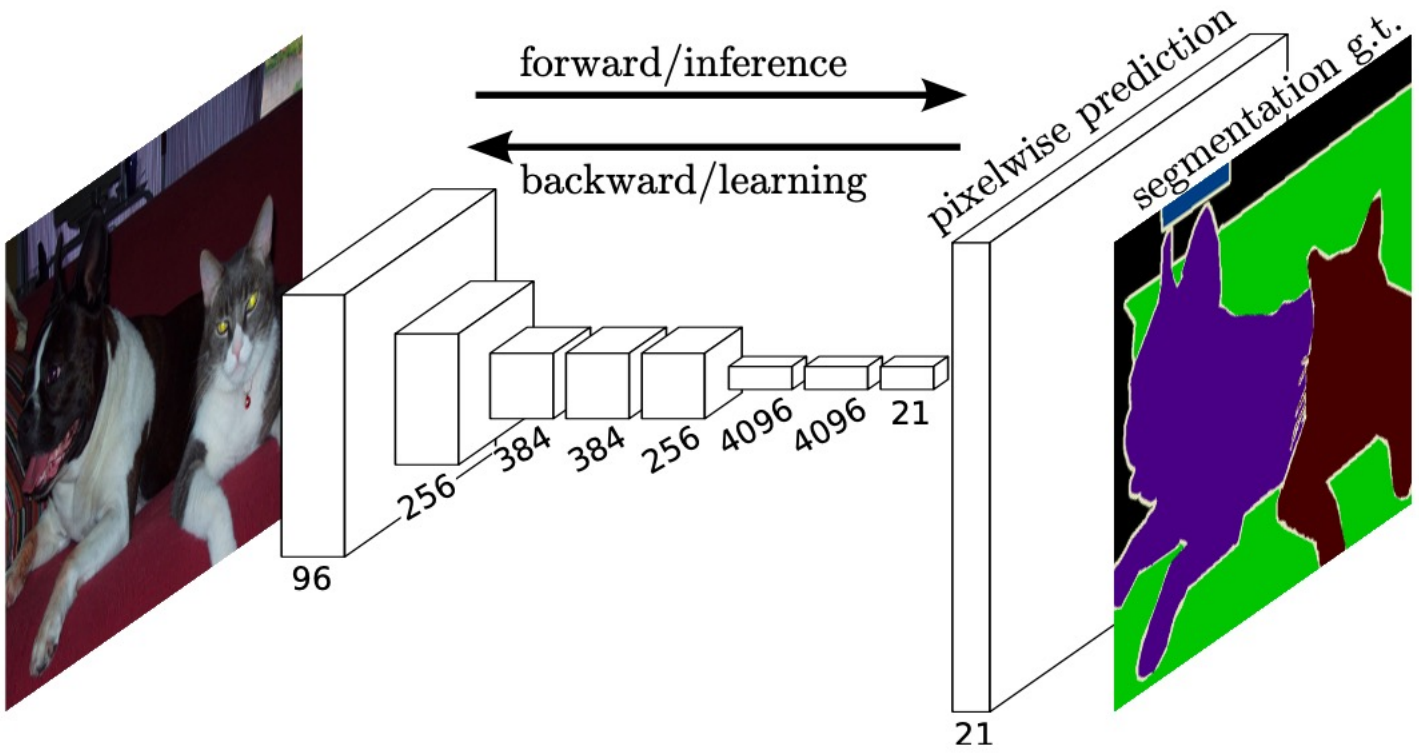
- Mask R-CNN adopts the same two-stage procedure, with an identical first stage (which is RPN). In the second stage, *in parallel* to predicting the class and box offset, Mask R-CNN also outputs a binary mask for each RoI.
- During training, we define a multi-task loss on each sampled RoI as $L = L_{cls} + L_{box} + L_{mask}$.
- The mask branch has a Km^2-dimensional output for each RoI, which encodes K binary masks of resolution $m \times m$, one for each of the K classes. We apply a per-pixel sigmoid, and define L_{mask} as the average binary cross-entropy loss.

Mask R-CNN

- L_{mask} allows the network to generate masks for every class without competition among classes. This decouples mask and class prediction. This is different from common practice when applying FCNs to semantic segmentation, which typically uses a per-pixel softmax and a multinomial cross-entropy loss.
- We predict an $m \times m$ mask from each RoI using an FCN. This allows each layer in the mask branch to maintain the explicit $m \times m$ object spatial layout without collapsing it into a vector representation that lacks spatial dimensions.

Mask R-CNN

- Fully convolutional network



RoIAlign

- RoIPool introduces misalignments between the RoI and the extracted features.
- RoIAlign layer
  - We avoid any quantization of the RoI boundaries or bins (i.e., we use x/16 instead of [x/16]). We use bilinear interpolation to compute the exact values of the input features at four regularly sampled locations in each RoI bin, and aggregate the result (using max or average).

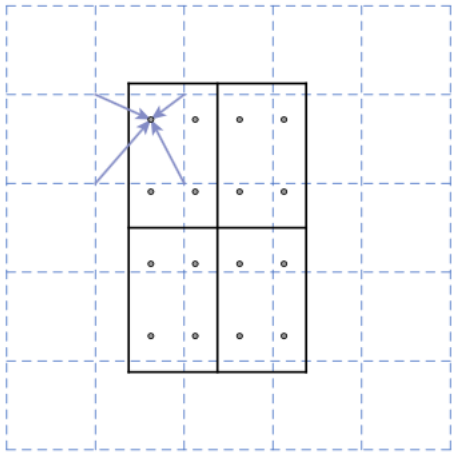
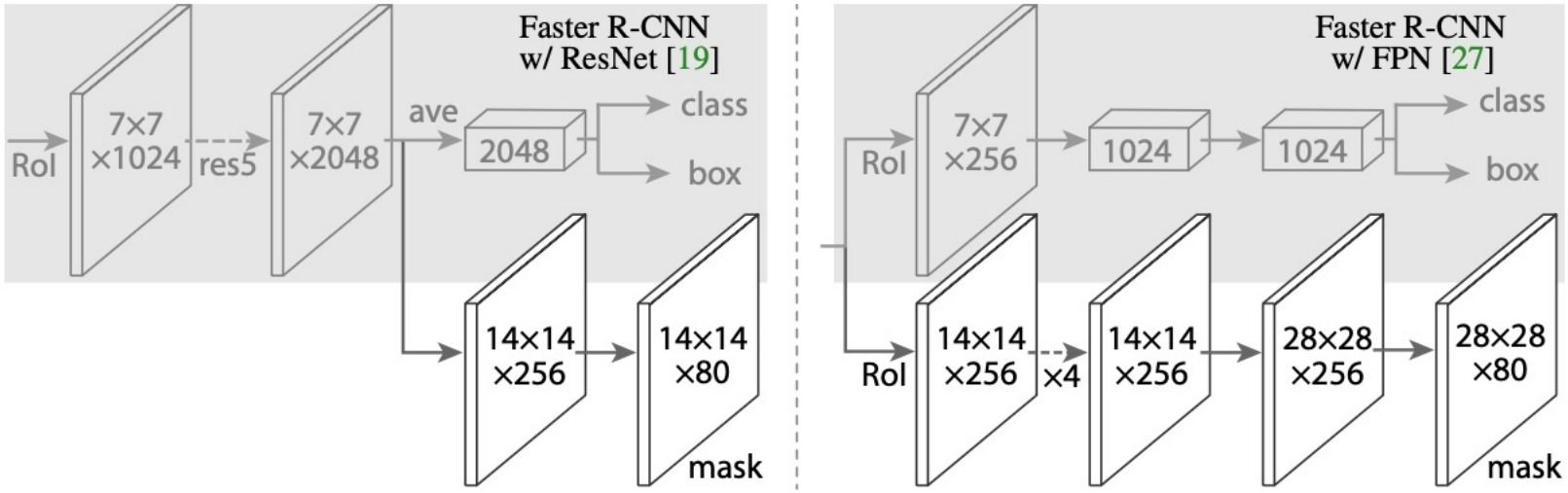


Figure 3. **RoIAlign:** The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.

Network Architecture

- The convolutional *backbone* architecture used for feature extraction over an entire image
- The network *head* for bounding-box recognition (classification and regression) and mask prediction.



Experiments

- We report the standard COCO metrics including AP (averaged over IoU thresholds), AP_{50}, AP_{75} , and AP_S , AP_M , AP_L (AP at different scales).
- Unless noted, AP is evaluating using *mask* IoU.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

Table 1. **Instance segmentation** *mask* AP on COCO `test-dev`. MNC [10] and FCIS [26] are the winners of the COCO 2015 and 2016 segmentation challenges, respectively. Without bells and whistles, Mask R-CNN outperforms the more complex FCIS+++, which includes multi-scale train/test, horizontal flip test, and OHEM [38]. All entries are *single-model* results.

Experiments

Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

Experiments

Figure 5. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).

Experiments

- FCIS+++ exhibits systematic artifacts on overlapping instances, suggesting that it is challenged by the fundamental difficulty of instance segmentation. Mask R-CNN shows no such artifacts.

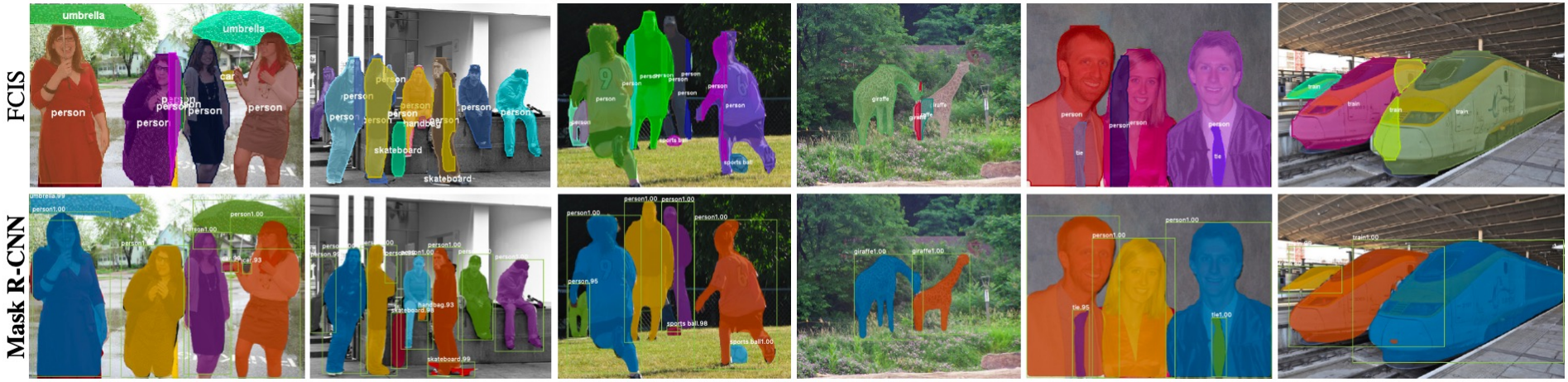


Figure 6. FCIS+++ [26] (top) *vs*. Mask R-CNN (bottom, ResNet-101-FPN). FCIS exhibits systematic artifacts on overlapping objects.

Experiments

- Ablation studies

net-depth-features	AP	AP_{50}	AP_{75}
ResNet-50-C4	30.3	51.2	31.5
ResNet-101-C4	32.7	54.2	34.3
ResNet-50-FPN	33.6	55.2	35.3
ResNet-101-FPN	35.4	57.3	37.5
ResNeXt-101-FPN	**36.7**	**59.5**	**38.9**

(a) **Backbone Architecture**: Better backbones bring expected gains: deeper networks do better, FPN outperforms C4 features, and ResNeXt improves on ResNet.

	AP	AP_{50}	AP_{75}
softmax	24.8	44.1	25.1
sigmoid	**30.3**	**51.2**	**31.5**
	+5.5	*+7.1*	*+6.4*

(b) **Multinomial *vs*. Independent Masks** (ResNet-50-C4): *Decoupling* via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

Experiments

- Bounding box detection
  - Even though the full Mask R-CNN model is trained, only the classification and box outputs are used at inference
  - As a further comparison, we trained a version of Mask R-CNN but without the mask branch, denoted by "Faster R-CNN, RoIAlign"

	backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_{S}	AP^{bb}_{M}	AP^{bb}_{L}
Faster R-CNN+++ [19]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [27]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [21]	Inception-ResNet-v2 [41]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [39]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	**39.8**	**62.3**	**43.4**	**22.1**	**43.2**	51.2

Table 3. **Object detection** *single-model* results (bounding box AP), *vs.* state-of-the-art on `test-dev`. Mask R-CNN using ResNet-101-FPN outperforms the base variants of all previous state-of-the-art models (the mask output is ignored in these experiments). The gains of Mask R-CNN over [27] come from using RoIAlign (+1.1 AP^{bb}), multitask training (+0.9 AP^{bb}), and ResNeXt-101 (+1.6 AP^{bb}).

HRNet

Wei-Ta Chu

Jingdong Wang , Ke Sun , Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu , Yadong Mu , Mingkui Tan , Xinggang Wang , Wenyu Liu , and Bin Xiao, “Deep High-Resolution Representation Learning for Visual Recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 10, 2021.

Introduction

- High-resolution representations are needed for position-sensitive tasks.
- The previous methods adopt the high-resolution recovery process to raise the representation resolution from the low-resolution representation outputted by a classification or classification-like network.

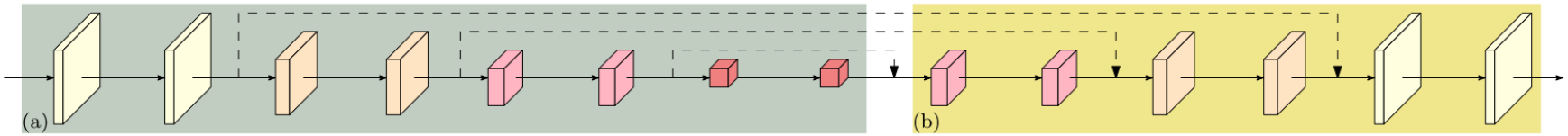


Fig. 1. The structure of recovering high resolution from low resolution. (a) A low-resolution representation learning subnetwork (such as VGGNet [102] and ResNet [40]), which is formed by connecting high-to-low convolutions in series. (b) A high-resolution representation recovering subnetwork, which is formed by connecting low-to-high convolutions in series. Representative examples include SegNet [3], DeconvNet [87], U-Net [97], Hourglass [85], encoder-decoder [92], and SimpleBaseline [126].

Introduction

- High-Resolution Net (HRNet) is able to maintain high-resolution representations through the whole process.
- Our approach connects high-to-low resolution convolution streams in parallel rather than in series.
- We repeat multi-resolution fusions to boost the high-resolution representations with the help of the low-resolution representations

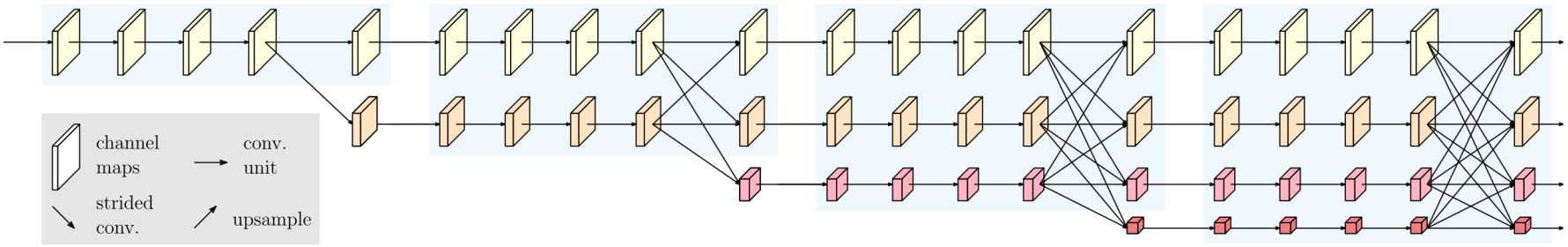


Fig. 2. An example of a high-resolution network. Only the main body is illustrated, and the stem (two stride-2 3×3 convolutions) is not included. There are four stages. The 1st stage consists of high-resolution convolutions. The 2nd (3rd, 4th) stage repeats two-resolution (three-resolution, four-resolution) blocks. The detail is given in Section 3.

High-Resolution Networks

- Parallel Multi-Resolution Convolution
  - We start from a high-resolution convolution stream as the first stage, gradually add high-to-low resolution streams one by one, forming new stages, and connect the multi-resolution streams in parallel.

$$
\begin{array}{ccccccc}
\mathcal{N}_{11} & \rightarrow & \mathcal{N}_{21} & \rightarrow & \mathcal{N}_{31} & \rightarrow & \mathcal{N}_{41} \\
 & \searrow & \mathcal{N}_{22} & \rightarrow & \mathcal{N}_{32} & \rightarrow & \mathcal{N}_{42} \\
 & & & \searrow & \mathcal{N}_{33} & \rightarrow & \mathcal{N}_{43} \\
 & & & & & \searrow & \mathcal{N}_{44},
\end{array}
$$

High-Resolution Networks

- Repeated Multi-Resolution Fusion
  - The goal of the fusion module is to exchange the information across multi-resolution representations.

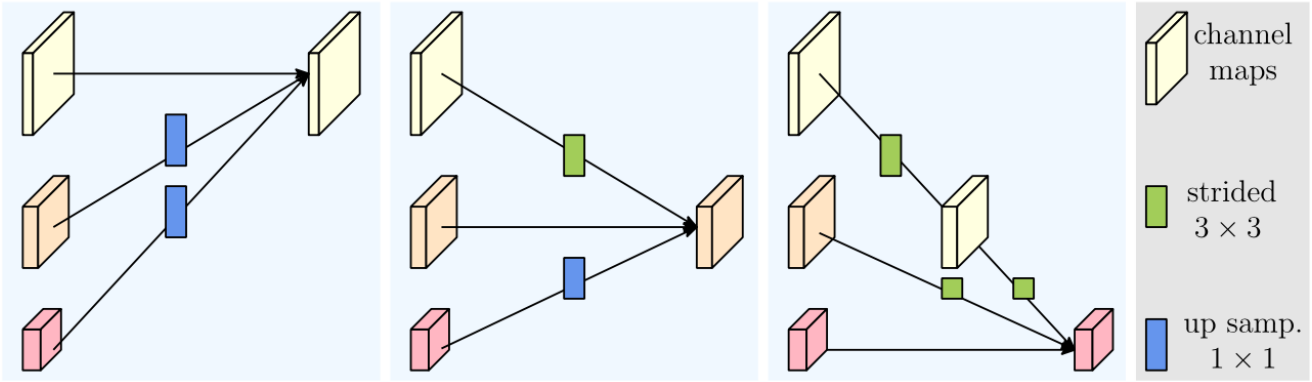


Fig. 3. Illustrating how the fusion module aggregates the information for high, medium and low resolutions from left to right, respectively. Right legend: strided 3×3 = stride-2 3×3 convolution, up samp. 1×1 = bilinear upsampling followed by a 1×1 convolution.

High-Resolution Networks

- Repeated Multi-Resolution Fusion
  - Each output representation is the sum of the transformed representations of the three inputs:

$$\mathbf{R}_r^o = f_{1r}(\mathbf{R}_1^i) + f_{2r}(\mathbf{R}_2^i) + f_{3r}(\mathbf{R}_3^i)$$

  - The choice of the transform function $f_{xr}()$ is dependent on the input resolution index x and the output resolution index r. If $x = r$, $f_{xr}(\mathbf{R}) = \mathbf{R}$. If $x < r$, $f_{xr}(\mathbf{R})$ downsamples the input representation $\mathbf{R}$ through (r-s) stride-2 3 by 3 convolutions.
  - If $x > r$, $f_{xr}(\mathbf{R})$ upsamples the input representation $\mathbf{R}$ through the bilinear upsampling followed by a 1 by 1 convolution for aligning the number of channels.

High-Resolution Networks

- Representation Head
  - HRNetV1. The output is the representation only from the high-resolution stream.
  - HRNetV2. We rescale the low-resolution representations through bilinear upsampling without changing the number of channels to the high resolution, and concatenate the four representations, followed by a 1 by 1 convolution to mix the four representations.

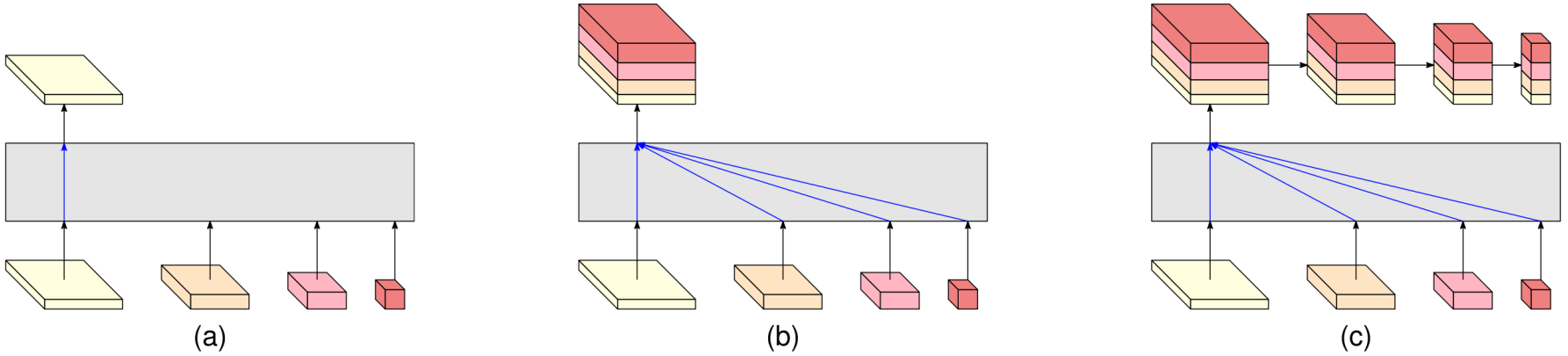


Fig. 4. (a) HRNetV1: only output the representation from the high-resolution convolution stream. (b) HRNetV2: concatenate the (upsampled) representations that are from all the resolutions (the subsequent 1×1 convolution is not shown for clarity). (c) HRNetV2p: form a feature pyramid from the representation by HRNetV2. The four-resolution representations at the bottom in each sub-figure are outputted from the network in Fig. 2, and the gray box indicates how the output representation is obtained from the input four-resolution representations.

High-Resolution Networks

- Representation Head
  - HRNetV2p. We construct multi-level representations by downsampling the high-resolution representation output from HRNetV2 to multiple levels.
  - Applying HRNetV1 to human pose estimation, HRNetV2 to semantic segmentation, and HRNetV2p to object detection

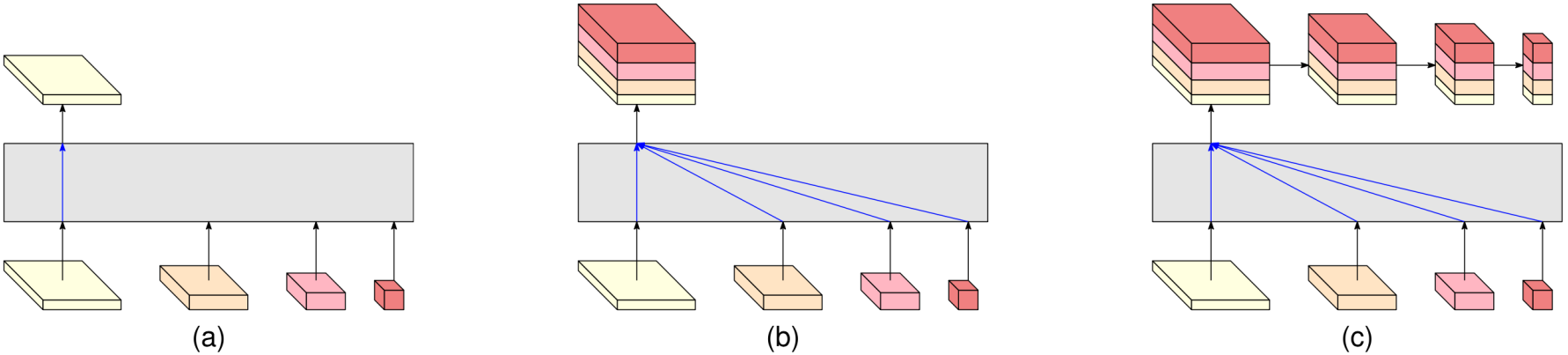


Fig. 4. (a) HRNetV1: only output the representation from the high-resolution convolution stream. (b) HRNetV2: concatenate the (upsampled) representations that are from all the resolutions (the subsequent 1×1 convolution is not shown for clarity). (c) HRNetV2p: form a feature pyramid from the representation by HRNetV2. The four-resolution representations at the bottom in each sub-figure are outputted from the network in Fig. 2, and the gray box indicates how the output representation is obtained from the input four-resolution representations.

Semantic Segmentation

- We feed the input image to HRNetV2 and then pass the resulting $15C$-dimensional representation at each position to a linear classifier with the softmax loss to predict the segmentation maps. The segmentation maps are upsampled (4 times) to the input size by bilinear upsampling for both training and testing.
- Scene parsing datasets, PASCAL-Context and Cityscapes, and a human parsing dataset, LIP.
- The mean of class-wise intersection over union (mIoU) is adopted as the evaluation metric.

Semantic Segmentation

- Cityscapes. The Cityscapes dataset contains 5,000 high quality pixel-level finely annotated scene images. The finely-annotated images are divided into 2975, 500, and 1525 images for training, validation and testing.

TABLE 3
Semantic Segmentation Results on Cityscapes `val`
(Single Scale and no Flipping)

	backbone	#param.	GFLOPS	mIoU
UNet++ [155]	ResNet-101	59.5M	748.5	75.5
Dilated-ResNet [40]	D-ResNet-101	52.1M	1661.6	75.7
DeepLabv3 [16]	D-ResNet-101	58.0M	1778.7	78.5
DeepLabv3+ [18]	D-Xception-71	43.5M	1444.6	79.6
PSPNet [148]	D-ResNet-101	65.9M	2017.6	79.7
HRNetV2	HRNetV2-W40	45.2M	493.2	80.2
HRNetV2	HRNetV2-W48	65.9M	696.2	81.1
HRNetV2 + OCR [139]	HRNetV2-W48	70.3M	1206.3	**81.6**

The GFLOPS is calculated on the input size 1024×2048. *The small model HRNetV2-W40 with the smallest GFLOPS performs better than two representative contextual methods (Deeplab and PSPNet). Our approach combined with the recently-developed object contextual representation (OCR) scheme [139] gets further improvement. D-ResNet-101 = Dilated-ResNet-101.*

Semantic Segmentation

- PASCAL-Context. The PASCAL-Context dataset includes 4,998 scene images for training and 5,105 images for testing with 59 semantic labels and 1 background label.

TABLE 5
Semantic Segmentation Results on PASCAL-Context

	backbone	mIoU (59)	mIoU (60)
FCN-8s [101]	VGG-16	-	35.1
BoxSup [24]	-	-	40.5
HO_CRF [2]	-	-	41.3
Piecewise [73]	VGG-16	-	43.3
DeepLab-v2 [15]	D-ResNet-101	-	45.7
RefineNet [72]	ResNet-152	-	47.3
UNet++ [155]	ResNet-101	47.7	-
PSPNet [148]	D-ResNet-101	47.8	-
Ding et al. [26]	ResNet-101	51.6	-
EncNet [141]	D-ResNet-101	52.6	-
DANet [31]	D-ResNet-101	52.6	-
ANN [158]	D-ResNet-101	52.8	-
SVCNet [27]	ResNet-101	53.2	-
CFNet [142]	D-ResNet-101	54.0	-
APCN [37]	D-ResNet-101	55.6	-
HRNetV2	HRNetV2-W48	54.0	48.3
HRNetV2 + OCR [139]	HRNetV2-W48	**56.2**	**50.1**

Semantic Segmentation

- LIP. The LIP dataset contains 50,462 elaborately annotated human images, which are divided into 30,462 training images and 10,000 validation images. The methods are evaluated on 20 categories (19 human part labels and 1 background label).

TABLE 6
Semantic Segmentation Results on LIP

	backbone	extra.	pixel acc.	avg. acc.	mIoU
Attention+SSL [34]	VGG16	Pose	84.36	54.94	44.73
DeepLabV3+ [18]	D-ResNet-101	-	84.09	55.62	44.80
MMAN [82]	D-ResNet-101	-	-	-	46.81
SS-NAN [150]	ResNet-101	Pose	87.59	56.03	47.92
MuLA [86]	Hourglass	Pose	**88.50**	60.50	49.30
JPPNet [69]	D-ResNet-101	Pose	86.39	62.32	51.37
CE2P [80]	D-ResNet-101	Edge	87.37	63.20	53.10
HRNetV2	HRNetV2-W48	N	88.21	67.43	55.90
HRNetV2 + OCR [139]	HRNetV2-W48	N	88.24	**67.84**	**56.48**

Conclusion

- HRNet is a stronger backbone for computer vision problems.
- (i) Connect high and low resolution convolutions in parallel other than in series;
- (ii) Maintain high resolution through the whole process instead of recovering high resolution from low resolution; and
- (iii) Fuse multi-resolution representations repeatedly, rendering rich high-resolution representations with strong position sensitivity.

Fig. 7. Qualitative segmentation examples from Cityscapes (left two), PASCAL-Context (middle two), and LIP (right two).