Appendix

In this appendix, we present additional results of our proposed approach in Sec. A, and provide additional training and implementation details in Sec. B (both for our model, and the strong MNC baseline [10]).

A Additional Results

In our main paper, we reported our AP\textsubscript{r} results averaged over all classes. Fig. 5 visualises the per-class results of our best model at different IoU thresholds. Fig. 6 displays the success cases of our method, while Fig. 7 shows examples of failure cases. Furthermore, we illustrate the strengths and weaknesses of our part instance segmentation method in comparison to MNC [10] in Fig. 8, and compare our instance-level human segmentation results, which we obtain by the simple mapping described in Sec. 3.4 of our main paper, to MNC in Fig. 9.

Finally, we attach an additional video. We run our system offline, on a frame-by-frame basis on the entire music video, and show how our method is able to accurately parse humans at both category and instance level on internet data outside the Pascal dataset. Instance-level segmentation of videos requires data association. We use a simple, greedy method which operates on a frame-by-frame basis. Segments from one frame are associated to segments in the next frame based on the IoU, using the same method we use for our loss function as described in Sec. 3.3 of the main paper.

![Figure 5: Visualisation of per-class results for different IoU thresholds on the Pascal Person-Parts test set.](image)

The heatmap shows the per-class AP\textsubscript{r} of our best model at IoU thresholds from 0.1 to 0.9 in increments of 0.1 on the Pascal Person-Parts test set. It shows that our method achieves best instance accuracy for the head category, and finds lower arms and lower legs most challenging to segment correctly. This is likely because of the thin shape of the lower limbs which is known to pose difficulty for semantic segmentation.
<table>
<thead>
<tr>
<th>Input</th>
<th>Semantic Segmentation</th>
<th>Instance Segmentation</th>
<th>Ground Truth</th>
</tr>
</thead>
</table>

Figure 6: **Success cases of our method.** The first column shows the input image and the input detections we obtained from training the R-FCN detector [11]. The second column shows our final semantic segmentation (as described in Sec. 3.4 of the main paper). Our proposed method is able to leverage an initial category-level segmentation network and human detections to produce accurate instance-level part segmentation as shown in the third column.
Figure 7: Failure cases of our method. First three rows: a missing human detection confuses the instance-level segmentation module. Fourth and fifth row: overlapping detection bounding boxes lead to incorrect instance label assignment when the overlapping region are visually similar. Sixth row: although our method is robust against false positive detections, two small regions on the leftmost person’s left arm and left knee are assigned to the false positive detection.
Figure 8: **Comparison to MNC on the Pascal Person-Parts [8] test set.** *First row:* Unlike MNC which predicts for each part instance independently, our method reasons globally and jointly. As a result, MNC predicts two instances of lower legs for the same lower leg of the second and third person from the left. Furthermore, with a dedicated category-level segmentation module, we are less prone to false negatives, whereas MNC misses the legs of the rightmost person, and the lower arm of the second person from the right. *Second row:* While we can handle poor bounding box localisation because of our global potential term, MNC is unable to segment regions outside the bounding boxes it generates. Consequently, only one lower arm of the person on the left is segmented as the other one is outside the bounding box. The square corners of the segmented lower arm correspond to the limits imposed by the bounding box which MNC internally uses (box generation is the first stage of the cascade [10]). *Third row:* By analysing an image globally and employing a differentiable CRF, our method can produce more precise boundaries. As MNC does not perform category-level segmentation over the entire image, it has no incentive to produce a coherent and continuous prediction. Visually, this is reflected in the gaps of “background” between body parts of the same person. *Fourth row:* MNC predicts two instances of lower leg for the second person from the right, and fails to segment any lower arms for all four people due to the aforementioned problems.
Figure 9: **Comparison to MNC on the Pascal Person-Parts [8] test set for instance-level human segmentation.**

To generate the results in the second column, we run the public MNC model trained on VOC 2011/SBD [19] using the default parameters and extract only its human instance predictions. In contrast with proposal-driven methods such as MNC, our approach assigns each pixel to only one instance, is robust against non-ideal bounding boxes, and often produces better boundaries due to the Instance CRF which is trained end-to-end. **First and second row:** since MNC predicts instances independently, it is prone to predicting multiple instances for a single person. **Third row:** due to the global potential term, we can segment regions outside of a detection bounding box which fails to cover the entire person, whereas MNC is unable to recover from such imperfect bounding boxes, leading to its frequent occurrences of truncated instance predictions. **Fourth row:** a case where MNC and our method show different failure modes. MNC predicts three people where there are only two, and our method can only predict one instance due to a missing detection.
Figure 9 (Continued): **Comparison to MNC on the Pascal Person-Parts [8] test set for instance-level human segmentation.** *First row:* MNC is unable to recover from a false positive detection and predicts two people. *Second row:* while both MNC and our method start off with poor bounding box localisation that does not cover the whole instance, we are able to segment the entire person, whereas MNC is bounded by its flawed region proposal. *Third row:* MNC performs better in this case as it is able to segment the infant, whereas we miss her completely due to a false negative person detection.
B Additional information

We detail our initial category-level segmentation module and compare it to DeepLab-v2 [6] in Sec. B.1, present our network training details in Sec. B.2, and finally describe how we train the MNC model which serves as our baseline in Sec. B.3.

B.1 Details of the category-level segmentation module

As shown in Fig 10b, the structure of our category-level segmentation module consists of a ResNet-101 backbone, and a classifier that extracts multi-scale features from the ResNet-101 output by using average pooling with different kernel sizes. While our category-level segmentation module and the Deeplab-v2 network (Fig. 10a) of Chen et al. [6] both attempt to exploit multi-scale information in the image, the approach of [6] entails executing three forward passes for each image, whereas we only need a single forward pass.

In comparison to Deeplab-v2, our network saves both memory and time, and achieves better performance. To carry out a single forward pass, our network uses 4.3GB of memory while Deeplab-v2 [6] needs 9.5GB, 120% more than ours. Speed-wise, our network runs forward passes at 0.255 seconds per image (3.9 fps), whereas Deeplab-v2 takes 55% longer, at 0.396 seconds per image (2.5 fps) on average. When Deeplab-v2 adds a CRF with 10 mean-field iterations to post-process the network output, it gains a small improvement in mean IoU by 0.54% [6], but it requires 11.2GB of memory to make a forward pass (140% of the total amount used by our full network including the instance-level segmentation module), and takes 0.960 seconds per image (1.0 fps), almost a quarter of our frame rate. Tests are done on a single GeForce GTX Titan X (Maxwell) card. Overall, we are able to achieve better segmentation accuracy (as shown in Tab. 3 of our main paper) and is more memory- and time-efficient than Deeplab-v2.

B.2 Training our proposed network

B.2.1 Training the category-level segmentation module

We initialise our semantic segmentation network with the COCO pre-trained ResNet-101 weights provided by [6]. Training is first performed on the Pascal VOC 2012 training set using the extra annotations from [19], which combine to a total of 9012 training images. Care is taken to ensure that all images from the Pascal Person-Parts test set is excluded from this training set. A polynomial learning rate policy is adopted such that the effective learning rate at iteration $i$ is given by $l_i = l_0 (1 - i / i_{\text{max}})^p$, where the base learning rate, $l_0$, is set to $6.25 \times 10^{-4}$, the total number of iterations, $i_{\text{max}}$, is set to 30k, and the power, $p$, is set to 0.9. A batch size of 16 is used. However, due to memory constraints, we simulate this batch size by “accumulating gradients”: We carry out 16 forward and backward passes with one image per iteration, and only perform the weight update after completing all 16 passes. We use a momentum of 0.9 and weight decay of $1 \times 10^{-4}$ for these experiments. After 30k of iterations are completed, we take the best performing model and fine-tune on the Pascal Person-Parts training set using the same training scheme as described above. Note that the parameters of the batch normalisation modules are kept unchanged in the whole learning process.

Online data-augmentation is performed during training to regularise the model. The training images are randomly mirrored, scaled by a ratio between 0.5 and 2, rotated by an angle between -10 and 10 degrees, translated by a random amount in the HSV colour space, and blurred with a randomly-sized Gaussian kernel, all on-the-fly. We observe that these techniques are effective at reducing the accuracy gap between training and testing, leading to overall higher test accuracies.
Figure 10: Comparison of the Deeplab-v2 network structure which achieves 64.9% IoU on the Pascal Person-Parts dataset [6] and our network structure. The numbers following the layer type denote the kernel size and number of filters. For pooling layers, only their kernel sizes are shown as the number of filters is not applicable. The upsampling ratios can be inferred from the context. Fig. 10a: in the Deeplab-v2 architecture, a $513 \times 513 \times 3$ input image is downsampled by two different ratios (0.75 and 0.5) to produce multi-scale input at three different resolutions. The three resolutions are independently processed by a ResNet-101-based network using shared weights (shown by the individually coloured paths). The output feature maps are then upsampled where appropriate, combined by taking the elementwise maximum, and finally upsampled back to $513 \times 513$. Fig. 10b: the category-level segmentation module proposed in this paper forwards an input image of size $521 \times 521 \times 3$ through a ResNet-101-based CNN, producing a feature map of resolution $66 \times 66 \times 2048$. This feature map is average-pooled with four different kernel sizes, giving us four feature maps with spatial resolutions $1 \times 1$, $2 \times 2$, $3 \times 3$, and $6 \times 6$ respectively. Each feature map undergoes convolution and upsampling, before being concatenated together with each other and the $66 \times 66 \times 2048$ ResNet-101 output. This is followed by a convolution layer that reduces the dimension of the concatenated features to 512, and a convolutional classifier that maps the 512 channels to the size of label space in the dataset. Finally, the prediction is upsampled back to $521 \times 521$. In both Fig. 10a and 10b, the ResNet-101 backbone uses dilated convolution such that its output at $\text{res5c}$ is at $1/8$ of the input resolution, instead of $1/32$ for the original ResNet-101 [22]. The convolutional classifiers (coloured in purple) output $C$ channels, corresponding to the number of classes in the dataset including a background class. For the Pascal Person-Parts Dataset, $C$ is 7. Best viewed in colour.

**B.2.2 Training the instance-level segmentation module**

In our model, the pairwise term of the fully-connected CRF takes the following form:

$$\psi_{\text{Pairwise}}(v_i, v_j) = \mu(v_i, v_j)k(f_i, f_j)$$

where $\mu(\cdot, \cdot)$ is a compatibility function, $k(\cdot, \cdot)$ is a kernel function, and $f_i$ is a feature vector at spatial location $i$ containing the 3-dimensional colour vector $I_i$ and the 2-dimensional position vector $p_i$ [24].
We further define the kernel as follows:

\[
k(f_i, f_j) = w^{(1)} \exp \left( -\frac{|p_i - p_j|^2}{2\theta^2_\alpha} - \frac{|I_i - I_j|^2}{2\theta^2_\beta} \right) + w^{(2)} \exp \left( -\frac{|p_i - p_j|^2}{2\theta^2_\gamma} \right)
\]  

(6)

where \(w^{(1)}\) and \(w^{(2)}\) are the linear combination weights for the bilateral term and the Gaussian term respectively. In order to determine the initial values for the parameters in the Instance CRF to train from, we carry out a random search. According to the search results, the best prediction accuracy is obtained by initialising \(w^{(1)} = 8, \ w^{(2)} = 2, \ \theta_\alpha = 2, \ \theta_\beta = 8, \ \theta_\gamma = 2\). Furthermore, we use a fixed learning rate of \(1 \times 10^{-6}\), momentum of 0.9, and weight decay of \(1 \times 10^{-4}\) for training both the instance-level and category-level segmentation modules jointly. Although we previously use the polynomial learning rate policy, we find that for training the instance-level segmentation module, a fixed learning rate leads to better results. Furthermore, our experiments show that a batch size of one works best at this training stage. Using this scheme, we train for 175k iterations, or approximately 100 epochs.

### B.3 Training Multi-task Network Cascades (MNC)

We use the publicly available Multi-task Network Cascades (MNC) framework \cite{10}, and train a new model for instance-level part segmentation using the Pascal Person-Parts dataset. The weights are initialised with the officially released MNC model\(^1\) which has been trained on Pascal VOC 2011/SBD \cite{19}. The base learning rate is set to \(1 \times 10^{-3}\), which is reduced by 10 times after 20k iterations. A total of 25k training iterations are carried out. A batch size of 8, momentum of 0.9 and weight decay of \(5 \times 10^{-4}\) are used. These settings are identical to the ones used in training the original MNC and provided in their public source code. Using these settings, we are also able to reproduce the experimental results obtained in the original MNC paper \cite{10}, and hence we believe that the MNC model we have trained acts as a strong baseline for our proposed approach.

\(^1\)https://github.com/daijifeng001/MNC