Contents lists available at ScienceDirect

Medical Image Analysis

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

Contour proposal networks for biomedical instance segmentation

Eric Upschulte[a,b,*], Stefan Harmeling[c], Katrin Amunts[a,d], Timo Dickscheid[a,b,c]

[a] *Institute of Neuroscience and Medicine (INM-1), Research Centre Jülich, Wilhelm-Johnen-Str., Jülich 52428, Germany*
[b] *Helmholtz AI, Research Centre Jülich, Wilhelm-Johnen-Str., Jülich 52428, Germany*
[c] *Institute of Computer Science, Heinrich Heine University Düsseldorf, Universitätsstr. 1, Düsseldorf 40225, Germany*
[d] *Cécile & Oscar Vogt Institute for Brain Research, University Hospital Düsseldorf, Heinrich Heine University Düsseldorf, Merowingerplatz 1A, Düsseldorf 40225, Germany*

A R T I C L E   I N F O

Article history:
Received 25 March 2021
Revised 12 January 2022
Accepted 13 January 2022
Available online 22 January 2022

MSC:
92C55
68T07
68T45
68U10

Keywords:
Cell detection
Cell segmentation
Object detection
CPN

A B S T R A C T

We present a conceptually simple framework for object instance segmentation, called *Contour Proposal Network* (CPN), which detects possibly overlapping objects in an image while simultaneously fitting closed object contours using a fixed-size representation based on Fourier Descriptors. The CPN can incorporate state-of-the-art object detection architectures as backbone networks into a single-stage instance segmentation model that can be trained end-to-end. We construct CPN models with different backbone networks and apply them to instance segmentation of cells in datasets from different modalities. In our experiments, CPNs outperform U-Net, Mask R-CNN and StarDist in instance segmentation accuracy. We present variants with execution times suitable for real-time applications. The trained models generalize well across different domains of cell types. Since the main assumption of the framework is closed object contours, it is applicable to a wide range of detection problems also beyond the biomedical domain. An implementation of the model architecture in PyTorch is freely available.

1. Introduction

1.1. Motivation

Instance segmentation is the task of labeling each pixel in an image with an index that represents distinct objects of predefined object classes. This is different from semantic segmentation, which assigns the object class itself to each pixel, and does not distinguish objects of the same type if their shapes touch or overlap. A common instance segmentation problem in biomedical imaging is the detection of cells in microscopic images, in particular for quantitative analysis. While the pixel accuracy of recent cell segmentation methods has become sufficient for many imaging setups, detection accuracy often remains a bottleneck, especially wrt. handling of touching and overlapping objects. In many biomedical applications, both accurate object detection and realistic recovery of object shape are both desirable. However, many instance segmentation methods define one unique object index per pixel, referring to the foreground object only. This results in an incomplete capture of partially superimposed objects, and consequently leads to a misrepresentation of their actual shape (as in e.g. Fig. 5g top) which in turn might impair shape-sensible downstream tasks like morphological cell analysis. To avoid such problems, instance segmentation methods with appropriate modeling of object boundaries are required.

Segmentation models should also generalize well to variations in the data distribution.This is important for small variations, which inevitably occur in practical lab settings due to different tissue samples as well as fluctuations of histological protocols and digital scanning processes (Stacke et al., 2019; Yagi, 2011). Generalizability is also important at the scale of data domains, in order to allow transfer of trained models to different experiments with manageable annotation efforts.

1.2. Related work

Pixel classifiers Instance segmentation can be achieved using a dense pixel classifier such as the U-Net (Ronneberger et al., 2015), and can be cast from a semantic segmentation solution to an instance-agnostic approach using a grouping strategy such as connected component labeling (CCL). This will group multiple pixels of the same class into non-overlapping instances. To distinguish touching instances as well, one may introduce narrow back-

[*] Corresponding author at: Institute of Neuroscience and Medicine (INM-1), Research Centre Jülich, Wilhelm-Johnen-Str., Jülich 52428, Germany.
E-mail address: e.upschulte@fz-juelich.de (E. Upschulte).

https://doi.org/10.1016/j.media.2022.102371

Table 1
Instance segmentation results for selected datasets and methods. The *F1 score* F1_τ is reported for a range of *Intersection over Union* (IoU) thresholds τ and as the average $\text{F1}_{\text{avg}} = 1/9 \sum_{\tau \in \text{T}} \text{F1}_\tau$ for thresholds $\text{T} = (0.5, 0.55, 0.6, \ldots 0.9)$.

Model	Backbone	F1_{avg}	$\text{F1}_{\tau=0.15}$	$\text{F1}_{\tau=0.6}$	$\text{F1}_{\tau=0.7}$	$\text{F1}_{\tau=0.8}$	$\text{F1}_{\tau=0.9}$
Neuronal Cell Bodies							
$\text{CPN}_{\mathcal{R}4}$-U22	U-Net	**0.55**	**0.80**	**0.74**	**0.62**	**0.40**	**0.10**
$\text{CPN}_{\mathcal{R}0}$-U22	U-Net	0.51	**0.80**	0.73	0.58	0.33	0.05
$\text{CPN}_{\mathcal{R}4}$-R50-FPN	ResNet-50-FPN	0.43	0.74	0.65	0.49	0.23	0.02
$\text{CPN}_{\mathcal{R}0}$-R50-FPN	ResNet-50-FPN	0.42	0.73	0.64	0.48	0.22	0.02
U-Net	U-Net	0.47	0.71	0.63	0.51	0.33	**0.10**
StarDist	U-Net	0.46	0.75	0.68	0.53	0.28	0.04
Mask R-CNN	ResNet-50-FPN	0.34	0.70	0.55	0.34	0.11	0.00
BBBC039							
$\text{CPN}_{\mathcal{R}4}$-U22	U-Net	**0.91**	**0.96**	**0.95**	**0.93**	**0.91**	**0.76**
$\text{CPN}_{\mathcal{R}0}$-U22	U-Net	0.90	**0.96**	**0.95**	**0.93**	0.90	0.72
$\text{CPN}_{\mathcal{R}4}$-R50-FPN	ResNet-50-FPN	0.90	0.95	0.94	**0.93**	0.90	0.74
$\text{CPN}_{\mathcal{R}0}$-R50-FPN	ResNet-50-FPN	0.90	0.95	0.94	**0.93**	0.89	0.71
U-Net	U-Net	0.89	0.95	0.93	0.92	0.88	0.71
StarDist	U-Net	0.88	0.95	0.94	0.91	0.88	0.71
Mask R-CNN	ResNet-50-FPN	0.86	0.94	0.93	0.92	0.89	0.52
Synthetic Shapes							
$\text{CPN}_{\mathcal{R}4}$-U22	U-Net	**0.90**	**0.98**	**0.98**	**0.96**	**0.89**	**0.64**
$\text{CPN}_{\mathcal{R}0}$-U22	U-Net	0.89	**0.98**	**0.98**	**0.96**	0.88	0.51
$\text{CPN}_{\mathcal{R}4}$-R50-FPN	ResNet-50-FPN	0.88	**0.98**	0.97	0.94	0.86	0.54
$\text{CPN}_{\mathcal{R}0}$-R50-FPN	ResNet-50-FPN	0.86	**0.98**	0.97	0.94	0.84	0.47
U-Net	U-Net	0.87	0.96	0.95	0.92	0.85	0.59
StarDist	U-Net	0.87	0.97	0.96	0.93	0.85	0.52
Mask R-CNN	ResNet-50-FPN	0.85	0.96	0.90	0.85	0.72	0.36

ground gaps between objects with careful per pixel loss weightings (Ronneberger et al., 2015). Improved versions define border pixels as an additional class (Chen et al., 2016; Guerrero-Pena et al., 2018; Zabawa et al., 2020). Such models have demonstrated to segment the borders of isolated objects very precisely. However, in case of crowded images, already a few falsely classified pixels can merge close-by instances and critically impair the detection result (Caicedo et al., 2019). Furthermore, this approach misrepresents object shapes in case of overlap.

Pixel classifiers coupled with shape models. To reach better robustness on crowded images, some authors proposed to couple active contour models with CNN-based segmentation models. Thierbach et al. (2018) first employed a dense pixel classifier to predict probability maps of object centroids. These maps were then thresholded to initialize a subsequent active contour segmentation. Zhang et al. (2018) suggested a scheme where a CNN is trained to explicitly predict the energy function for fitting an active contour model to a given object. The contour computation is here attached as a black box to the learning loop, so that the conversion from pixel to shape space and back is invisible to the network training, and thus not part of the actual learning. Gur et al. (2019) proposed to use a neural renderer as a differentiable domain transition from polygon to pixels, allowing a full learning path. This way they train a U-Net-like CNN that produces 2D displacement fields for polygonal contour evolution with a loss that addresses the segmentation as well as ballooning and curvature minimizing forces in the pixel domain. While this allows end-to-end training, the actual boundary representation remains hidden and is not accessible to downstream tasks.

Dense vs sparse detectors. The above-mentioned solutions have in common that they learn object masks in the pixel domain under a hidden or decoupled shape model, producing a dense classification by assigning a label to each pixel of an image. An alternative is to perform object detection by directly estimating the parameters of a contour model in its embedding space, and attaching a pixel location to the shape descriptor. This way the bounds of an entire object are concentrated at a single pixel, leading to a sparse detection scheme and forcing the model to develop an explicit internal understanding of instances. This is the main motivation of the presented work. For closed contours, pixel masks can then be obtained by rasterization. Giving direct emphasis (and possibly supervision) to the shape model, such an approach could provide a more efficient problem representation.

Bounding box regression. The de facto, standard for modeling boundaries in object detection networks are bounding boxes (Ren et al., 2017; Liu et al., 2016; Lin et al., 2017b; Redmon and Farhadi, 2018; Bochkovskiy et al., 2020; Yang et al., 2020). Here, the models predict at least four outermost object locations. This approach captures little information about the object instance beyond location, scale and aspect ratio (Jetley et al., 2017). The most established approach from this category is the Mask R-CNN (He et al., 2017). It first detects bounding boxes by regression, and then gathers image features inside the bounding box to produce pixel masks.

Regression of shape representations. More detailed shape representations have been proposed in recent years as well (Jetley et al., 2017; Schmidt et al., 2018; Miksys et al., 2019; Xie et al., 2020). Closest to our work is the approach of Jetley et al. (2017), who combined the popular YOLO architecture (Redmon et al., 2016) with an additional regression of a decodable shape representation for each object proposal. They showed that the integration of a higher-order shape reasoning into the network architecture improves generalization. In particular, it allowed to predict plausible masks for previously unseen object classes. Jetley et al. (2017) evaluated three different shape representations, namely fixed-sized binary shape masks, a radial representation, and a learned shape encoding. The binary shape masks show quantitatively worse results than the other two representations. The radial representation defines a series of offsets between an anchor pixel and points on its contour, and turned out to be inferior for common object classes in natural images. It has also been applied for cell nuclei detection in the *StarDist* architecture (Schmidt et al., 2018), which showed good detection accuracy but stays behind the pixel precision achieved by U-Nets (Ronneberger et al., 2015). StarDist was extended as *PolarMask* (Xie et al., 2020) to be applicable to multiclass problems,

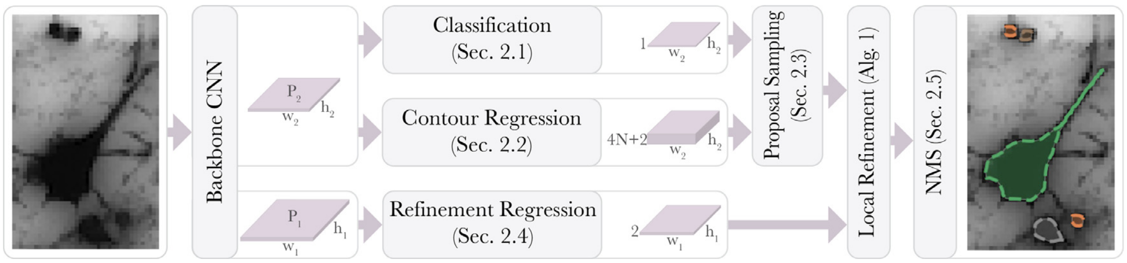
Backbone CNN
P2
w2
h2
P1
w1
h1
Classification (Sec. 2.1)
Contour Regression (Sec. 2.2)
Refinement Regression (Sec. 2.4)
1
4N+2
2
Proposal Sampling (Sec. 2.3)
Local Refinement (Alg. 1)
NMS (Sec. 2.5)

Fig. 1. The Contour Proposal Network (*CPN*) setup for instance segmentation. An initial backbone network computes feature maps P_1 and P_2. Based on the low-resolution P_2 a classification head determines for each pixel if an object is present or not, while the contour regression heads generate object contours, defined in the frequency domain, at each pixel. All contour representations that are classified to represent an object are extracted and converted to pixel space using Eq. 1. The high-resolution P_1 is used to regress a refinement tensor that is used during a Local Refinement step (Algorithm 1) that maximizes pixel accuracy. Finally, non-maximum suppression (*NMS*) is applied to remove redundant detections.

such as the COCO dataset (Lin et al., 2014). Also, it was coupled with a different loss and evaluated with multiple backbone architectures. In general, the applicability of the radial model is limited to the “star domain”, which excludes many non-convex shapes (Dietler et al., 2020). Predicting radial representations also involves a predefined number of rays, leading to possibly suboptimal sampling and limited precision of the contour (Schmidt et al., 2018). As a third representation, Jetley et al. (2017) train an auto-encoder on the Caltech-101 silhouettes dataset to learn a shape embedding for the detection network. For this encoder-decoder approach, the authors report difficulties in controlling the detail level of the embedding: Although enlarging the embedding space resulted in better preservation of shape details and reduced reconstruction errors, it had a negative effect on the overall instance segmentation performance, in fact performing below the chosen baseline approach. Notably, the same baselines were outperformed by Mask R-CNN (He et al., 2017). Miksys et al. (2019) extended this approach with an additional distance transform acting as a proxy between decoder and shape image, which allows to superimpose “discs” at every pixel location and hence mitigate the impact of falsely predicted pixels. They also considered the inclusion of the decoder in the training process. Despite showing quantitative improvements, the pixel precision remains suboptimal.

1.3. The contour proposal network

Based on existing strengths and weaknesses in the field, we here introduce the *Contour Proposal Network* (CPN). Similar in spirit to the approach of Jetley et al. (2017), it models instance segmentation as a sparse detection problem by performing regression of object shape representations at single pixel locations. The model architecture is depicted in Fig. 1: A backbone network derives feature maps from an input image. For each pixel of the feature map, regression heads generate a contour representation, while a classification head determines whether an object is present at a given location. Based on the classifications, a proposal sampling stage then extracts a sparse list of contour representations. By converting these to the pixel domain using the fully differentiable Fourier sine and cosine transformation, we implicitly enforce the contour representations to be defined in the frequency domain, inspired by Elliptical Fourier Descriptors (Kuhl and Giardina, 1982). The resulting contour coordinates are optimized by a local refinement procedure to further maximize pixel precision using a residual field, produced from an additional regression head. This is similar in spirit to the displacements fields used by Gur et al. (2019), but integrates more naturally as the CPN already operates with near-final contour proposals in the pixel domain at this stage. The complete framework is trained end-to-end across all these stages. As a final inference step, non-maximum suppression removes redundant detections from the object proposals.

It is straightforward to set up a CPN model with a different contour model (Sec. 2.2). However, Fourier Descriptors are particularly well suited, as they fulfill highly desirable properties for biomedical instance segmentation: They provide a low dimensional and fixed-size representation, can describe complex non-convex shapes with an intuitive adjustment of the level of detail (cf. Fig. 2), overcome typical sampling problems, ensure closed and continuous contours, can be scaled across different image resolutions in a lossless manner, and can be decoded into 2d coordinate space without requiring additional decoder networks.

We train CPNs that outperform U-Net, Mask R-CNN and StarDist in instance segmentation accuracy in different experiments. We demonstrate that inference speed of selected CPNs is suitable for real-time applications, especially when considering automatic mixed precision (*amp*). The trained models generalize well to datasets from different families of biological cells.

2. Methods

The Contour Proposal Network (*CPN*) uses five basic building blocks (Fig. 1). Initially, dense feature maps $P_1 \in \mathbb{R}^{w_1 \times w_1 \times c_1}$ (high-resolution) and $P_2 \in \mathbb{R}^{w_2 \times w_2 \times c_2}$ (low-resolution) are generated by a *backbone CNN* which can be freely chosen. From the latent feature map P_2, a *classifier head* detects objects, while parallel *regression heads* jointly generate explicit contour representations. The classification scores generated by the classifier estimate whether an object exists at the given locations. Contours are modelled as a series of 2d coordinates by applying the Fourier sine and cosine transformation of degree N to the latent outputs of the contour regression head, resulting in a fully differentiable, fixed-sized format for boundary regression (Sec. 2.2). The contour proposals are a dense map on the pixel grid, with an $h_2 \times w_2 \times (4N+2)$ tensor of shape descriptors and an $h_2 \times w_2 \times 1$ tensor of corresponding object classification scores. The output resolution $h_2 \times w_2$ can be different and even lower than the input resolution without sacrificing accuracy of the final contour. It effectively defines the maximum number of objects that can be detected in the image. All representations that are deemed to describe an object are extracted as a list of contour proposals and mapped to pixel space using Eq. 1. The proposals are then processed by a trainable *refinement block* to maximize fit of contours with image content using high-resolution features P_1. The last building block filters redundant detections using non-maximum suppression. The entire model is trained end-to-end.

2.1. Detection

A classification head produces a detection score for each contour representation and states whether it represents a present ob-

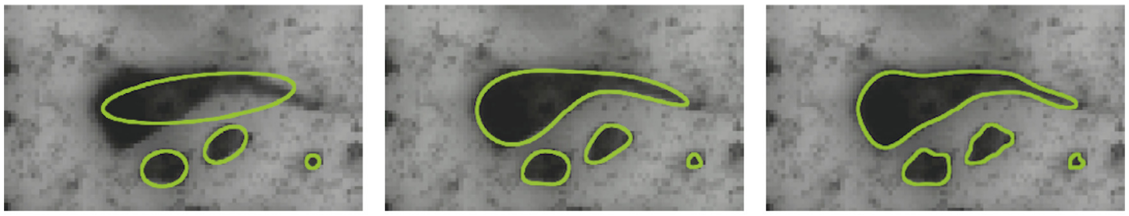

Fig. 2. Contour representation with different settings of the order hyperparameter N. It defines the vector size of the descriptor that is given by $4N + 2$. The higher the order, the more detail is preserved. The 2d contour coordinates are sampled from the descriptor space with Eq. 1. Even small settings of N yield good approximations of odd and non-convex shapes, in this case human neuronal cells, including a curved apical dendrite.

ject or not. In the present work we focus on the binary case, and do not distinguish different object categories.

2.2. Contour representation

Following Kuhl and Giardina (1982), we define a contour of degree N as a series of 2d coordinates $(x_N(t_1), y_N(t_1)), \dots (x_N(t_S), y_N(t_S))$ with $t_s < t_{s+1}$, using the Fourier sine and cosine transformation

$$x_N(t) = a_0 + \sum_{n=1}^{N} \left(a_n \sin\left(\frac{2n\pi t}{T}\right) + b_n \cos\left(\frac{2n\pi t}{T}\right) \right)$$
$$y_N(t) = c_0 + \sum_{n=1}^{N} \left(c_n \sin\left(\frac{2n\pi t}{T}\right) + d_n \cos\left(\frac{2n\pi t}{T}\right) \right) \quad (1)$$

For legibility we omit the subscript of x_N and y_N in the following. The evolution of the x-coordinate along the contour $x(t)$ is parameterized by two series of coefficients $\boldsymbol{a} = a_0, a_1, \dots a_N$ and $\boldsymbol{b} = b_1, \dots b_N$, with a_0 determining the spatial offset of the contour on the pixel grid. Accordingly, $y(t)$ is parameterized by coefficients $\boldsymbol{c}$ and $\boldsymbol{d}$. The parameter vector $[\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\ \boldsymbol{d}] \in \mathbb{R}^{4N+2}$ hence determines a 2D object contour. The location parameter $t_s \in [0, 1]$ with interval length $T = 1$ determines at which fraction of the contour a coordinate is sampled. The *order* hyperparameter N determines the smoothness of the contour, with larger N adding higher frequency coefficients and thus allowing closer approximations of object contours (Fig. 2). This formulation always produces closed contours. It is differentiable, and both the contour representation and the sampled contour coordinates are fixed in size, given an order N and a sample size S. Thus, we can directly regress the parameters of this representation to predict closed object contours with convolutional neural networks.

The CPN employs two separate regression heads for predicting contour shape $\{a_i, b_i, c_i, d_i | 1 \le i \le N\}$ and localization in the image (a_0, c_0). By isolating regression of shape and location, we intend to preserve translational invariance of the contour representation and equivariance of the offset regression.

The CPN can employ other representations than Fourier Descriptors. Generally, it only requires a contour description in vector format that is either defined in a 2d coordinate space, or can be converted to it in a differentiable fashion to ensure end-to-end learning. The coordinate representation is essential to perform the local refinement. The possibly simplest alternative to Fourier Descriptors would be given by bounding boxes, which provide a very coarse contour description defined by two vertices. In this case, the CPN predicts vectors with four values that define the two edge points of a bounding box. A list of box proposals can be sampled, refined and filtered with non-maximum suppression in the same way as Fourier Descriptor proposals, which is described in the following sections.

2.3. Proposal sampling

The classification head decides for each pixel whether an object is present. Proposal sampling then extracts all contour representations from object locations that were identified. During inference this is done using the classification results, and during optimization using the training targets of the classifier. The result is a list of contour proposals, still in Fourier space. As the subsequent stages in the CPN work with 2d coordinates, Eq. (1) is used to convert the sampled Fourier representations to pixel coordinates.

2.4. Local refinement

Some cues for optimal boundary placement which can be inferred from the training data are more efficiently encoded at the pixel level than in a shape space. As a trivial example, many segmentation problems imply that boundaries should be placed at high gradients, independent of the boundary shape. To provide an additional mechanism of encoding such knowledge efficiently and further maximizing the pixel-precision of estimated contours, we propose a *local refinement* of predicted contour coordinates in the pixel domain as part of the model architecture, as defined in Algorithm 1.

Algorithm 1 Local Refinement. Iteratively refine a single contour coordinate x, y using refinement tensor $\boldsymbol{v} \in \mathbb{R}^{w \times h \times 2}$ and maximum correction margin σ, assuming $1 \le x \le w$ and $1 \le y \le h$. Rounding is denoted by $\lfloor \cdot \rceil$.

```
1: procedure REFINE(x, y, $\boldsymbol{v}$, r, $\sigma$)
2:     for $r$ iterations do
3:         $[x \quad y] \leftarrow [\lfloor x \rceil \quad \lfloor y \rceil] + \sigma \tanh(\boldsymbol{v}_{\lfloor x \rceil, \lfloor y \rceil})$
       return $x, y$
```

The local refinement provides an additional trainable mechanism to further fit the predicted series of 2d coordinates towards their targeted ground truth coordinates. To this end we use a regression head (e.g. 2 convolutional layers) to generate a two-channel feature map $\boldsymbol{v} \in \mathbb{R}^{w \times h \times 2}$ and effectively look up the values of $\boldsymbol{v}$ at the location described by each respective coordinate. Given a single contour coordinate x, y, the 2d vector $\boldsymbol{v}_{\lfloor x(t) \rceil, \lfloor y(t) \rceil}$ can be added to the rounded coordinate vector $\lfloor x \rceil, \lfloor y \rceil$.[1] As this is optimized to shift coordinates closer to the target, $\boldsymbol{v}$ is implicitly trained to compensate for remaining misalignments of boundaries, as illustrated in Fig. 3. To limit the influence of the refinement, we use a *tanh* activation, scaled by σ. The entire local refinement procedure, consisting of a small regression head and Algorithm 1, is trained simultaneously with all other trainable components of the CPN. This way the CPN can be trained efficiently and end-to-end.

[1] Note that the use of rounded coordinates prevents contour proposal heads from being influenced by the refinement head during training. Also the rounding provides a consistent starting point for the refinement.

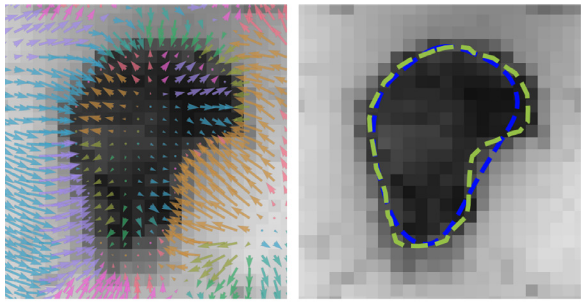
(a) Refinement tensor $\boldsymbol{v}$
(b) Refinement: before (*blue*), after (*green*)

Fig. 3. Local refinement example. (a) illustrates the learned refinement tensor $\boldsymbol{v}$ as a vector field, superimposed with the input image. (b) shows a contour proposal before and after refinement. The refinement tensor learned to shift contour coordinates to maximize pixel-precision. (Best viewed in color).

When applied multiple times, the corrective refinement step shifts contour coordinates towards a targeted location incrementally. A contour coordinate has then reached its final position once its correction yields zero for all spatial dimensions. Even if applied multiple times, the two channel refinement tensor $\boldsymbol{v}$ is only computed once. The actual refinement becomes tractable since the CPN directly outputs boundary coordinates. In fact, the procedure can be efficiently implemented using *fancy indexing*, keeping the computational cost for this procedure relatively low. The local refinement reduces localization errors of the contour regression and can compensate the exclusion of higher contour frequencies by choosing small values for the order hyperparameter N.

2.5. Non-maximum suppression

Similar to other object detection methods (He et al., 2017; Redmon et al., 2016; Lin et al., 2017b) the CPN generates dense proposals, thus multiple pixels of the produced output grid may represent the same object. To remove redundant detections during inference, we apply bounding-box non-maximum suppression (*NMS*). NMS specifically keeps proposals with a high detection score, but suppresses proposals with lower scores and a bounding box *IoU* (Intersection over Union) that exceeds a given threshold. The required bounding boxes can be retrieved efficiently for each contour proposal $(x(t_1), y(t_1)), \dots (x(t_S), y(t_S))$ as $b = [\min x(\boldsymbol{t}), \min y(\boldsymbol{t}), \max x(\boldsymbol{t}), \max y(\boldsymbol{t})]$. It turns out that the redundant proposals produced by the CPN are typically almost identical, so that the choice of the NMS scheme is not critical (see Sec. 3.7).

2.6. Loss functions

We define objectives for two components: Detection score and contour prediction. For legibility we present objectives per pixel.

Detection score. The detection head performs binary classification for each pixel, producing a score that states whether an object instance is present or not at the pixel location. The loss $\mathcal{L}_{\text{inst}}$ for this task is the standard Binary Cross Entropy (*BCE*).

Contour coordinate loss. At each pixel where a contour should be attached, we apply a loss that minimizes the distance between ground truth contour coordinates and estimated coordinates. For a single coordinate it is given by

$$\mathcal{L}_{\text{coord}}(x, y, \hat{x}, \hat{y}) = \frac{1}{2}(|x - \hat{x}|_1 + |y - \hat{y}|_1). \tag{2}$$

The contour proposal prediction is trained using

$$\mathcal{L}_{\text{contour}} = \frac{1}{S}\sum_{s=1}^{S} \mathcal{L}_{\text{coord}}(x(t_s), y(t_s), \hat{x}(t_s), \hat{y}(t_s)) \tag{3}$$

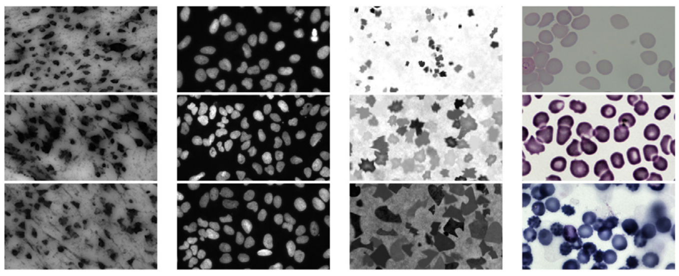

Fig. 4. Examples from the used datasets (Sec. 3.1).

with ground truth contour coordinate $x(t_s), y(t_s)$ and estimated $\hat{x}(t_s), \hat{y}(t_s)$, at random positions $t_s \in [0, 1]$. Coordinates are defined as in Eq. 1 given targets $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}$ and estimates $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}$ and $\hat{\boldsymbol{d}}$. Local refinement is trained accordingly with

$$\mathcal{L}_{\text{refine}} = \frac{1}{S}\sum_{s=1}^{S} \mathcal{L}_{\text{coord}}\Big(x(t_s), y(t_s), Refine(\hat{x}(t_s), \hat{y}(t_s))\Big) \tag{4}$$

substituting $\hat{x}(t_s), \hat{y}(t_s)$ with refined coordinates using Algorithm 1.

Representation loss. Additionally, we can directly supervise the shape parameters in the frequency domain using

$$\mathcal{L}_{\text{repr}} = |\boldsymbol{\beta} \odot (\boldsymbol{a} - \hat{\boldsymbol{a}})|_1 + |\boldsymbol{\beta}_{-0} \odot (\boldsymbol{b} - \hat{\boldsymbol{b}})|_1 + |\boldsymbol{\beta} \odot (\boldsymbol{c} - \hat{\boldsymbol{c}})|_1 + |\boldsymbol{\beta}_{-0} \odot (\boldsymbol{d} - \hat{\boldsymbol{d}})|_1 \tag{5}$$

with $\boldsymbol{\beta}_{-0}$ denoting the exclusion of β_0 from $\boldsymbol{\beta} = [\beta_0, \dots \beta_N]$. While the objective is already well defined without this representation loss, it provides additional regularization of the shape space and enables to emphasize specific detail levels by applying individual factors β_n. An intuitive setting decreases β_n with growing n to put more relative emphasis on the coarse contour outlines represented by low frequency coefficients.

CPN loss. Combining the components above, the overall per pixel loss is given by

$$\mathcal{L}_{\text{CPN}} = \mathcal{L}_{\text{inst}}(o) + o(\mathcal{L}_{\text{contour}} + \mathcal{L}_{\text{refine}} + \lambda\mathcal{L}_{\text{repr}}) \tag{6}$$

with $o = 1$ for pixels that represent an object and $o = 0$ otherwise.

3. Experiments and results

We evaluate the instance segmentation performance of the CPN on three datasets (NCB, BBBC039, SYNTH) and compare the results with U-Net and Mask R-CNN as baseline models. Also, the cross-dataset generalization performance is examined by training models on BBBC039 and testing them on a fourth dataset, BBBC041. To better understand the effects of employing the CPN feature space, this experiment includes a U-Net that is first trained as the backbone of a CPN. Finally, we compare inference speeds of different models.

3.1. Datasets

NCB - Neuronal Cell Bodies. This dataset consists of 82 grayscale image patches from microscopic scans of cellbody-stained brain tissue sections, with annotations of approximately 29,000 cell bodies. Fig. 4a shows examples. It includes significant variations in cell shape, intensity, and object overlap, as well as challenging configurations like occlusions, noise, varying contrast and histological artifacts. Brain samples come from the body donor program of the Anatomical Institute of Düsseldorf in accordance with legal and ethical requirements.[2] Tissue sections were stained using

[2] ethics approval #4863

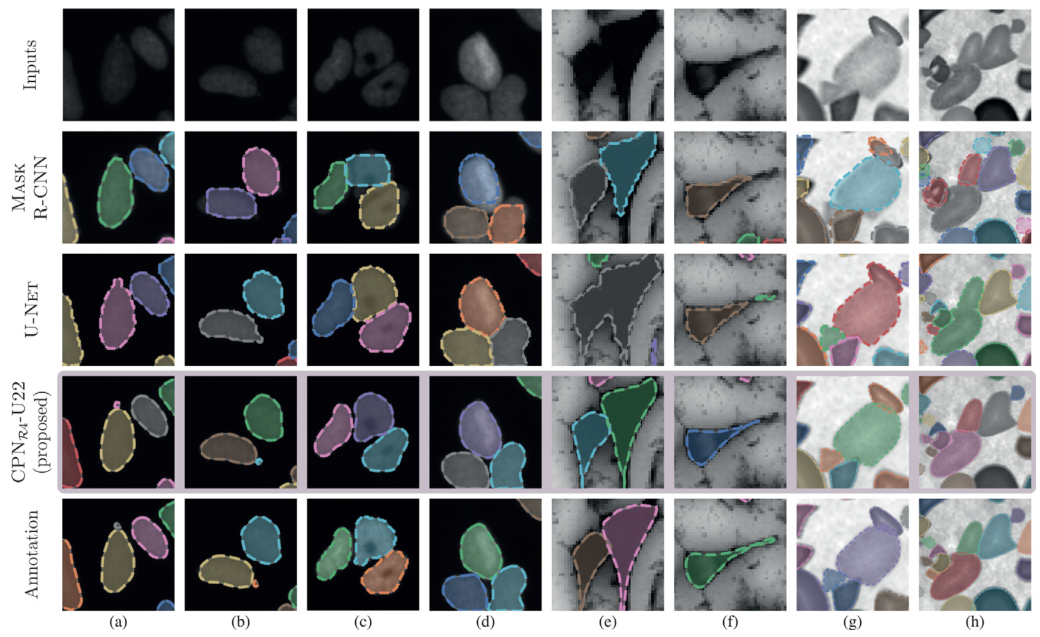

Fig. 5. Example patches from different datasets with reference annotations (5th row) and detections computed by the proposed CPN$_{\mathcal{R}4}$-U22 (4th row), U-Net (3rd row) and Mask R-CNN (2nd row) models.

a modified Merker stain (Merker, 1983). Each tissue section has an approximate thickness of 20 μ and is captured with a resolution of 1 μ using a high-throughput light-microscopic scanner (TissueScope HS, Huron Digital Pathology Inc.). Note that cell bodies in this dataset are always continuously and fully annotated even under occlusions, in order to allow a model to learn mostly realistic morphologies. Image patches were manually labeled for cell body instances by a group of experts in our institute. This was performed using a custom web-based annotation software, which allowed to enter overlapping pixel labels and to inspect the 3d context provided by depth focusing. To minimize highly subjective annotations of ambiguous cases, the software includes collaborative feedback features that allow consensus among multiple experts during annotation. The dataset is publicly accessible via the associated GitHub repository.[3]

BBBC039 - *Nuclei of U2OS cells in a chemical screen.* This is a dataset from the *Broad Institute Bioimage Benchmark Collection* (Ljosa et al., 2012). It consists of 200 grayscale images from a high-throughput chemical screen on U2OS cells, depicting approximately 23,000 annotated nuclei. Fig. 4b shows examples.

BBBC041 - *P. vivax (malaria) infected human blood smears.* Also from the Broad Bioimage Benchmark Collection (Ljosa et al., 2012), this dataset consists of 1364 images depicting approximately 80,000 Malaria infected human blood smear cells, annotated with bounding boxes. Fig. 4d shows examples.

SYNTH - *Synthetic shapes.* This dataset consists of 4129 grayscale images that show a large variety of different synthetic shapes in different sizes. It contains approximately 1,305,000 annotated objects. Fig. 4c shows examples. Shapes include simple structures, such as circles, ellipses and triangles, as well as more complex non-convex structures. Objects and background vary in intensity and texture, with objects showing mostly darker intensities than background. Similar to the NBC dataset mentioned above, objects can overlap and are fully annotated, even if occluded. Thus, a single pixel can belong to more than one instance.

[3] git.io/JOnWX

3.2. Baseline methods

We evaluate performance against the following baseline methods:

- U-Net (Ronneberger et al., 2015) is an encoder-decoder network with lateral skip-connections and a de facto standard for biomedical image segmentation (cf. Sec. 1.2). In addition to its original definition we use batch normalization after each convolutional layer. Following Caicedo et al. (2019), the network classifies each pixel into one of three classes: cell, background and boundary.
- Mask R-CNN (He et al., 2017) is a widely used instance segmentation method that proposes bounding boxes for each object, filters proposals by non-maximum suppression and finally produces masks based on proposed bounding box regions (cf. Sec. 1.2). The implementation used in our experiments is based on *torchvision*, a Python package that includes popular model architectures and is part of *PyTorch*.
- StarDist (Schmidt et al., 2018) is a popular cell detection method that uses star-convex polygons to sparsely represent object shapes (cf. Sec. 1.2). Similar to CPN and Mask R-CNN, it applies non-maximum suppression to remove redundant detections. We use the official implementation of Schmidt et al. (2018) that is available on GitHub.

3.3. CPN training

For comparability, we instantiate CPN models with the same backbone architectures as the baseline models, and train them with the same number of epochs, data size, batch size and data augmentation. In particular, we use four CPN variants:

- CPN$_{\mathcal{R}4}$-R50-FPN uses a Feature Pyramid Network (*FPN*) (Lin et al., 2017a) with a 50 layer residual architecture (He et al., 2016, ResNet-50) as its backbone and applies 4 iterations of contour refinement (Sec. 2.4)
- CPN$_{\mathcal{R}0}$-R50-FPN is CPN$_{\mathcal{R}4}$-R50-FPN with contour refinement disabled

Table 2
Cross-dataset evaluation of object detection performance. We report F1 scores for models trained on BBBC039 dataset and tested on BBBC041 dataset. Results are based on bounding boxes using same metrics as Table 1. The pretrained U-Net was trained as part of CPN$_{\mathcal{R}4}$-U22.

Model	Backbone	$F1_{avg}$	$F1_{\tau=0.5}$	$F1_{\tau=0.6}$	$F1_{\tau=0.7}$	$F1_{\tau=0.8}$	$F1_{\tau=0.9}$
CPN$_{\mathcal{R}4}$-U22	U-Net	**0.54**	**0.83**	**0.81**	**0.70**	0.29	0.02
U-Net	Pretrained U-Net	0.52	0.72	0.71	0.67	**0.39**	**0.07**
U-Net	U-Net	0.45	0.62	0.60	0.57	0.33	0.06
Mask R-CNN	ResNet-50-FPN	0.49	0.77	0.75	0.66	0.24	0.02

- CPN$_{\mathcal{R}4}$-U22 uses a 22 layer U-Net as a backbone, which is set up like the baseline described in Sec. 3.2, but omitting its final output layer. It uses 4 iterations of contour refinement
- CPN$_{\mathcal{R}0}$-U22 is CPN$_{\mathcal{R}4}$-U22 with contour refinement disabled

For assessing inference speed, we use additional backbone architectures (Sec. 3.6).

We supervise both the contour representation and the sampled contour coordinates. As the contour representation is well defined, we calculate ground truth representations on the fly and use them to guide the network during training. Using Eq. 1 with uniform sampling $t_1, \ldots t_S$ ($t_s \in [0, 1]$) we retrieve contour coordinates from both ground truth and prediction for supervision. The sample size hyperparameter S influences precision and performance by fixing the number of coordinates used during training. We choose $S = 64$ here.

While it is also possible to use the derivable and non-parametric formula from Kuhl and Giardina (1982) to derive the contour representation from another latent space, we did not observe any benefits and thus omit this possibility.

During training, we dedicate the center region of each object to contain the contour representations. The size of such a region is defined as a fraction of the object size. Overlapping regions are characterized as ambiguous and excluded. Generally, it is necessary to assign at least one pixel in proximity to each object.

3.4. Detection and segmentation performance

To evaluate the detection performance and the shape quality of the produced contours we use the harmonic mean of precision and recall $F1_\tau = TP_\tau / (TP_\tau + \frac{1}{2}(FP_\tau + FN_\tau))$ for different *Intersection over Union* (IoU) thresholds τ. The IoU threshold $\tau \in [0, 1]$ defines the minimal IoU that is required for two shapes to be counted as a match. Each ground truth shape can be a match for at most one predicted shape. A true positive (TP) is a predicted shape that matches a ground truth shape, a false positive (FP) is a shape that does not match any ground truth shape and a false negative (FN) is a ground truth shape that does not match any predicted shape. $F1_\tau$ scores with a small $\tau = 0.5$ quantify the coarse detection performance of a model, yielding good scores if the model correctly infers object presence along with a roughly matching contour. $F1_\tau$ scores with a larger $\tau = 0.9$ quantify the fine detection performance, allowing little deviance from the target shape. We define $F1_{avg} = 1/9 \sum_{\tau \in T} F1_\tau$ for thresholds $T = (0.5, 0.55, 0.6, \ldots 0.9)$, to measure the average performance for different thresholds.

Table 1 shows quantitative results of the CPN, U-Net, Mask R-CNN and StarDist on three different datasets. The CPN with local refinement yields highest scores on all datasets. Local refinement further increases the average F1 scores, especially for high thresholds τ, thus increasing the quality of the contours as expected. On the datasets BBBC039 and SYNTH CPN$_{\mathcal{R}4}$-U22 outperforms the baseline models for all thresholds.

3.5. Cross-dataset generalization

We assessed how well the baseline and CPN models generalize to variations in the input data distribution as follows: Models are trained for instance segmentation on the BBBC039 dataset. Without any retraining or adaptation, models are then applied to BBBC041. Generalization capabilities are then evaluated with the F1 score on the basis of bounding boxes derived from the respective instance segmentation results. This provides a quantitative characterization of detection and segmentation performance under transfer to different data domains. To comply with the basic characteristics of BBBC039, we converted the images to inverted grayscale images and applied a fixed contrast adjustment and downscaling.

Results are shown in Table 2. CPN models consistently show higher scores than baseline methods. For small IoU thresholds such as $\tau = 0.50$, the scores of CPN and Mask R-CNN models are particularly distinguished from U-Net. Fig. 6 shows detected instances from different methods on two typical examples, illustrating the problems in cross-dataset generalization. CPN$_{\mathcal{R}4}$-U22 tends to detect conservatively, preferring some false negatives for avoiding false positives. U-Net shows more false positives, sometimes seemingly detecting noise. For Mask R-CNN, contours are less precise than for the others, and overall less true instances are detected.

Reusing trained CPN backbones for different tasks We also examined the generalization performance of a U-Net when its encoder and decoder are trained as the backbone of CPN$_{\mathcal{R}4}$-U22, and then reused with a new final prediction layer in a U-Net to output segmentation results. For retraining, the encoder part is frozen, to ensure that the CPN feature space is kept intact. This case is reported in the second row of Table 2. All scores are significantly higher compared to the U-Net without such pre-training. For high IoU thresholds, e.g. $\tau \in \{0.8, 0.9\}$, this variant even provides overall highest scores. However, as the generalization performance increased, the $F1_{avg}$ score on the BBBC039 test set dropped from 0.89 to 0.87.

3.6. Inference speed

We computed the number of frames per seconds (*FPS*) on the BBBC039 test set for different models. Each image has a size of 520×696. To improve the precision of the measurement we reiterated over the test set multiple times. Pre- and post-processing steps were excluded from the timings, as well as initial warm-up runs. This experiment was performed in single-precision (*float32*) and automatic mixed precision (*amp*) via PyTorch's *autocast* feature. The latter automatically selects CUDA operations to run in half-precision (*float16*) to improve performance while aiming to maintain accuracy.

Results are presented in Table 3. In terms of inference speed, CPN$_{\mathcal{R}4}$-R50-FPN outperforms both Mask R-CNN-R50-FPN and U-Net when applied with normal single-precision (*float32*). As this CPN reached 29.9 FPS, it qualifies for many online video processing applications. When applied with automatic mixed precision (*amp*)

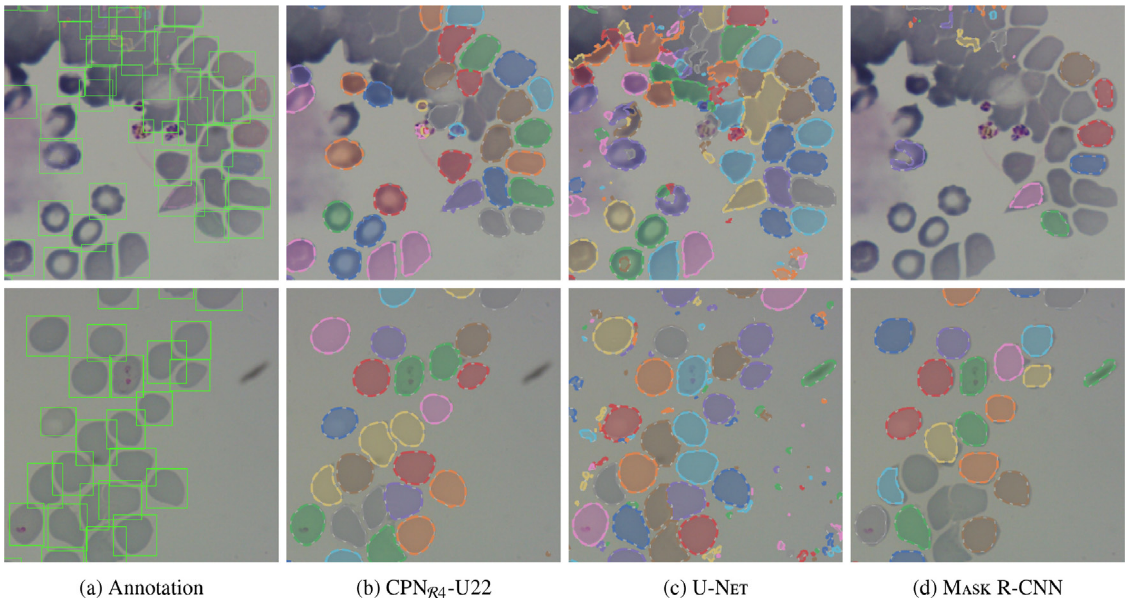

Fig. 6. Cross-dataset generalization examples from three different models. The models were trained on BBBC039 and applied to images from BBBC041 without retraining or adaptation. Two samples are depicted above.

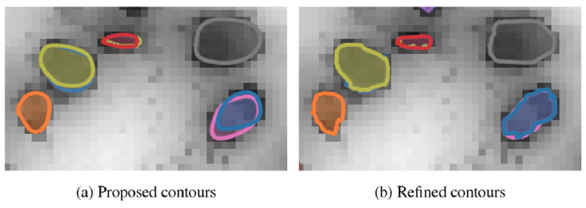

Fig. 7. Local refinement equalizes redundant contour proposals. Both images illustrate results of a CPN without non-maximum suppression. All contour proposals are superimposed. (a) shows proposals before refinement, (b) shows the same proposals after refinement. Redundant contour proposals that are initially only similar are typically almost identical after the refinement step. (Best viewed in color).

Table 3
Inference speeds of different models. We report the number of frames per second (*FPS*) for the BBBC039 test set with an image size of 520 × 696. We measure times with single-precision (*float32*) and and automatic mixed precision (*amp*). The initial run and possible post-processing steps are excluded. All models were implemented and executed as PyTorch models on an NVIDIA A100. We denote ResNet by 'R', ResNeXt by 'X' and U-Net by 'U' for brevity.

Model	FPS	FPS (amp)
CPN$_{\mathcal{R}0}$-R50-FPN	30.19	37.57
CPN$_{\mathcal{R}4}$-R50-FPN	29.86	36.17
CPN$_{\mathcal{R}0}$-X50-FPN	27.20	36.83
U-Net	23.42	77.71
CPN$_{\mathcal{R}4}$-U22 (P_2 stride 2)	15.41	42.20
Mask R-CNN-R50-FPN	13.74	-
CPN$_{\mathcal{R}4}$-X101-FPN	13.39	25.66
CPN$_{\mathcal{R}4}$-U22	12.71	26.72

the CPN$_{\mathcal{R}4}$-U22, that uses a stride of 2 in the classification and regression head, achieved 42.2 FPS - the highest performance of the tested CPN models and the second highest overall performance among the tested models. U-Net, which shares the same backbone, showed the best inference speed performance using amp.

The influence of local refinement on inference speed was evaluated for the R50-FPN based CPN, for which four refinement iterations reduced the result by 0.33 FPS, when used with single-precision (*float32*).

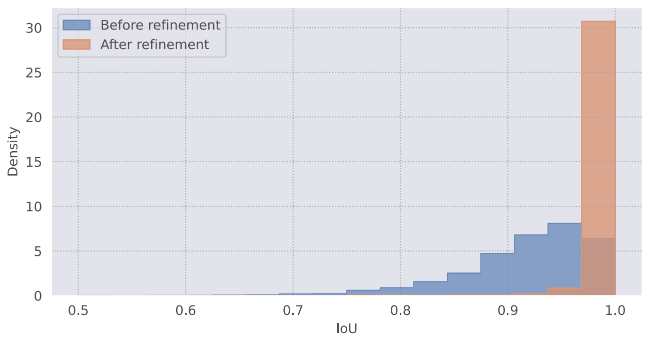

Fig. 8. Similarity of redundant contour proposals. For each contour that is selected by non-maximum suppression we measure the average Intersection over Union with respect to redundant proposals, both before and after refinement. The histogram illustrates the results for 11k neuronal cell bodies. The refinement equalizes redundant contours, shifting the distribution of IoUs towards 1. This reduces the influence that choices of NMS might have on the model's quantitative performance.

3.7. Influence of non-maximum suppression

Non-maximum suppression is an integral part of many object detection pipelines and its choices can have significant impact on a model's quantitative performance (Bodla et al., 2017). To understand the influence that the NMS scheme may have on a particular CPN model, we investigate the similarity of contours that are deemed to be redundant. This is relevant, as choices between similar contour proposals are less critical than choices between dissimilar ones. Fig. 8 shows results for the detection of neuronal cell bodies. For each contour that is selected by non-maximum suppression, we measured the average Intersection over Union with respect to redundant proposals both before and after refinement. The histogram shows that contour proposals are often similar with a majority of IoUs larger than 0.85 before refinement is applied. The refinement further equalizes similar contours with a majority of IoUs close to the maximum of 1. This effect is also illustrated in Fig. 7.

4. Discussion and conclusion

We present the Contour Proposal Network (CPN), a framework for segmenting object instances by proposing contours which are encoded as fixed-sized representations based on Fourier Descriptors. CPN models can be constructed with different backbone CNN architectures to produce image features. We assessed the performance of four different CPN variants, employing both U-Net and ResNet-FPN backbones, against U-Net, Mask R-CNN and StarDist as baselines. All U-Net based CPNs outperformed the U-Net counterpart in terms of $F1_{avg}$ instance segmentation performance on all three tested datasets, both with and without local refinement. Given that CPN and U-Net share the same backbone architecture U22, the results indicate that the CPN provides a more effective problem description. This is also supported by the comparison of the CPN and Mask R-CNN in our experiments. For both tested backbone architectures, the CPNs showed consistently higher $F1_{avg}$ than Mask R-CNN-R50-FPN. Given the same U-Net backbone, CPN outperformed StarDist, both with and without local refinement. This confirms that the Fourier based contour representation is a suitable choice for the CPN.

The CPN models employ a highly entangled representation of object shapes and sparse detections: The networks are effectively forced to concentrate the description of a complete object into a single pixel by anchoring the boundary representation to a specific coordinate. This requires them to form an intrinsic spatial relationship between whole objects and their parts, which encourages compact and robust representations with good generalization properties. This principle shares some commonalities with *Capsule Networks* (Sabour et al., 2017), which also aim to condense instances of objects or object parts into vector representations coupled with a detection score. The effects can be observed when looking at examples such as the ones depicted in Fig. 5: If boundaries are invisible or poorly defined, CPN models exploit the learned knowledge of boundary shapes to find highly plausible separations (e.g. Fig. 5 e, g, h). Very small and touching objects, which are often overseen by pixel-based methods, are well detected (e.g. Fig. 5 a, b). Separation of clusters of kissing objects is typically modelled quite accurately, reproducing gaps between touching shapes very consistently (e.g. Fig. 5 c, d). Thin structures, such as dendrites, can be modeled accurately (e.g. Fig. 5f). Discontinuities that may occur with pixel-based methods can be avoided, as proposed contours are continuous and closed by design.

While leading to accurate object representations, experiments on cross-dataset generalization showed that the learned shape priors are not overly restrictive and transfer well to different data distributions. CPN models were even able to produce plausible contours for previously unseen objects as long as their basic morphology is consistent with the training examples. In particular the $F1_{\tau=0.50}$ margin between $CPN_{\mathcal{R}4}$-U22 and U-Net of 0.21 suggests that the better performing CPN formed a more universal intrinsic understanding of what an instance is. In this context, we also observed that the CPN training has a positive influence on the backbone CNNs to produce a feature space with good generalization properties: A pixel-classifying U-Net showed significantly better performance on our cross-dataset evaluation when its encoder and decoder were trained as a CPN backbone.

By modeling the contour representation in the frequency domain, CPNs can bypass several sampling problems occurring in previous works, like selecting the optimal sampling rate in pixel space. Instead, by setting the order of the Fourier series, the user can specify different levels of contour complexity in a natural way. Furthermore, the representation allows to generate arbitrary output resolutions without compromising detection accuracy.

The local refinement step, which is an integrated and fully trainable part of the CPN, supports contour proposals to achieve high pixel precision despite the regularization imposed by the shape model. We measured a notable increase in performance for high IoU thresholds when applying refinement, indicating that high contour frequencies can be modeled efficiently using a residual field. While the refinement can improve contour details, it only had a minor influence on inference speed in our experiment. We see the refinement as an important complementary module of the CPN framework.

Non-maximum suppression (NMS) is an important part of the CPN, as it removes redundant proposals. In general, there is no guarantee that proposals with a high detection score are the best fitting ones. Bodla et al. (2017) show that NMS choices can in fact significantly influence quantitative model performance. However, the results in Sec. 3.7 demonstrate that for the CPN, local refinement leads to a strong equalization of redundant contour proposals in practice. As a consequence, the NMS step is here mostly required to pick one of several almost identical proposals. The particular choices of NMS thus have only little influence on the model performance for the CPN.

In terms of inference speed, $CPN_{\mathcal{R}4}$-R50-FPN outperforms all other tested methods when applied with normal single-precision (*float32*). With 29.9 FPS and an image size of 520×696 it is even suitable for online processing tasks, especially as it produces ready-to-use object instance descriptions, not requiring additional post-processing steps like connected component labeling. The experiments also showed that local refinement adds little time overhead, in the case of R50-FPN based CPN four refinement iterations increased pixel-precision while costing less than half a frame per second. For automatic mixed precision $CPN_{\mathcal{R}4}$-U22 with strided heads showed fastest inference speed among all CPNs with 42.2 FPS.

Since the only assumption of the proposed approach are closed object contours, it is applicable to a wide range of detection problems, also beyond the biomedical domain, that have not been investigated in the present work.

An implementation of the model architecture in PyTorch is available at https://github.com/FZJ-INM1-BDA/celldetection.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests.

CRediT authorship contribution statement

Eric Upschulte: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Stefan Harmeling:** Writing – review & editing, Supervision. **Katrin Amunts:** Conceptualization, Resources, Writing – review & editing, Supervision, Funding acquisition. **Timo Dickscheid:** Conceptualization, Methodology, Investigation, Resources, Writing – original draft, Project administration, Funding acquisition, Supervision.

Acknowledgments

This project received funding from the European Union's Horizon 2020 Research and Innovation Programme, grant agreement 945539 (HBP SGA3), and Priority Program 2041 (SPP 2041) "Computational Connectomics" of the German Research Foundation (DFG), as well as the Helmholtz Association's Initiative and Networking Fund through the Helmholtz International BigBrain Analytics and Learning Laboratory (HIBALL) under the Helmholtz International Lab grant agreement InterLabs-0015. Computing time was granted through JARA-HPC on the supercomputer JURECA at Juelich Supercomputing Centre (JSC) as part of the project CJINM14.

References

Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y. M., 2020. Yolov4: optimal speed and accuracy of object detection. 2004.10934.

Bodla, N., Singh, B., Chellappa, R., Davis, L.S., 2017. Soft-nms - improving object detection with one line of code. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 5562–5570. doi:10.1109/ICCV.2017.593.

Caicedo, J.C., Roth, J., Goodman, A., Becker, T., Karhohs, K.W., Broisin, M., Molnar, C., McQuin, C., Singh, S., Theis, F.J., et al, 2019. Evaluation of deep learning strategies for nucleus segmentation in fluorescence images. Cytom. Part A 95 (9), 952–965. doi:10.1002/cyto.a.23863.

Chen, H., Qi, X., Yu, L., Heng, P.-A., 2016. Dcan: deep contour-aware networks for accurate gland segmentation. 1604.02677.

Dietler, N., Minder, M., Gligorovski, V., Economou, A.M., Joly, D.A.H.L., Sadeghi, A., Chan, C.H.M., Koziski, M., Weigert, M., Bitbol, A.-F., Rahi, S.J., 2020. A convolutional neural network segments yeast microscopy images with high accuracy. Nat. Commun. 11 (1), 5723. doi:10.1038/s41467-020-19557-4. http://www.nature.com/articles/s41467-020-19557-4

Guerrero-Pena, F.A., Marrero Fernandez, P.D., Ing Ren, T., Yui, M., Rothenberg, E., Cunha, A., 2018. Multiclass weighted loss for instance segmentation of cluttered cells. In: Proceedings of the 25th IEEE International Conference on Image Processing (ICIP) doi:10.1109/icip.2018.8451187. doi: 10.1109/ICIP.2018.8451187

Gur, S., Shaharabany, T., Wolf, L., 2019. End to end trainable active contours via differentiable rendering. arXiv:1912.00367.

He, K., Gkioxari, G., Dollar, P., Girshick, R., 2017. Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). IEEE, pp. 2980–2988. doi:10.1109/ICCV.2017.322. http://ieeexplore.ieee.org/document/8237584/

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. doi:10.1109/CVPR.2016.90.

Jetley, S., Sapienza, M., Golodetz, S., Torr, P.H.S., 2017. Straight to shapes: real-time detection of encoded shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 4207–4216. doi:10.1109/CVPR.2017.448. http://ieeexplore.ieee.org/document/8099931/

Kuhl, F.P., Giardina, C.R., 1982. Elliptic fourier features of a closed contour. Comput. Graph. Image Process. 18, 236–258.

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017a. Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Honolulu, HI, pp. 936–944. doi:10.1109/CVPR.2017.106. http://ieeexplore.ieee.org/document/8099589/

Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P., 2017b. Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). IEEE, pp. 2999–3007. doi:10.1109/ICCV.2017.324.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft coco: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (Eds.), Proceedings of the Computer Vision – ECCV 2014. Springer International Publishing, Cham, pp. 740–755.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C., 2016. SSD: Single Shot MultiBox Detector. In: Lecture Notes in Computer Science, Vol. 9905. Springer International Publishing, pp. 21–37. http://link.springer.com/10.1007/978-3-319-46448-0_2

Ljosa, V., Sokolnicki, K.L., Carpenter, A.E., 2012. Annotated high-throughput microscopy image sets for validation. Nat. Methods 9 (7), 637. doi:10.1038/nmeth.2083. http://www.nature.com/articles/nmeth.2083

Merker, B., 1983. Silver staining of cell bodies by means of physical development. J. Neurosci. Methods 9 (3), 235–241. doi:10.1016/0165-0270(83)90086-9.

Miksys, L., Jetley, S., Sapienza, M., Golodetz, S., Torr, P. H. S., 2019. Straight to shapes++: real-time instance segmentation made more accurate. 1905.11358.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 779–788. doi:10.1109/CVPR.2016.91. http://ieeexplore.ieee.org/document/7780460/

Redmon, J., Farhadi, A., 2018. Yolov3: an incremental improvement. 1804.02767.

Ren, S., He, K., Girshick, R., Sun, J., 2017. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. 39 (6), 1137–1149. doi:10.1109/TPAMI.2016.2577031.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Lecture Notes in Computer Science, Vol. 9351. Springer International Publishing, pp. 234–241. http://link.springer.com/10.1007/978-3-319-24574-4_28

Sabour, S., Frosst, N., Hinton, G.E., 2017. Dynamic routing between capsules. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Curran Associates Inc., USA, pp. 3859–3869.

Schmidt, U., Weigert, M., Broaddus, C., Myers, G., 2018. Cell Detection with Star-Convex Polygons. In: Lecture Notes in Computer Science, Vol. 11071. Springer International Publishing, pp. 265–273. http://link.springer.com/10.1007/978-3-030-00934-2_30

Stacke, K., Eilertsen, G., Unger, J., Lundström, C., 2019. A closer look at domain shift for deep learning in histopathology. CoRR abs/1909.11575. 1909.11575

Thierbach, K., Bazin, P.-L., Back, W.d., Gavriilidis, F., Kirilina, E., Jger, C., Morawski, M., Geyer, S., Weiskopf, N., Scherf, N., 2018. Combining Deep Learning and Active Contours Opens The Way to Robust, Automated Analysis of Brain Cytoarchitectonics. In: Lecture Notes in Computer Science, Vol. 11046. Springer International Publishing, pp. 179–187. http://link.springer.com/10.1007/978-3-030-00919-9_21

Xie, E., Sun, P., Song, X., Wang, W., Liu, X., Liang, D., Shen, C., Luo, P., 2020. Polarmask: single shot instance segmentation with polar representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 12190–12199. doi:10.1109/CVPR42600.2020.01221. https://ieeexplore.ieee.org/document/9157078/

Yagi, Y., 2011. Color standardization and optimization in whole slide imaging. Diagn. Pathol. 6 (S1), S15. doi:10.1186/1746-1596-6-S1-S15.

Yang, L., Ghosh, R.P., Franklin, J.M., Chen, S., You, C., Narayan, R.R., Melcher, M.L., Liphardt, J.T., 2020. Nuset: a deep learning tool for reliably separating and analyzing crowded cells. PLoS Comput. Biol. 16 (9), e1008193. doi:10.1371/journal.pcbi.1008193.

Zabawa, L., Kicherer, A., Klingbeil, L., Tpfer, R., Kuhlmann, H., Roscher, R., 2020. Counting of grapevine berries in images via semantic segmentation using convolutional neural networks. ISPRS J. Photogramm. Remote Sens. 164, 73–83. doi:10.1016/j.isprsjprs.2020.04.002.

Zhang, L., Bai, M., Liao, R., Urtasun, R., Marcos, D., Tuia, D., Kellenberger, B., 2018. Learning deep structured active contours end-to-end. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, pp. 8877–8885. doi:10.1109/CVPR.2018.00925. https://ieeexplore.ieee.org/document/8579023/