DeepFH Segmentations for Superpixel-based Object Proposal Refinement

Christian Wilms[a], Simone Frintrop[a]

[a] *University of Hamburg, Department of Informatics, Vogt-Koelln-Str. 30, Hamburg, 22527, Germany*

ARTICLE INFO

Keywords:
Object proposals
Image segmentation
Superpixels

ABSTRACT

Class-agnostic object proposal generation is an important first step in many object detection pipelines. However, object proposals of modern systems are rather inaccurate in terms of segmentation and only roughly adhere to object boundaries. Since typical refinement steps are usually not applicable to thousands of proposals, we propose a superpixel-based refinement system for object proposal generation systems. Utilizing precise superpixels and superpixel pooling on deep features, we refine initial coarse proposals in an end-to-end learned system. Furthermore, we propose a novel DeepFH segmentation, which enriches the classic Felzenszwalb and Huttenlocher (FH) segmentation with deep features leading to improved segmentation results and better object proposal refinements. On the COCO dataset with LVIS annotations, we show that our refinement based on DeepFH superpixels outperforms state-of-the-art methods and leads to more precise object proposals.

1. Introduction

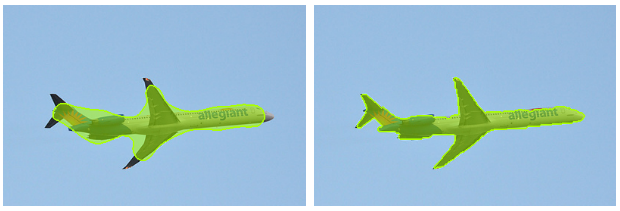
Object proposal generation serves as the basis for many object detection or instance segmentation systems [9, 32, 11, 5, 39]. The task describes the class-agnostic generation of possible object locations, which reduce the search space for subsequent methods. Since the task is class-agnostic, different to, e.g., instance segmentation, no class information is utilized. This leads to better generalization to unseen classes [27]. Since the emergence of deep learning, a number of CNN-based methods for object proposal generation were proposed [29, 30, 16, 37, 8, 21]. However, most of these systems suffer from imprecise segmentations, especially systems that propose segmentation masks as visible in Fig. 1a.

A major reason for the imprecise segmentations is the CNN, which leads to a low resolution in the system's segmentation stage. For instance, in [37] segmentations are based on 10×10 feature maps, independent of the object's size. However, those low resolution feature maps are semantically rich and important for the systems' overall success. Thus, systems suffer from coarse and imprecise segmentations, despite generating state-of-the-art overall results.

In tasks like semantic segmentation or salient object detection, this problem is tackled by utilizing encoder-decoder architectures [20, 6, 22] or CRFs [15, 23]. However, this is computationally expensive if applied to 1000 proposals per image. Other lines of work include dilated convolutions [6, 42] or iterative refinement [43, 18]. Most of these approaches are computationally expensive as well if applied to all proposals or many layers of a network. Thus, a refinement system which precisely captures the object contours and shares computations between the proposals is necessary.

Superpixels, introduced by [33], capture fine details of objects and serve as an abstraction of the pixels by reducing the number of entities. Furthermore, superpixels can be

*Corresponding author
wilms@informatik.uni-hamburg.de (C. Wilms); frintrop@informatik.uni-hamburg.de (S. Frintrop)



(a) without our refinement (b) with our refinement

Figure 1: AttentionMask object proposal without (a) and with (b) our superpixel-based refinement. Note the precise segmentation of fine details after applying our refinement.

shared between the proposals of an image, since they only rely on the image content. However, applying superpixels in CNNs is not straightforward, since they lack a lattice structure. However, [13, 19, 28] proposed semantic segmentation approaches with information aggregation across superpixels in CNNs.

In this paper, we follow this aggregation approach and propose a superpixel-based refinement for object proposals. We start from the coarse proposals of the state-of-the-art object proposal generation system AttentionMask [37] and add two streams to the system as visualized in Fig. 2. The first stream generates precise superpixel segmentations on input image resolution, while the second stream extracts features from the backbone network. In our proposed superpixel refinement module, for each proposal, we aggregate the information from the coarse AttentionMask proposal and the extracted features per superpixel utilizing our superpixel pooling module following [13, 19, 28]. For each proposal, this leads to a feature vector per superpixel, which is classified as background or object part with our new superpixel classifier. The evaluation shows superior overall results compared to state-of-the-art systems. Furthermore, the masks adhere better to the object boundaries (see Fig. 1), which is outlined by our results on the COCO dataset [24] using LVIS annotations [10].

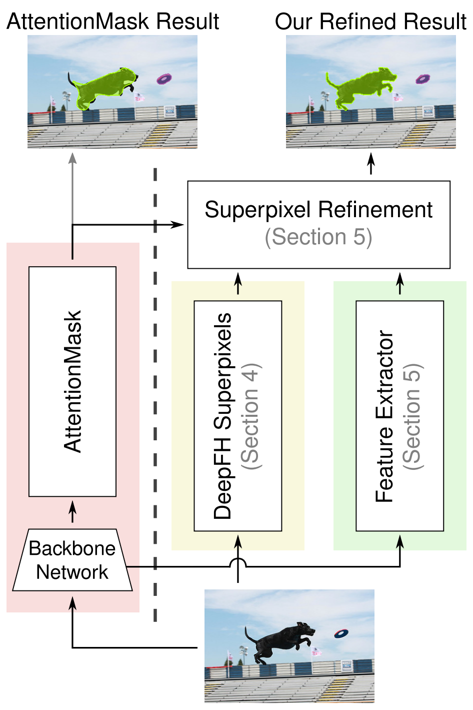


Figure 2: Building blocks of our proposed system. We refine object proposals with deep features and superpixels generated with our proposed DeepFH segmentation (cf. Sec. 4.2). The streams are combined in our superpixel refinement module (cf. Sec. 5) utilizing superpixel pooling and a superpixel classifier leading to precise proposals. The images on the top show the original proposal (left) and our refined version (right).

A typical problem when utilizing superpixels for object proposal refinement are compact superpixels. They lead to artificially splitting up uniform areas and increases the number of superpixels without significant gain in segmentation accuracy. Furthermore, superpixels, which cover large areas, have a larger support for information aggregation. Few approaches [7, 26, 36] have been proposed for generating non-compact superpixels, since it is difficult to control the number of superpixels for evaluation [34].

As a second novelty, we propose the novel DeepFH segmentation. DeepFH extends the segmentation algorithm by Felzenszwalb and Huttenlocher (FH) [7], which relies on RGB features for pixel similarity, with semantically richer deep features. We generate deep features with a lightweight encoder-decoder architecture based on a variation of ResNet [12]. In our evaluation, we show that DeepFH outperforms the original FH and further improves the downstream superpixel-based object proposal refinement. Since FH's general structure is unchanged, FH's properties including non-compact superpixels stay unchanged.

Our main contributions are a new superpixel-based refinement system for object proposals with precise proposal masks outperforming state-of-the-art and a new deep learning-based segmentation algorithm to support the refinement system. Overall, this paper presents an extended version of our conference paper [38]. We extend that paper in three major ways:

- Utilizing deep features with a tailored encode-decoder architecture, leading to novel DeepFH segmentations based on classic FH.
- An advanced superpixel-based refinement approach for object proposal generation systems combining DeepFH and superpixel pooling for improved results and better boundary adherence.
- A more detailed evaluation of our overall refinement system w.r.t. the superpixel segmentations.

2. Related Work

This section reviews related approaches for object proposal generation and superpixel segmentation.

2.1. Object Proposal Generation

Class-agnostic object proposal generation systems produce bounding boxes [8, 21] or pixel-precise segmentation masks [29, 30, 16, 37] alongside an objectness score per proposal. [14] give an overview of approaches not utilizing CNNs. Since we propose a system with pixel-precise masks, we focus on these approaches here.

Pre CNN systems usually grouped superpixels based on low-level or mid-level image cues. For instance, [31] propose a hierarchical image segmentation with subsequent merging and boundary classification.

Utilizing CNNs, [29] propose DeepMask, a model with individual heads for inferring objectness and the segmentation mask on top of a backbone network. For generating multiple proposals per image, DeepMask runs on many crops from an image pyramid (multi-shot approach). Due to subsampling in the backbone, the masks are generated on a coarse resolution, leading to imprecise proposals. SharpMask [30] tackles this problem by adding a decoder path to refine the segmentation masks. Still, SharpMask utilizes the multi-shot approach.

Feeding overlapping image crops through the backbone is redundant. To increase efficiency, [16] propose FastMask with a feature pyramid inside the CNN (one-shot approach). [16] generate the feature pyramid based on the backbone's final layer utilizing pooling. All possible fixed sized windows are extracted from the pyramid to capture objects of different size. For segmentation and objectness scoring, [16] propose individual heads. Due to the pyramid on top of the backbone, the feature maps for segmentation are coarse, resulting in imprecise masks.

Extending FastMask, [37] propose AttentionMask. AttentionMask improves the efficiency with attention at each pyramid level. As a result, only relevant windows are extracted, omitting background areas. Utilizing the freed resources, [37] add a new level to the pyramid for capturing small objects. The segmentation masks are still generated on 40×40 maps, independent of the object's size. Thus, despite improved results and efficiency, the masks are still coarse.

Complementing those approaches, we propose a superpixel-based refinement to enhance the quality of coarse segmentation masks.

2.2. Superpixel Segmentation

Since the seminal work of [33], many superpixel segmentation approaches were proposed [1, 4, 41, 25, 7, 35, 17, 40]. [34] give an overview of approaches not utilizing CNNs. Many superpixel algorithms start from an initial segmentation, which is refined to fit the image content, leading to compact superpixels of uniform size. [1] start from a uniform grid and apply a local k-means clustering to the pixels. [4] also start from an initial grid and move pixels or pixel blocks between superpixels. Similarly, [41] propose an MRF approach with a topology preserving energy term.

In contrast, [25] and [7] merge individual pixels to generate a segmentation. [25] merge pixels based on the entropy of random walks to generate superpixels of roughly uniform size. The FH algorithm [7] merges pixels based on color or intensity differences. The termination depends on internal and external dissimilarities of pixel clusters. Thus, the superpixel size is not regularized, leading to non-compact superpixels of varying size. [36] propose an approach to get a similar effect with other segmentation algorithms.

Few approaches generate superpixels utilizing CNNs. [35] compute pixel affinities and apply [25] on those affinities. [17] propose a differentiable approach of [1] with CNN features. [40] learn associations between each pixel and superpixels of a uniform grid to infer superpixels.

Different to those approaches, we utilize deep features to enhance the FH segmentation. Thus, the segmentation allows non-compact superpixels and utilizes deep features, which improves our downstream application.

3. System Overview

Our overall system consists of four building blocks as visualized in Fig. 2. The first part is the state-of-the-art object proposal system AttentionMask [37] (red area in Fig. 2), which we use to generate coarse object proposals. Second, we propose a novel DeepFH superpixel segmentation, described in Sec. 4, based on deep features and the classic FH algorithm (yellow area Fig. 2). Third, we extract features (see Sec. 5) from AttentionMask's backbone (green area Fig. 2). Finally, we combine these streams in our novel superpixel-based refinement module described in Sec. 5 (top area in Fig. 2). The refinement module takes AttentionMask's coarse proposals and refines them using the deep features and the new DeepFH superpixels utilizing superpixel pooling and our novel superpixel classifier. The output of the overall system are refined object proposals, which better adhere to object boundaries.

4. DeepFH Segmentation

This section introduces our novel DeepFH segmentation as a variation of the segmentation algorithm by Felzenszwalb and Huttenlocher (FH) [7] based on CNN-features.

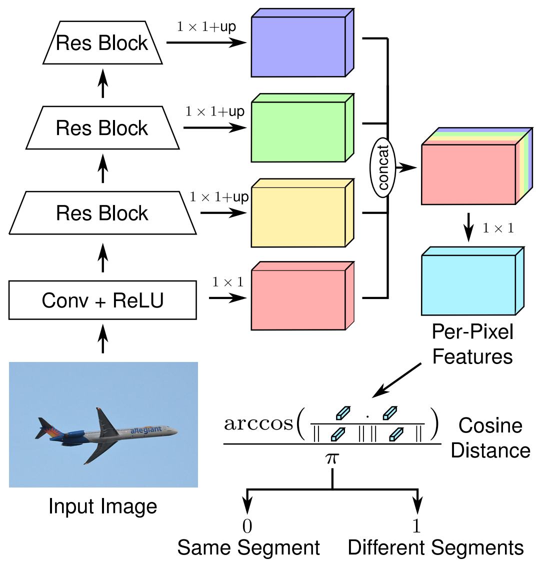


Figure 3: Feature extractor for our DeepFH segmentation. The encoder consists of ResNet-18 blocks (2 convolution layers with skip connection). After applying a 1×1 convolution per stage, the features of different stages (different colors) are concatenated and the final features are extracted. Per pixel, these features are used for pixel affinity calculation based on cosine distance.

For generating a DeepFH segmentation, we run an image through a lightweight encoder-decoder (cf. Fig. 3). This results in semantically rich per-pixel features at the decoder's output layer (cyan box in Fig. 3) with input image resolution. We use these features to enrich the color distance between pixels in the classic FH algorithm with the cosine distance in the deep feature space for a combined distance. The rest of the FH algorithm (see Sec. 4.2) is unchanged to keep the algorithm's properties like non-compact superpixels.

4.1. Feature Extraction

For extracting semantically rich CNN-based features, we follow the works on pixel affinity by [3, 2] for semantic segmentation as well as the approaches of [35, 17, 40] on CNN-based superpixel segmentation. We propose a simple encoder-decoder with a 4-stage ResNet-18-based backbone for feature extraction as visible on the left in Fig. 3. In contrast to ResNet-18, we use only one residual block per stage in the encoder. This is in line with [35, 17, 40], which propose shallow networks for feature extraction. Furthermore, this architecture generates better results compared to networks with more layers like ResNet-18 or ResNet-34 (see top part of Tab. 1).

The decoder part (right part in Fig. 3) consists of a per-stage feature extraction with a bilinear upsampling, a concatenation across all stages and the final feature extraction layer leading to a 128 dimensional feature vector per pixel. This is similar to the decoder architectures in pixel affinity [3, 2, 35] and is more powerful than the step-wise integration of feature maps in [40] (see bottom part of Tab. 1).

Table 1
Architectures for our feature extraction network's encoder (top) and decoder (bottom), evaluated w.r.t. the downstream object proposal refinement task. AR@n denotes the average recall for the first n proposals.

Architecture	AR@10↑	AR@100↑	AR@1000↑
Ours	**0.104**	**0.223**	**0.330**
ResNet-18	0.102	0.222	0.324
ResNet-34	0.102	0.223	0.325
Step-wise [40]	0.102	0.221	0.323

Table 2
Cosine distance and Euclidean distance as deep feature distance in DeepFH evaluated w.r.t. the downstream object proposal refinement task. AR@n denotes the average recall for the first n proposals.

Distance	AR@10↑	AR@100↑	AR@1000↑
Cosine	**0.104**	**0.223**	**0.330**
Euclidean	0.102	0.218	0.320

On top of the final layer, the cosine distance between feature vectors of adjacent pairs of pixels $(\mathbf{f}_i, \mathbf{f}_j)$ is calculated:

$$\delta_{\cos}(\mathbf{f}_i, \mathbf{f}_j) = \frac{1}{\pi} \arccos\left(\frac{\mathbf{f}_i \cdot \mathbf{f}_j}{\|\mathbf{f}_i\| \|\mathbf{f}_j\|}\right). \tag{1}$$

The result of the cosine distance is the classification result for the pair of adjacent pixels. If the distance is low, the pixels belong to the same segment and vice versa. Applying the cosine distance is different to [3, 2], which use Euclidean distance. However, since the feature vectors are high-dimensional, cosine distance is favorable (cf. Tab. 2).

4.2. Extension of FH Algorithm

The original FH algorithm [7] is a graph-based segmentation method, in which adjacent pixels or pixel clusters are greedily merged. As criterion for merging, simple color or intensity difference is used. For RGB images, the difference between the pixels $\mathbf{p}_i$ and $\mathbf{p}_j$ representing the RGB values as vector is calculated as

$$\delta_{\text{FH}} = \|\mathbf{p}_i - \mathbf{p}_j\|_2. \tag{2}$$

During merging, the largest dissimilarity within a cluster of pixels and the smallest dissimilarity between two adjacent clusters of pixels are compared. If a certain threshold-based criterion is met, the clusters are merged. The algorithm terminates once no pair of clusters meets the criterion. Since all steps are based on the initial pixel distance, by only enhancing this distance calculation, the outcome of the FH algorithm can be improved.

Enhancing the distance calculation for our DeepFH, we utilize the CNN-features $\mathbf{f}_i$, and $\mathbf{f}_j$ for adjacent pixels $\mathbf{p}_i$ and $\mathbf{p}_j$ extracted from the encoder-decoder (see Sec. 4.1). The features are extracted from the final feature map (cyan box in Fig. 3). Since the features were learned based on cosine distance, we add the cosine distance between pixels' features as a second term to the original distance δ_{FH}:

$$\delta_{\text{DeepFH}} = (1-\alpha)\|\mathbf{p}_i - \mathbf{p}_j\|_2 + \alpha\delta_{\cos}(\mathbf{f}_i, \mathbf{f}_j). \tag{3}$$

This is convenient, since both distances are in the interval $[0, 1]$. Thus, the parameter α only controls the influence of the distances. We set $\alpha = 0.2$ for best results on the downstream object proposal refinement task. The rest of the FH algorithm stays unchanged. Thus, the runtime is still $\mathcal{O}(n \log n)$ and the properties regarding the segmentation quality proven in [7] are kept.

4.3. Training

For training the feature extractor, we use binary crossentropy loss to assess the feature vectors' cosine distance between adjacent pixels. For optimization, we use Adam with an initial learning rate of 0.01, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. Similar to [35, 17, 40], our network is trained from scratch without pre-trained weights, since pre-training did not improve the results.

The overall goal of our system is to generate better object proposals. Therefore, we did not train on typical segmentation datasets, but generated a segmentation dataset from the COCO dataset [24] training images with LVIS annotations [10]. The LVIS annotations are significantly more precise, which is important in our pixel affinity-like setup, and contain one binary segmentation per annotated object. As ground truth, we combine all binary segmentations per image, generating an oversegmentation of the image retaining all object boundaries. We train the network on this dataset for 8 epochs. The integration of DeepFH segmentations into the superpixel-based proposal refinement is described in Sec. 5.3.

5. Superpixel-based Object Proposals

This section introduces our superpixel-based refinement for object proposal generation. The refinement connects the object proposal generation system AttentionMask [37] (see Sec. 2.1) and the DeepFH segmentation described in Sec. 4.2. It starts from the coarse proposals of AttentionMask, which are 40×40 windows per proposal. AttentionMask is visualized in the red part of Fig. 4, which depicts our overall refinement system. The 40×40 windows contain per-pixel probabilities for being part of an object and represent only a certain area of the input image. In parallel, we generate precise DeepFH superpixel segmentations for each level of AttentionMask's feature pyramid with different resolutions, capturing fine details of the input image (yellow part in Fig. 4). In a third stream, displayed in the green part of Fig. 4, features are extracted from AttentionMask's backbone for each level of the feature pyramid to support the refinement.

We apply the refinement, visualized based on an example in Fig. 5, to each proposal. The refinement starts by combining the upsampled AttentionMask proposal and

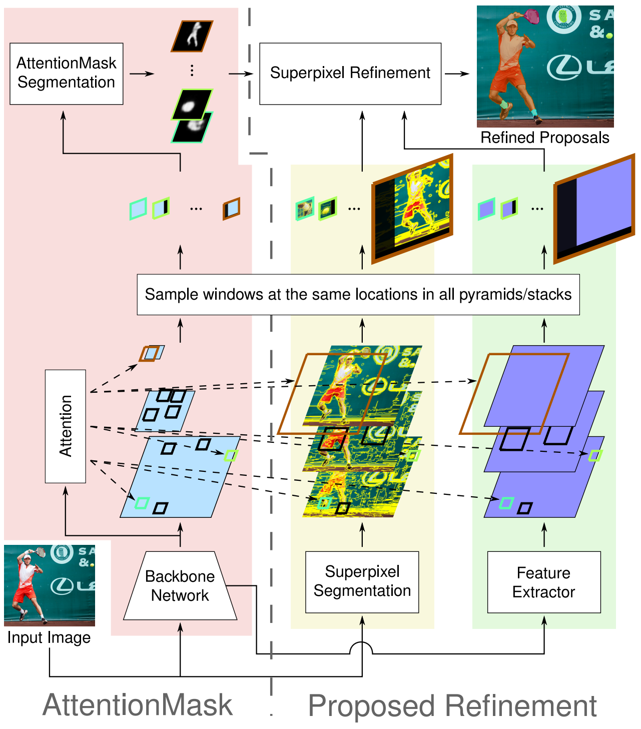


Figure 4: Overview of our proposed superpixel-based refinement system. The general structure follows AttentionMask (red area) and is extended by our DeepFH superpixel segmentations (yellow area), our feature extraction (green area) as well as our superpixel refinement module. The system's three streams share the same pyramid/stack structure and are connected by AttentionMask's attention module. The module selects interesting areas in the pyramids/stacks, which are cropped from all streams (colored windows). However, indicated by the different sizes, the windows cropped from the segmentation and feature extraction streams have a higher resolution, leading to more detailed results. Finally, the cropped windows from the three streams are utilized in our superpixel refinement module (see Fig. 5).

the respective superpixel segmentation window. For each superpixel, the average probability, called mask prior, is computed using our superpixel pooling. This process is visualized in the upper left part of Fig 5. Utilizing the extracted features and the segmentations, a per superpixel feature vector is extracted applying our superpixel pooling again as visualized in the bottom part of Fig. 5 (see Sec. 5.1). Subsequently, mask prior and respective superpixel features are concatenated for each superpixel overlaying with the AttentionMask proposal window (upper central part of Fig. 5). Finally, these superpixels are classified with our new superpixel classifier (see Sec. 5.2) as background or object superpixels. The object superpixels constitutes the refined object proposal. This process is visualized in the top right of Fig. 5. The final proposals combine the advantages of detailed superpixels, deep features and semantically rich but coarse AttentionMask object proposals. See Sec. 5.3 for details of the refinement's integration with AttentionMask and DeepFH.

5.1. Superpixel Pooling

Following [13, 19, 28], we propose a superpixel pooling module, to combine CNNs and superpixels. Directly integrating superpixels and CNNs is non-trivial, as outlined in the introduction. Superpixel pooling is similar to standard pooling, however, the final value is pooled from an arbitrarily shaped neighborhood (superpixel). Thus, the result has no lattice structure anymore. It is one feature or feature vector per superpixel, which has to be processed individually. Backpropagation is possible, similar to standard pooling.

We utilize our superpixel pooling module twice in our refinement, as visualized in Fig. 5. First, we use the superpixel pooling to create the mask prior, a superpixelized version of the coarse AttentionMask proposal, shown in the upper left of Fig.5. The AttentionMask proposals are 40×40 windows representing parts of the image with different sizes in the original image. The proposals contain per-pixel probabilities for being part of an object. These windows are traced back to a level and spatial position of AttentionMask's feature pyramid by the attention system in AttentionMask. Given that level and the window's spatial position, we extract a crop from a suitable high-resolution superpixel segmentation, covering the same area in the input image as shown by the colored windows in Fig. 4.

The superpixel segmentation's resolution depends on the level in AttentionMask's feature pyramid the proposal originates from. For proposals covering small objects, fine segmentations are used and vice versa. The segmentation crop's size is not fixed, but represents the input image area in high resolution. Thus, the segmentation can cover all details of the object. Given this segmentation crop and one proposal, we use superpixel pooling to create an average value of the proposal per superpixel. The value represents the probability of the superpixel belonging to the object. This is called mask prior and is applied individually per proposal.

Furthermore, we utilize our new superpixel pooling to compose a feature vector from the extracted features (green part in Fig. 4) for each superpixel of each level. Different from the mask prior computation described above, this computation is shared between all proposals from the same feature pyramid level, leading to an efficient execution. Per proposal, the mask prior and the feature vector for each superpixel overlaying with the proposal window are concatenated (see central part in Fig. 5). This constitutes a batch of superpixel representations per proposal for classification.

5.2. Superpixel Classification

After applying superpixel pooling twice, each coarse AttentionMask proposal is a batch of superpixel feature vectors. These vectors represent precise superpixels with a resolution suitable for the proposal size. To determine the final refined proposal, each of the superpixel feature vectors is classified as background or object part (upper right in Fig. 5). For this classification, we propose a lightweight superpixel classifier visualized in Fig. 6 with only four fully-connected layers. The light-weight architecture is beneficial in terms of results compared to other structures as Tab. 3 indicates.

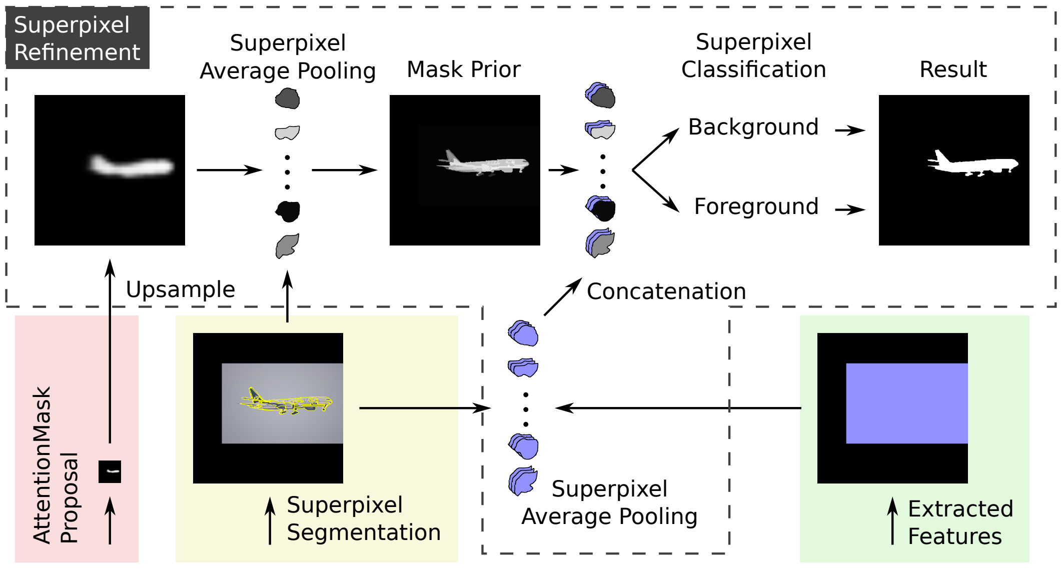


Figure 5: Detailed view of our proposed superpixel refinement module showing the workflow for an individual proposal. The coarse AttentionMask proposal (red area) is upsampled and superpixel pooling is applied using the respective window from the segmentation stack (yellow area). This leads to a precise superpixelized version of the proposal called mask prior. Similarly, superpixel pooling is applied to the respective window of the feature stack (green area), yielding a feature vector per superpixel. Finally, mask prior and feature vector per superpixel are concatenated and classified as background or object part, leading to a precise proposal.

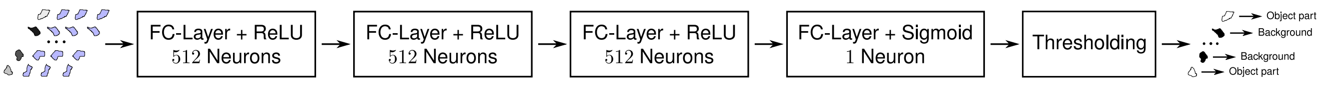

FC-Layer + ReLU 512 Neurons
FC-Layer + ReLU 512 Neurons
FC-Layer + ReLU 512 Neurons
FC-Layer + Sigmoid 1 Neuron
Thresholding
Object part
Background
Background
Object part

Figure 6: Architecture of our superpixel classifier with fully-connected layers denoted as FC-layer. The input to the classifier is a batch of superpixels representing a proposal. For each superpixel, it consists of one mask prior value (gray) from the AttentionMask result and 512 learned features (bluish). The classifier determines if a superpixel is part of an object (white) or background (black).

Table 3
Architectures for the proposed superpixel classifier. The first column indicates the amount of neurons per fully-connected layer. AR@n denotes the average recall for the first n proposals.

Architecture	AR@10↑	AR@100↑	AR@1000↑
512 – 512 – 512	**0.104**	**0.223**	**0.330**
1024 – 1024 – 1024	0.104	0.221	0.325
256 – 256 – 256	0.102	0.220	0.327
512 – 512 – 512 – 512	0.099	0.223	0.328
512 – 512	0.101	0.224	0.328

Table 4
Results of the proposed refinement approach using features from different ResNet stages for feature extraction. AR@n denotes the average recall for the first n proposals.

ResNet-block	AR@10↑	AR@100↑	AR@1000↑
res1	0.101	0.223	0.326
res2	**0.104**	**0.223**	**0.330**
res3	0.103	0.224	0.323
res4	0.098	0.221	0.324

Altogether, the classifier takes a batch of superpixel features and returns the label object or background per superpixel. Superpixels that do not overlap with the proposal window are set as background to reduce computation.

5.3. Combination with AttentionMask and DeepFH

Combining the proposed superpixel-based refinement with AttentionMask into one end-to-end system leads to a change in the backbone. The original AttentionMask backbone, a 4-stage ResNet-50 is replaced by a 4-stage ResNet-34, to fit the memory restrictions on GPUs. To extract features from the backbone for per-superpixel features, we evaluated different backbone stages (see Tab. 4) and chose the *res2* stage. The feature extractor itself is a 1×1 convolution. Altogether, AttentionMask, the feature extractor and the superpixel classifier are trained end-to-end in one system.

Outside this network, we generate DeepFH segmentations (see Sec. 4). As described in Sec. 4.3, we train DeepFH on the COCO dataset with LVIS annotations. Subsequently, DeepFH is used to generate one segmentation for each level of AttentionMask's feature pyramid. We optimize the parameter t in FH as well as the parameter α from Eq. 3 to generate on average 500 superpixels on the level of the largest objects and 8000 superpixels on the level of the smallest objects, with regular intermediate steps for the intermediate levels. Directly setting a desired number of superpixels is impossible for DeepFH. We present a comparison to other segmentation algorithms and numbers of superpixels in Sec. 6.2.1.

Finally, after generating the refined proposals based on the DeepFH superpixels, we apply a three-step post-processing to each proposal. First, we use bilinear filtering on superpixel level, using the superpixels' mean colors. Second, to remove segmentation artifacts, we apply morphological opening and closing to the proposals. This removes or fills tiny antenna-like artifacts in the proposal masks. Finally, we apply non-maximum suppression (NMS) with a high IoU of 0.95 to remove almost identical proposals. Since all proposals are represented by few superpixels, identical proposals are more common than with pixel-based proposals. The post-processing is evaluated in Sec. 6.2.4.

5.4. Training

We train our superpixel-based refinement with AttentionMask end-to-end using SGD with a learning rate of 0.0001 for 13 epochs. The ResNet-34 backbone is initialized with ImageNet weights, while the rest of the network is learned from scratch. Apart from AttentionMask, the feature extractor and the superpixel classifier are trained. The superpixel classifier is trained with a binary crossentropy loss ($\mathcal{L}_{spx}$), while the feature extractor is automatically trained by the overall system. Thus, it does not require distinct supervision. Combined with the AttentionMask [37] losses $\mathcal{L}_{objn}, \mathcal{L}_{ah}, \mathcal{L}_{seg}$, and $\mathcal{L}_{att_n}$, the overall loss is defined as

$$\begin{aligned}\mathcal{L} =& w_{objn}\mathcal{L}_{objn} + w_{ah}\mathcal{L}_{ah} + w_{seg}\mathcal{L}_{seg} + \\ & w_{att}\sum_{n}\mathcal{L}_{att_n} + w_{spx}\mathcal{L}_{spx}\end{aligned} \tag{4}$$

with weights controlling the losses' influence. $w_{spx} = 1$ throughout our experiments, while the other weights stay unchanged compared to [37].

As training data we use the training images of the COCO dataset [24] with the COCO annotation for a fair comparison. The ground truth contains polygon annotations for objects of 80 classes. However, generating ground truth for the superpixel classifier is non-trivial, since the annotation boundaries do not necessarily match the superpixel boundaries. Thus, for each annotated object, we generate an optimal set of superpixels yielding a maximum IoU between the set and the ground truth annotation. This process starts from all superpixels completely contained in the annotated object and greedily adds superpixels that increase the IoU. These superpixels are positive samples, while all other superpixels form a pool of negative samples.

6. Evaluation

For evaluating our new superpixel-based refinement approach combined with our novel DeepFH segmentations on top of AttentionMask, we mostly follow standard object proposal evaluation pipelines. We assess the quality of the results based on average recall (AR), which combines recall as well as the proposals' segmentation quality and is frequently used in object proposal literature [14, 29, 30, 16, 37]. AR is calculated for different numbers of proposals, e.g., AR@100 for the first 100 proposals, and for different sizes of objects. To only assess the segmentation quality of the proposals, we use standard segmentation measures boundary recall (BR) undersegmentation error (UE), and oversegmentation error (OE).

Finally, we introduce the Achievable Intersection over Union (AIoU) as a new measure to assess the quality of a segmentation for superpixel-based object proposal generation. Since the ground truth for our system was superpixelized for training (see Sec. 5.4), we measure given a superpixel segmentation the average IoU between the optimal set of superpixels per annotated object and the annotated object itself. If the superpixel segmentation perfectly captures the annotated object, the AIoU is 1. Otherwise, the AIoU decreases to a value between 0 and 1. Thus, the AIoU is an upper bound for the quality of a superpixel-based proposal algorithm in terms of average IoU (AVGIoU) per ground truth object. Comparing AIoU and AVGIoU allows us to assess how much of the segmentation quality is utilized.

The dataset for our evaluation is the COCO [24] dataset with more detailed LVIS [10] annotations. This is different to other works [29, 30, 16, 37]. However, it is important for our evaluation, since we focus on precise object proposals. Thus, we use the LVIS data for validation and testing, while all approaches were trained on COCO data. An evaluation of our baseline AttentionMask compared to other methods on COCO is presented in [37]. We compare our method to the CNN-based methods DeepMask [29], SharpMask [30], FastMask [16], and AttentionMask [37] described in Sec. 2.1 as well as the method MCG [31], which does not utilize CNNs and does not suffer from CNN-based downsampling.

6.1. Results on LVIS

First, we present the results of our proposed system on the object proposal generation task using the COCO dataset with LVIS annotations. From the quantitative results in Tab. 5 it is visible that our proposed superpixel-based refinement with FH or DeepFH outperforms all other methods. In terms of AR@100, we surpass the other methods by 0.024 to 0.082. Thus, the refinement improves the results despite using a smaller backbone. For instance, the results of AttentionMask using a ResNet-34 backbone are slightly worse compared to original AttentionMask. Our refinement improves the results significantly even surpassing original AttentionMask as well as FastMask, SharpMask, DeepMask and MCG.

Comparing the numbers for objects of different sizes, we can show an improvement across all object sizes. Interestingly, for large objects (AR^{L}@100) SharpMask and the not DL-based MCG outperform original AttentionMask. This could not be shown using the COCO annotations [37].

Table 5
Results on the COCO dataset with LVIS annotations using average recall (AR). AR^S, AR^M and AR^L denote results on small, medium and large objects. Best and second-best results marked with **bold font** and *italic font* respectively.

Method	Backbone	AR@10↑	AR@100↑	AR@1000↑	AR^S@100↑	AR^M@100↑	AR^L@100↑
MCG [31]	-	0.048	0.131	0.237	0.031	0.204	0.462
DeepMask [29]	ResNet-50	0.069	0.147	0.214	0.014	0.314	0.430
SharpMask [30]	ResNet-50	0.073	0.154	0.229	0.014	0.327	0.460
FastMask [16]	ResNet-50	0.069	0.161	0.256	0.055	0.296	0.386
AttentionMask [37]	ResNet-50	0.073	0.189	0.284	0.081	0.312	0.446
AttentionMask [37]	ResNet-34	0.076	0.185	0.271	0.083	0.305	0.423
Ours with FH [7]	ResNet-34	*0.092*	*0.206*	*0.290*	*0.092*	*0.340*	*0.473*
Ours with DeepFH	ResNet-34	**0.094**	**0.213**	**0.304**	**0.098**	**0.349**	**0.480**

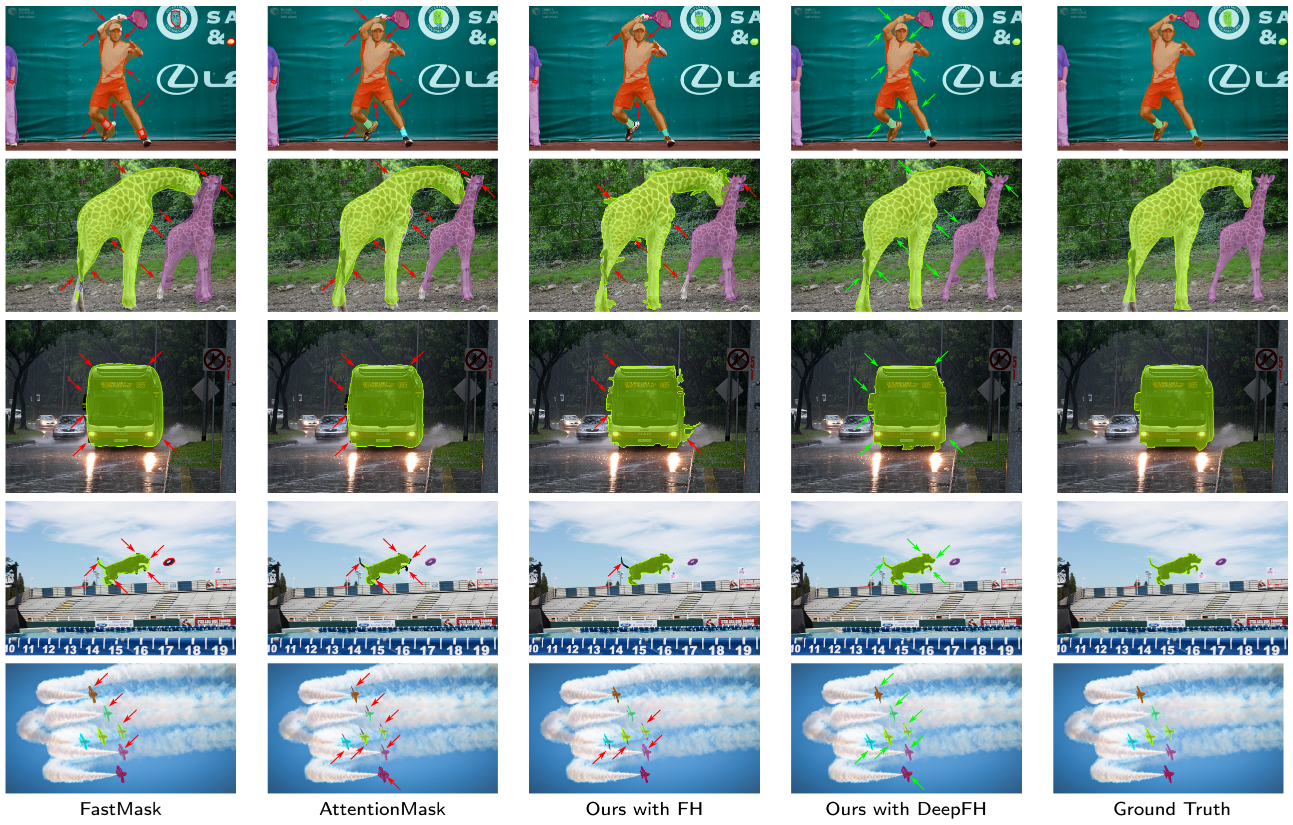


Figure 7: Qualitative results of FastMask [16], AttentionMask [37] and our approach with FH [7] and our new DeepFH superpixels on the COCO dataset with LVIS annotations. The filled colored contours denote found objects, while not filled red contours denote missed objects. The green arrows highlight locations showing the strength of our proposals adhering well to object boundaries, while the red arrows denote the same locations in the results of selected other systems.

However, it is not surprising, since FastMask and AttentionMask heavily utilize subsampling. Yet, our proposed refinement improves AttentionMask's results for large objects, outperforming SharpMask and MCG. Finally, comparing the bottom rows, it is visible that our DeepFH segmentations improve the results compared to classic FH segmentations. We discuss this difference in more detail in our ablation studies.

Investigating the proposal's boundary adherence, we generate a segmentation per found object for each of the systems using the best proposal per found object. We compare these segmentations to the segmentations generated from the ground truth objects. The results of this experiment (see Tab. 6) allow to analyze the proposal masks using typical segmentation measures like BR and UE. The numbers show that both of our proposed systems outperform all other DL-based systems by a large margin (e.g., +0.153 in BR compared to AttentionMask). Thus, our proposals fit the ground truth objects better in terms of BR and UE. Only the not DL-based method MCG generates similar results in terms of BR, however, missing a larger amount of objects (see Tab. 5).

Table 6
Comparison of the best proposals per image as segmentations using segmentation measures boundary recall (BR) and under-segmentation error (UE) on the LVIS dataset. Best and second-best results marked with **bold font** and *italic font* respectively.

Method	Backbone	BR↑	UE↓
MCG [31]	-	*0.685*	0.073
DeepMask [29]	ResNet-50	0.488	0.087
SharpMask [30]	ResNet-50	0.561	0.080
FastMask [16]	ResNet-50	0.510	0.084
AttentionMask [37]	ResNet-50	0.568	0.070
AttentionMask [37]	ResNet-34	0.547	0.075
Ours with FH [7]	ResNet-34	0.681	*0.068*
Ours with DeepFH	ResNet-34	**0.700**	**0.066**

Comparing our refinement using FH and DeepFH segmentations shows that the DeepFH segmentations further improve BR (+3%) and UE (−3%). Thus, our refined proposals based on DeepFH adhere better to object boundaries compared to all other methods.

The qualitative results in Fig. 7 support these findings. It is clearly visible that our proposed refinement improves the boundary adherence compared to other systems. In the first row, e.g., the tennis player's limbs are only precisely captured using our refinement with DeepFH. Similarly, in the second row the giraffes' heads and limbs are captured in more detail using our refinement with DeepFH compared to the imprecise segmentations of FastMask and AttentionMask. Even the small ears and most of the ossicones are precisely captured. Compared to the refinement with FH, less superpixels were misclassified. Similar effects are visible along the contour of the bus in the third row, which is not segmented as a blob like in the AttentionMask and FastMask results. Focusing on smaller objects, the final two rows show the precise segmentation of a jumping dog even capturing its thin tail and the detailed segmentations of tiny airplanes using our refinement. Therefore, as the results in Tab. 5 and Tab. 6 indicated, our proposed refinement with DeepFH captures objects of all scales in more detail than other approaches.

6.2. Ablation Studies

After discussing the main results, we present ablation studies regarding the segmentation and the post-processing of our system. Other studies regarding design choices were already presented in Sec. 4 and Sec. 5. All studies, except for the analysis of the post-processing, were carried out on our COCO validation set using LVIS annotations.

6.2.1. Segmentation Algorithms for Refinement

In this study, we evaluate the use of different segmentation algorithms and numbers of superpixels for the object proposal refinement based on AR. First, we compare our DeepFH-based refinement system with different numbers of superpixels. The results for 8000 to 500 superpixels and 4000 to 250 superpixels are presented in the first part of

Table 7
Results of different segmentation methods and numbers of superpixels used in our refinement approach. Best result without and with ground truth are marked with **bold font** and *italic font* respectively

Segmentation	#Superpixels	AR@100↑	AR@1000↑
DeepFH	8000 – 500	**0.223**	**0.330**
DeepFH	4000 – 250	0.219	0.322
DeepFH with GT	8000 – 500	0.244	0.361
DeepFH with GT	4000 – 250	*0.272*	*0.405*
SpxFCN [40]	8000 – 500	0.207	0.302
SSN [17]	8000 – 500	0.220	0.319
SEAL [35]	8000 – 500	0.215	0.310
FH [7]	8000 – 500	0.217	0.318
ETPS [41]	8000 – 500	0.214	0.311
ERS [25]	8000 – 500	0.205	0.302
SEEDS [4]	8000 – 500	0.200	0.284
SLIC [1]	8000 – 500	0.196	0.292

Tab. 7. It is visible that more superpixels have a positive influence on the results. However, more than 8000 superpixels are impossible due to memory limitations. In a second experiment, we add all ground truth edges to both variations. The second part of Tab. 7 shows the results. Interestingly, a smaller number of superpixels generates significantly better results. Thus, a strong oversegmentation is not desirable for our proposed refinement system.

As a third step, we evaluate other segmentation methods presented in Sec. 2.2 with our refinement system. The results are shown in the third and fourth part of Tab. 7. The segmentation methods in the fourth part do not utilize deep features. The results show that the methods ETPS, ERS, SLIC and SEEDS generating rather compact superpixels of uniform size perform worse than original FH. Our DeepFH outperforms all those methods by utilizing deep features. Comparing DeepFH to other DL-based superpixel approaches like SSN, SEAL and SuperFCN, the results also show a better performance of DeepFH in object proposal refinement. This is in line with the non DL-based results, since SSN, SEAL and SuperFCN generate rather compact superpixels of uniform size. Thus, for the proposed refinement system a FH-based segmentation and specifically DeepFH are better suited than other segmentation methods.

6.2.2. Segmentation Quality

After showing the superiority of DeepFH segmentations in object proposal refinement, we evaluate the different segmentations in terms of segmentation measures BR, UE and OE on the LVIS annotations. Ground truth segmentations represent the joint segmentations of the annotated objects. The results are presented in Fig. 8. SSN and SEAL perform best in terms of BR (see Fig. 8a) and UE (see Fig. 8b). This is in line with the results of [17, 35] on segmentation datasets. Thus, those segmentations adhere generally better to the object boundaries. Both, FH and DeepFH do not perform

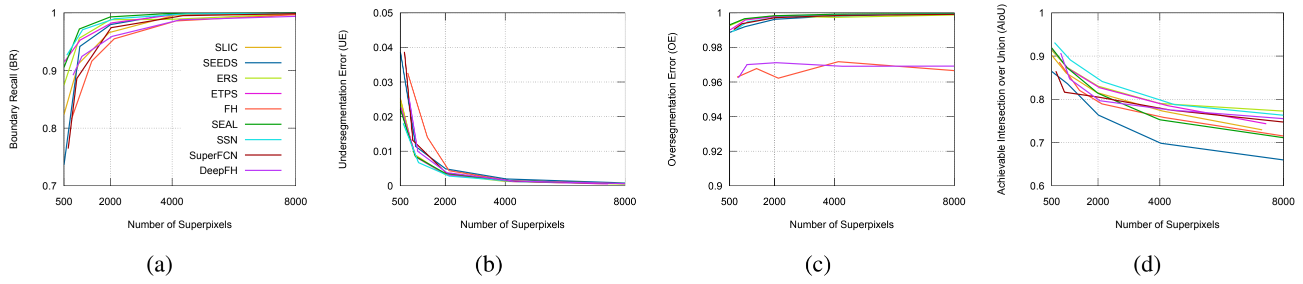


Figure 8: Results of different segmentation algorithms on the LVIS annotations for varying numbers of superpixels in terms of BR, UE, OE and AIoU.

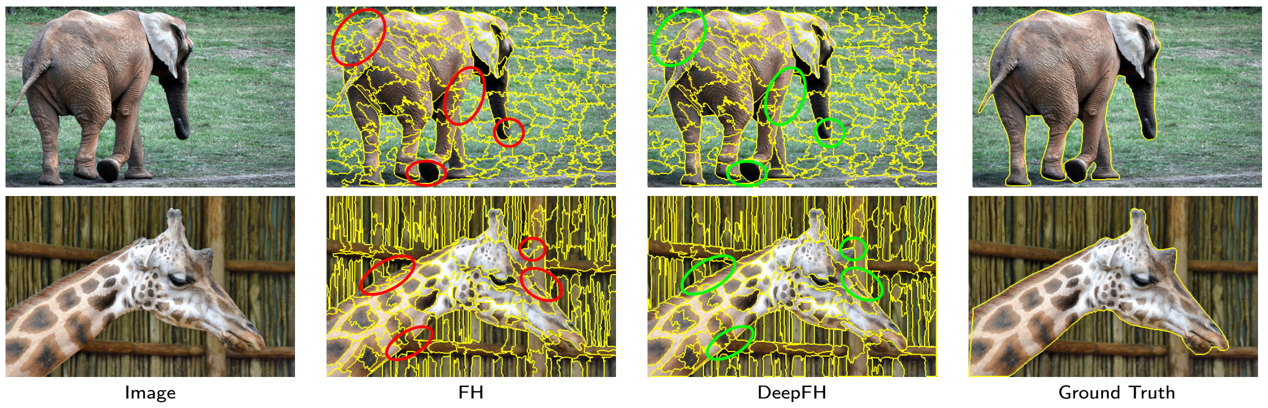


Figure 9: Qualitative results of FH [7] and our proposed DeepFH segmentation on COCO images with LVIS annotations. Red circles denote segmentation errors in FH, while green circles denote the same location in the DeepFH results.

very well in terms of BR and UE. However, evaluating the strength of oversegmentation in terms of OE (see. Fig. 8c), FH and DeepFH outperform all other methods. Thus, due to the design of FH and DeepFH, both segmentations do not artificially oversegment uniform areas as strongly. This leads to a much smaller OE and to better results on the downstream task, since many superpixels covering larger parts of background reach the object boundary. Therefore, our superpixel refinement prefers segmentations with a smaller OE, despite a slightly worse segmentation accuracy in terms of BR and UE.

Evaluating a segmentation's suitability for superpixel-based refinement of object proposals, we proposed AIoU as an upper limit for the average IoU between proposals and annotations. The results in terms of AIoU are presented in Fig. 8d. Again, SSN performs very well, while FH and DeepFH generate mediocre results. This is also supported by the results in Tab. 8, which show the average AIoU across the scales for selected segmentations. Tab. 8 also shows the AVGIoU, the average IoU between an annotation and the best fitting proposal. In contrast to the AIoU, these numbers favor DeepFH and FH segmentations. Thus, those segmentations realize a larger portion of their potential (AIoU), leading to a larger efficiency in terms of AVGIoU/AIoU. Still, it is clearly visible that all segmentations realize only slightly above 50% of their AIoU.

Table 8
Results of selected segmentations with the proposed refinement approach on top of AttentionMask. Evaluation in terms of average IoU (AVGIoU) between a ground truth object and the best fitting proposal, the achievable IoU for a given segmentation (AIoU) as well as the efficiency AVGIoU/AIoU. Best and second-best results marked with **bold font** and *italic font* respectively.

Segmentation	AVGIoU↑	AIoU↑	Efficiency↑
DeepFH	**0.448**	0.790	*0.568*
SSN [17]	*0.443*	*0.809*	0.547
SEAL [35]	0.438	0.777	0.563
FH [7]	0.441	0.766	**0.577**
ETPS [41]	0.424	0.799	0.530
ERS [25]	0.436	**0.811**	0.538

6.2.3. *FH vs. DeepFH Segmentations*

We already quantitatively evaluated the difference between our proposed DeepFH and the original FH in terms of BR, UE, OE and AIoU on the LVIS annotations. Most of these measures showed an improved performance for DeepFH compared to FH. Highlighting this improvement, Fig.9 shows qualitative segmentation results on the COCO dataset with LVIS annotations. In both images, it is visible that the FH segmentation results in segmentation errors along the object's contour. For instance, in the first image

Table 9
Results of the proposed refinement approach using different post-processing steps on the test dataset. Best result marked with **bold font**.

Post-processing	AR@10↑	AR@100↑	AR@1000↑
none	0.076	0.189	0.288
bilateral filtering	0.079	0.196	0.304
+ opening	0.080	0.197	0.306
+ closing	0.080	0.199	**0.307**
+ NMS	**0.094**	**0.213**	0.304

along the back of the elephant, multiple superpixels cover both object and background. This can be attributed to the low contrast between object and background. Similarly, the bottom of the elephant's lifted foot is not segmented properly. Those effects are fixed in the DeepFH segmentation. The image in the second row shows a highly textured environment, where object and background share similar colors. The FH segmentation reflects this with several segmentation errors along the giraffes' neck and head. DeepFH properly distinguishes between object and background by utilizing semantics in deep features.

6.2.4. Post-Processing

In Sec. 5.3, multiple post-processing steps on top of the refinement system were introduced. Tab. 9 presents the detailed results of each post-processing step on the final result. First, the bilateral filtering on superpixel-level improves the results significantly, as the numbers indicate. This step mainly adjusts the results for adjacent superpixels with similar color. The next two steps apply opening and closing to the proposals eliminating segmentation artifacts. This has a rather large effect in qualitative results, however, the quantitative results also show an improvement. The final NMS for near duplicate removal reduces the AR@1000 slightly as expected, however, the more relevant AR@10 and AR@100 significantly improve. Therefore, Tab. 9 shows improvements for all individual post-processing steps.

7. Conclusion

In this paper, we tackle the problem of class-agnostic object proposal generation as a typical first step in object detection pipelines. We introduced the problem of imprecise object proposals due to subsampling in CNN-based systems and showed that typical solutions are not feasible with thousands of proposals. Therefore, we propose a superpixel-based refinement approach on top of the AttentionMask system. The refinement utilizes our novel DeepFH segmentation and superpixel pooling on deep features. Thereby, we combine coarse initial proposals and precise superpixel segmentations to generate precise object proposals.

The results on the COCO dataset with LVIS annotations showed the improvements of our overall system compared to state-of-the-art object proposal systems. We further showed the increased segmentation quality of the proposals with typical segmentation measures. Finally, we evaluated the influence of the chosen segmentation method in detail. For best performance, a segmentation with less artificial over-segmentation of uniform areas is necessary, like FH and our proposed DeepFH. In addition, we showed an improved performance of our DeepFH segmentation compared to FH for the object proposal refinement task and the general superpixel segmentation task. However, the results also indicated that just above 50% of the segmentation's potential is utilized for the object proposal refinement. Thus, a more sophisticated superpixel classification is needed to reach the superpixels' potential on this task.

References

[1] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., 2012. SLIC superpixels compared to state-of-the-art superpixel methods. TPAMI 34, 2274–2282.

[2] Ahn, J., Cho, S., Kwak, S., 2019. Weakly supervised learning of instance segmentation with inter-pixel relations, in: CVPR, pp. 2209–2218.

[3] Ahn, J., Kwak, S., 2018. Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation, in: CVPR, pp. 4981–4990.

[4] Van den Bergh, M., Boix, X., Roig, G., de Capitani, B., Van Gool, L., 2012. SEEDS: Superpixels extracted via energy-driven sampling, in: ECCV, pp. 13–26.

[5] Chen, L.C., Hermans, A., Papandreou, G., Schroff, F., Wang, P., Adam, H., 2018. MaskLab: Instance segmentation by refining object detection with semantic and direction features, in: CVPR, pp. 4013–4022.

[6] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L., 2017. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. TPAMI 40, 834–848.

[7] Felzenszwalb, P.F., Huttenlocher, D.P., 2004. Efficient graph-based image segmentation. IJCV 59, 167–181.

[8] Gidaris, S., Komodakis, N., 2016. Attend refine repeat: Active box proposal generation via in-out localization, in: BMVC, pp. 90.1–90.13.

[9] Girshick, R., 2015. Fast R-CNN, in: ICCV, pp. 1440–1448.

[10] Gupta, A., Dollar, P., Girshick, R., 2019. LVIS: A dataset for large vocabulary instance segmentation, in: CVPR, pp. 5356–5364.

[11] He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017a. Mask R-CNN, in: ICCV, pp. 2961–2969.

[12] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: CVPR, pp. 770–778.

[13] He, Y., Chiu, W.C., Keuper, M., Fritz, M., 2017b. STD2P: RGBD semantic segmentation using spatio-temporal data-driven pooling, in: CVPR, pp. 4837–4846.

[14] Hosang, J., Benenson, R., Dollár, P., Schiele, B., 2015. What makes for effective detection proposals? TPAMI 38, 814–830.

[15] Hou, Q., Cheng, M.M., Hu, X., Borji, A., Tu, Z., Torr, P.H., 2017. Deeply supervised salient object detection with short connections, in: CVPR, pp. 3203–3212.

[16] Hu, H., Lan, S., Jiang, Y., Cao, Z., Sha, F., 2017. FastMask: Segment multi-scale object candidates in one shot, in: CVPR, pp. 991–999.

[17] Jampani, V., Sun, D., Liu, M.Y., Yang, M.H., Kautz, J., 2018. Superpixel sampling networks, in: ECCV, pp. 352–368.

[18] Kirillov, A., Wu, Y., He, K., Girshick, R., 2020. PointRend: Image segmentation as rendering, in: CVPR, pp. 9799–9808.

[19] Kwak, S., Hong, S., Han, B., 2017. Weakly supervised semantic segmentation using superpixel pooling network, in: AAAI, pp. 4111–4117.

[20] Li, G., Yu, Y., 2016. Deep contrast learning for salient object detection, in: CVPR, pp. 478–487.

[21] Li, H., Liu, Y., Ouyang, W., Wang, X., 2017. Zoom out-and-in network with recursive training for object proposal. arXiv preprint arXiv:1702.05711 .

[22] Li, R., Li, K., Kuo, Y.C., Shu, M., Qi, X., Shen, X., Jia, J., 2018. Referring image segmentation via recurrent refinement networks, in: CVPR, pp. 5745–5753.

[23] Lin, G., Milan, A., Shen, C., Reid, I., 2017. RefineNet: Multi-path refinement networks for high-resolution semantic segmentation, in: CVPR, pp. 1925–1934.

[24] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., 2014. Microsoft COCO: Common objects in context, in: ECCV, pp. 740–755.

[25] Liu, M.Y., Tuzel, O., Ramalingam, S., Chellappa, R., 2011. Entropy rate superpixel segmentation, in: CVPR, pp. 2097–2104.

[26] Luengo, I., Basham, M., French, A.P., 2016. SMURFS: Superpixels from multi-scale refinement of super-regions, in: BMVC, pp. 4.1–4.12.

[27] Ošep, A., Voigtlaender, P., Weber, M., Luiten, J., Leibe, B., 2020. 4D generic video object proposals, in: ICRA, pp. 10031–10037.

[28] Park, H., Jeong, J., Yoo, Y.J., Kwak, N., 2017. Superpixel-based semantic segmentation trained by statistical process control, in: BMVC, pp. 78.1–78.13.

[29] Pinheiro, P.O., Collobert, R., Dollár, P., 2015. Learning to segment object candidates, in: NIPS, pp. 1990–1998.

[30] Pinheiro, P.O., Lin, T.Y., Collobert, R., Dollár, P., 2016. Learning to refine object segments, in: ECCV, pp. 75–91.

[31] Pont-Tuset, J., Arbelaez, P., Barron, J.T., Marques, F., Malik, J., 2017. Multiscale combinatorial grouping for image segmentation and object proposal generation. TPAMI 39, 128–140.

[32] Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks, in: NIPS, pp. 91–99.

[33] Ren, X., Malik, J., 2003. Learning a classification model for segmentation, in: ICCV, pp. 10–17.

[34] Stutz, D., Hermans, A., Leibe, B., 2018. Superpixels: An evaluation of the state-of-the-art. Computer Vision and Image Understanding 100, 1–27.

[35] Tu, W.C., Liu, M.Y., Jampani, V., Sun, D., Chien, S.Y., Yang, M.H., Kautz, J., 2018. Learning superpixels with segmentation-aware affinity loss, in: CVPR, pp. 568–576.

[36] Wilms, C., Frintrop, S., 2017. Edge adaptive seeding for superpixel segmentation, in: GCPR, pp. 333–344.

[37] Wilms, C., Frintrop, S., 2018. AttentionMask: Attentive, efficient object proposal generation focusing on small objects, in: ACCV, pp. 678–694.

[38] Wilms, C., Frintrop, S., 2020. Superpixel-based refinement for object proposal generation, in: ICPR, pp. 4965–4972.

[39] Wilms, C., Heid, R., Sadeghi, M.A., Ribbrock, A., Frintrop, S., 2020. Which airline is this? airline logo detection in real-world weather conditions, in: ICPR, pp. 4996–5003.

[40] Yang, F., Sun, Q., Jin, H., Zhou, Z., 2020. Superpixel segmentation with fully convolutional networks, in: CVPR, pp. 13964–13973.

[41] Yao, J., Boben, M., Fidler, S., Urtasun, R., 2015. Real-time coarse-to-fine topologically preserving segmentation, in: CVPR, pp. 2947–2955.

[42] Yu, F., Koltun, V., 2016. Multi-scale context aggregation by dilated convolutions. ICLR .

[43] Zhang, C., Lin, G., Liu, F., Yao, R., Shen, C., 2019. CANet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning, in: CVPR, pp. 5217–5226.