Improving boundary alignment by regularizing superpixels in encoder-decoder models

Jort de Jong (1397885)
Eindhoven Univeristy of Technology
j.m.d.jong@student.tue.nl

***Abstract*—Fully convolutional networks (FCNs) have proven effective at the task of semantic segmentation. FCNs and succeeding models employ an encoder-decoder structure. Recently, encoder-decoder segmentation models have adopted superpixels. Superpixels represent an image with local clusters of pixels. The decoder is substituted for superpixel based upsampling. The superpixels are responsible for the boundary alignment in the model output. When the class boundaries in the model output do not align with the class boundaries in the ground truth, the boundary alignment is considered poor. Applications within quality assessment and photo/video editing rely on accurate boundary alignment. Poor boundary alignment is especially prevalent when training labels are coarse. Coarse labels contain unlabeled pixels, with the unlabeled pixels concentrated around the class boundaries. In this work we propose to regularize the superpixels employed by encoder-decoder models. The proposed regularization term encourages SLIC-superpixels. SLIC-superpixels are based on pixel color and position, independent of the segmentation label. The regularization term can be applied to any encoder-decoder model that employs superpixel based upsampling. We evaluate the impact of our regularization term on FCN-16 with superpixel based upsampling and find it effective at improving boundary alignment. The impact of regularization is not limited to coarse labels, but extends to fine labels. Boundary recall improves up to 15.9% on the Cityscapes dataset and up to 60.3% on the SUIM dataset.**

I. Introduction

With fully convolutional networks (FCN) [14], Long *et al.* showed the effectiveness of the encoder-decoder structure for semantic segmentation, where the objective is to categorise each pixel into a class. The decoder is tasked with upsampling the low resolution encoder prediction. Many popular segmentation models employ this encoder-decoder structure [14, 12, 2, 17, 16]. Recently, T. Suzuki [21] proposed to substitute the decoder for superpixel based upsampling. Superpixels are a useful representation to reduce the complexity of image data and are common in many computer vision tasks. A superpixel is a local cluster of pixels. In the method proposed by T. Suzuki, superpixels are learned based on the encoder features. Incorporating superpixels into FCN-16 results in the HCFCN-16 architecture, depicted in figure 3. The low resolution encoder prediction is upsampled with the superpixels. Substituting the decoder for superpixel based upsampling increases increases accuracy or reduces inference time.

Most literature on semantic segmentation focuses on fine labels. With fine labels, each pixel in the image is labeled. Fine segmentation labels require significant labour to collect. Subsequently, many data collection efforts prefer coarse labels over fine labels. Coarsely annotating an image involves drawing simple polygons. Each pixel within a polygon is assigned a common class. Coarse labels contain unlabeled pixels, with the unlabeled pixels concentrated around the class boundaries. Figure 1 depicts a fine and a coarse label. Models trained on coarse labels often suffer from imprecise segmentations, where the predicted class boundaries do not align with the ground truth class boundaries. This is referred to as poor boundary alignment and is measured with boundary recall.

Precise segmentations are desirable in applications. For some applications, precise boundaries are critical. Applications within photo/video editing rely on precise segmentations. The task of blurring a segment of the image/frame relies on precise segmentations. Like blurring a licence plate or the background in a video conference. Only the desired segment should be blurred and therefore segmented precisely. Applications within quality assessment require precise segmentations, like in the motherboard manufacturing process. Motherboards are constructed as follows. First a layer of solder paste is printed on the base circuit board. A series of robots place components, like resistors and capacitors, in place. The components become attached to the board when an oven melts the solder. Components can shift out of place when the solder melts. At this stage, a photo is taken from the board for quality assurance. A segmentation of the image is compared against the design. A precise segmentation allows for detection of misplaced components. An imprecise segmentation does not align with the design, leading to false positives. These applications motivate an approach that produces precise segmentations, even when training labels are coarse.

A supervised loss relies on the pixel labels to train a segmentation model. Regularization has proven effective in minimizing the impact of missing data. This motivates the use of regularization to cope with the missing pixel labels in coarse labels. In encoder-decoder models, that employ superpixels, the superpixels are responsible for boundary alignment. In this work we propose to regularize

these superpixels. We investigate the following research question: Can regularization be effective in significantly improving the boundary alignment of an encoder-decoder segmentation model when trained on coarse labels? We propose a regularization term that encourages superpixels to be SLIC-superpixels. SLIC-superpixels are based on pixel color and position, independent of the segmentation label.

We demonstrate significant improvements to boundary alignment by regularizing the superpixels in HCFCN-16. The improved boundary alignment is not limited to coarse labels and extends to fine labels. Moreover, we demonstrate significant improvements to pixel accuracy in limited settings. Our contributions are as follows:

- We propose a regularization term that encourages SLIC-superpixels in encoder-decoder models that employ superpixel based upsampling.
- We demonstrate significant improvements to boundary alignment and pixel accuracy on the Cityscapes and SUIM datasets when regularizing HCFCN-16.
- We provide a PyTorch implementation of the superpixel downsampling operation required by the regularization term.

II. Related Work

Semantic segmentation. Many applications, like autonomous driving, terrain segmentation and photo editing, rely on semantic segmentation. Semantic segmentation is the task of assigning a class to each pixel in an image. The Cityscapes [5] and the COCO-Stuff [13] datasets are popular semantic segmentation benchmarks. These datasets provide fine segmentation labels. With fine segmentation labels, each pixel is labeled. The Cityscapes dataset also provides coarse labels. Coarsely annotating an image involves drawing simple polygons. With coarse labels some pixels are left unlabeled. Deep learning approaches have proven effective on the task of semantic segmentation.

Encoder-decoder models. Many segmentation models employ an encoder-decoder structure. The encoder-decoder models [14, 12, 2, 17, 16] use an encoder to extract high level features from the input image. Fully convolutional encoders, like ResNet [8], are used to maintain the spatial structure. The resulting feature map has a lower spatial resolution than the input image. The decoder is tasked with upsampling this feature map to the desired segmentation output. Fully convolutional networks (FCNs) [14] have set the ground work for encoder-decoder models. FCNs use convolutional layers and downsampling operations to extract high level features from an image. The encoder in FCN-16 reduces the spatial resolution to 1/16 the input resolution. A fully convolutional classifier brings the high dimensional features down to the desired number of classes. Finally, bilinear interpolation increases the spatial resolution to the final segmentation output. FCNs have proven effective for semantic segmentation, although they are prone to output imprecise segmentations. Bilinear interpolation is unable to recover much of the fine detail lost by the encoder. More recent approaches [12, 17, 16] employ learnable parameters in the decoder. These approaches are able to recover more detail, often with increased computational budget.

Superpixels. Superpixels are local clusters of pixels that share similar features. Superpixels are often used as a substitute for pixels as an efficient image representation. Simple Linear Iterative Clustering (SLIC) [1] is an effective algorithm for generating superpixels. SLIC employs K-means clustering to group nearby pixels into superpixels based on position and color features. Specifically, pixels are clustered in 5-D space defined by the x, y pixel coordinates and L, a, b values of the CIELAB color space. The superpixel size and compactness are both hyperparameters.

Several methods aim to integrate superpixels into neural networks. [6] and [9] use pre-computed superpixels to increase model efficiency. SpixelFCN [22] is a fully convolutional network that can be trained to output superpixels. In SpixelFCN a pixel does not belong to a single superpixel. Instead a pixel belongs to nine local superpixels with various levels of association. SpixelFCN outputs an association map $\mathrm{Q} \in \mathbb{R}^{H \times W \times 9}$. The association map holds a distribution for each pixel. The distribution quantifies the pixel association to the surrounding superpixels. Two different choices of loss function are available to train SpixelFCN. The first loss function relies on segmentation labels to learn superpixels. The second loss function imitates the SLIC objective and does not rely on segmentation labels. Fengting Yang *et al.* propose a downsampling/upsampling scheme to utilize the predicted superpixels in downstream tasks. Instead of feeding a high resolution image to a CNN. The image is first downsampled using the superpixels provided by SpixelFCN. The lower resolution image is used by the CNN in the downstream task. Finally the CNN output is upsampled using the same superpixels to recover the lost resolution. Fengting Yang *et al.* demonstrate the effectiveness of SpixelFCN by applying the scheme to PSMNet [3] for the application of stereo matching.

In [21] T. Suzuki proposes the use of superpixels in encoder-decoder segmentation models. T. Suzuki uses the notion of assignment matrices and hierarchical clusters. The superpixels are given by the assignment matrices. The notion of superpixels and clusters can be used interchangeably. This work uses the notion of superpixels. T. Suzuki proposes to hierarchically group pixels together at downsampling layers. At each downsampling layer the model has two additional weight matrices, W and $\tilde{W}$. The feature maps before and after downsampling are multiplied by W and $\tilde{W}$ respectively. Following equation 1 an assignment matrix is obtained. This

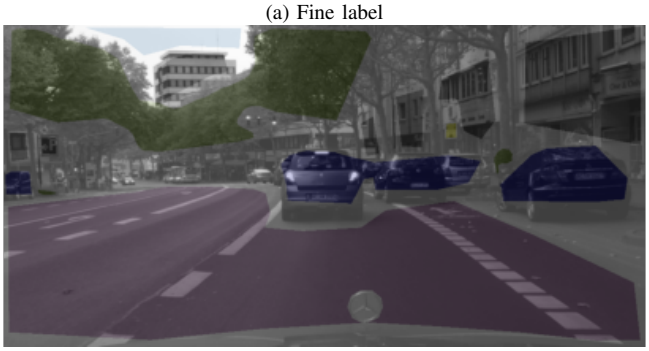
assignment matrix assigns each pixel in the feature map before downsampling to a pixel in the feature map after downsampling. Each downsample layer has an assignment matrix. This way, each pixel in the original image is assigned to a pixel in the low resolution encoder output. The pixels in the low resolution encoder output are the superpixel seeds. Every pixel in the original image is assigned to a superpixel seed. A superpixel consists of all the pixels assigned to the corresponding superpixel. The classifier receives the encoder output and returns a prediction for each superpixel seed. Finally the low resolution prediction is upsampled with the superpixels. Upsampling with the superpixels is done by upsampling with the assignment matrices. Superpixel based upsampling corresponds with sharing the prediction for each superpixel seed among the pixels associated with that superpixel. In practice the superpixels are soft superpixels. A pixel is not assigned to a single superpixel seed. Instead a pixel is assigned to nine local superpixel seeds with various level of association. The method proposed by T. Suzuki significantly reduces inference time with a slight reduction in pixel accuracy when applied to PSPNet [23] and DeepLabv3 [4]. Applying the proposed method to FCN, referred to as HCFCN, increases the pixel accuracy at the cost of inference time. Figure 3 depicts the architecture of HCFCN-16. This work adopts the superpixels as proposed by T. Suzuki and continues with HCFCN-16 as the baseline.

Coarse labels. Supervised learning of convolutional neural networks is known to be data intensive. Often, many images are required to achieve satisfactory results. Semantic segmentation labels in particular are very costly to collect. Each pixel in a fine segmentation label is assigned a class. The object boundaries in particular are difficult to annotate accurately [18]. As a result, many data collectors opt for coarsely annotated images. Coarsely annotating an image involves drawing simple polygons. Each pixel within this polygon is assigned a common class. In figure 1, a fine and a coarse label are shown. Fine annotations take over three times as much time as coarse annotations [24].

Aleksandar Zlateski et al. [24] investigate the importance of label quality for semantic segmentation. Training on fine labels yields superior results compared to training on coarse labels. However, investing time in producing a large number of fine labels is not the best data collection strategy. Ideally more time should be spent on producing coarse labels than time spent on producing fine labels. This strategy yields more labels and results in superior pixel accuracy.

Segmentation methods for coarse labels. Recently more methods have been proposed for the weakly-supervised setting. With weakly-supervised segmentation the dataset consists of annotations that are relatively easy to obtain. Annotations such as labels of objects present in the image, coarse labels, bounding boxes or scribble annotations. Still, literature aimed at coarse labels is limited. In [11] Ning Jia *et al.* propose an

(a) Fine label



(b) Coarse label

Fig. 1: A fine and a coarse label from the Cityscapes dataset

approach to refine coarse labels for the segmentation of dot-matrix batchcodes. Batchcodes are printed on the packaging of many products. Ning Jia *et al.* find that the coarse labels, that roughly correspond to the location of batchcodes, are insufficient for training a reliable segmentation model. Ning Jia *et al.* propose a label refinement process called Maximally Stable Global Region (MSGR). The workflow of generating compact box from coarse label using MSGR is depicted in figure 2.

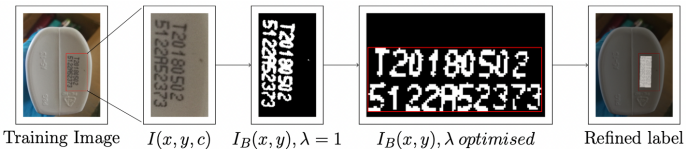


Fig. 2: The workflow of generating compact box from coarse label using MSGR.

Shima Nofallah *et al.* propose an approach for the segmentation of skin biopsy images with coarse and sparse annotations [19]. The proposed approach segments larger and smaller entities separately. First, an image goes to stage 1. In stage 1 a segmentation model generates a segmentation mask with the following entities: background, stratum corneum, epidermis, dermis and unlabeled. Stage 2 consists of two separate segmentation models that each receive a modified image. The stage 2-Dermis segmentation model receives the image with the epidermis entity masked. The stage 2-Epidermis segmentation model receives the image with the dermis entity masked. Each segmentation model in stage 2

is tasked with segmenting a different entity. Stage 2-Dermis generates the segmentation mask of the entity, dermal nests, present in the dermis. Stage 2-Epidermis generates the segmentation mask of the entity, epidermal nests, present in the epidermis. In the final stage, the stage 2-Dermis and stage 2-Epidermis segmentation masks are overlaid on the stage 1 mask, and the final tissue-level segmentation mask is generated. Both methods discussed [11, 19] are domain specific. This work aims to propose a segmentation method that is domain agnostic.

In this work, we aim to improve the boundary alignment of encoder-decoder models, that utilize superpixels as in [21], when training labels are coarse. The following section proposes a regularization term that encourages the superpixels to be SLIC-superpixels.

III. Proposed Method

As discussed, T. Suzuki proposes to integrate superpixels into existing encoder-decoder models, like FCN. FCN with superpixel based upsampling as the decoder is referred to as HCFCN. The superpixels are responsible for class boundaries. Poor boundary alignment in the segmentation output is a result of poor superpixels. When training labels are coarse, the pixels around the class boundaries are often unlabeled. As such, the supervised loss provides little to no incentive for precise boundaries. This is a problem of missing data, motivating the use of regularization. As the superpixels are responsible for the predicted class boundaries, we aim to regularize the superpixels. A bias for superpixels needs to be independent of the segmentation labels. Ideally, the pixels within a superpixel should share a common class. SLIC-superpixels are a good bias for superpixels. SLIC-superpixels do not depend on the segmentation labels. And, locally, pixel color correlates with pixel class. We propose a regularization term that encourages SLIC-superpixels in encoder-decoder models that employ superpixels as in [21].

A. Preliminary

Let $I \in \mathbb{R}^{HW \times 3}$ be an RGB image where H and W denote the image height and width. Let $x_i^{(s)} \in \mathbb{R}^N$ be a N-dimension feature vector of the i-th pixel in the feature map $x^{(s)}$. Superscript s denotes an output stride, namely, the resolution of $x^{(s)}$ is (HW/s^2). Fully convolutional networks (FCNs) consists of convolutional blocks, with convolutional layers and ReLU activations. Let $g^{(s)}$ be a convolutional block, with input resolution (HW/s^2). The convolutional blocks are separated by downsampling layers such as max-pooling or strided convolutional layers. The downsampling layers reduce the spatial resolution of the feature maps throughout the FCN. Assume, without loss of generality, that a downsampling layer, d, reduces the resolution by half. Let $z^{(2s)}$ be the feature map obtained by downsampling $x^{(s)}$. Following T. Suzuki [21], pixels are hierarchically grouped together at downsampling layers in the encoder to form superpixels. The superpixels do not impact the encoder output in the forward pass and are only used by the decoder. In figure 3 the architecture of HCFCN-16 is depicted with the described notation.

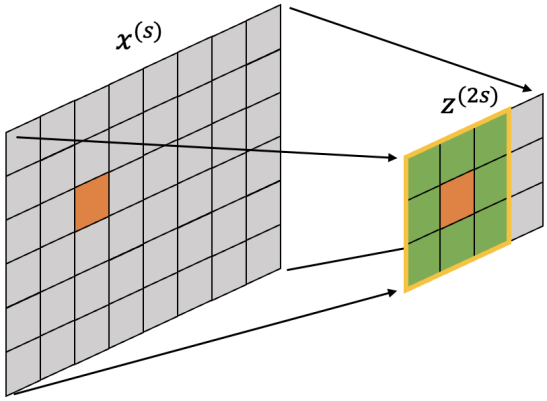


Fig. 4: Illustration of the candidate seeds. Feature map $x^{(s)}$ is downsampled to feature map $z^{(2s)}$. The assignment from the orange pixel in $x^{(s)}$ to the seeds in $z^{(2s)}$ is restricted to the nine candidate seeds within the yellow frame.

The superpixels are given by the set of assignment matrices A. At each downsampling layer an assignment matrix $A^{(s)} \in \mathbb{R}^{\frac{HW}{s^2} \times 9}$ can be learned. In figure 3 we can see that the assignment matrix $A^{(s)}$ is obtained from the feature map $x^{(s)}$ before the downsampling layer and the feature map $z^{(2s)}$ after the downsampling layer. Every pixel in the feature map $x^{(s)}$ has a soft assignment to the pixels in $z^{(2s)}$. We refer to the pixels in $z^{(2s)}$ as seeds. Not all seeds in $z^{(2s)}$ are considered for assignment, as this is not computationally applicable. Pixel $x_i^{(s)}$ is assigned to a subset of seeds in $z^{(2s)}$ called the candidate seeds. In line with existing superpixel segmentation methods [1, 21, 22] the candidate seeds are restricted to the nine surrounding seeds, as illustrated by figure 4. The assignment from $x_i^{(s)}$ to $z_j^{(2s)}$ is defined as follows:

$$A_{ij}^{(s)} = \frac{exp(\mathcal{S}(W^{(s)}x_i^{(s)}, \tilde{W}^{(2s)}z_{\mathcal{N}(i,j)}^{(2s)})/\tau)}{\sum_{k=1}^{9} exp(\mathcal{S}(W^{(s)}x_i^{(s)}, \tilde{W}^{(2s)}z_{\mathcal{N}(i,k)}^{(2s)})/\tau)}, \quad (1)$$

where τ is a temperature parameter, set at 0.07 following [21]. $W^{(s)} \in \mathbb{R}^{K \times N}$ and $\tilde{W}^{(2s)} \in \mathbb{R}^{K \times M}$ are learnable weight matrices which map the feature maps into a K-dimensional space. In this work K is set at 48, as this resulted in the highest accuracy by HCFCN-16 over the Cityscapes validation set when trained on fine training labels. The map $\mathcal{N}(i,j)$ returns the index in $z^{(2s)}$ corresponding to the j^{th} neighbor seed of pixel i in $x^{(s)}$. $\mathcal{S}$ denotes a similarity function. Following [21], the cosine similarity is used in this work.

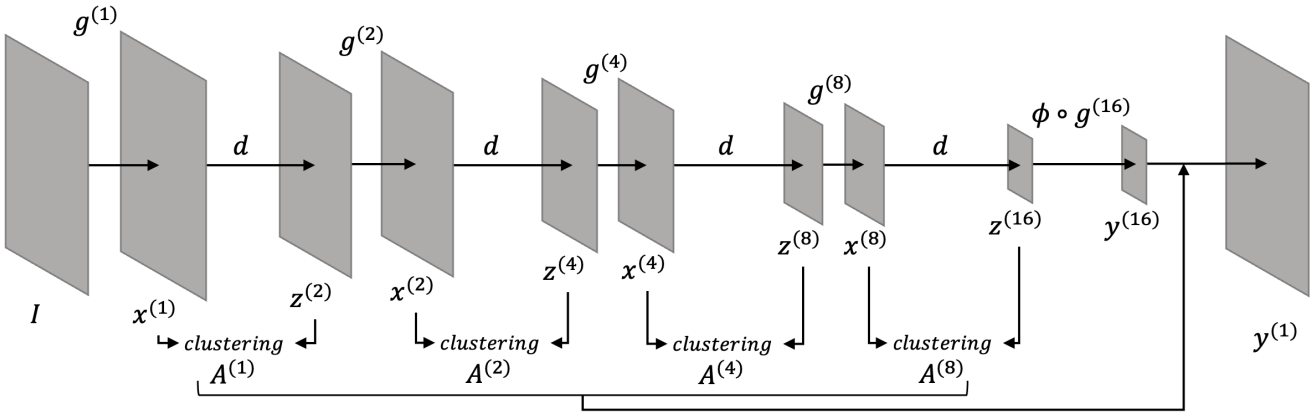


Fig. 3: HCFCN-16 architecture with convolutional layers $g^{(s)}$ and downsampling operation d. Assignment matrix $A^{(s)}$ is obtained at the layer that downsamples $x^{(s)}$ to $z^{(2s)}$. The low resolution output $y^{(16)}$ is upsampled by the assignment matrices A [21].

The classifier ϕ takes the low resolution encoder output $z^{(16)}$ and outputs a class prediction $y^{(16)}$ for each superpixel seed. Finally, the superpixels are used to upsample the low resolution classifier output $y^{(16)}$ to the original resolution. Upsampling y^{16} to y^1 is done as follows:

$$y^{(1)} = \prod_{s'=\{8,4,2,1\}} A^{(s')} y^{(16)}. \tag{2}$$

Superpixel upsampling corresponds with sharing the prediction for each superpixel seed among the pixels associated with that superpixel. It is not necessary to learn assignment matrices at each downsampling layer. T. Suzuki found little to no improvements in learning assignment matrices at the first two downsampling layers. Following [21], in our experiments only assignment matrices $A^{(8)}$ and $A^{(4)}$ are obtained. Upsampling y^{16} to y^4 is done using the assignment matrices. Upsampling from y^4 to y^1 is done with bilinear upsampling.

B. Regularization Term

The SLIC algorithm clusters nearby pixels into superpixels based on CIELAB color and positional features. In [22] Fengting Yang *et al.* propose a loss function which imitates the SLIC objective. The loss function encourages the superpixels outputted by SpixelFCN to be SLIC-superpixels. The loss function proposed by Fengting Yang *et al.* does not apply to the superpixels in this work, because the superpixels in this work are embedded in the assignment matrices A. We propose a regularization term that encourages the superpixels, given by the assignment matrices A, to be SLIC-superpixels.

Let us first define a general function $q(x, A)$ required by the regularization term. The function $q(x, A)$ takes as input a full resolution feature map x and superpixels A. The function $q(x, A)$ returns a full resolution feature map where each pixel contains the weighted average feature of its corresponding superpixels. First, the full resolution feature map x is downsampled using the superpixels. The result is a low resolution feature map where each pixel is a superpixel seed. Each superpixel seed contains the weighted average feature computed over the pixels assigned to that superpixel. Second, this low resolution feature map is upsampled with the same superpixels A. The result is a full resolution feature map. Each pixel in this feature map contains the weighted average feature of its corresponding superpixels. The function $q(x, A)$ is given by equation 3 and illustrated in figure 5. For illustration purposes each pixel is assigned to a single superpixel instead of a soft assignment to nine superpixels.

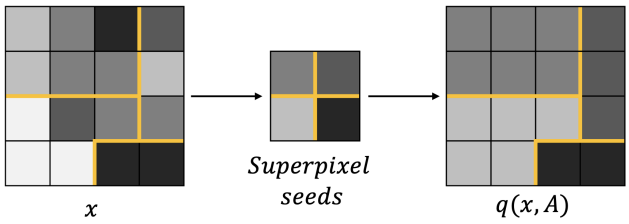


Fig. 5: Illustration of the function $q(X, A)$ defined by equation 3. The superpixel boundaries are highlighted in yellow. The feature map X is downsampled using the the superpixels before upsampling with the same superpixels. For illustration purposes each pixel is assigned to a single superpixel instead of a soft assignment to nine superpixels.

$$q(x, A) = \prod_{s'=\{8,4,2,1\}} A^{(s')} \prod_{s'=\{1,2,4,8\}} A^{(s')} x \tag{3}$$

Equation 3 consists of two distinct parts. Downsampling a full resolution feature map x with the assignment matrices A. And upsampling the resulting low resolution feature map with the same assignment matrices. For the latter step, given by equation 2, a PyTorch implementation is provided in [21]. In appendix VII-C we provide a PyTorch implementation to

downsample a feature map with an assignment matrix.

Finally we use the function q to construct our regularization term as follows. Assuming the image I is an RGB image, we first need to map I to the CIELAB color space. The CIELAB color image is given by $f(I)$, where the function f maps an RGB image to the CIELAB color space. The CIELAB color image is downsampled before being upsampled with the superpixels A, using function q. Each pixel in the resulting feature map, $q(f(I), A)$, holds the average CIELAB color features of its corresponding superpixels. For each pixel we minimize the ℓ_2 distance between the CIELAB color features and the average CIELAB color features of its corresponding superpixels.

The original SLIC algorithm has a hyperparameter to encourage compact superpixels. This hyperparameter persists in the regularization term. We represent a pixel's position by its image coordinates $\mathbf{p} = [x, y]^T$, with $\mathbf{P}$ denoting the map of pixel coordinates for each pixel. Using function q, we obtain the feature map $q(\mathbf{P}, A)$. Each pixel in the feature map $q(\mathbf{P}, A)$ holds the average position of its corresponding superpixels. To encourage compact superpixels, the ℓ_2 distance between the pixel position and the average position of its corresponding superpixels is minimized. Finally, the resulting regularization term is given by equation 4. The hyperparameter m determines the compactness of the superpixels and is set to 0 in our experiments.

$$L_{SLIC}(I, A) = \sum_{\mathbf{p}} ||f(I)[\mathbf{p}] - q(f(I), A)[\mathbf{p}]||_2 + m * ||\mathbf{p} - q(\mathbf{P}, A)[\mathbf{p}]||_2. \tag{4}$$

The regularization term is added to the cross-entropy loss. The regularization strength is set by λ. The complete loss is given by equation 5. The final inference and training pipeline is depicted in figure 6.

$$L_{total} = L_{cross-entropy} + \lambda * L_{SLIC}. \tag{5}$$

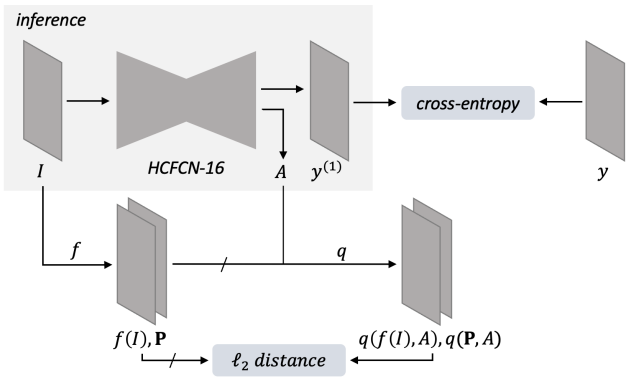


Fig. 6: The inference and training pipeline for the regularized HCFCN-16.

Dataset	Resolution	Training size	Validation size	Test size
Cityscapes	1024 × 512	2369	606	500
SUIM	512 × 512	1290	150	110

TABLE I: Comparison of the Cityscapes and SUIM dataset properties.

IV. Experimental Results

We train HCFCN-16 with the proposed regularization term and compare against FCN-16 and HCFCN-16 without regularization.

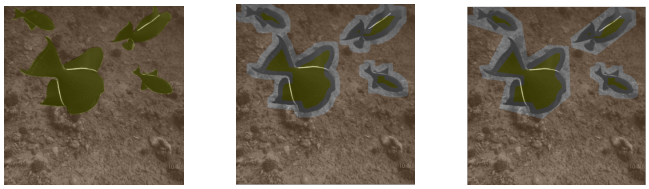
A. Experimental Setup

Architecture. The proposed regularization term can be used in combination with any encoder-decoder segmentation model that employs superpixels as proposed by T. Suzuki. This work is limited to HCFCN-16. All models in this work employ a ResNet34 [8] encoder. The final layer, *conv5_x*, is excluded to ensure an output stride of 16. Following T. Suzuki, all downsampling operations are substituted with the modulated deformable convolution (DCNv2) [16] with a stride of two. As suggested by T. Suzuki, only assignment matrices $A^{(4)}$ and $A^{(8)}$ are learned. Upsampling from y^4 to y^1 is done with bilinear upsampling. The architectures of HCFCN-16 with and without regularization are identical.

Training. We employ an unbiased universal training method for all approaches. The results in the ablation study, section IV-C, are obtained using the following training method. Cross-entropy is used as the supervised training loss. The regularization term m is set to 0 and λ is set to 0.075 and 0.05 on the Cityscapes and SUIM datasets respectively. The regularization hyperparameter choices are motivated in section IV-C, the ablation study. During training, images are randomly cropped to a size of 512 by 512 pixels to fit into memory. At random, images are horizontally flipped. The optimizer of choice is Adam with a learning rate of 0.0005. Training is done for 50 and 75 epochs on the Cityscapes and SUIM datasets respectively. Training for 50 and 75 epochs allows the models to converge. The model with the highest validation accuracy is returned.

The results in section IV-B are obtained by training on the training set as described above before fine-tuning on the training and validation sets. Fine-tuning is done following algorithm 7.3 in [7]. After training on the training set, the model is fine-tuned on the training and validation sets until the loss on the validation set matches the loss on training set before fine-tuning. As there is no guarantee that this criteria will be reached, fine-tuning is limited to 15 epochs.

Evaluation. All methods are evaluated with two semantic segmentation datasets. The Cityscapes dataset [5] and the SUIM dataset [10]. The Cityscapes dataset contains urban street scenes with detailed semantic segmentation labels.



(a) Fine label (b) Eroded label (c) Coarse label

Fig. 7: Generating a coarse label from a fine label on the SUIM dataset. The fine label (a) is eroded by 12 pixels resulting in the eroded label (b). Finally the Douglas-pecker algorithm is used to approximate the eroded label with a polygon to obtain the coarse label (c).

The SUIM dataset contains semantic segmentation labels for underwater images. An example of the Cityscapes and SUIM dataset can be seen in figure 8. The Cityscapes dataset offers fine and coarse labels. All Cityscapes images are resized to 1024 by 512 pixels. Next to the Cityscapes dataset, we evaluate on the SUIM dataset. The proposed regularization term is expected to perform well on the SUIM dataset given its vibrant color palette. The resolution of the SUIM dataset is reduced by half to 512 by 512 pixels. The SUIM dataset does not provide coarse labels. As such, coarse labels are derived from the fine labels, following [24]. First, each class in the SUIM fine labels is eroded by 12 pixels. Then, the Douglas-pecker algorithm is used to approximate the eroded label with a polygon. The resulting coarse labels imitate the coarse labels of the Cityscapes dataset. The process of generating coarse labels is depicted in figure 7. The SUIM datasets provides 110 test samples. In this work, 150 samples from the provided *train_val* set are reserved for validation. This leaves 1290 samples in the training set. For the Cityscapes dataset, the provided validation set is used as the test set. The cities *Aachen*, *Monchengladbach*, *Stuttgart* and *Weimar* from the training set are chosen at random to be reserved for validation. Table I shows the number of samples in each set for both datasets. All results in section IV-B are obtained by evaluating over the test set. In section IV-C, the ablation study, all results are obtained by evaluating over the validation set. All methods are evaluated using pixel accuracy as given by equation 6. Boundary recall [15] (BR) is used to evaluate boundary alignment and is computed following equation 7. The function $b(y, r)$ returns the set of pixels within a $(2r + 1) \times (2r + 1)$ neighborhood of the boundary pixels in y. Following [20] r is set to 0.0025 times the image diagonal. Finally, Achievable Segmentation Accuracy (ASA) is computed by equation 8 to directly evaluate the superpixels. All results are obtained by averaging over five training and evaluation runs. When a difference between methods is described as significant, a student *t*-test with an α of 0.05 supports this statement.

$$ACC(y^{(1)}, y) = \frac{\sum_{i=1}^{HW} \mathbb{1}\{y_i^{(1)} = y_i\}}{\sum_{i=1}^{HW} \mathbb{1}\{y_i\}} \tag{6}$$

	Fine training labels		Coarse training labels	
Method	Pixel acc.	BR	Pixel acc.	BR
FCN-16	87.9 ± 0.3	48.0 ± 1.09	84.5 ± 0.36	34.0 ± 0.44
HCFCN-16	**88.2 ± 0.28**	61.9 ± 0.72	84.8 ± 0.19	47.1 ± 1.89
Regularized HCFCN-16	88.2 ± 0.20	**63.6 ± 0.92**	**85.3 ± 0.18**	**54.6 ± 0.90**

TABLE II: Average results on the Cityscapes test set. Methods are trained on fine and coarse training labels and fine-tuned on the training and validation sets.

	Fine training labels		Coarse training labels	
Method	Pixel acc.	BR	Pixel acc.	BR
FCN-16	71.1 ± 1.00	6.6 ± 0.43	68.4 ± 0.78	4.8 ± 0.24
HCFCN-16	70.8 ± 1.08	14.4 ± 1.05	63.5 ± 4.67	7.8 ± 2.09
Regularized HCFCN-16	**72.9 ± 0.34**	**18.7 ± 0.70**	**70.4 ± 0.66**	**12.5 ± 0.63**

TABLE III: Average results on the SUIM test set. Methods are trained on fine and coarse training labels and fine-tuned on the training and validation sets.

$$BR(y^{(1)}, y) = \frac{n(b(y^{(1)}, 0) \cap b(y, r))}{n(b(y^{(1)}, 0) \cap b(y, r)) + n(b(y, 0) - b(y^{(1)}, r))} \tag{7}$$

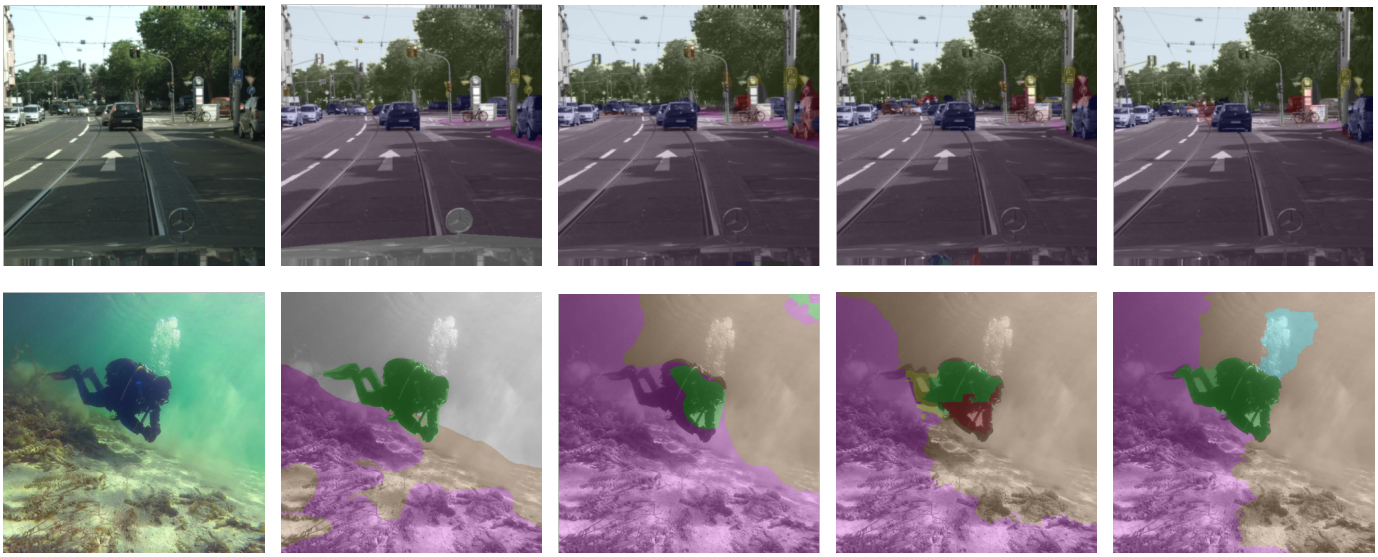
$$ASA(A, y) = ACC(q(y, A), y) \tag{8}$$

B. Results

In tables II and III all methods are evaluated on the Cityscapes test set and SUIM test set respectively. The regularized HCFCN-16 achieves significantly higher pixel accuracy than FCN-16 and HCFCN-16 without regularization when trained on Cityscapes coarse labels, SUIM coarse labels or SUIM fine labels. When trained on Cityscapes fine labels the impact of regularization on pixel accuracy is insignificant. The improvements in boundary alignment are more substantial. The regularized HCFCN-16 achieves significantly higher boundary recall than FCN-16 and HCFCN-16 without regularization on both datasets. Both pixel accuracy and boundary recall show a discrepancy between the Cityscapes and SUIM datasets. The regularization term is more effective on the SUIM dataset than on the Cityscapes dataset. In figure 8 an example of each method for each dataset is shown. The differences between methods are more apparent on the SUIM sample than on the Cityscapes sample. FCN-16 and HCFCN-16 both fail to provide precise segmentations on the SUIM dataset. On the Cityscapes sample all methods could be considered reasonably precise. This is in line with the results seen in tables II and III.

C. Ablation Study

Hyperparameter λ. We study the impact of λ when trained on Cityscapes coarse labels. With λ the regularization strength can be adjusted. In figure 9 we can see that pixel accuracy starts to degrade when λ surpasses 0.075. In figure 10 we can see boundary recall still benefiting



(f) Image (g) Ground truth (h) FCN-16 (i) HCFCN-16 (j) Regularized HCFCN-16

Fig. 8: Example results on the Cityscapes (top) and the SUIM (bottom) test sets. All methods are trained on coarse labels and fine-tuned on the training and validation set. The Cityscapes images are cropped to match the SUIM images. Examples are chosen to highlight the impact of regularization.

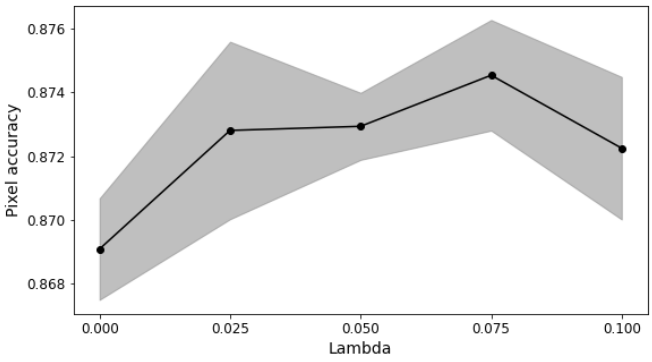


Fig. 9: Average accuracy and standard deviation on the Cityscapes validation set for various levels of λ. HCFCN-16 is trained on Cityscapes coarse training labels.

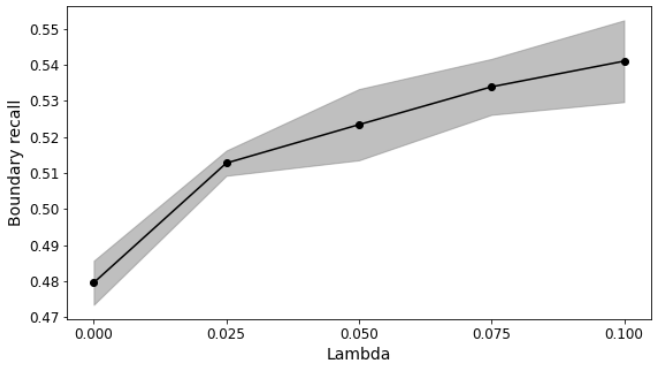


Fig. 10: Average boundary recall and standard deviation on the Cityscapes validation set for various levels of λ. HCFCN-16 is trained on Cityscapes coarse training labels.

from an increased λ. For the Cityscapes dataset, λ is set to 0.075 to maximise pixel accuracy. A similar relation is concluded on the SUIM dataset. On the SUIM dataset the maximum pixel accuracy is achieved when λ is set to 0.05.

Hyperparameter *m*. Increasing the hyperparameter m, encourages superpixels to be more compact. Low values for m often leads to scattered superpixels. In figure 11 and 12 the pixel accuracy and boundary recall for various levels of m are plotted. The regularization term is most effective when m is set to zero. This relation holds for the SUIM dataset as well. Setting m to zero does not lead to scattered superpixels. This could be explained by the supervised loss encouraging compact superpixels.

Boundary alignment. The regularization term aims to improve boundary alignment. The improved boundary recall suggests the regularization term is effective in doing so. We investigate the role of the superpixels in improving the boundary alignment. In figure 13 a region of a Cityscapes sample is highlighted. The label on the left shows the intricate class boundaries of this sample. The predicted boundaries of HCFCN-16 do not align with the ground truth boundaries exactly. Regularization arguably improves boundary alignment in this example. The superpixels of both the HCFCN-16 and the regularized HCFCN-16 show reasonable boundary alignment. The imperfect boundary alignment is a result of misclassified superpixels and not misaligned superpixels.

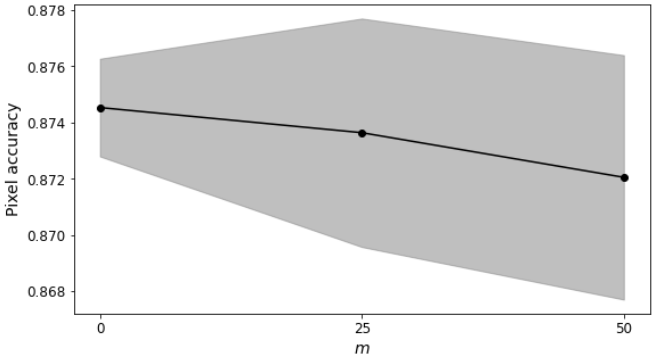


Fig. 11: Average accuracy and standard deviation on the Cityscapes validation set for various levels of m. HCFCN-16 is trained on Cityscapes coarse training labels.

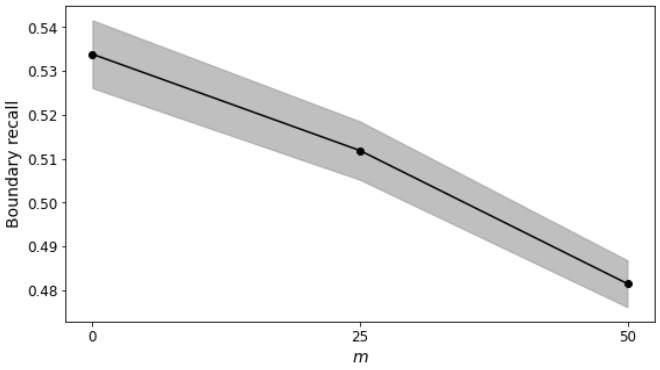
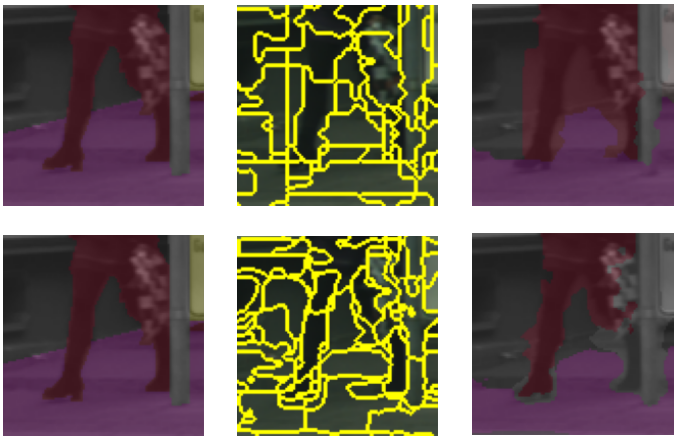


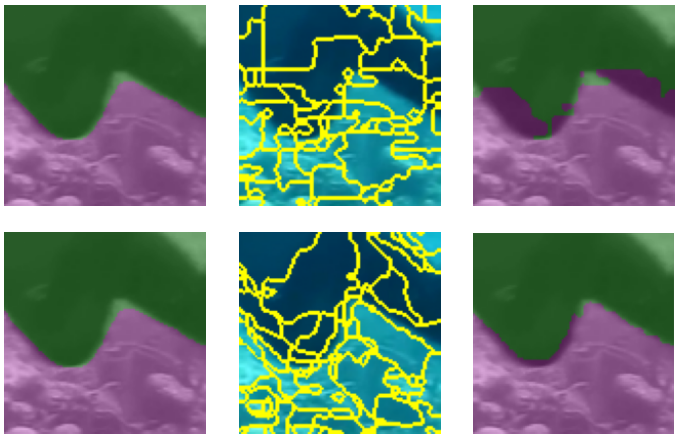
Fig. 12: Average boundary recall and standard deviation on the Cityscapes validation set for various levels of m. HCFCN-16 is trained on Cityscapes coarse training labels.



(d) Label (e) Superpixels (f) Output

Fig. 13: Example of boundary alignment on the Cityscapes validation set. HCFCN-16 is shown on the top and the regularized HCFCN-16 is shown on the bottom.

In figure 14 a region of a SUIM sample is shown. The ground truth class boundary is simple compared to the Cityscapes sample. Still HCFCN-16 fails to align the predicted boundary with the ground truth boundary. The superpixels, plotted in the middle, explain the imprecise segmentation. The superpixels in HCFCN-16 do not align with the ground truth boundary. The superpixels of the regularized HCFCN-16 do align with the ground truth boundary. As a result the predicted boundary in the the regularized HCFCN-16 output aligns with the ground truth boundary.



(d) Label (e) Superpixels (f) Output

Fig. 14: Example of boundary alignment on the SUIM validation set. HCFCN-16 is shown on the top and the regularized HCFCN-16 is shown on the bottom.

SLIC-superpixels. The regularization term encourages the superpixels to be SLIC-superpixels. We are interested in the SLIC-superpixels the regularization term encourages. We train HCFCN-16 on the regularization term exclusively, without the supervised loss. The superpixels of the resulting model are considered SLIC-superpixels. We compare these SLIC-superpixels to the superpixels when trained on fine labels, coarse labels and coarse labels with regularization. Achievable Segmentation Accuracy (ASA) is used to compare the superpixels. ASA is commonly used to evaluate superpixels. ASA measures the maximum accuracy that can be achieved given the superpixels and the label. The ASA for each method on both datasets can be seen in table IV. The SLIC-superpixels on the SUIM dataset achieve the highest ASA among the different methods, even outperforming HCFCN-16 trained on fine labels. This suggest the SLIC-superpixels are very appropriate for the SUIM dataset. The SLIC-superpixels on the Cityscapes dataset manage an ASA of 96.2 ± 0.12, matching the ASA of the superpixels when trained on coarse labels. The HCFCN-16 trained on fine labels achieves the highest ASA. The regularized HCFCN-16 trained on coarse labels results in an ASA of 97.0 ± 0.01. These results are inline with the results seen in tables II and III.

Method (label set)	Cityscapes	SUIM
HCFCN-16 (fine)	**97.3 ± 0.08**	97.5 ± 0.08
HCFCN-16 (coarse)	96.2 ± 0.19	96.8 ± 0.08
Regularized HCFCN-16 (coarse)	97.0 ± 0.01	**98.6 ± 0.01**
SLIC-superpixels (no labels)	96.2 ± 0.12	**98.6 ± 0.01**

TABLE IV: Achievable Segmentation Accuracy on the validation set of the superpixels of each method compared. The SLIC-superpixels method is obtained by training on the regularization term exclusively.

Classification error. As stated before, ASA measures the maximum accuracy that can be achieved given the superpixels. The realised accuracy is given by pixel accuracy. This allows us to define *classification error* as the difference between ASA and pixel accuracy. The regularization term directly impacts ASA, as the regularization term is aimed at the superpixels. We investigate the indirect impact of the regularization term on the classification error. In table V we can see, on the Cityscapes dataset, the regularization term has slightly increased the classification error. Suggesting that the encoder has learned features that allow for SLIC-superpixels to be learned, but are not optimal for classification. In table VI the classification errors on the SUIM dataset are given. On the SUIM dataset, the regularization term has reduced the classification error. The regularization term does not directly incentivize the encoder to improve classification. We suggest the following hypothesis to explain the improved classification: The supervised cross-entropy loss does propagate backwards through the superpixels to the encoder. Poor superpixels could restrict the encoder and classifier in learning to output accurate predictions. We can conclude that on the SUIM dataset the regularization term improves the pixel accuracy beyond the improvements to the superpixel alignment.

Method	ASA	Pixel acc.	Class. error
HCFCN-16	96.2 ± 0.19	86.9 ± 0.16	9.3 ± 0.25
Regularized HCFCN-16	97.0 ± 0.01	87.5 ± 0.17	9.5 ± 0.17

TABLE V: Classification error computed over the Cityscapes validation set. Both methods are trained on the Cityscapes coarse training labels.

Method	ASA	Pixel acc.	Class. error
HCFCN-16	96.8 ± 0.08	64.9 ± 0.49	31.9 ± 0.50
Regularized HCFCN-16	98.6 ± 0.01	70.7 ± 0.8	27.9 ± 0.80

TABLE VI: Classification error computed over the SUIM validation set. Both methods are trained on the SUIM coarse training labels.

V. Discussion

Regularizing the superpixels in HCFCN-16 significantly improves boundary alignment, as measured by boundary recall and shown in section IV-B. The improvements to boundary alignment are not limited to coarse labels, but extend to fine labels. Boundary recall improves up to 15.9% on the Cityscapes dataset and up to 60.3% on the SUIM dataset. Figure 14 shows the improved boundary alignment by the regularized HCFCN-16 on the SUIM dataset. Without regularization, HCFCN-16 suffers from poor boundary alignment on the SUIM dataset. Regularizing the superpixels impacts overall pixel accuracy as well. On the SUIM dataset, the regularized HCFCN-16 has a lower classification error than HCFCN-16. The regularization term improves pixel accuracy beyond the direct improvements to the superpixels. The impact on training time is minimal. The regularized HCFCN-16 takes 3.8% longer to complete a single epoch over the Cityscapes training set than HCFCN-16 without regularization using a NVIDIA Tesla T4 GPU.

We aim to explain the discrepancy between the effectiveness of regularization on the Cityscapes and SUIM datasets. The SUIM dataset contains fewer training samples. A smaller training set could increase the effectiveness of the regularization term. In appendix VII-A we investigate the impact of training set size on the effectiveness of regularization. The difference in training set size fails to explain the difference in regularization effectiveness. This leaves us with two explanations. The first explanation is a result of the SLIC-superpixels. SLIC-superpixels are obtained by training HCFCN-16 on the regularization term exclusively. The SLIC-superpixels manage a higher ASA than the superpixels trained on fine labels on the SUIM dataset. The SLIC-superpixels are appropriate for the SUIM dataset. On the Cityscapes dataset, the ASA of the SLIC-superpixels is not significantly higher than the ASA of HCFCN-16 trained on coarse labels. The discrepancy between the SLIC-superpixel ASA on the Cityscapes and SUIM datasets can explain the discrepancy between the effectiveness of regularization on the Cityscapes and the SUIM dataset. The second explanation is based on the examples in figures 8, 13 and 14. We can conclude that HCFCN-16 is allready reasonably precise without regularization on the Cityscapes dataset. There is simply less room for improvement to boundary alignment when trained on the Cityscapes dataset. This conclusion is supported by the results in tables II and III. Regularizing the superpixels is effective when the model is suffering from poor boundary alignment.

Section I introduced the application of optical quality assurance in the motherboard manufacturing process. The proposed regularization term can be beneficial to this application. With the regularization term a precise segmentation allows for detection of misplaced components. Without regularization the segmentation model has significantly worse boundary alignment. This will lead to an increase in false positive detection of misplaced components.

The impact of regularization varies significantly between the two datasets included in this work. Future works should aim to provide a better understanding of when and why the proposed superpixel regularization is effective. Future work should explore more datasets and apply regularization to

more encoder-decoder models. Future work should provide a set of criteria that identify the context in which the proposed superpixel regularization is effective.

VI. Conclusion

Boundary alignment is relevant to many applications within quality assurance, photo and video editing. In this work we proposed a regularization term that encourages SLIC-superpixels in encoder-decoder models that employ a superpixel based decoder. The proposed regularization term significantly improves the boundary alignment of HCFCN-16. Boundary recall improves up to 15.9% on the Cityscapes dataset and up to 60.3% on the SUIM dataset. When HCFCN-16 does not suffer from poor boundary alignment the impact of regularization in minimal.

References

[1] Radhakrishna Achanta et al. "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2274–2282. DOI: 10.1109/TPAMI.2012.120.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. 2015. DOI: 10.48550/ARXIV.1511.00561. URL: https://arxiv.org/abs/1511.00561.

[3] Jia-Ren Chang and Yong-Sheng Chen. "Pyramid Stereo Matching Network". In: *CoRR* abs/1803.08669 (2018). arXiv: 1803.08669. URL: http://arxiv.org/abs/1803.08669.

[4] Liang-Chieh Chen et al. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *CoRR* abs/1706.05587 (2017). arXiv: 1706.05587. URL: http://arxiv.org/abs/1706.05587.

[5] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[6] Raghudeep Gadde et al. *Superpixel Convolutional Networks using Bilateral Inceptions*. 2015. DOI: 10.48550/ARXIV.1511.06739. URL: https://arxiv.org/abs/1511.06739.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[8] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[9] Shengfeng He et al. "SuperCNN: A Superpixelwise Convolutional Neural Network for Salient Object Detection". In: *International Journal of Computer Vision* 115 (Apr. 2015). DOI: 10.1007/s11263-015-0822-0.

[10] Md Jahidul Islam et al. "Semantic Segmentation of Underwater Imagery: Dataset and Benchmark". In: *CoRR* abs/2004.01241 (2020). arXiv: 2004.01241. URL: https://arxiv.org/abs/2004.01241.

[11] Ning Jia et al. "Coarse Annotation Refinement for Segmentation of Dot-Matrix Batchcodes". In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. 2019, pp. 2001–2007. DOI: 10.1109/ICMLA.2019.00320.

[12] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *CoRR* abs/1612.03144 (2016). arXiv: 1612.03144. URL: http://arxiv.org/abs/1612.03144.

[13] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *ECCV*. Sept. 2014. URL: https://www.microsoft.com/en-us/research/publication/microsoft-coco-common-objects-in-context/.

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2014. DOI: 10.48550/ARXIV.1411.4038. URL: https://arxiv.org/abs/1411.4038.

[15] D.R. Martin, C.C. Fowlkes, and J. Malik. "Learning to detect natural image boundaries using local brightness, color, and texture cues". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.5 (2004), pp. 530–549. DOI: 10.1109/TPAMI.2004.1273918.

[16] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. *Learning Deconvolution Network for Semantic Segmentation*. 2015. DOI: 10.48550/ARXIV.1505.04366. URL: https://arxiv.org/abs/1505.04366.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. DOI: 10.48550/ARXIV.1505.04597. URL: https://arxiv.org/abs/1505.04597.

[18] Bryan C. Russell et al. "LabelMe: A Database and Web-Based Tool for Image Annotation". In: *International Journal of Computer Vision* 77 (2007), pp. 157–173.

[19] Wenjun Wu Shima Nofallah Mojgan Mokhtari. "Segmenting Skin Biopsy Images with Coarse and Sparse Annotations using U-Net". In: *Journal of digital imaging* (2022), pp. 1–12. DOI: 10.1007/s10278-022-00641-8.

[20] David Stutz, Alexander Hermans, and Bastian Leibe. "Superpixels: An Evaluation of the State-of-the-Art". In: *CoRR* abs/1612.01601 (2016). arXiv: 1612.01601. URL: http://arxiv.org/abs/1612.01601.

[21] Teppei Suzuki. "Implicit Integration of Superpixel Segmentation into Fully Convolutional Networks". In: *CoRR* abs/2103.03435 (2021). arXiv: 2103.03435. URL: https://arxiv.org/abs/2103.03435.

[22] Fengting Yang et al. "Superpixel Segmentation With Fully Convolutional Networks". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 13961–13970. DOI: 10.1109/CVPR42600.2020.01398.

[23] Hengshuang Zhao et al. “Pyramid Scene Parsing Network”. In: *CoRR* abs/1612.01105 (2016). arXiv: 1612.01105. URL: http://arxiv.org/abs/1612.01105.

[24] Aleksandar Zlateski et al. “On the Importance of Label Quality for Semantic Segmentation”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1479–1487. DOI: 10.1109/CVPR.2018.00160.

VII. Appendix

A. Impact dataset size

The effectiveness of regularization is expected to correlate inversely with the number of training samples. The Cityscapes dataset has more samples than the SUIM dataset. Furthermore, each sample in the Cityscapes dataset has double the horizontal resolution compared to the SUIM dataset. This could explain the discrepancy in the effectiveness of regularization between the Cityscapes and SUIM datasets. We test this hypothesis by training HCFCN-16 with and without regularization on various subsets of the Cityscapes dataset. In figure 15 we can see the average accuracy for the HCFCN-16 with and without regularization subject to the number of training samples. At 645 samples the Cityscapes subset matches the total number of pixels in the SUIM training set. The impact of regularization does not increase as the number of training samples decreases. The difference in dataset size between the Cityscapes and SUIM datasets fails to explain the discrepancy in effectiveness of regularizing the superpixels.

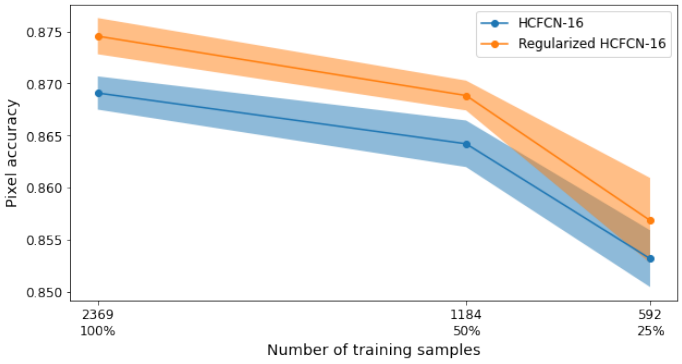


Fig. 15: Average accuracy and standard deviation on the Cityscapes validation set when trained on various subsets of the Cityscapes coarse training set.

B. Varying level of coarseness

On the SUIM dataset, coarse labels are derived from the fine labels. First, each class in the fine labels is eroded by 12 pixels. Then, the Douglas-pecker algorithm is used to approximate the eroded label with a polygon. The resulting coarse labels imitate the coarse labels of the Cityscapes dataset. In this section we investigate the impact of intermediate levels of coarseness. The intermediate levels of coarseness are obtained by varying the number of eroded pixels. In figure 16 we can see the impact on pixel accuracy when trained on increasingly coarse labels. In figure 17 the impact on boundary recall is seen instead.

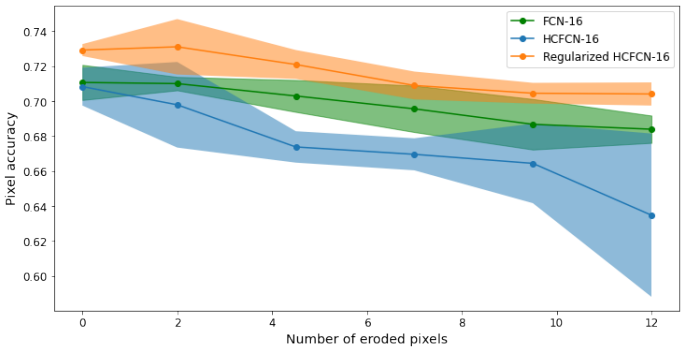


Fig. 16: Pixel accuracy on the SUIM test set when trained on coarse labels with increasing number of eroded pixels. Methods are trained on coarse training labels and fine-tuned on the coarse training and validation sets.

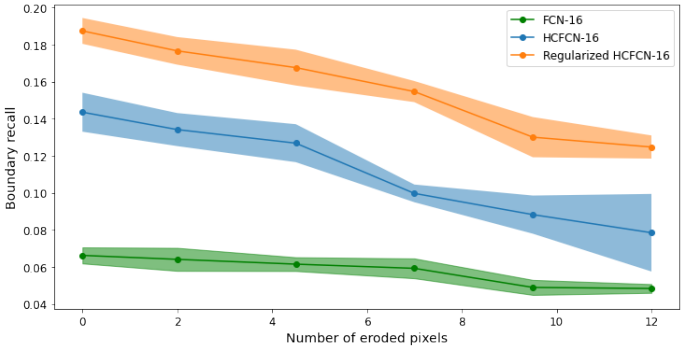


Fig. 17: Boundary recall on the SUIM test set when trained on coarse labels with increasing number of eroded pixels. Methods are trained on coarse training labels and fine-tuned on the coarse training and validation sets.

C. Example of PyTorch implementation of superpixel based downsampling (Eq. 3)

Equation 3 downsamples a full resolution feature map x with the assignment matrices A before upsampling with the same assignment matrices. A pixel in the resulting feature map contains the average features of its corresponding superpixels. This function is essential to the regularization term. Equation 3 consists of two distinct steps. Downsampling a full resolution feature map x with the assignment matrices A. And upsampling the resulting low resolution feature map with the same assignment matrices. In [21] T. Suzuki provides a PyTorch implementation for the latter step. In table VII we provide a PyTorch implementation to downsample a feature map with a given assignment matrix. Downsampling a feature map x with the assignment matrices A is done by recursively downsampling the feature map x with $A^{(s')}$ for $s' \in \{1, 2, 4, 8\}$. The provided downsample function is compatible with the PyTorch code provided in [21].

```python
import torch
import torch.nn.functional as F
eps = 1e-8

def downsample(features, assignment):
    """Downsamples the features with the given assignment matrix"""
    b, nr_feat, h, w = features.shape

    #Unfold the features
    features = F.unfold(features, kernel_size=2, stride=2).reshape(b * nr_feat, 4, int(h/2), int(w/2))
    features = F.unfold(features, kernel_size=3, padding=1).reshape(b, nr_feat, 4, 9, int(h/2), int(w/2))

    #Unfold the assignment
    assignment = F.unfold(assignment, kernel_size=2, stride=2).reshape(b, 36, int(h/2), int(w/2))
    assignment = F.unfold(assignment, kernel_size=3, padding=1)
    assignment = assignment.reshape(b, 9, 4, 9, int(h/2), int(w/2)).permute(0, 1, 3, 2, 4, 5)

    #Flip to take the diagonal from right to left
    assignment = torch.flip(assignment, dims=[1])
    assignment = torch.diagonal(assignment, dim1=1, dim2=2).permute(0, 1, -1, 2, 3)
    assignment = assignment.view(b, 1, 4, 9, int(h/2), int(w/2)).repeat(1, nr_feat, 1, 1, 1, 1)

    #Downsample features
    down_features = torch.sum(features * assignment, dim=(2, 3))
    down_features = torch.div((down_features), (torch.sum(assignment, dim=(2, 3)) + eps))

    return down_features
```

TABLE VII: PyTorch implementation of superpixel based downsampling (Eq. 3)