Fast Semantic Feature Extraction using Superpixels for Soft Segmentation

Shashikant Verma, Rajendra Nagar, and Shanmuganathan Raman

Indian Institute of Technology, Gandhinagar, Gujarat
{shashikant.verma, rajendra.nagar, shanmuga}@iitgn.ac.in

Abstract. In this work, we address the problem of extracting high dimensional, soft semantic feature descriptors for every pixel in an image using a deep learning framework. Existing methods rely on a metric learning objective called multi-class N-pair loss, which requires pairwise comparison of positive examples (same class pixels) to all negative examples (different class pixels). Computing this loss for all possible pixel pairs in an image leads to a high computational bottleneck. We show that this huge computational overhead can be reduced by learning this metric based on superpixels. This also conserves the global semantic context of the image, which is lost in pixel-wise computation because of the sampling to reduce comparisons. We design an end-to-end trainable network with a loss function and give a detailed comparison of two feature extraction methods: pixel-based and superpixel-based. We also investigate hard semantic labeling of these soft semantic feature descriptors.

Keywords: Feature Extraction, Semantic Representation, Image Segmentation, Superpixels.

1 Introduction

Automatic object detection and characterization is a difficult problem. Challenges arise due to the fact that the appearance of the same object can differ heavily in terms of orientation, texture, color, etc. Therefore, it is necessary that feature description for pixels belonging to the same class should be such that it can cope-up with all variations. There has been wide research in this field using traditional approaches. Features such as SIFT [1], SURF [2] are crafted in such a way that they remain scale and orientation invariant. The problem of scale variance of features can be explained by scale-space theory [3]. Though various traditional approaches work well for a small set of images or a particular shape of an object, their efficiency in capturing characteristics for a large number of classes with respective variations has not been explored much. The current state-of-the-art methods rely on deep learning for feature extraction, which outperforms traditional approaches. Deep learning based segmentation methods extract a feature map of the image and then assign a probability measure on it. Hence, we obtain a hard label for every pixel signifying its class. In soft segmentation, a pixel can belong to more than one segments. Therefore, it

represents soft transitions between the boundaries of objects. These soft transitions have a wide range of applications in image matting, deblurring, editing, and compositing [4] [5].

Deep learning techniques use various metric learning methods to establish a relation between the input and the output. Deep networks can learn these complex non-linear metrics by loss functions such as contrastive Loss, triplet Loss, etc. These losses are often used in obtaining discriminatory features maps for applications in image retrieval and face recognition [6]. The loss incurred in these frameworks are computed only considering one negative example, and hence, these methods lead to slow convergence. Addressing this issue authors of [7] proposed multi-class N-pair loss extending triplet loss in which a positive example is compared with all possible negative examples.

The metric learning technique in [7] can be used for extracting distinct semantic features such that objects similar in semantic context tend to have similar features. As there can be a large number of both pixels and class labels in an image, N-pair loss computation can be a computational bottleneck. To deal with this, [4] adopts a sampling approach. They iteratively sample a sizeable number of pixels from a subset of randomly chosen classes and compute pairwise loss on these selections. A major drawback of this approach is that the global context of image diversity is not captured and all negative class examples are not employed for the loss calculation.

To address this problem, we leverage the property of superpixels in representing a set of similar pixels. We employ a modified form of N-pair loss on all superpixels instead of pixels. This also preserves the global context of semantic diversity in an image. In section 3, we discuss the detailed implementation of the layer which determines feature vector for superpixels and propose the architecture of an end-to-end trainable network. The model learns to extract similar features for superpixels if they represent the same semantic class and dissimilar otherwise. In section 4, we discuss about the complexity of our methodology during the forward and backward pass, assess semantic hard labeling of obtained feature descriptors and their application in obtaining soft segmentation using matting method of [4].

2 Related Work

There are many widely used loss functions for deep metric learning, such as Euclidean loss, softmax loss, contrastive loss, N-pair loss, etc. For extracting features, contrastive and triplet losses impose a margin parameter such that features of different classes are distant from each other at least with the margin imposed. Both the loss functions suffer from slow convergence as they employ only one negative example at a time for learning [7]. Moreover, to reduce computational complexity, they require data sampling for positive and negative samples to accelerate training [8] [4]. We use a modified form of N-pair loss as in [7] for our feature extraction, which addresses the problem of these losses while maximizing inter-class separability.

Image segmentation techniques have significantly improved with deep learning methods. Most of these methods incorporate softmax loss for classification of each pixels to their respective context [9][10][11]. Though softmax losses can efficiently predict the probabilities of a feature to be classified in which class, they fail to obtain the largest separable discriminatory feature map for positive and negative examples. Inspired from margin imposition in contrastive and triplet losses, [12] proposes a combination of margin term with softmax losses. Softmax loss alone tends to fail in extracting discriminatory features, specific to problems addressed [13][14] combine both of the losses for learning.

For soft segmentation, per-pixel feature extraction is an important step. [4] adopts discriminatory feature learning metric by imposing loss in hard negative data mining style. They further use the semantic discrimination ability of these features to obtain soft segments using spectral matting technique. Some other soft segmentation methods use matting with the color information of the pixels [15][16].

3 Method

Our method has the following steps: Feature extractor CNN, which extracts the feature map with the weights of kernels learned during training by CNN. We over-segment the input image to obtain superpixels and define a mapping function on the extracted feature map to determine feature descriptor for every superpixel. We employ a modified form of N-pair loss on these superpixels to update the weights of the network. We discuss specific details of architecture and feature extraction, mapping function to obtain feature of superpixels, loss function in 3.1 and 3.2, respectively.

3.1 Model Architecture

We extract features for semantic soft segmentation by a neural network, as shown in Fig. 1. We use cascaded ResNet bottle-neck block [17] as the baseline of the network for feature extraction and downsample the map up to approximate one-third size of initial input. Output feature map at different layers contain different contextual information. Lower level features are object contours and edge aware while higher level features are context aware. In the end, we concatenate all these features as discussed in [18]. Thus, the obtained map has information about semantic context and contour of objects and regions in an image.

As the network grows deeper, convolutional neural networks face the problem of vanishing gradients, and this can be addressed by making skip connections in the architecture. A basic ResNet bottleneck block serves this purpose [17]. We downsize the feature map output of ResNet blocks by the max-pooling operation. In the end, we concatenate feature maps from various layers of the network by bi-linearly upsampling them to match with original image shape. We perform a metric learning operation on these concatenated features using super-pixels instead of the per-pixel approach used by [4]. This increases the computational

performance of the network and makes the model aware of global semantic information in an image. We generate superpixels by simple linear iterative clustering Algorithm [19], which uses LAB features and spatial information per pixel to over segment the image.

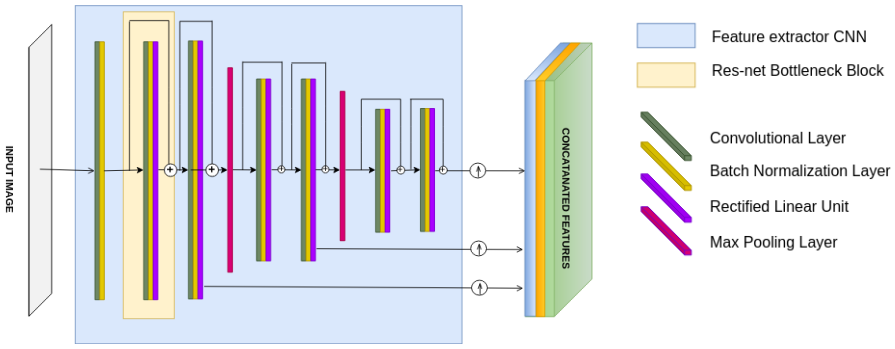


Fig. 1: Architecture of Feature extraction CNN.

Feature descriptor for superpixels. For every superpixel generated by SLIC algorithm, first we need to associate each of them with a ground truth label for supervised learning using CNN. As every superpixel is a collection of pixels which are similar to each other in the local context, we assign ground truth label of a superpixel with the labels of pixels constituting it. Due to inaccuracy along edges and other factors, it may occur that a superpixel may contain differently labeled pixels. In such a case, the label having a majority among all pixels is assigned as ground truth label, refer Equation (1).

For an image $\mathcal{I}$ containing N_p pixels with its ground truth labeled in N_c semantic classes. Let $\mathfrak{P}= \{p_i\}_{i=1}^{N_p}$ be the set of all pixels, $\mathfrak{L} = \{l_i\}_{i=1}^{N_p}$ be their corresponding labels where $l_i \in \{1, 2, \ldots, N_c\}$, and $\mathfrak{S}= \{\mathcal{S}_i\}_{i=1}^{N_s}$ be the set of all N_s superpixels generated by SLIC. We define the sets $\mathcal{P}_i \subset \mathfrak{P}$ and $\mathcal{L}_i \subset \mathfrak{L}$ to be the set of pixels and the set of labels that a superpixel $\mathcal{S}_i$ contains, respectively. Assume $\mathcal{S}_i$ contains total of k pixels such that cardinality $|\mathcal{P}_i| = k = |\mathcal{L}_i|$ and $\mathcal{P}_i = \{p_j\}_{j=1}^{k}$, $\mathcal{L}_i = \{l_j\}_{j=1}^{k}$ with pixel p_j having label l_j. The ground truth label $l_{\mathcal{S}_i}$ for superpixel $\mathcal{S}_i$ is determined by Equation (1).

$$l_{\mathcal{S}_i} = \max(\{C(l_i) \mid C(l_i) = \sum_{j=1}^{k} \delta(l_j, l_i)\}_{i=1}^{k}) \tag{1}$$

$$\mathfrak{l}_{\mathfrak{S}} = [l_{\mathcal{S}_1}, l_{\mathcal{S}_2}, \ldots, l_{\mathcal{S}_{N_s}}]^T \tag{2}$$

where, $C(l_i)$ represents the number of occurrences of l_i labelled pixels in superpixel $\mathcal{S}_i$, $\mathfrak{l}_{\mathfrak{S}}$ is a vector representing labels of all N_s superpixels, and $\delta(i, j)$ is a Kronecker delta function which takes the value 1 if $i = j$ and 0 otherwise.

Let us represent final concatenated features from feature extractor by $\mathfrak{F}$ with $\mathfrak{F} \in \mathbb{R}^{D \times H \times W}$ and input image $\mathcal{I} \in \mathbb{R}^{3 \times H \times W}$, where D,H,W are the dimensions

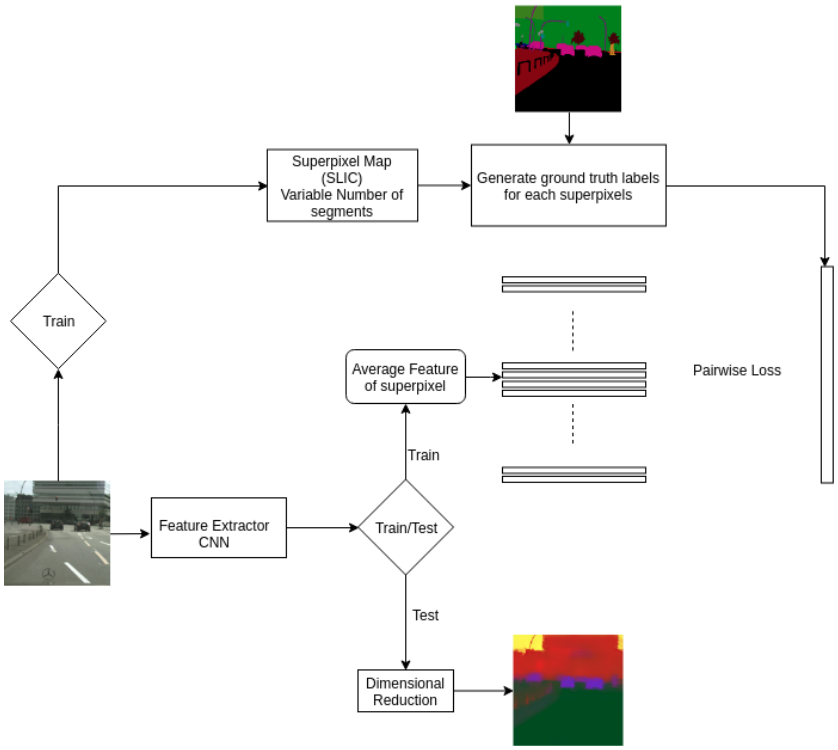


Fig. 2: Flowchart

of the extracted feature map, height and width of the image, respectively. Note that for every N_p pixels, their exist N_p number of D length feature vectors in the feature map $\mathfrak{F}$. For each pixel $p \in \mathfrak{P}$, we represent its feature vector as $\mathcal{F}_p$ such that $\mathcal{F}_p \in \mathfrak{F}$. We find a single dimensional feature vector $\mathcal{F}_{\mathcal{S}_i}$ for superpixel $\mathcal{S}_i \in \mathfrak{S}$ as the average of features of all pixels in $\mathcal{P}_i$ by Equation (3). Note that $\mathcal{P}_i$ is the set of all pixels that are contained in superpixel $\mathcal{S}_i$ with $|\mathcal{P}_i| = k$ and $\mathcal{P}_i \subset \mathfrak{P}$.

$$\mathcal{F}_{\mathcal{S}_i} = \frac{1}{|\mathcal{P}_i|} \sum_j \mathcal{F}_{p_j}, \quad \forall p_j \in \mathcal{P}_i \subset \mathfrak{P} \ , \ i \in \{1, 2, \ldots, N_s\} \tag{3}$$

$$\mathcal{F}_{\mathfrak{S}} = [\mathcal{F}_{\mathcal{S}_1}, \mathcal{F}_{\mathcal{S}_2}, \ldots, \mathcal{F}_{\mathcal{S}_{N_s}}]^T \tag{4}$$

Using Equation (3), we determine the averaged feature for all superpixels and obtain a superpixel feature map $\mathcal{F}_{\mathfrak{S}}$ with $\mathcal{F}_{\mathfrak{S}} \in \mathbb{R}^{N_s \times D}$ and corresponding ground truth label $\mathfrak{l}_{\mathfrak{S}}$ with $\mathfrak{l}_{\mathfrak{S}} \in \mathbb{R}^{N_s}$ from Equations (4) and (2), respectively.

3.2 Loss function

For learning, we use modified form of the N-pair loss [7] where we use L2 distance between superpixel feature vectors instead of inner product in similar way as in [4]. Consider two feature vectors of superpixels $\mathcal{S}_p$ and $\mathcal{S}_q$ represented by $\mathcal{F}_{\mathcal{S}_p}, \mathcal{F}_{\mathcal{S}_q} \in \mathfrak{F}_{\mathfrak{S}}$ with labels $l_{\mathcal{S}_p}, l_{\mathcal{S}_q} \in \mathfrak{l}_{\mathfrak{S}}$ respectively, where $p, q \in \{1, 2, \ldots, N_s\}$. We define loss function such that $\mathcal{F}_{\mathcal{S}_p}$ and $\mathcal{F}_{\mathcal{S}_q}$ are similar to each other if $l_{\mathcal{S}_p} = l_{\mathcal{S}_q}$ and dissimilar otherwise by Equation (5).

$$\begin{aligned} L_{pq} = \frac{1}{|\mathfrak{S}|} \sum_{p,q \in |\mathfrak{S}|} \mathbb{I}[l_{\mathcal{S}_p} = l_{\mathcal{S}_q}] \log\Big(\Big(1 + \exp(\, \big\|\mathcal{F}_{\mathcal{S}_p} - \mathcal{F}_{\mathcal{S}_q}\big\| \,)\Big)/2\Big) \\ + \sum_{p,q \in |\mathfrak{S}|} \mathbb{I}[l_{\mathcal{S}_p} \neq l_{\mathcal{S}_q}] \log\Big(1 + \exp(\, - \big\|\mathcal{F}_{\mathcal{S}_p} - \mathcal{F}_{\mathcal{S}_q}\big\| \,)/2\Big) \end{aligned} \tag{5}$$

where $|\mathfrak{S}| = N_s =$ total number of superpixels generated using SLIC. $\mathbb{I}[.]$ is an indicator function which is 1 if statement holds true and is 0 otherwise. $\|.\|$ represents L2 distance between the feature vectors. Note that if $l_{\mathcal{S}_p} = l_{\mathcal{S}_q}$ and both superpixels have dissimilar features then $log\Big(\Big(1 + exp(\, \big\|\mathcal{F}_{\mathcal{S}_p} - \mathcal{F}_{\mathcal{S}_q}\big\| \,)\Big)/2\Big)$ term in Equation (5) evaluates to a larger value and contributes to total loss which needs to be minimized by back-propagation through the network. Moreover, if features of both the superpixels are similar then the value of this term approaches zero and no loss is induced. Fig. 2 shows the flowchart of adopted methodology. Note that superpixel supervision is only needed for loss calculation and is no more needed once the training is over. Since from ground truth information, we only use the fact that whether two features under consideration belong to the same category or not, learning is class agnostic. In Section 4, we assess semantic hard labelling of the learned feature descriptors.

Feature extraction and selection We fix the size of the input to CNN by resizing the images to 224×224. Then, we obtain a 128-dimensional feature map $\mathfrak{F}$. For loss computation, we over-segment the input image into $N_s = 500$ superpixels and compute N-pair loss as defined in Equation (5). Using Equations (3) and (4), we obtain 128 dimensional feature vector $\mathcal{F}_{\mathfrak{S}} \in \mathbb{R}^{128 \times 500}$ for every superpixel. This feature map for every super pixel has enough capacity to capture diverse contextual data of objects. We perform guided filtering [20] on the obtained feature map with the guidance of input image. This lets features to adhere more towards boundary and contours present in an image. To reduce dimensionality and select the dominant feature from $\mathfrak{F}$, we use principal component analysis (PCA)[21] and generate a 3D feature map corresponding to the three largest eigenvalues. We show a comparison of both selected dominant features and few random 3D projections of filtered 128-D map $\mathfrak{F}$ in Fig. 3 and 4, respectively. To show feature descriptors of superpixels, we define a mapping on $\mathcal{F}_{\mathfrak{S}}$ as $\mathcal{M} : \mathcal{F}_{\mathfrak{S}} \rightarrow \widetilde{\mathfrak{F}}$, where $\widetilde{\mathfrak{F}} \in \mathbb{R}^{128 \times 224 \times 224}$ as in Equations (6) and (7).

$$\mathcal{M}(\mathcal{F}_{\mathcal{S}_i}) = \mathcal{F}_{\mathcal{S}_i} \tag{6}$$

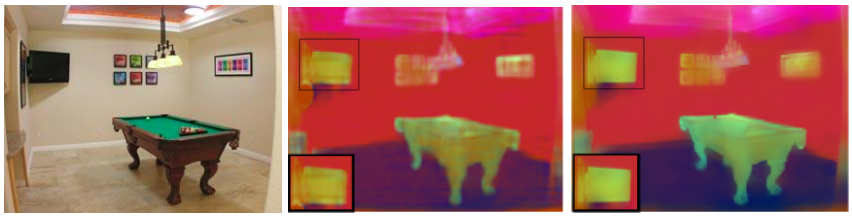
$$\widetilde{\mathcal{F}_p} = \mathcal{M}(\mathcal{F}_{\mathcal{S}_i}) \;\mid\; \forall p \in \mathcal{P}_i, \;\; i \in \{1, 2, \ldots, N_s\} \tag{7}$$

where $\widetilde{\mathcal{F}_p} \in \widetilde{\mathfrak{F}}$ is the mapped 128 dimensional feature vector for pixel location p. Note that $\mathcal{P}_i$ is the set of all pixels that the superpixel $\mathcal{S}_i$ contains. We show a comparison of mapped superpixel features $\widetilde{\mathfrak{F}}$ vs feature map $\mathfrak{F}$ in Fig. 5. Observe that feature map $\mathfrak{F}$ is smoother and continuous than $\widetilde{\mathfrak{F}}$ in Fig. 5. Hence, we use $\mathfrak{F}$ for feature selection using PCA and further processing. This also reduces the complexity of CNN once training is over as superpixel supervision is no more needed. In Fig. 2, we show a flowchart of approach during training and testing of network.

4 Experimental Analysis

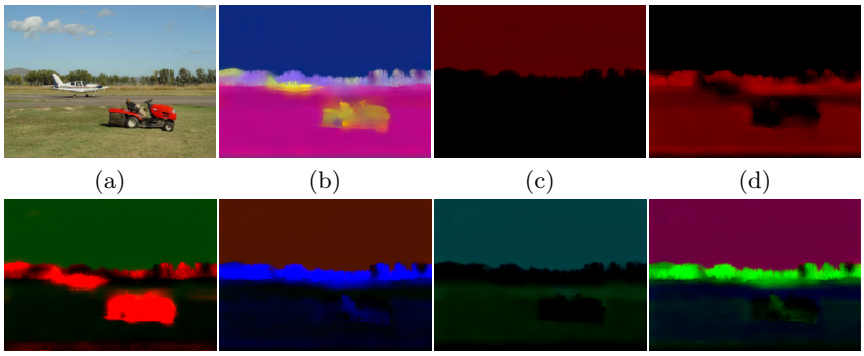
We trained our network on ADE20k dataset [22], which contains 150 labeled semantic classes. We used basic bottleneck block from ResNet as a building block of our feature extractor CNN. Weights of the network were initialized by Xavier initialization[24]. We start with a learning rate of 1×10^{-3} and use stochastic gradient descent optimizer with the momentum of 0.9, weight decay of 5×10^{-4} and poly learning rate of 0.9 as suggested in [25]. We generated $N_s = 500$ number of superpixels for input image by SLIC algorithm [19] to estimate loss during training session. Note that few pixels may remain disjoint after over segmentation using SLIC. To ensure connectivity, post-processing of superpixel is done so that every disjoint pixel is assigned to a nearby superpixel. Due to this operation, we may obtain a lesser amount of superpixels than N_s. We populate $\mathcal{F}_{\mathfrak{S}}$ and $\mathfrak{l}_{\mathfrak{S}}$ with redundant data to match the dimensions of incoming batches and compute loss using equation 5. We train for 60K iterations with a batch size of four over ADE20k training split which has 20210 images. It takes about 12 hours on NVIDIA Titan Xp 12GB GPU. We show the semantic soft segmentation by matting method proposed in [4] on our features and comparison in Fig. 6.

Loss computational complexity. It is easy to notice that over segmenting an image into superpixels reduces the number of data points from total pixels in image $|\mathfrak{P}| = N_p$ to $|\mathfrak{S}| = N_s$. As mentioned by authors of SLIC algorithm [19], the complexity to compute superpixels is $\mathcal{O}(N_p)$ and for computing L2 distance between feature descriptors for every possible pairs of superpixels, complexity is of order $\mathcal{O}(N_s^2)$. Thus, our method achieves total complexity of $\mathcal{O}(N_p) + \mathcal{O}(N_s^2)$. In our experiment we use $N_s = 500$ and image of size 224×224 having total pixel count $N_p = 50176$. We compare our method of loss estimation with [4], which employ sampling based approach to reduce complexity. They randomly select N_{inst} number of instances out of N_c labelled classes from image, sample N_{samp} amount of pixels from every selected instance and compute L2 loss between each possible pixel pairs. They repeat this process N_{iter} times. Let us define the total number of feature descriptors sampled as $N_{total} = N_{iter} \times N_{inst} \times N_{samp}$. The computational complexity for this method becomes $\mathcal{O}(N_{total}^2)$. Numerical values reported in [4] are as $N_{iter} = 10$, $N_{inst} = 3$ and $N_{samp} = 1000$ having complexity of $\mathcal{O}(30000^2)$ compared to $\mathcal{O}(500^2) + \mathcal{O}(50176)$ by our method. Note that in our



(a) (b) (c)

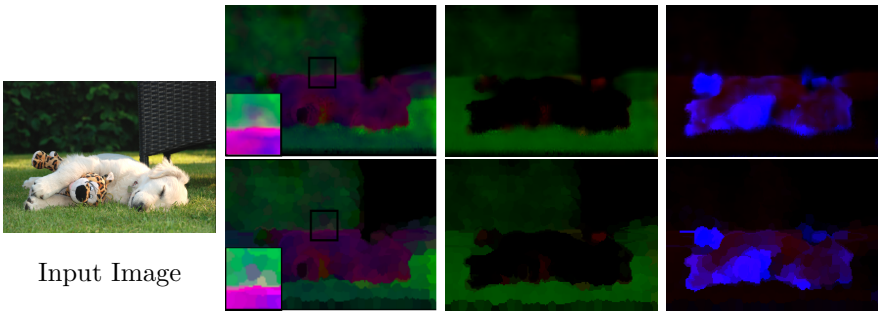
Fig. 3: For an input image (a), we show effect on features obtained after PCA with and without guided filtering in (b),(c) respectively. Notice the adherence of features towards edges in (c) due to guided filtering operation. Image taken from ADE20k dataset [22]



(a) (b) (c) (d)

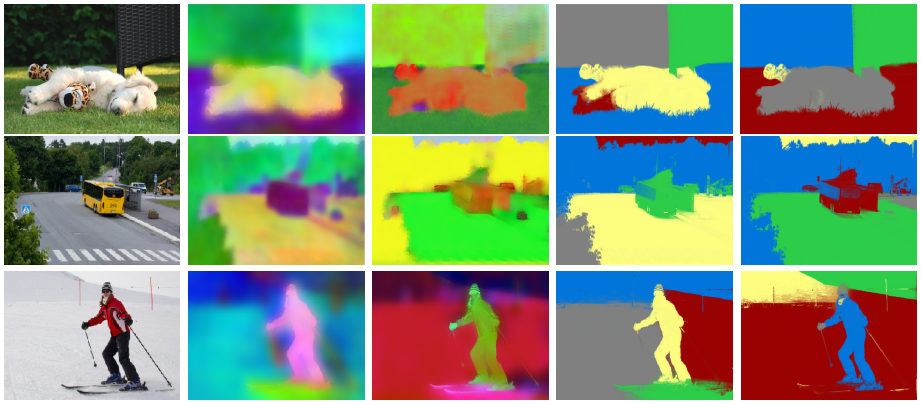
(e) (f) (g) (h)

Fig. 4: For an input image in (a) we extract 128-dimensional feature map $\mathfrak{F}$ and show randomly sampled 3d maps from it in (c,d,e,f,g,h). (b) shows 3-dimensional feature selection on $\mathfrak{F}$ using PCA. Notice the semantic context embedded in different channels of map. Image taken from ADE20k dataset [22]



Input Image

Fig. 5: For an input image, Row 1 shows randomly sampled 3D maps from $\mathfrak{F}$ and Row 2 shows superpixel feature descriptors by mapping $\mathcal{M}$ defined in 6. Observe the smoothness of $\mathfrak{F}$ compared to $\widetilde{\mathfrak{F}}$. Zoom in caption is enhanced for better visualization. Image taken from coco stuff dataset [23]



(a) (b) (c) (d) (e)

Fig. 6: (a) Three different input images from coco stuff dataset [23] (b) Features extracted in [4] (c) Features extracted using our method (d) Semantic soft segmentation results reported in [4] (e) Semantic soft segmentation results produced by our features using matting method of [4].

method, superpixel computation does not cause any back-propagation overhead. During training time, gradients accumulation in the graph is only due to N_s^2 data points while the method in [4] needs an accumulation of gradients for all N_{total} points. In Table 1, we summarize the comparison of both the methods.

Method	Iterations	Instances	Points	Complexity	Parameters
Our method	NA	NA	N_s	$\mathcal{O}(N_s^2) + \mathcal{O}(N_p)$	3.7 M
Aksoy et al[4]	N_{iter}	N_{inst}	N_{total}	$\mathcal{O}(N_{total}^2)$	68.6 M

Table 1: Computational complexity of two methods. Note that in our method $\mathcal{O}(N_p)$ don't add any overhead to gradient computations during backpropagation

Semantic hard segmentation. To assess the hard labeling of obtained feature descriptors, we add layers to reduce the dimension of feature map to a total number of labeled classes N_c in ground truth for evaluating cross entropy loss. We employ architecture shown in Fig. 7 on CityScape dataset [26] which has a total of $N_c = 19$ labelled classes. We use feature extractor CNN in the test mode to obtain soft semantic feature map of 128 dimensions. We train the model over training split of the dataset and form a confusion matrix for every pixel with its predicted label and ground truth label to compute mean Intersection over Union(IoU), pixel accuracy, class accuracy and frequency weighted IoU. We report standard metrics of semantic segmentation and some results on validation split in table 2 and Fig. 8, respectively. Note that, we use lower level features

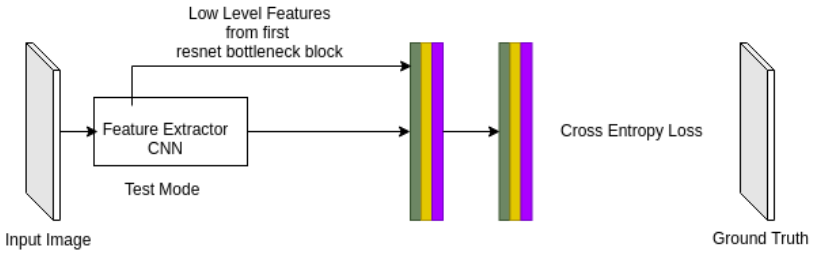


Fig. 7: Semantic hard labelling of soft feature descriptors by employing cross entropy loss. Note that we use Feature extractor CNN in test mode and only learn weights for added convolution and batch-normalization layers. Color legend for layers is same as in Fig. 1.

from first ResNet block inspired from [18] to acquire contours and edges of slim objects like poles, traffic lights which seem to be lost due to in-efficiency of SLIC. From Table 2, we observe that soft features extracted posses enough diversity to be used for semantic segmentation purposes by integrating it with various state-of-the-art methods. We compare the results with benchmarks on CityScapes dataset in Table 3.

Pixel Accuracy	88.12%
Class Accuracy	58.65%
mIoU	40.37%
fwIoU	81.66%

Table 2

Method	mIoU
SegNet basic [11]	57.0%
FCN-8s [10]	65.3%
DeepLab [25]	63.1%
DeepLab-CRF [25]	70.4%
Proposed Approach	**40.37 %**

Table 3

Table 2 shows the metric evaluation on CityScapes[26] dataset, where mIoU, fwIoU refers to mean and frequency weighted intersection Over union. Table 3 compares the results with state of the art methods on this dataset.

5 Conclusion

We have proposed a deep metric learning method to extract a feature descriptor per pixel in an image. We have designed a layer which can be back-propagated to determine features for SLIC superpixels and have proposed an end-to-end trainable model architecture. We have employed the multi-class N-pair loss on superpixels instead of pixels, thus reducing the complexity of loss computation and backpropagation overhead. We have shown that the feature

map learned has diverse semantic context of an input image and can be used for various applications like semantic soft and hard segmentation.

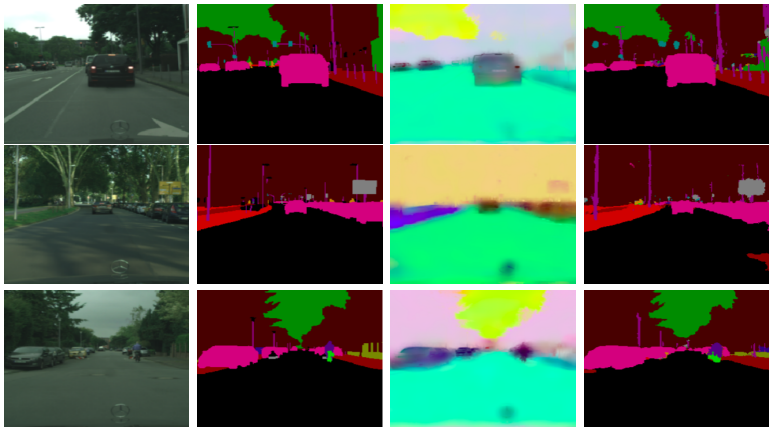


Fig. 8: Semantic segmentation by employing cross entropy loss on obtained 128D feature map.Soft features shown are selected 3D features using PCA.

References

1. M. Brown and D. G. Lowe, "Invariant features from interest point groups." in *BMVC*, vol. 4, 2002.
2. H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.
3. T. Lindeberg, *Scale-space theory in computer vision*. Springer Science & Business Media, 2013, vol. 256.
4. Y. Aksoy, T.-H. Oh, S. Paris, M. Pollefeys, and W. Matusik, "Semantic soft segmentation," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 72, 2018.
5. J. Pan, Z. Hu, Z. Su, H.-Y. Lee, and M.-H. Yang, "Soft-segmentation guided object motion deblurring," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 459–468.
6. S. Chopra, R. Hadsell, Y. LeCun *et al.*, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR (1)*, 2005, pp. 539–546.
7. K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in Neural Information Processing Systems*, 2016, pp. 1857–1865.
8. F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.
9. H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

10. J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.
11. V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
12. W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks." in *ICML*, vol. 2, no. 3, 2016, p. 7.
13. Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in neural information processing systems*, 2014, pp. 1988–1996.
14. X. Zhang, F. Zhou, Y. Lin, and S. Zhang, "Embedding label structures for fine-grained feature representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1114–1123.
15. Y. Aksoy, T. Ozan Aydin, and M. Pollefeys, "Designing effective inter-pixel information flow for natural image matting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 29–37.
16. D. Singaraju and R. Vidal, "Estimation of alpha mattes for multiple image layers," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 7, pp. 1295–1309, 2010.
17. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
18. G. Bertasius, J. Shi, and L. Torresani, "High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 504–512.
19. R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.
20. K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 6, pp. 1397–1409, 2012.
21. P. Bickel, P. Diggle, S. Fienberg, U. Gather, I. Olkin, and S. Zeger, "Springer series in statistics," 2009.
22. B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
23. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.
24. K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
25. L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.
26. M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.