

**SPARSE SHAPE ENCODING
FOR IMPROVED INSTANCE SEGMENTATION**

KEYI LIU

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

MARCH, 2023

©KEYI LIU, 2023

ABSTRACT

Neurophysiological studies suggest that neurons in the intermediate visual area V4 of the primate cortex encode a sparse representation of object shape. While there are metabolic arguments for such sparse representations, there are also potential advantages for inference. Here we explore whether sparse shape encoding can yield benefits for instance segmentation. Specifically, we encode 2D object shape using a Distance Transform Map (DTM) and learn a sparse basis for this representation. To make use of this encoding, we design an instance segmentation head to estimate the sparse coefficients of each object, and then recover the shape from the zero-crossing level set of the corresponding DTM. Our novel **SparseShape** encoding approach produces fewer topological errors than the state of the art, yields competitive mask AP on the MS COCO benchmark, and exhibits superior generalization performance on the Cityscapes traffic instance segmentation task.

ACKNOWLEDGEMENTS

I would like to thank the following people, without whom I would not have been able to complete this research, and without whom I would not have made it through my master's degree.

The Elder Lab at York University, especially to my supervisor Dr. James Elder, whose insight and knowledge into the subject matter steered me through this research. My supervisory committee member Dr. Marcus Brubaker, who took the time to review my research proposal and thesis. Special thanks to Dr. Bob Hou, whose technical support allowed my studies to go smoothly when the campus was shut down due to the COVID-19. My colleagues at the Elder Lab, who have supported me and had to put up with my stresses and moans for the past two years of study. Dr. Steven Waslander and his team at University of Toronto for all these excellent discussions.

This research was supported by the ORF-RE project, Intelligent Systems for Sustainable Urban Mobility (ISSUM), the Vision: Science to Applications (VISTA) program, and by a sub-contract from the University of Toronto and Huawei Technologies Canada.

Last but not least, my biggest thanks to my family for all the support you have shown me through this research, the culmination of two years of distance learning. For my wife Xuyao Lang, thanks for all your support, without which I would have stopped these studies due to the pandemic. You have been amazing, and I will now clear all the papers off the kitchen table as I promised.

Table of Contents

ABSTRACT	ii
ACKNOWLEDGEMENTS	iii
Table of Contents	iv
List of Figures	vii
List of Tables	ix
1 Introduction	1
2 Related Work	6
2.1 One-Stage and Two-Stage Methods	6
2.1.1 Two-stage Instance Segmentation	6
2.1.2 One-stage instance segmentation	7
2.2 Shape Representations	8
2.2.1 Region-Based Instance Segmentation	8
2.2.2 Contour-Based Instance Segmentation	9
2.2.3 Coefficient-Based Instance Segmentation	10
2.3 The Distance Transform	11
2.4 Relationship to Our Approach	12
3 Methodology	14
3.1 Distance Transform Representation	14
3.2 Sparse Coding	14

3.3	Flowchart of the Proposed Framework	15
3.4	Network Architecture	19
3.4.1	One-stage Object Detector	19
3.4.2	Instance Segmentation	20
3.5	Training	23
3.6	Inference	25
4	Experiments	26
4.1	Instance Segmentation Datasets	26
4.2	Evaluation Metric	27
4.3	Implementation Details	29
4.3.1	Sparse Coding Dictionary Learning	29
4.3.2	Network Training and Inference	30
4.4	System Optimizations	32
4.4.1	DTM Resolution	32
4.4.2	Sparse Coding Parameters	33
4.4.3	Loss Function for Sparse Coefficient Estimation	35
4.4.4	Shape Representation and Encoding Method	35
4.4.5	Feature Fusion and Mask Refinement	36
4.4.6	Mask Quality Modulation	38
4.4.7	Sparsity of the Estimated DTM Coefficients	39
4.5	Comparison to State-of-the-Art	40
4.5.1	Mask AP	40
4.5.2	Topological Accuracy	42
4.5.3	Generalization	45

4.5.4	Computational Time	46
4.6	Qualitative Results	46
4.6.1	Topological Improvements	46
4.6.2	Additional Challenging Examples from MS COCO . .	47
4.6.3	Examples with Improved Segmentation	49
4.6.4	Generalization to Cityscapes	51
5	Analysis of Limitations	55
5.1	Limitation of Dictionary Adaptation	55
5.2	Limitation of SparseShape	57
5.2.1	Detection Error	57
5.2.2	Over-Smoothing	60
5.2.3	Instances with Thin and Elongated Parts	66
5.2.4	Incorrect Separation between Objects	66
6	Conclusion	68
7	Future Work	71
7.1	Leverage High Resolution Semantic Features	71
7.2	Amodal Completion	72

List of Figures

1.1	Common Shape Representations	2
3.1	Distance Transform and Sparse Coding	16
3.2	Proposed Stage 1: Dictionary Learning	17
3.3	Proposed Stage 2: Instance Segmentation Network Training.	18
3.4	Proposed Stage 3: Network Inference	18
3.5	General One-stage Instance Segmentation Network Architec- ture.	20
3.6	Proposed Instance Segmentation Head	21
4.1	Sparse Coding Experiments	34
4.2	Qualitative Comparison on Diverse Topologies	48
4.3	Qualitative Comparison on Challenging Cases	50
4.4	Additional Qualitative Results	51
4.5	Challenging Instances for CondInst	52
4.6	Challenging Instances for BlendMask	53
4.7	Qualitative Results on Cityscapes	54
5.1	Sparse Shape DTM Dictionaries Learned from MS COCO and Cityscapes	55
5.2	Example of basis functions	56
5.3	Hard Cases for SparseShape	58
5.4	Limitations: Detection Error	59
5.5	Gaussian Noise Perturbation	62
5.6	Histogram Equalization Examples	63
5.7	Standard and Log DTMs	64

5.8	Mask Refinement Module Effectiveness	65
5.9	Limitation: Thin and Elongated Parts	67
7.1	Leverage Higher Resolution Semantic Features for Mask Re- finement	71

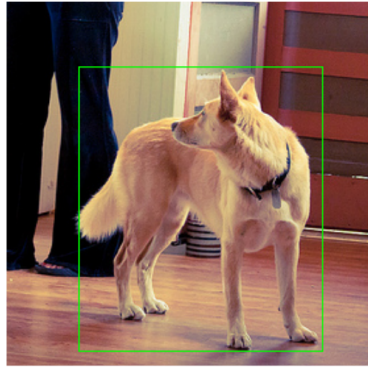
List of Tables

4.1	Dataset Size for Dictionary Learning	30
4.2	Grid Search on the Dimension of DTM	33
4.3	Ablation on Sparse Coefficient Loss Functions.	35
4.4	Mask vs DTM and PCA vs Sparse Coding	36
4.5	Feature Fusion Improves Classification	37
4.6	Effectiveness of Feature Fusion and Mask Refinement.	38
4.7	Mask Quality Estimation	38
4.8	Sparsity Analysis on Estimated Coefficients	39
4.9	Comparison to state-of-the-art methods using a ResNet-50- FPN backbone	41
4.10	Comparison to the state-of-the-art: R101	42
4.11	Topological MAE on MS COCO	44
4.12	Topological distribution of ground truth and estimated masks on the COCO [28] validation set. KL is the KL-divergence from ground truth.	45
4.13	Performance on Cityscapes	46
4.14	Computational Time	47
5.1	Dependency on Dictionary Employed	57

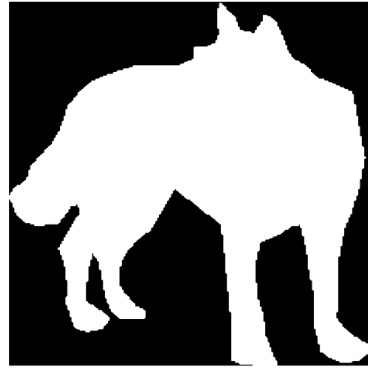
1 Introduction

Convolutional neural networks have achieved impressive human-level performance on many tasks through their ability to gather low-level features from images and build increasingly more global and complex conjunctions of these features in higher layers [22]. In primate visual cortex, an analogous increase in receptive field complexity is observed in intermediate and higher areas, one purpose of which is to compute representations of object shape, known to be crucial for object recognition and other important visual tasks [35, 9]. While some early studies [23, 41] have argued that shape processing in human cortex and deep networks is similar, more recent work has suggested that networks trained only on classification tasks rely less on global shape and more on texture and local shape features compared to humans [14, 2, 13]. This suggests that more explicit shape training might be needed to produce networks that are more sensitive to global shape, like the human brain.

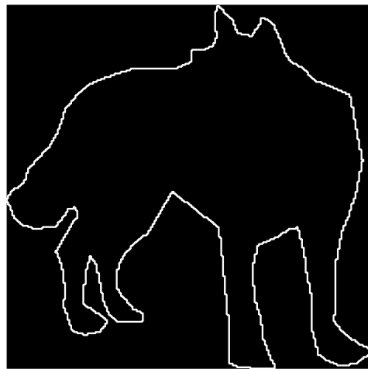
Instance segmentation can be treated as a local pixel-level task (“*Is this pixel inside or outside the 2D boundary of the object*”), without any explicit attempt to represent its global shape (Fig. 1.1(b)). This *mask*-based approach is attractive in its simplicity and aligns well with the pixel-based intersection over union (IoU) metric typically used to evaluate instance segmentation tasks. Note, however, that such binary pixel representations are highly redundant and carry shape information only implicitly: Knowing whether a single pixel is a 1 or a 0 says very little about the global shape of the object.



(a) A dog



(b) Mask



(c) Contour



(d) Distance transform map

Figure 1.1: Common shape representations for instance segmentation. (a) An object with its bounding box; (b) Resized mask representation within the bounding box; (c) A bounding contour derived from the mask in (b); (d) A distance transform map (DTM) representation derived from the mask in (b).

An alternative is to represent a mask region by its bounding contour (Fig. 1.1(c)). This representation is potentially more efficient as it compresses the 2D region into a 1D contour. It can also be argued that shape information (e.g., curvature) is more directly computable from this contour representation. Unfortunately, the region-contour duality breaks down for higher-order shape topologies, e.g. segmentation with multiple disconnected components, or regions with holes. Prior contour-based approaches have avoided this problem by limiting the family of allowable shapes: PolarMask [47] assumes shapes are functions of polar angle without holes, ESE-Seg [48] develops a way to fill holes and connect disconnected regions with morphological operations. Alternatively, additional post-hoc computations can be used to fix topological errors: DeepSnake [36] uses an extra part detector to infer disconnected shape components first and perform instance segmentation on each detected component.

We address these challenges by explicitly modeling shape in a novel representational framework that can easily handle complex topologies. First, we propose to replace the binary pixel representation employed by standard region (mask)-based approaches with a distance transform map (DTM) representation of shape. Like the binary mask representation, this region-based representation allows for the representation of complex topologies as the union of zero-crossing level sets, and also provides richer shape information at the pixel level: Since each pixel value encodes the distance to the nearest point on the boundary, it quantifies information about the local degree of “insideness” (Fig. 1.1(d)).

As for the standard binary representation, the DTM representation is

highly redundant (a DTM has the same dimensionality as a mask). Inspired by neurophysiological evidence that the primate brain uses a sparse coding strategy to encode shape more efficiently [6], we learn a sparse DTM dictionary that allows a shape to be represented as a linear combination of a relatively small number of basis functions, and show that a deep network head can be trained to infer the coefficients of this representation. Once these coefficients are estimated, the object segmentation can be computed from the zero-crossing level set of the reconstructed DTM.

Our sparse DTM representation can be easily incorporated into typical object detection frameworks. To demonstrate the idea, we adopt the one-stage FCOS detector [43] which has been the detector of choice for most recent instance segmentation systems [51, 42, 25, 7, 4].

In summary, the major contributions of this thesis are:

1. A distance transform encoding that improves neural network ability to estimate shape by encoding richer shape information at the local pixel level. To the best of our knowledge, this is the first work that incorporates the distance transform, as a direct and explicit shape representation, into a one-stage instance segmentation framework.
2. A novel sparse coding method that encodes each shape as a linear combination of basis shapes, thus reducing redundancy.
3. A segmentation head employing a novel feature fusion strategy to estimate the sparse coefficients of this representation.
4. A novel shape reconstruction pipeline that uses these coefficients to

estimate the distance transform as a weighted combination of sparse basis functions, identifies the shape as the union of the zero-crossing level set of this distance transform, and then refines the result using a light-weight residual network (Figure 3.6(b)).

5. A mask quality score is adopted in the network to not only promote training on hard segmentation examples, but also calibrate detection confidence scores during inference (Figure 3.6(c)).
6. A demonstration that this novel sparse shape method leads to substantial improvements in topological accuracy while maintaining competitive mask average precision (AP) over baselines on the MS COCO benchmark [28] and better generalization performance on the Cityscapes benchmark [10]. These results suggest that the sparse shape coding strategy observed in primate cortex [6] can be useful for instance segmentation systems.
7. Limitations of the proposed method are identified, and future work has been suggested.

2 Related Work

2.1 One-Stage and Two-Stage Methods

Given its crucial role for image understanding, a huge amount of effort has been made to develop robust and accurate instance segmentation systems. Related closely to the object detection systems, instance segmentation can be divided into two-stage and one-stage methods, depending upon whether the detection and segmentation happen in a cascaded manner (known as the two-stage method) or in parallel (known as the one-stage method).

2.1.1 Two-stage Instance Segmentation

Mask RCNN [17] is one of the pioneering two-stage systems built upon the two-stage object detection network Faster RCNN [38]. By carefully aligning instance features in deep networks using the ROIAlign, Mask RCNN is able to propose instance locations and estimate segmentation masks within the proposed bounding boxes. Various improvements have raised the performance of two-stage methods. For example, Mask Scoring RCNN [18] proposes to estimate a mask quality score together with each instance mask. The quality score reflects the Intersection over Union (IoU) between a predicted mask and its ground truth mask, which can better penalize predictions with high classification confidence score but poor mask quality. PANet [30] boosts the feature extraction ability of its feature pyramid network by adding efficient shortcut connections between the lower and the top

pyramidal features. Features in higher levels are enriched because the path for propagating accurate localization signals between lower levels and higher levels are shortened.

2.1.2 One-stage instance segmentation

To gain real-time performance in real-world applications, instance proposal generation and segmentation estimation have been fused into a single stage, a.k.a. one-stage system, maintaining reasonable accuracy while reducing inference time. One-stage systems are inspired by the success of semantic segmentation, and are conceptually simpler and faster than most two-stage methods.

Kirillov et al [21] segment objects from an instance-agnostic feature map by learning instance-specific boundaries. The Sequential Grouping Network [29] is able to estimate points located on the boundaries of objects and gradually group them into complete segments. These methods are composed of a deep network and a post-processing module, but the post-processing steps are usually computationally intensive. Subsequent methods shift these computations from a learning-free post-processing module to a network that can be jointly optimized [11, 15, 8]. More recent work has produced more compact systems by taking advantage of the structural advances in one-stage object detection frameworks [54, 45, 44, 51, 42, 7, 47]. Recent advances in representative one-stage systems will be discussed in detail in the following sections.

2.2 Shape Representations

2.2.1 Region-Based Instance Segmentation

Mask region estimation is most commonly posed as a direct pixel-level classification problem [8, 18, 11], and this approach can be elaborated to better handle variations in object scale and overlap or near-overlap of object instances. CenterMask [46], for example, separately estimates mask shape and scale, and employs a parallel saliency branch to estimate a global saliency map. The final mask of an instance is jointly determined by its mask shape and a crop from the global saliency map to facilitate the separation of neighbouring objects. CondInst [42] produces instance-aware convolution kernels to find instance segmentation masks also with the aid of a similar global semantic segmentation map. Specifically, a controller network is proposed to dynamically estimate convolutional kernel weights according to the bounding box features assigned to the current instance. Kernels populated by these dynamically estimated weights then convolve with a collaterally estimated semantic map to generate final instance masks.

Beyond direct region mask estimation, a number of approaches attempt to decompose mask regions as linear combinations of prototypes or components, and estimate the coefficients of this linear combination. YOLACT [4], for example, employs a “Protonet” to estimate k region masks and a parallel prediction head to estimate coefficients on these prototypes for the shape of objects. Note, however, that each output channel of the “Protonet” is not really coding instance shape but rather a complete segmentation for the entire image (joint shape of all detected objects in all categories). BlendMask [7]

adopts a similar strategy (adapt from Protonet) to YOLACT, but replaces the coefficients with attention maps that are better able to capture finer details. These attention maps have lower resolution than the basis masks produced by the Protonet but are sufficient to capture coarse instance shape and pose. ShapeMask [24] reconstructs a coarse shape from bounding box features by estimating a weighted combination of a set of shape priors (acquired offline via clustering method). A finer mask is then estimated via two refinement steps that use feature embeddings extracted from the coarse shape.

Perhaps most similar to our approach is MEInst [51], which is based on a pre-learned principal component (PCA) subspace encoding of individual object region masks. The segmentation head of the network then estimates the coefficients of this PCA shape encoding. If dimensionality of the subspace is constrained, this PCA approach could help to address the redundancy of region masks. Note, however, that while PCA will minimize the squared error on the training dataset, it will not necessarily minimize error on unseen data, due to overfitting. Further, if the distribution of shapes is not joint Gaussian, the components will not necessarily be statistically independent, and this could make it harder for the network to estimate coefficients from pixel data.

2.2.2 Contour-Based Instance Segmentation

Contour-based methods tackle the instance segmentation task by either directly estimating the coordinates of the vertices on the boundary or estimating a set of coefficients to encode the contour. PolarMask [47] and

ESE-Seg [48] both constrain shapes to be functions of polar angle. While PolarMask directly estimates the length of rays from the polar center, ESE-Seg employs a lower-dimensional Chebyshev polynomial basis and trains a network head to estimate the coefficients for each detected object instance.

The polar shape constraint is unfortunately not satisfied for many shapes of interest (e.g., human bodies). DeepSnake [36] relaxes the polar assumption and estimates shapes by recurrent deformation of boundary vertices, disconnected components are handled independently with the aid of a part detector. Dense RepPoints [50] uses a distance transform to sample a dense set of points on either side of the object boundary to form a regression target, while the loss is determined by the Chamfer distance between ground truth and estimated point sets. While enjoying some compression by targeting the boundary rather than the region mask, DeepSnake and Dense RepPoints do not apply any subspace encoding method on the boundary and are thus still fairly high-dimensional.

2.2.3 Coefficient-Based Instance Segmentation

Coefficient-based approaches often work with the mask region or contour shape representation to segment instances. These approaches compress the representation in the original shape space to some lower dimensional encoding spaces, and estimate a set of coefficients in the compressed space. Some methods are more explicit and treat the encoding coefficients as the regression learning target. For example, ESE-Seg [48] encodes shape contours into a lower-dimensional Chebyshev polynomial space and estimates these polynomial coefficients directly. MEInst [51] encodes each mask region into

a PCA subspace and directly estimates these PCA coefficients.

While other methods are more implicit, in that the coefficients and basis are dynamically generated with the input shapes, and the learning targets are still the final masks. For example, YOLACT [4] and BlendMask [7] both build the subspace basis dynamically with the inputs, and simultaneously estimate either the coefficients or attention maps throughout a deep network respectively to assemble these basis. ShapeMask [24] learns through the neural network a set of weights to linearly combine shape priors that are acquired offline via clustering methods.

2.3 The Distance Transform

The distance transform has been widely adopted for semantic segmentation to reduce blurring of boundaries [20, 19, 49]. Distance transforms can also be estimated in parallel with binary masks as an auxiliary task, and then fused to improve semantic segmentation [34, 12]. However, distance transforms have not been fully explored for instance segmentation.

Bai [1] did propose a modified watershed transform method for instance segmentation that is strongly related to the distance transform. It first uses a pre-trained semantic segmentation system to gate an image and estimate the gradient of the distance transform. The gradient map is then fed to a watershed network that generates a discretized energy map. Instances can be extracted as energy basins by a cut at a single energy level. Dense RepPoints [50] uses a distance transform to sample a dense set of points on either side of the object boundary to form a regression target. Since the shape representation is still contour rather than the distance transform

itself, the benefits from these distance transforms are quite limited.

2.4 Relationship to Our Approach

MEInst [51] uses PCA to address redundancy of binary region-mask representations, while ShapeMask [24] uses a set of shape priors learned by an offline K-means clustering method. Our SparseShape approach can be seen as intermediate between the two, in that our sparse components are expected to be more statistically independent and more articulated than the principal components used by MEInst, but are not necessarily complete shapes as in ShapeMask. We hypothesize that this may provide superior results through better regularization and handling of non-Gaussian shape distributions, while allowing more flexible reconstruction of novel shapes. On the other hand, while ShapeMask trains the network using a loss on the estimated mask and the network requires two extra steps to refine the shape, both MEInst and SparseShape use a loss directly on the coefficients, which we find to be more effective. Also, we apply this sparse coding approach to DTM representations of shape rather than binary masks to take advantage of the richer pixel-level shape information afforded by the distance transform.

YOLACT [4] and BlendMask [7] encode the joint shape of all objects in an image. We posit that this might limit generalization, since objects may occur in different combinations and different relative positions in novel datasets, and so we encode shapes individually. Unlike the deep watershed method of Bai [1], our SparseShape approach does not require a separate semantic segmentation network, and can be incorporated into one-stage object detection systems. Dense RepPoints [50] leverages a DTM only to sample

points near the object boundary, whereas SparseShape directly uses the DTM as a shape representation space.

3 Methodology

3.1 Distance Transform Representation

Given a mask within the bounding box of an object i , let G_i denote the set of all foreground pixels in the mask. The Distance Transform Map (DTM) of this object, $d_i(x)$, is defined as,

$$d_i(x) = \begin{cases} \min_{y \notin G_i} \|x - y\|_2 & \text{if } x \in G_i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $\|x - y\|_2$ is the Euclidian distance between two pixels x and y . To produce better segmentation performance, we make a few changes to this definition. We first resize each mask into a fixed $m \times m$ pixel window. Second, we normalize each DTM to have maximum distance 1. Finally, to improve the crispness of the boundary, we set background pixels to -1 instead of 0. Our new DTM definition thus becomes,

$$d_i(x) = \begin{cases} \min_{y \notin G_i} \frac{\|x - y\|_2}{d_i^{max}} & \text{if } x \in G_i \\ -1 & \text{otherwise} \end{cases} \quad (2)$$

where d_i^{max} is the maximum distance in the original DTM.

3.2 Sparse Coding

Sparse coding is a linear coding method in which a signal is reconstructed by a linear combination of a set of basis functions (a.k.a. atoms or components). The collection of these basis functions is called a dictionary. To

apply it to DTM encoding, we represent each $m \times m$ DTM as a vector $X_i \in \mathbb{R}^M$, where $M = m \times m$. The sparse representation $\hat{X}_i$ of X_i is then given by $\hat{X}_i = \mathbf{D}\alpha_i$ where the dictionary $\mathbf{D} \in \mathbb{R}^{M \times C}$, contains the C basis vectors and $\alpha_i \in \mathbb{R}^C$ is a vector of linear coefficients over these basis vectors.

We learn the dictionary $\mathbf{D}$ from a training dataset of N shapes by optimizing the following regularized objective,

$$\min_{\mathbf{D}, \alpha_i} \frac{1}{N} \sum_{i=1}^N \frac{1}{M} \|X_i - \mathbf{D}\alpha_i\|_2^2 + \lambda \|\alpha_i\|_1 \quad (3)$$

where $\lambda \in \mathbb{R}$ is a parameter that controls the importance of the regularization term. We solve for the dictionary $\mathbf{D}$ and coefficients α_i iteratively using online dictionary learning [31] and the fast ISTA algorithm (FISTA) [3].

Figure 3.1 shows a dictionary learned from the MS COCO [28] instance segmentation dataset (train2017 split), and an example of DTM reconstruction using this dictionary. Figure 3.1(b) shows one set of sparse coefficients generated by optimizing the objective in Equation 3. To demonstrate the sparsity of the representation, a histogram of the coefficients is plotted in 3.1(c). 3.1(d) and (e) illustrates the reconstruction performance by the sparse coding model. Specifically, a linear combination of basis functions in 3.1(a) with corresponding weights in 3.1(b) yields 3.1(e), which is an approximation of 3.1(d).

3.3 Flowchart of the Proposed Framework

Figure 3.2, 3.3, and 3.4 demonstrate all three stages of solving the instance segmentation task with our proposed method. Figure 3.2 illustrates

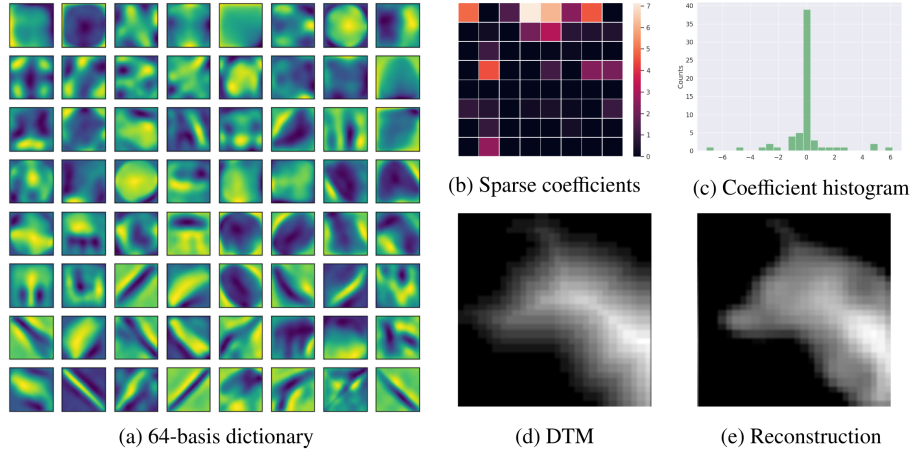


Figure 3.1: Distance transform and sparse coding. (a) A dictionary of 64 DTM basis functions learned from MS COCO dataset [28]; (b) A set of sparse coefficients of DTM (d) with respect to the dictionary in (a); (c) Histogram of the sparse coefficients in (b); (d) A DTM generated from a dog mask; (e) Reconstruction of the DTM derived by linear combination of the basis functions in (a) with the weights in (b). Note that the background pixels are clamped to 0 for visualization purpose. The size of the map is 28-by-28 for calculation, but is zoomed in for a better view.

the process of preparing ground truth sparse shape coefficients as the regression target of the network. Given a dataset of images and their corresponding annotations, we first crop and resize each ground truth instance mask into a fixed 28-by-28 resolution. Then, we normalize each mask according to Equation 2 and calculate the DTM. Finally, the dictionary and sparse coefficients are jointly learned via Equation 3. These ground truth sparse coefficients provide the supervision for training the instance segmentation head of our network (in Fig. 3.3(b) and Fig. 7.1 mask estimation head).

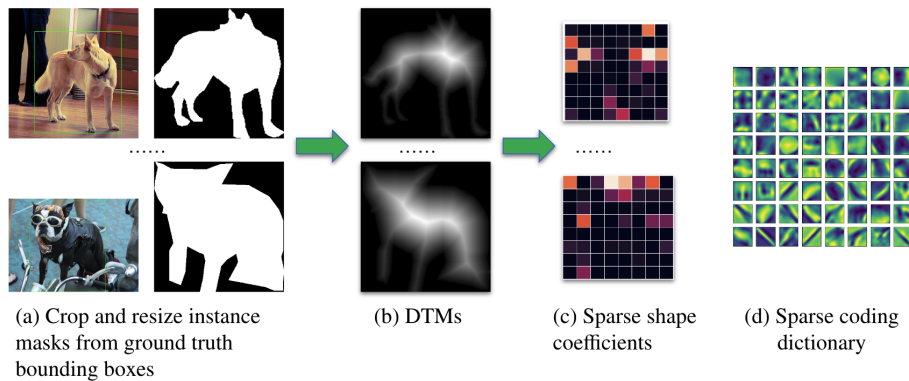


Figure 3.2: Stage 1: Dictionary Learning. (a) We first crop and resize each ground truth instance mask from its bounding box. (b) We compute the DTM from each resized mask. The DTM basis functions (d) and corresponding coefficients (c) are jointly learned via Equation 3. Green arrows mark the flow of the ground truth information used to generate the ground truth coefficients, which constitute the instance segmentation network training target.

At inference, we feed the input image through a one-stage object detector (we adopt the FCOS [43] detector in this work). For each detected object, we estimate the set of sparse coefficients through the instance segmentation

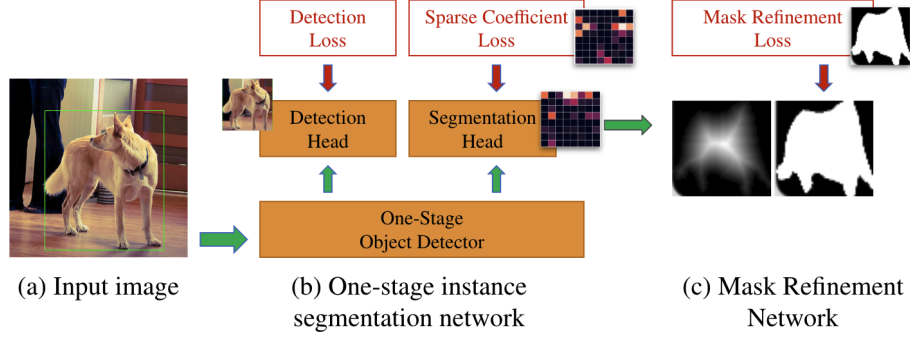


Figure 3.3: Stage 2: Instance Segmentation Network Training. The green arrows show the direction of a forward pass. A set of bounding boxes will be generated at the detection head and a set of estimated sparse coefficients will be the output of the instance segmentation head. The estimated sparse coefficients will be decoded to DTMs and then masks. The red arrows show the supervision signal coming back from the ground truth to optimize the corresponding network components jointly.

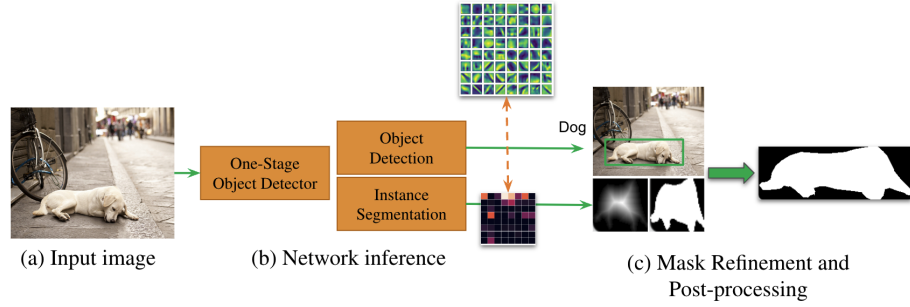


Figure 3.4: Stage 3: Inference. The input image (a) is fed through the instance segmentation network (b), which estimates the category, bounding box, and sparse coefficients in parallel. A mask refinement and post-processing step (c) first decodes the sparse coefficients for the DTM using the dictionary in Stage 1 (Figure 3.2). The mask can then be recovered by finding the zero-crossing level set of the DTM. The recovered mask will be further refined by a mask refinement module (to be discussed in Figure 3.6). Finally, the mask is resized and repositioned to align with the actual image resolution and object location using the estimated bounding box.

head. The associated DTM is then computed as a linear combination of the dictionary basis functions with these estimated coefficients as weights. The associated segmentation mask is then recovered from the zero-crossing level set of the DTM. Finally, we shift and scale the recovered mask according to the estimated bounding box location and size to get the actual instance segmentation mask. Additional details of the network will be discussed in the following sections.

3.4 Network Architecture

In this section, We will introduce the architecture of a one-stage instance segmentation framework (built upon the one-stage FCOS object detector [43]) as in Figure 3.5. I will briefly introduce the key components of this FCOS detector, and then discuss how to incorporate a conventional instance segmentation head, which will be used as our baseline to validate the proposed modules and shape encoding. Finally, We will discuss key innovations to this baseline instance segmentation system.

3.4.1 One-stage Object Detector

One-stage object detectors have recently demonstrated superior performance over two-stage methods while being faster and conceptually simpler [52, 43, 53]. Most recent approaches to instance segmentation (e.g., [51, 47, 42, 7, 25]) adopt the FCOS detector [43] as a base as it runs faster and performs better than Faster RCNN [38].

FCOS consists of a backbone for feature extraction, a feature pyramid network for better object localization across a wide range of different scales, a

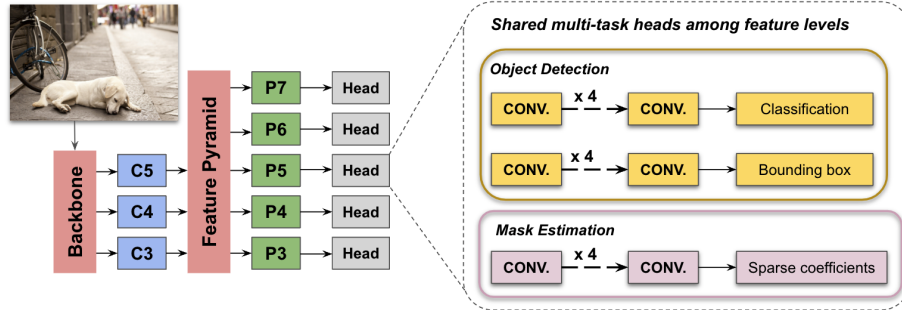


Figure 3.5: General architecture of a one-stage instance segmentation framework, e.g. FCOS [43], with an additional instance segmentation head. From left to right, we have 3 cascaded blocks: A backbone for feature extraction; A feature pyramid network (FPN) for feature enhancement across object scales; Shared multi-task heads, for each feature pyramidal level, consists of classification, bounding box regression and sparse coefficient estimation head; An additional post-processing step (not shown here) will be applied to recover the mask from the estimated sparse coefficients as shown in Figure 3.4(c).

classification head and a localization head (illustrated in Figure 3.5). These heads share the same set of parameters across different pyramidal levels except a single scalar learned from data and applied to the output of different pyramid level for bounding box size adjustment.

3.4.2 Instance Segmentation

To add the function of instance segmentation, one of the most straightforward ways is adding a separate head for this particular task with exactly the same configuration as the other two heads except the final output layer. This instance segmentation head can be trained to estimate shape representations. For example, PolarMask [47] estimates the length of rays from each object center to its bounding contour, and MEInst [51] estimates PCA

coefficients that can be decoded into instance masks.

In this work, our instance segmentation system consists of two cascaded stages: A coarse segmentation stage with feature fusion (Fig. 3.6(a)), and a mask refinement stage (Fig. 3.6(b)) that also estimates a mask quality score (Fig. 3.6(c)).

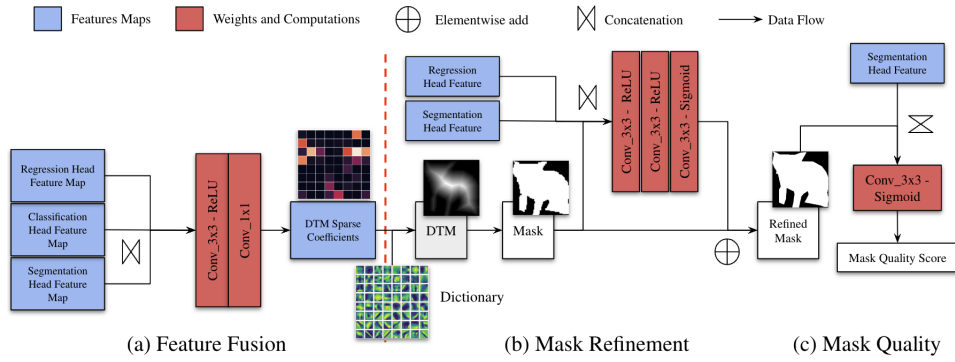


Figure 3.6: Proposed instance segmentation head: Feature fusion (a) mask refinement module (b) and mask quality score estimation (c). (a) The last feature map from the classification and regression head are concatenated with the segmentation head features to estimate the sparse coefficients of the DTM; (b) The predicted DTM is decoded by linearly combining the estimated sparse coefficients and the pre-learned dictionary. The zero-crossing level set of the DTM forms the initial estimate of the shape. A residual block refines this estimate with guidance from concatenated regression and segmentation features; (c) A lightweight quality scoring layer (3x3 convolution with a Sigmoid activation) takes as input the segmentation head features and the refined mask and estimates the IoU between the estimated and ground truth mask.

Coarse Segmentation Stage with Feature Fusion We employ a segmentation head that produces a feature map in parallel with classification and bounding box regression maps. In a standard single-stage instance seg-

mentation framework [51, 47, 42] these three heads do not share features, which ignores dependencies between class, bounding box and segmentation. To address this issue, we fuse features from all three heads with a small network consisting of a 3-by-3 convolutional fusion layer and a 1-by-1 convolutional output layer that produces estimates of the DTM sparse coefficients (Fig. 3.6(a)).

Mask Refinement Module The mask refinement module accepts the estimated sparse DTM coefficients from the feature fusion block and outputs a refined object instance mask based upon these coefficients. First, the DTM is reconstructed from the estimated coefficients and an initial estimate of the mask is identified as the zero-crossing level set of this DTM. To improve fine details of the output mask, we employ a lightweight mask refinement module with a single residual block (Figure 3.6(b)). This 3-layer residual block accepts as input the current estimated mask and concatenated segmentation and regression features, and produces a refined mask estimate.

Mask Quality Score Estimation Scoring Mask R-CNN [18] incorporates a module in its two-stage instance segmentation framework that estimates a quality score for each predicted mask. This strategy calibrates the misalignment between mask quality and object confidence scores, improving instance segmentation performance by prioritizing more accurate mask predictions during inference. We adopt this idea for inference, and also use this score in a novel way during training to promote hard examples.

In particular, a ground truth mask quality score is calculated as the IoU

between the current mask estimation and its corresponding ground truth. To estimate this quantity, we concatenate segmentation head features and the mask predicted by the mask refinement module together as input to a novel quality scoring layer consisting of a 3-by-3 convolution followed by a Sigmoid nonlinearity. This ground truth mask quality can also be used to re-weight each training example to emphasize harder cases (Fig. 3.6(c)).

3.5 Training

Our training loss is a weighted combination of five components:

$$\mathcal{L} = \lambda_d \mathcal{L}_{det} + \lambda_c \mathcal{L}_{coef} + \lambda_s \mathcal{L}_{sparsity} + \lambda_m \mathcal{L}_{mask} + \lambda_q \mathcal{L}_{quality} \quad (4)$$

We use the detection loss $\mathcal{L}_{det}$ adopted by FCOS [43], which includes a focal loss [27] for classification, generalized IoU loss [40] for bounding box regression, and binary cross-entropy loss for the centerness estimation. The sparse coefficient loss $\mathcal{L}_{coef}$ is an L^2 loss on the error of the sparse coefficients. $\mathcal{L}_{sparse}$ is an L^1 regularization applied on the estimated coefficients to encourage sparsity. $\mathcal{L}_{mask}$ is the dice loss [32] for the mask refinement module. Dice loss was originally proposed to overcome foreground-background imbalance for semantic segmentation. In our case, since the estimated masks are within bounding boxes, most of the pixels are foreground pixels, so the dice loss helps here as well.

Note that loss is applied at two different points in the segmentation head. The coefficient loss $\mathcal{L}_{coef}$ and sparsity loss $\mathcal{L}_{sparsity}$ are applied at the output of the feature fusion block as an intermediate supervision signal,

while the mask loss $\mathcal{L}_{mask}$ and the mask quality estimation loss $\mathcal{L}_{quality}$ are applied at the output of the mask refinement module. The gradients of the mask refinement module can be propagated all the way back to the very first layer of the network given that the DTM decoding procedure between the feature fusion block and the mask refinement module is fully differentiable, so the network is trained end to end. For all experiments, we adopt the weights of each component within the detection loss $\mathcal{L}_{det}$ from FCOS [43], and manually set the weights of other components λ_d , λ_c , and λ_q to 1, $\lambda_m = 2.5$, and $\lambda_s = 0.01$. These weights were optimized through a set of grid search experiments.

To make use of the mask quality during training, we up-weight the loss for mask estimates that have a low IoU with ground truth (hard examples). Specifically, the dice loss of each positive training example was re-weighted by the reciprocal of the IoU between the current mask estimation and its ground truth:

$$\mathcal{L}_m = \frac{1}{\mathbf{IOU}(\hat{\mathcal{M}}, \mathcal{M}) + \sigma} \mathcal{L}_{dice}(\hat{\mathcal{M}}, \mathcal{M}) \quad (5)$$

where $\hat{\mathcal{M}}_i$ and $\mathcal{M}_i$ represent the estimated and ground truth segmentation mask respectively. $\mathbf{IOU}(\cdot)$ is the function calculating the intersection over union, and $\alpha = 0.25$ is a margin parameter that imposes an upper bound on the training instance weights. Note that the weights will be in the range $(0.8, 4)$. Following Mask Scoring RCNN [17], we also use an L^2 loss to regress the mask quality scores.

3.6 Inference

The inference process remains mostly the same as FCOS [43] since we only append two more predictions to the predicted objects, i.e. masks and mask quality scores. Specifically, an input image is fed through the network and then predicts bounding boxes with several attributes, including classification confidence scores, sparse coefficients, masks, and mask quality scores. The final mask is calculated according to its corresponding bounding box size and location. As in Scoring Mask RCNN [18], before feeding the output masks to the non-maximum suppression function, we also use the estimated mask quality score to re-calibrate classification scores generated from the classification head. Specifically, the final score of each object is the product of its classification confidence and its estimated mask quality.

4 Experiments

4.1 Instance Segmentation Datasets

MS COCO: Common objects in context MS COCO [28] is one of the most widely-used benchmarks for object detection, instance segmentation, panoptic segmentation, and body keypoint estimation. Given that this thesis is closely related to both 2D object detection and instance segmentation, and all the above-mentioned state-of-the-art work has been evaluated on this dataset, we will also primarily evaluate our results on this dataset. The COCO 2017 dataset contains about 117K images for training with an average of 7.3 instances per image. Instance masks are available for objects in a total of 80 categories. These masks are annotated according to the visible pixels of the objects. No occlusion or 3D information is currently available.

Cityscapes Cityscapes [10] is an important benchmark for road user (including cars, trucks, buses, motorcycles and their riders, bicycles and their riders, and pedestrians) detection and segmentation. There are 5K traffic images taken by an on-board calibrated stereo camera with an average of 18.8 instances per image. Instance segmentation masks are available for visible pixels only. The amount of data in this dataset is relatively small, but enough for fine-tuning and evaluation purposes. The dataset has been divided into 3000 training, 500 validation, and 1500 test images. There are 8 classes, seven of which are also in the MS COCO instance segmentation category. The eighth class is *rider*, which represents people who are cycling or riding a motorcycle, whereas MS COCO groups all people into the *per-*

son category no matter what their activities are. All images are 2048×1024 pixels. Note that the original Cityscapes dataset has both fine and coarse segmentation labels. We follow current practice and only use the fine labels for fine-tuning.

4.2 Evaluation Metric

The primary evaluation metric used in this work is mask AP, introduced with the MS COCO [28] dataset in 2014. Mask AP is the area under the precision-recall curve, averaged over multiple IoU thresholds $\tau \in (0.5, 0.95)$ with an interval of 0.05 (see below), and is also averaged over all object categories.

During evaluation, the mask AP for each category is calculated first. For category C , the set of predicted masks $\hat{M}_i$ and their associated confidence scores $\hat{S}_i$ will be evaluated against all the ground truth masks M_i for this category C . The **IoU** between every pair of estimated and ground truth mask is calculated as,

$$IoU(\hat{M}_i, M_k) = \frac{\mathbf{A}(\hat{M}_i \cap M_k)}{\mathbf{A}(\hat{M}_i \cup M_k)} \quad (6)$$

where $\mathbf{A}$ is the function returning the area of a mask. A prediction is then scored against ground truth according to,

$$\text{Result} = \begin{cases} \text{Hit,} & \text{if } \mathbf{IoU}(\hat{M}_i, M_k) \geq \tau_j \\ \text{Miss,} & \text{otherwise} \end{cases} \quad (7)$$

If multiple estimated masks exceed the **IoU** threshold against the same ground truth, the mask with the highest confidence score $\hat{S}_i$ will be considered as the hit. Once a hit is determined, both the estimated and the ground truth mask will be consumed, so any prediction will not match with more than one ground truth, and vice versa. After a complete enumeration of all ground truth masks, the estimated masks that do not match with any ground truth and all ground truth masks that do not match with any predictions will be considered as false positives and false negatives respectively.

The mask AP for category C at threshold τ_j is estimated as the average of maximum precision at 101 recall levels uniformly sampled between 0 and 1,

$$\begin{aligned} AP_C^j &= \frac{1}{101} \sum_{r \in (0, 0.01, \dots, 0.99, 1)} AP_r \\ &= \frac{1}{101} \sum_{r \in (0, 0.01, \dots, 0.99, 1)} AP_{interp}(r) \end{aligned} \quad (8)$$

where

$$AP_{interp}(r) = \max_{\tilde{r} \geq r} AP(\tilde{r}) \quad (9)$$

such that we replace the precision value with the maximum precision for any recall that is greater than or equal to $\tilde{r}$. The intention in interpolating the precision and recall curve in this way (Equ. 9) is to reduce the impact of the “wiggles” in the precision and recall curve, caused by small variations in the ranking of examples.

In order to compute the performance over all categories, the mean Average Precision (mAP) is computed as the mean of the APs from all categories. Finally, these mAP values are averaged over all IoU thresholds τ_j to get the

final evaluation metric (referred to as “mask AP” across the thesis).

We also evaluate different aspects of our system and competing methods with other metrics, which we will discuss later with specific experiments.

4.3 Implementation Details

4.3.1 Sparse Coding Dictionary Learning

To learn a class-agnostic dictionary, we draw segmented shapes from all 80 object categories in the training partition of the MS COCO [28] dataset. For categories with 1,000 or fewer instances, we use all instances. For categories with more than 1,000 instances we randomly select 1,000 instances whose area is greater than 500 pixels (as these bigger masks provide more shape details to support dictionary learning).

In the end, we collected a shape dataset of around $78K$ resized masks. Each shape mask is first converted into its DTM of size 28×28 pixels using Euclidean distance, and then normalized according to Equation 2. We then vectorized all N DTMs, and put them into a matrix of size $N \times 784$. For dictionary learning and sparse coding, we selected 500 as the batch size, and optimized the objective function (Equation 3) for 2000 iterations.

In fact, the dictionary learning is a data-driven process, but it is not a data hungry one. To assess how our system depends upon the size of the dataset used to train the DTM sparse shape dictionary, we randomly sample subsets of images from the COCO training partition in the above mentioned manner and assess the accuracy of shape reconstruction on the validation partition, given ground truth shape targets, as well as the proportion of ac-

tive shape components, which indicates the sparsity of the encoding. Table 4.1 shows that, while reconstruction accuracy and sparsity improve asymptotically with the size of the training dataset, excellent results are obtained with only a few hundred random training shapes. This suggests that data requirements will be driven more by the needs of the deep segmentation network than by the shape dictionary.

Table 4.1: Ground truth reconstruction performance of our sparse shape coding method and the average activation rate (proportion of coefficients > 0.001), as a function of the size of the training set. We employed a dictionary with 128 shape components. Training sets were randomly drawn from the MS COCO training partition, and performance was evaluated on the MS COCO validation partition (*val2017* [28]).

Training images	mIoU	Activation rate
400	0.942	0.562
800	0.949	0.545
1600	0.954	0.528
3200	0.957	0.516
6400	0.958	0.505
12800	0.960	0.494

4.3.2 Network Training and Inference

Following the practice in previous work [51, 42, 47, 4, 7, 45], all our models are trained on the MS COCO *train2017* split, and all ablation experiments are evaluated on the *val2017* split. Comparisons with other state-of-the-art methods are reported on the *test-dev* split. Unless specified, we adopt the $1\times$ training strategy, single scale training and testing for all ab-

lation experiments, and $3\times$ training strategy with multi-scale training and single scale testing when comparing to the state-of-the-art (see below). In addition, we fine-tune and test our model on the Cityscapes benchmark following the same configurations in Mask RCNN [17] and CondInst [42].

MS COCO Experiments For the experiments on MS COCO, ResNet-50 and ResNet-101 [16] with FPN [26], pre-trained on ImageNet, are used as backbones, and all hyper-parameters are adopted from the FCOS object detector [43]. The detection heads follow the design employed by MEInst [51]. All the models are trained with stochastic gradient descent for 90K iterations (about 12 epochs) under the $1\times$ training strategy with an initial learning rate of 0.01. The learning rate is reduced by a factor of 10 at iteration 60K and at 80K. The mask width and height are both $m = 28$ pixels for all experiments, which is also consistent with MEInst. For the $3\times$ training, we train the model for 270K iterations (about 36 epochs), and the learning rate is reduced by a factor of 10 at 180K and 250K respectively. We use a mini-batch of 16 images and all models are trained on 2 GPUs with 8 images per GPU.

The input images for single scale training are resized to have their shorter sides either 400 pixels (for ablation experiments) or 800 pixels (for state-of-the-art comparison). If the resulting images after resizing have their longer side greater than 1,333 pixels, the longer side of the original image will be resized to 1,333 and the aspect ratio is preserved. For multi-scale training, multiple uniform choices of input size (minimum of horizontal and vertical dimensions) in the range of [640, 800] pixels are used with an interval of 64.

This and left-right flipping are the only data augmentation techniques used for training. When testing, the minimum dimension of the image is resized to 800 pixels.

Cityscapes Experiments For the experiments on the Cityscapes dataset, given the limited number of training images, the network must be trained on MS COCO first and fine-tuned on Cityscapes images. Specifically, we used our MS COCO-learned shape dictionary and network, and then fine-tuned the network on Cityscapes (with fine annotations only), using a batch size of 16, for $3K$ iterations. The classification head was initialized with weights from the COCO model for the 7 overlapping categories, and randomly initialized for the one non-overlapping category. We reduced the learning rate from 0.01 to 0.001 at $2K$ iterations.

4.4 System Optimizations

4.4.1 DTM Resolution

Table 4.2 shows the effect of DTM resolution on the ultimate segmentation performance of our method. We adjusted the sparsity constant λ in Equation 3 to maintain the proportion of active DTM components (coefficient magnitudes greater than 0.001) at roughly 50%. While the reconstruction IoU (given the ground truth mask) increases with both the DTM size and the dictionary size, the mask AP achieved by the network peaks at mask sizes of 28×28 and 42×42 pixels with 128 basis vectors. Since 28×28 masks are slightly more efficient, we use this resolution for the remaining experiments.

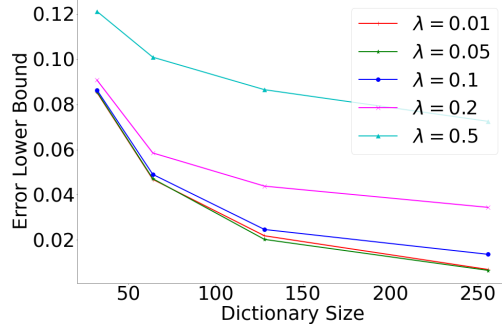
Table 4.2: Grid search on the dimensionality of DTM to be used for sparse coding and instance segmentation network training.

DTM Dimension	Dictionary Size	Reconstruction IoU	Mask AP
14×14	32	0.891	0.241
	64	0.938	0.25
28×28	64	0.959	0.253
	128	0.965	0.257
42×42	128	0.962	0.257
	256	0.983	0.256
56×56	256	0.981	0.254
	512	0.992	0.251

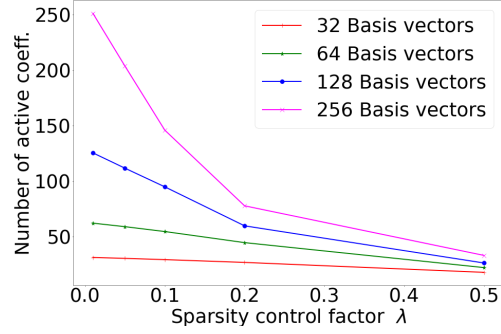
4.4.2 Sparse Coding Parameters

We first study the influence of dictionary size C and sparsity constant λ . Reconstruction error ($1 - \text{mean IoU}$, given the ground truth shape as a target) decreases for larger dictionaries, since larger dictionaries are more expressive, and for smaller sparsity constants, since this places more emphasis on the fidelity term of the loss (Fig. 4.1(a)). The number of active components (defined as coefficient magnitudes greater than 0.001) decreases as the sparsity constant is increased (Fig. 4.1(b)).

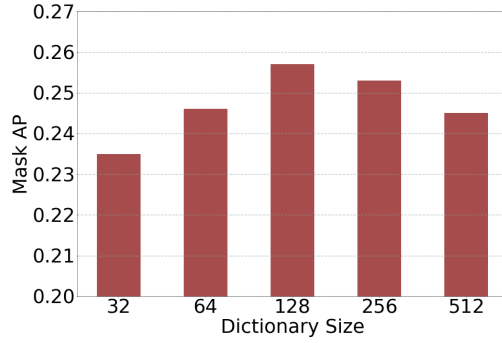
While reconstruction error given the ground truth as target monotonically decreases with dictionary size, this is not necessarily the case when coefficients must be estimated by the network, since larger dictionaries might require more network capacity to avoid underfitting, or more training data to avoid overfitting. Indeed, when fixing the sparsity constant to $\lambda = 0.2$



(a) Lower bound on error



(b) Sparsity



(c) Segmentation network performance

Figure 4.1: Effects of sparse coding parameters, evaluated on COCO [28] validation data. (a) Reconstruction error, defined as $1 - \text{mean IoU}$. (b) Number of active coefficients (c) Instance segmentation performance (mask AP) for $\lambda = 0.2$ (a trade-off between high sparsity and low reconstruction error).

we find that Mask AP on the validation data peaks at an intermediate dictionary size of $C = 128$ components (Fig. 4.1(c)). We use these parameters for our remaining experiments.

4.4.3 Loss Function for Sparse Coefficient Estimation

In Table 4.3, we show the effect of the loss function of the instance segmentation head on segmentation performance. Clearly, mean squared error ($L2$ Norm) loss significantly outperforms other candidates on Mask AP evaluated on the MS COCO *val2017* dataset [28].

Table 4.3: Evaluation of loss functions for sparse coefficient regression on the MS COCO *val2017* set [28].

Loss on Sparse Coefficients	Mask AP
Mean Squared Error (L2)	0.257
Mean Absolute Error (L1)	0.247
Smooth L1	0.249
Cosine Similarity	0.235

4.4.4 Shape Representation and Encoding Method

Table 4.4 compares the mask AP performance on the MS COCO validation set [28] for mask and DTM shape representation methods as well as PCA versus sparse coding in a 2×2 crossed design. We also assess the impact of the dimensionality of the encoding (64 vs 128) for the mask representation case. We do not include feature fusion or mask refinement

in these ablation experiments. Overall, we find that the DTM representation improves performance over the mask representation, and sparse coding outperforms PCA.

Table 4.4: Evaluation of Mask vs DTM shape representation and PCA vs sparse encoding on MS COCO *val2017* set [28].

Shape Representation	Encoding Method	Dimensionality	Mask AP
Mask	PCA	64	0.241
		128	0.245
	Sparse Coding	64	0.243
		128	0.249
DTM	PCA	128	0.250
	Sparse Coding	128	0.257

4.4.5 Feature Fusion and Mask Refinement

Table 4.5 shows the effectiveness of the proposed feature fusion strategy evaluated on improving the classification performance. FCOS [43] (the base detector we adopt) is an anchor-free object detector whose classification head outputs a heatmap (the last feature map) of object center locations with shape (C, H_l, W_l) , where $C = 80$ equals the number of categories in the MS COCO dataset, H_l , and W_l represents the height and width of the feature map at the l^{th} feature pyramidal level. To calculate the classification accuracy, we first take all the center locations of the ground truth bounding boxes and map these locations onto the last feature map of the classification head. From each of this ground truth location, we then extract the vector

along its C dimension and find the maximum value (in the SoftMax vector) and its index. Finally, we compare this index with the ground truth class label index and compute the classification accuracy.

While adding the basic segmentation head leads to a minor drop on the classification accuracy, adding the feature fusion to the segmentation does increase the classification accuracy. This is because the shortcut connections direct gradient information from the segmentation head to the classification head in a shorter path compared to the original connections via the backbone and feature pyramid in the original structure without feature fusion, and the shape-related segmentation information has discriminative capability to improve on classification.

Table 4.5: Effectiveness of our feature fusion strategy on classification performance.

Network Configuration	Classification Accuracy
FCOS (base detector [43])	0.794
FCOS + segmentation head	0.786
FCOS + segmentation head + feature fusion	0.807

Table 4.6 shows the effectiveness of the proposed feature fusion and mask refinement strategies evaluated by mask AP. A basic architecture with 3 parallel task heads is used as baseline. We see that features from both classification and bounding box regression make a contribution to instance segmentation performance, and our lightweight mask refinement module adds an additional boost. We include both of these strategies in our final system. We will use SparseShape to refer to our complete system in the rest of the

thesis.

Table 4.6: Effectiveness of our feature fusion and mask refinement strategies on mask AP.

Classification	Bounding Box	Mask Refinement	Mask AP
			0.257
✓			0.260
	✓		0.259
✓	✓		0.263 (+0.6%)
✓	✓	✓	0.267 (+0.4%)

4.4.6 Mask Quality Modulation

Table 4.7 shows the effectiveness of the mask quality score estimation during both training and inference. The baseline model does not use mask quality. Using the estimated mask quality at inference leads to an improvement of 0.2% on mask AP and incorporating the estimated mask quality as a re-weighting factor during training produces an additional 0.4% boost. We use mask quality at both training and inference for our complete SparseShape system.

Table 4.7: Effectiveness of the mask quality estimation for training and inference.

Method	Mask AP
Baseline	0.267
+ mask quality score (inference [18])	0.269 (+0.2%)
+ mask quality score (training + inference)	0.273 (+0.4%)

4.4.7 Sparsity of the Estimated DTM Coefficients

Our loss function 5 includes a sparsity term that encourages the estimated Distance Transform Map (DTM) coefficients to be sparse. To understand the effects of this term, we first assess how varying the weight λ_s of this term influences the dispersion (measured by variance), kurtosis and sparsity of the coefficients estimated by the network, and how this in turn influences segmentation performance (Table 4.8).

Table 4.8: Average variance, kurtosis, sparsity (proportion of absolute coefficients less than 0.001), and resulting segmentation performance (average AP) for a 128-vector DTM dictionary, over the MS COCO *val2017* dataset. [28].

λ_s	Std. Dev.		Kurtosis		Sparsity		mask AP
	GT	Est.	GT	Est.	GT	Est.	
0	0.87	0.72	8.49	3.9	50.4%	9.6%	0.254
0.005	0.87	0.7	8.49	4.0	50.4%	12.7%	0.257
0.01	0.87	0.69	8.49	4.3	50.4%	15.1%	0.257
0.02	0.87	0.65	8.49	5.5	50.4%	17.3%	0.253
0.05	0.87	0.59	8.49	6.1	50.4%	18.5%	0.247

We see that increasing λ_s does increase the kurtosis and sparsity of the estimated coefficients, driving them toward the ground truth values, but at the same time reduces the dispersion, driving it away from the ground truth value. Perhaps because of this tradeoff, we find that while performance (AP) initially increases as we increase λ_s , performance decreases for larger values, even though the estimated coefficients are not as sparse as they should be, in comparison to the coefficients that optimize the sparse objective function

3 given the ground truth target shape. For the experiments in the paper, we set $\lambda_s = 0.01$, the value we found empirically to maximize the mask AP performance and encourage the sparsity the most.

4.5 Comparison to State-of-the-Art

4.5.1 Mask AP

Table 4.9 and Table 4.10 compare the performance of our DTM-based SparseShape instance segmentation system with the state-of-the-art one-stage systems on the MS COCO *test-dev2017* dataset [28]. Since the backbone and number of training epochs greatly affect performance, these tables are organized to allow easy comparison between systems with the same backbone, trained for the same number of epochs, where possible. SparseShape outperforms prior contour-based approaches (e.g. PolarMask [47]) and prior coefficient-based approaches (e.g. ESE-Seg, MEInst, and YOLACT [48, 51, 4]). DeepSnake [36] is also a contour-based method but is trained with a double-stack hourglass backbone for 160 epochs, and reports a 0.303 mask AP on the MS COCO *test-dev*. Although using a more powerful backbone and training for more epochs, DeepSnake still under performs our SparseShape (complete, ResNet-50-FPN) system by 6.5%.

Comparing with region-based methods, SparseShape outperforms CM-SAM, ShapeMask, and CM-PR and produces competitive results with SOLO, but does not quite reach the performance of recent methods CondInst [42] and BlendMask [7] for 36 training epochs. These two methods employ high resolution global semantic maps in their network design, which potentially

Table 4.9: Comparison to state-of-the-art methods using a ResNet-50-FPN backbone. All results reported are on the MS COCO *test-dev 2017* partition. 36 epochs corresponds roughly to the $3\times$ training protocols with batch size of 16. “Basic” refers to a basic segmentation head without feature fusion, mask refinement or mask quality estimation, and “Complete” includes all these features.

Class	System	# Epochs	Mask AP
Contour	ESE-Seg[48]	36	0.307
Region	PANet[30]	36	0.366
	CM-SAM[25]	36	0.347
	CM-PR[46]	36	0.361
	SOLO[45]	36	0.368
	BlendMask[7]	36	0.370
	CondInst[42]	36	0.378
	SparseShape (Basic)	36	0.353
	SparseShape (Complete)	36	0.368

Table 4.10: Comparison to state-of-the-art on the MS COCO *test-dev 2017* split using a ResNet-101-FPN backbone. 36 and 48 epochs corresponds roughly to the $3\times$ and $4\times$ training protocols with batch size of 16. Models are all following standard protocols [43]. “Complete” refers to the segmentation head with feature fusion, mask refinement and mask quality estimation. DeepSnake adopts an Hourglass-104 backbone which has comparable model size to the ResNet-101 backbone.

Class	System	# Epochs	Mask AP
Contour	DeepSnake[36]	160	0.303
	PolarMask[47]	36	0.321
Region	YOLACT[4]	48	0.312
	ShapeMask[24]	36	0.374
	MEInst[51]	36	0.339
	SparseShape (Complete)	36	0.379

improves the separation among neighbouring objects and the accuracy of bounding contours.

4.5.2 Topological Accuracy

Mask AP measures overlap between estimated and ground-truth masks but does not tell us whether the topology of the estimated mask (e.g., number of connected components and holes) matches the topology of the ground truth. Arguably, getting the topology right can be even more important than getting the exact overlap for downstream tasks. Here we compare the topological accuracy of our SparseShape approach with leading SOTA one-stage segmentation approaches (SOLO [45], CondInst [42] and BlendMask [7]).

Topological MAE As a first measure of topological performance we assess how closely the number of connected components in the estimated mask matches the number in the ground truth mask. Specifically, we identify, for each ground truth mask in the COCO validation set, the estimated mask with correct label and highest IoU over a specified threshold. We define the *Topological MAE* (T-MAE) as $\mathbb{E} \left[\left| \hat{k} - k \right| / k \right]$, where $\hat{k}$ and k are the number of connected components in the estimated and ground truth masks, respectively. (If no estimate meets the class and IoU criteria, we ignore the object - missed detections are already assessed by the mask AP measure.)

Table 4.11 shows that SparseShape vastly outperforms (by a factor of 5-12) competing SOTA systems in identifying the correct number of connected components. This staggering difference is due primarily to an over-segmentation of objects by these competing methods, which tend to fragment a single object into multiple pieces. We note that the method with highest AP (CondInst [42]) also commits the most topological errors, suggesting that there may be a direct trade-off between pixel-level mask AP and topological performance.

Table 4.11 also shows that the principal basis for the superior topological performance of SparseShape is our core sparse DTM shape coding network (Basic). Feature fusion improves topological performance slightly, while mask refinement slightly lowers topological performance, possibly due to incorrect fragmentation of some objects into multiple components.

Topological Distribution As a second measure of topological performance we assess how closely the SparseShape model and other competing

Table 4.11: Topological MAE (error in number of connected components in mask), evaluated on the COCO [28] validation set.

Instance Segmentation System	Topological MAE	
	<i>IoU</i> @0.5	<i>IoU</i> @0.75
SOLO[45]	0.523	0.396
CondInst[42]	0.686	0.559
BlendMask[7]	0.519	0.431
SparseShape Basic	0.109	0.049
SparseShape Basic + FeatFusion	0.098	0.048
SparseShape Complete	0.101	0.055

methods are able to match the topological distribution in the ground truth masks. Table 4.12 breaks down COCO ground truth objects into four disjoint topological categories: 1C: Objects with a single connected component and no holes; 2C: Objects with two connected components and no holes; ≥ 3 C: Objects with three or more connected components and no holes; Hole: Objects with holes. We see that the vast majority of COCO objects (89.1%) consist of a single connected component with no holes. Competing SOTA methods greatly underestimate the number of single-component objects and overestimate the number of objects with 2 or more components and/or holes. While biased slightly in the opposite direction, overall SparseShape better matches the distribution of ground truth topologies, as measured by KL-divergence.

Table 4.12: Topological distribution of ground truth and estimated masks on the COCO [28] validation set. KL is the KL-divergence from ground truth.

	1C	2C	$\geq 3C$	Hole	KL
Ground Truth	89.1%	7.8%	2.8%	0.2%	-
SOLO[45]	74.8%	12.3%	11.0%	1.8%	0.113
CondInst[42]	80.6%	8.2%	9.3%	1.8%	0.072
BlendMask[7]	75.0%	10.2%	11.5%	3.2%	0.145
SparseShape	96.4%	2.7%	0.2%	0.7%	0.049

4.5.3 Generalization

Explicit incorporation of the statistics of shapes through our shape dictionary has the potential to serve as an inductive prior, which can lead to better generalization. To test this hypothesis, we evaluate how easily our SparseShape system and CondInst [42] can generalize to Cityscapes instance segmentation [10]. This is a challenging generalization task for several reasons. First, since MS COCO images were harvested from the web while the Cityscapes dataset is generated from a vehicle camera, the images are generally quite different. Second, while MS COCO contains 80 categories, the Cityscapes dataset contains only 8. Third, the Cityscapes dataset contains relatively few training samples, especially for truck, bus and train categories.

We follow the same procedure as CondInst [42] to transfer our system from MS COCO to Cityscapes, using our MS COCO-trained shape dictionary and network, and then fine-tuning only the network on Cityscapes, using a batch size of 16, for 3000 iterations. The new classification head was initialized with weights from the COCO model for the 7 overlapping

categories, and randomly initialized for the one non-overlapping category. The results are shown in Table 4.13. Our method outperforms CondInst on both of the two key evaluation metrics: by 0.4% on AP and 2.1% on AP_{0.5}. This improvement derives in part from better performance on rarer categories like truck, bus and train, where the inductive prior embodied by our sparse shape dictionary becomes more important.

Table 4.13: Performance on Cityscapes instance segmentation benchmark.

Method	AP	AP _{0.5}	person	rider	car	truck	bus	train	mcycle	bicycle
MRCNN[17]	0.365	0.622	0.348	0.288	0.53	0.343	0.57	0.375	0.223	0.238
CondInst[42]	0.369	0.632	0.35	0.284	0.555	0.374	0.575	0.363	0.221	0.233
SparseShape	0.373	0.653	0.344	0.284	0.538	0.388	0.582	0.396	0.224	0.228

4.5.4 Computational Time

Based on a ResNet-50 backbone with input shorter sides resized to 800 pixels, Table 4.14 compares the frame rates of our SparseShape system with competing systems on an Nvidia GTX 1080 Ti GPU. Our complete system is slightly slower but in the same range as competing systems. The frame rate of our system drops from 9.6 fps to 5.1 fps for a ResNet-101 backbone.

4.6 Qualitative Results

4.6.1 Topological Improvements

Figure 4.2 provides a qualitative comparison of SparseShape segmentations with SOTA methods CondInst [42] and BlendMask [7] on 4 example images from the COCO validation set, with a focus on objects with di-

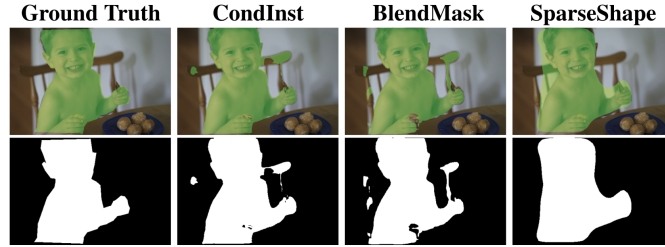
Table 4.14: Run-time comparison using a ResNet-50 backbone on an Nvidia GTX 1080Ti GPU.

Model	fps
SparseShape-Basic	12.7
SparseShape-Complete	9.6
CondInst	11.5
BlendMask	14.1
MEInst	12.6
SOLO	15.4

verse ground truth topologies (one connected component, two connected components, three connected components, and holes). While SparseShape correctly identifies the ground truth topologies, CondInst and Blendmask commit major topological errors. In Fig. 4.2(a), both CondInst and BlendMask generate multiple extraneous segments and holes. In Fig. 4.2(b), both CondInst and Blendmask miss one component and in (c) they both incorrectly join two components. In Fig. 4.2(d), CondInst fails to estimate the correct hole and BlendMask breaks it into two holes.

4.6.2 Additional Challenging Examples from MS COCO

Figure 4.3 shows additional qualitative comparisons with other SOTA methods, including MEInst [51], CondInst [42] and BlendMask [7], with a ResNet-50-FPN backbone, on a number of challenging examples. Overall, SparseShape yields segmentations that are more articulated than MEInst and less articulated than CondInst and Blendmask. While CondInst and



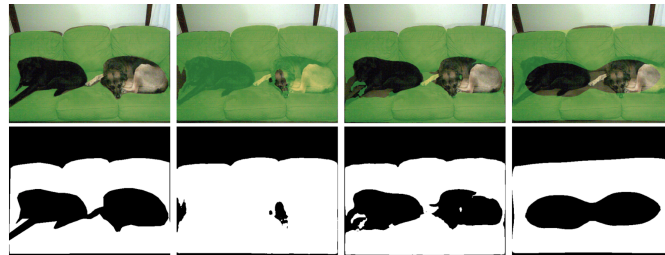
(a) Segmentation with 1 connected component



(b) Segmentation with 2 connected component



(c) Segmentation with 3 connected component



(d) Segmentation with holes

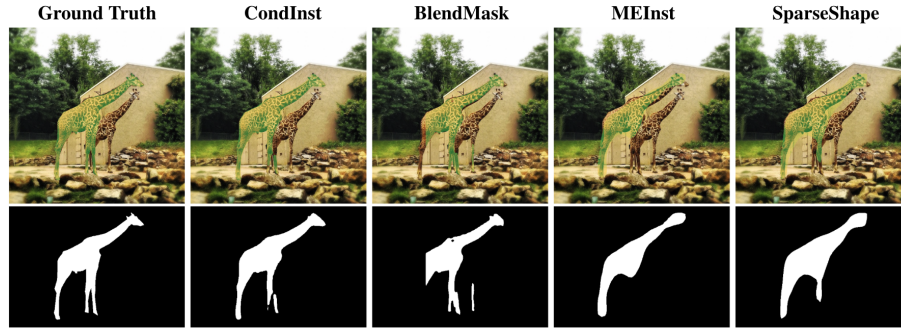
Figure 4.2: Comparison of SparseShape with SOTA (using a ResNet-50 FPN backbone) on diverse topologies from the COCO validation set.

BlendMask better capture some finer articulations, they also tend to commit more major errors. In Figure 4.3(a), MEInst completely misses the front legs and BlendMask misses the back legs. Both CondInst and BlendMask generate extra segments. In Figure 4.3(b), both CondInst and Blendmask produces badly fragmented and incorrect topologies. While CondInst assimilates much of the background into the segmentation, BlendMask misses much of the foreground object. In Figure 4.3(c), MEInst misses the left component of the bed, while both CondInst and BlendMask produce badly fragmented masks and confuse the boundary between the bed and the child.

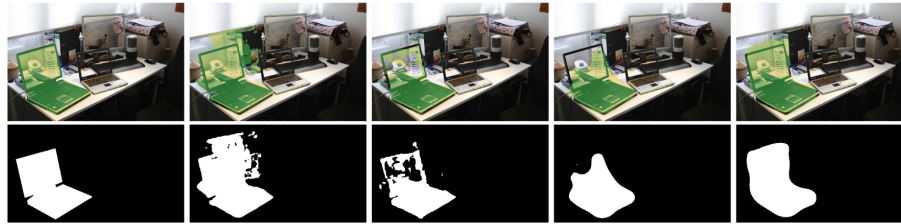
Figure 4.4 shows additional qualitative examples based on a ResNet-101 backbone [16]. These examples demonstrate the ability of our sparse DTM method to estimate complex segmentation with a variety of topologies in cluttered scenes.

4.6.3 Examples with Improved Segmentation

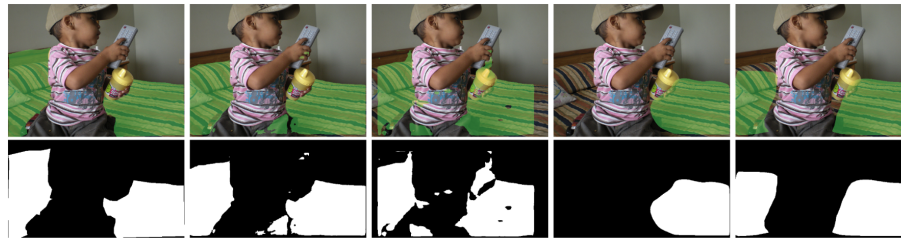
To gain further insight, Figure 4.5 and 4.6 compare the performance of SparseShape and leading methods CondInst [42] and BlendMask [7], respectively, on instances that the latter methods find challenging. SparseShape exhibits multiple benefits in these examples, generating the correct topologies, missing fewer object parts, exhibiting less confusion and fragmentation at the boundaries between objects and less assimilation of background into foreground.



(a) Segmentation between similar objects



(b) Segmentation in complex background



(c) Segmentation in complex appearance and occlusion

Figure 4.3: Qualitative comparison with leading methods for challenging cases based on a ResNet-50-FPN backbone. SparseShape generates a more complete segmentation than MEInst and avoids the fragmentation produced by CondInst and BlendMask.



Figure 4.5: Challenging instances for CondInst[42] (ResNet-50 backbone) from the MS COCO *val2017* dataset.



Figure 4.6: Challenging instances for BlendMask[7] (ResNet-50 backbone) from the MS COCO *val2017* dataset.

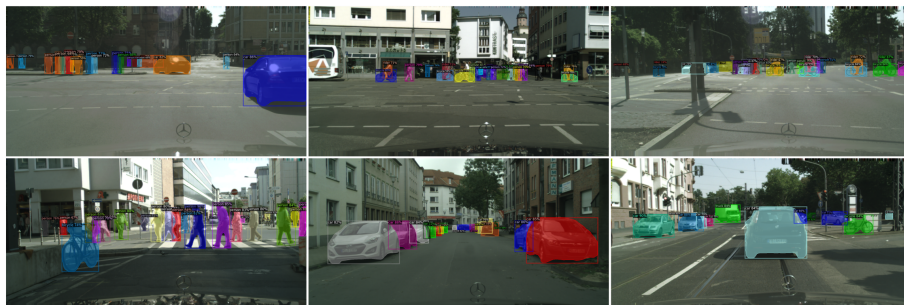


Figure 4.7: Qualitative results on Cityscapes images. Model has a ResNet-50-FPN backbone. We use our MS COCO-trained shape dictionary and network, and then fine-tune the network on Cityscapes, using a batch size of 16, for 3000 iterations.

5 Analysis of Limitations

5.1 Limitation of Dictionary Adaptation

Figure 5.1 compares the dictionary learned from MS COCO and Cityscapes datasets respectively. There are some notable differences between these two

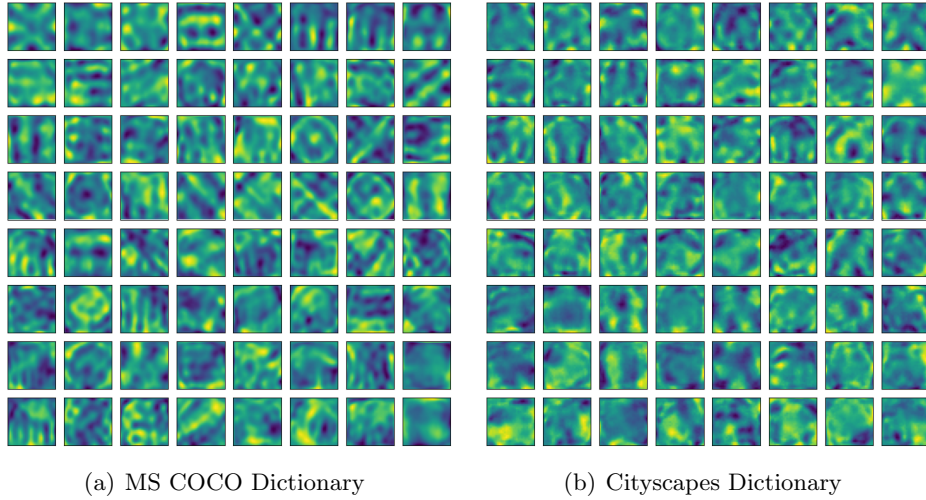


Figure 5.1: Sparse DTM dictionaries learned from MS COCO and Cityscapes datasets respectively. 128 basis vectors are used for pre-training and fine-tuning, only the first 64 most frequently used (descending order according to their frequency of usage) vectors are shown for comparison.

dictionaries. First, while each MS COCO basis function appears to be fairly unique, there are CityScape basis functions that appear quite similar to each other. Second, while it is hard to interpret any of the MS COCO basis functions as specific objects, there are several CityScapes basis functions that look like pedestrians (in Fig. 5.2).

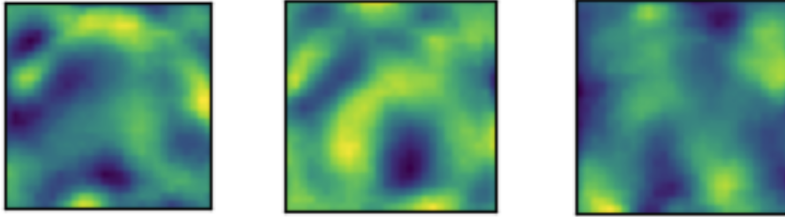


Figure 5.2: Cityscapes DTM basis functions that resemble pedestrians (selected from Figure 5.1).

MS COCO has shapes across 80 different categories, whereas Cityscapes only has 8. As a result, the MS COCO dictionary carries much more variance to accommodate a wider range of instance shapes, and it is hard to identify familiar shapes in these basis functions. However, it is relatively easy to find meaningful basis shapes in the Cityscapes dictionary. For example, there are a few basis functions that resembles pedestrians (Fig. 5.2). To summarize, the Cityscapes dictionary appears to be less diverse, more redundant and more interpretable than the MS COCO dictionary.

Given these differences, adapting a model trained on one dictionary to the other can be non-trivial. Table 5.1 evaluates how generalization to Cityscapes depends upon the dictionary used to pre-train the network on the source dataset (COCO) and the dictionary used to fine-tune the network on the target dataset (Cityscapes). While one might have expected performance on Cityscapes to improve if we use a dictionary trained on Cityscapes, we find that best results are achieved by consistently using a dictionary trained on COCO or a mix of COCO and Cityscapes images, rather than training the dictionary on Cityscapes. This is likely because pre-training of the network on the rich MS COCO dataset works best with

a matching MS COCO dictionary. Swapping in a new dictionary at the fine-tuning stage introduces a discontinuity in learning, as there is no correspondence between shape components in the two dictionaries.

Table 5.1: Generalization performance on Cityscapes as a function of the shape dictionary used for pre-training and fine-tuning. *Mixed* represents a random 80%/20% split of COCO and Cityscapes images equally distributed in all categories.

Dictionary Used		Mask AP	
Pre-training	Fine-tuning	Pre-training	Fine-tuning
COCO	COCO	0.367	0.373
COCO	Cityscapes	0.367	0.346
Cityscapes	Cityscapes	0.361	0.371
Mixed	Mixed	0.366	0.373

5.2 Limitation of SparseShape

Figure 5.3 shows some challenging images for SparseShape, drawn from the COCO validation dataset. We discuss in some detail these challenges and others below.

5.2.1 Detection Error

One source of segmentation error is poor localization and scaling of bounding boxes. If an estimated bounding box has the wrong location, size or aspect ratio, there is no way the segmentation mask can be accurate. For example, in Figure 5.3, the tail of the airplane in Example 2, and the

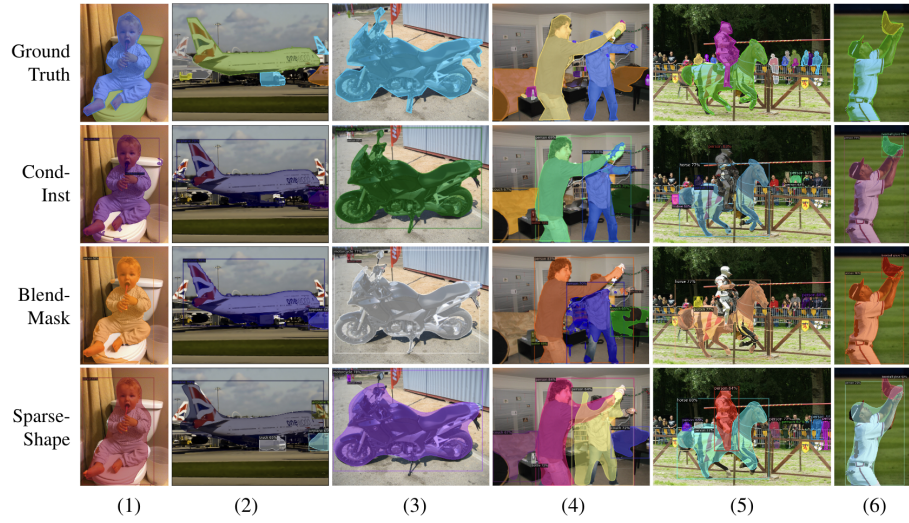


Figure 5.3: Hard cases for SparseShape (produced using the ResNet-50-FPN [16] backbone on the MS COCO *val2017* dataset). Errors are primarily due to over-smoothing of the boundary, which can lead to missing sharp concavities (e.g., gap between legs in Example 1, gap between front wheel and handlebars in Example 3) and narrow parts (e.g., kickstand in Example 3, legs of horse in Example 5). This over-smoothing sometimes results in over-thickening of the object to include neighbouring background (e.g., the toilet seat in Example 1, tail fin of jet in Example 2, the person on the right in Example 4).

person on the right in Example 4 are extended due to an overestimation of the bounding box size. Figure 5.4 shows additional examples of detection errors that make it impossible to produce a correct segmentation.

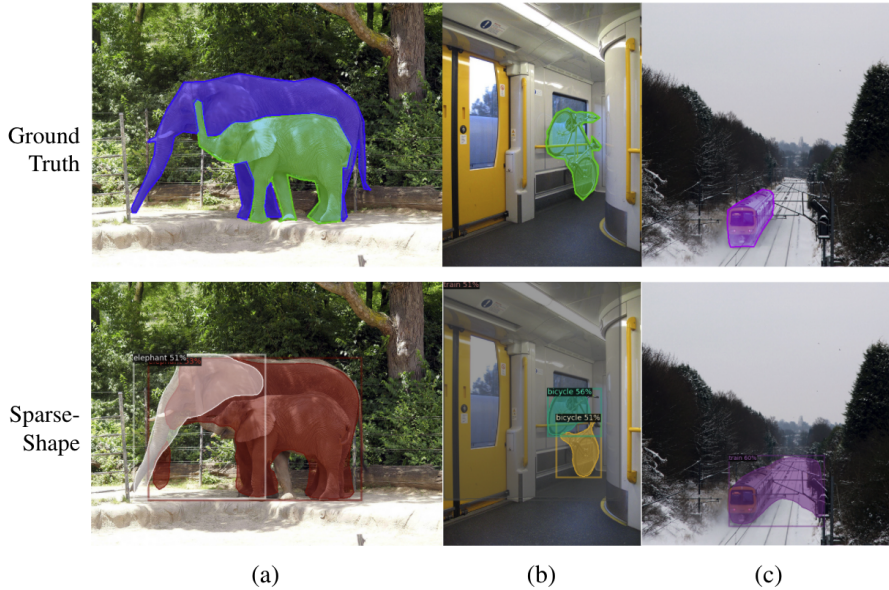


Figure 5.4: SparseShape Detection errors (a) While the baby elephant is not detected, the head and trunk of its mother are detected as a second elephant. (b) The bike is incorrectly identified as two objects. (c) The bounding box of the train is much too large.

The current SparseShape design couples the bounding box with mask estimation, making the performance of mask estimation dependent on the accuracy of the bounding box. One possible way of mitigating this type of error is to decouple the dependency at the mask refinement stage. Specifically, the masks could be resized and repositioned to their actual sizes and locations according to their estimated bounding boxes before entering the mask refinement module. Then the refinement step could be carried out

while allowing the mask to grow or shrink relative to the originally estimated bounding box.

5.2.2 Over-Smoothing

Another source of error for SparseShape is over-smoothing (Figure 5.3), which can result in the filling in of concave regions (e.g., gap between legs in Example 1, gap between front wheel and handlebars in Example 3), truncation of narrow parts (e.g., kickstand in Example 3, legs of horse in Example 5) and assimilation of background (e.g., the toilet seat in Example 1, tail fin of jet in Example 2, the person on the right in Example 4).

One contributor to over-smoothing is our pyramidal architecture from FCOS, which maps large objects to higher pyramid levels with lower spatial resolution. Our mask refinement module uses these lower spatial resolution feature maps, thus limiting the refinement module from recovering sharp features. CondInst and BlendMask also use similar pyramidal architectures, but their final mask is estimated from feature maps with finer spatial resolution (earlier in the feature pyramid).

One might be concerned that our DTM representation could cause over-smoothing, since interior pixels have larger distance values than pixels near the object boundary. However, evidence argues against this explanation. First, we note that reconstruction with ground truth coefficients faithfully recovers fine articulations (Fig. 5.5). Second, when we modify the dictionary to represent the log of the DTM rather than the DTM itself, thus attenuating the larger interior values, we find that this does not improve the articulation of fine detail (Fig. 5.7). Third, over-smoothing is also a problem for MEInst

(Fig. 4.3), which uses PCA reconstruction of binary masks, suggesting that the issue is not specific to sparse coding or the DTM.

Another possible cause of over-smoothing is error in the estimation of the sparse coefficients. We consider three types of coefficient error:

Random error in estimated coefficients Fig. 5.5 shows that as Gaussian noise is added to the coefficients, segmentations become less accurate but not systematically smoother. This suggests that random noise in coefficient values is not the primary cause of over-smoothing.

Systematic error in estimated coefficients Table 4.8 shows that estimated coefficients are systematically less diverse and less sparse than ground truth coefficients. In a new experiment (Fig. 5.6), we show that histogram equalization to match the distribution of estimated coefficients to the distribution of ground-truth coefficients results in slightly more articulated segmentations. For example, the gap between the legs of the bull (row 1) gets more extended, and the gap between the table side wall and leg (row 5) is better articulated. While overall this modification does not quantitatively improve mask AP, it suggests that systematic bias in estimated coefficients is one cause of over-smoothing.

Permutation of estimated coefficients Even if estimated coefficients had the right distribution they might not be assigned to the right sparse shape components. We can evaluate the ability of SparseShape to identify the right components by computing the Spearman rank correlation between the ground truth and estimated rank order of basis components for each

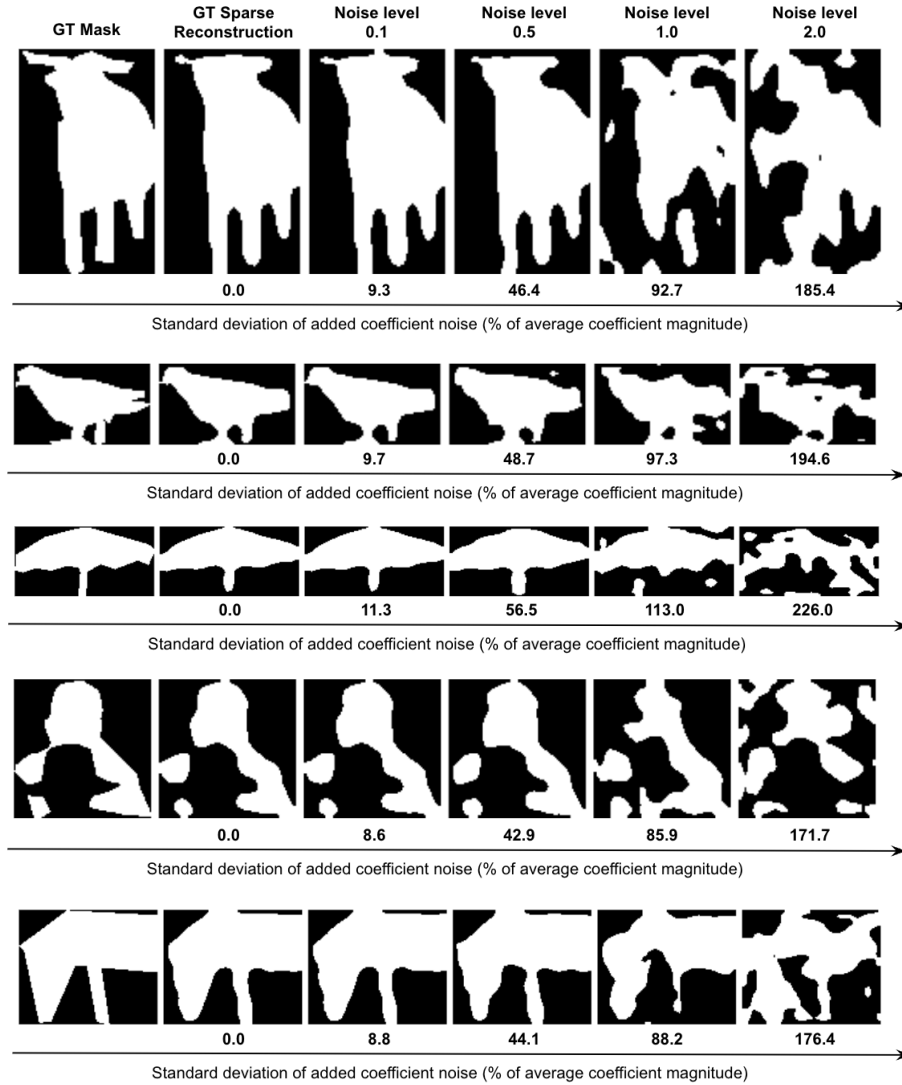


Figure 5.5: Example sparse reconstructions of 5 finely-articulated masks, and the effects of noise in coefficients. 4 levels (of zero-mean Gaussian noise) are applied to the ground truth sparse coefficients (Column 2). The number above each segmentation indicates the standard deviation of the added coefficient noise. The number below each segmentation indicates the standard deviation of the noise as a percentage of the mean coefficient magnitude.

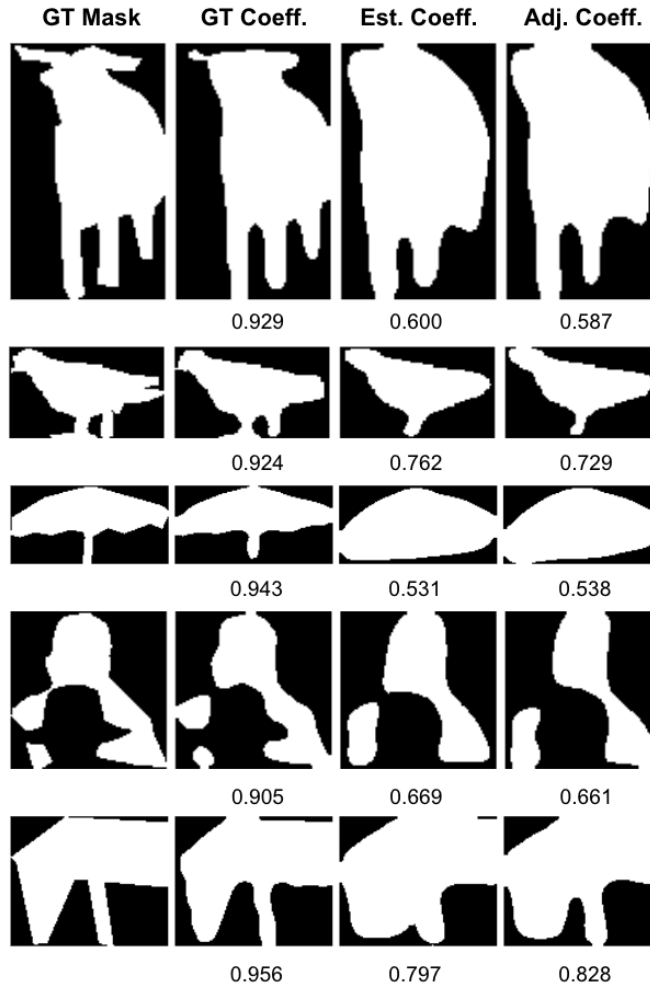


Figure 5.6: Example sparse reconstructions of 5 finely-articulated masks (before mask refinement) , and the effects of applying histogram equalization w.r.t. the MS COCO training set. From left to right: ground truth masks, reconstruction from ground truth sparse coefficients, reconstruction from estimated coefficients, and reconstruction from adjusted coefficients from histogram equalization. The IoU with the ground truth mask is shown below each subfigure.



Figure 5.7: Qualitative comparison of SparseShape using our standard DTM representation and a log DTM representation, on samples from the COCO validation dataset.

shape, by signed coefficient value. Spearman correlation between ground truth and estimated coefficients is 0.54 ± 0.0014 on the COCO validation set, indicating that SparseShape is very good but not perfect at identifying the right sparse components. To assess the impact of these permutation errors, we can reshuffle the estimated coefficients for each object to follow the same signed order as the ground truth coefficients. We find that this results in less accurate and less articulated reconstructions.

In sum, over-smoothing seems to result partly from systematic bias in estimated coefficients. While our refinement module does improve accuracy, it fails to fully eliminate this over-smoothing (Fig. 5.8), in part due to the pyramidal architecture of our backbone, which maps large objects to higher layers that do not provide sufficient resolution to recover these details. Resolving these issues is future research.

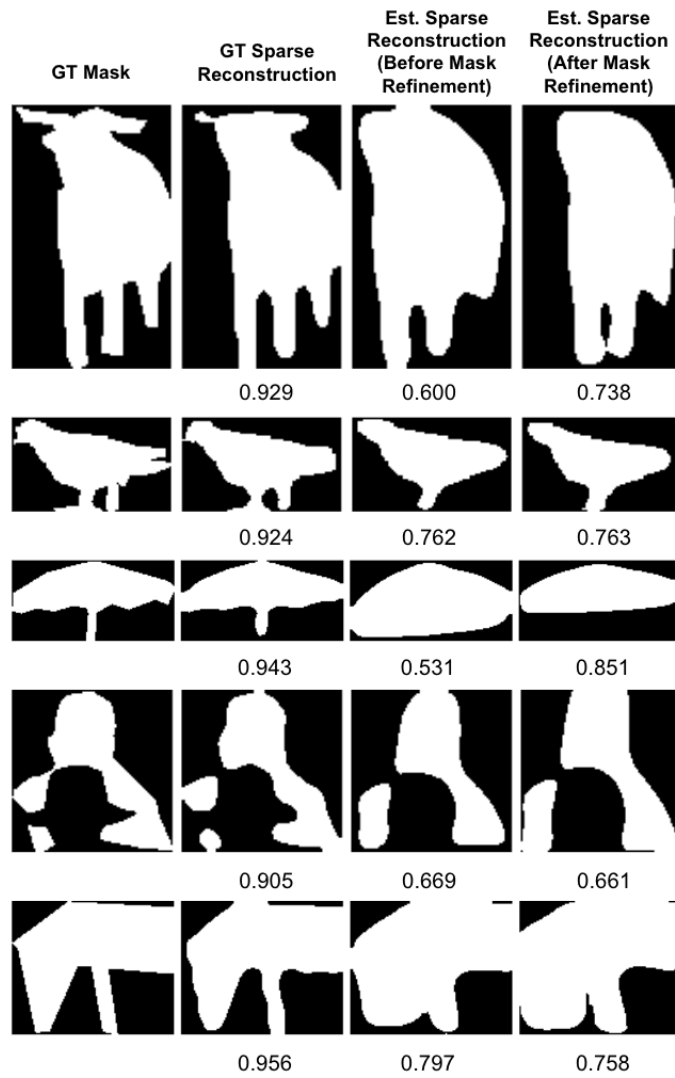


Figure 5.8: COCO validation examples showing the effect of the SparseShape mask refinement module.

5.2.3 Instances with Thin and Elongated Parts

Related to the problem of over-smoothing is the challenge of accurately segmenting thin, elongated parts. While SparseShape can in many cases estimate these thin, elongated parts (Fig. 4.4), it sometimes fails (Figure 5.9). While these do not have a large quantitative impact on mean AP, qualitatively they are serious errors and they would be more heavily penalized with a better segmentation metric.

Although these thin parts can typically be modelled well with our sparse dictionary (given ground truth, sparse reconstructions of the shapes in these images are highly accurate), our coefficient estimation network does not estimate these coefficients accurately enough. The mask refinement module also fails to resolve the problem, because these thin and elongated parts occupy only a small portion of the DTM by area, and therefore contribute only modestly to the IoU-based Dice loss. Finally, the features used for refinement simply do not carry the information of these missing parts due to their limited spatial resolution. Possible ways of mitigating this error are either using a higher resolution feature map for refinement or including a loss function on the mask that penalizes this type of error, e.g. the CM measure [33]. This is also future work.

5.2.4 Incorrect Separation between Objects

Figure 5.3 (Example 4 and 5) show cases where estimated instance segmentations overlap. While these overlapping segmentations make sense if the goal is to recover amodally-completed segmentations [55, 37], this is

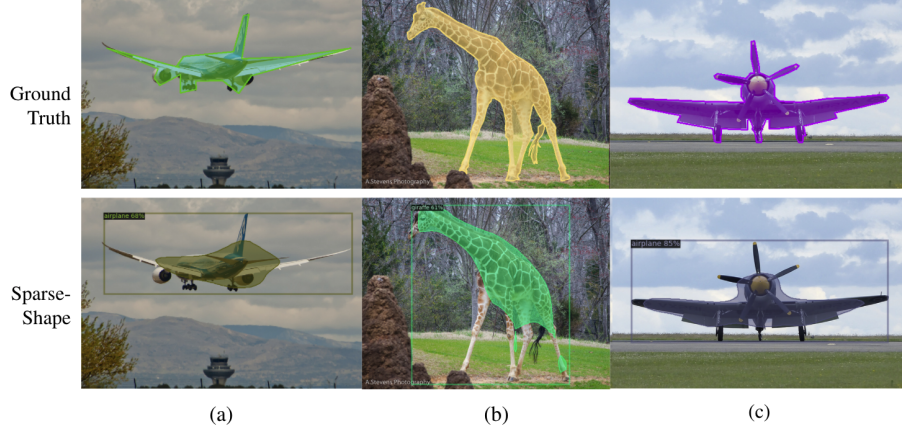


Figure 5.9: Thin, elongated parts sometimes pose a challenge for SparseShape. (a) The body of the airplane is over-smoothed, and the wings are missing. (b) The legs of the giraffe are poorly segmented. (c) The tips of the wing and the propeller are missing.

not the goal as defined MS COCO instance segmentation, and thus these overlapping segmentations lead to reduced mask AP scores.

The instance segmentation task is defined as the recovery of the *visible* portions of objects in the scene. As a result, the distribution of COCO shapes reflects both the statistics of object shape and the statistics of object occlusions. Given that the core principle of SparseShape is to employ a prior model of shape to properly regularize instance segmentation, it may be more natural to apply it to the problem of recovering *complete* shapes, including occluded regions. This is another great opportunity for future work.

6 Conclusion

Neurophysiological studies suggest that neurons in the intermediate visual area V4 of the primate cortex encode a sparse representation of object shape. In contrast, state-of-the-art instance segmentation methods rely on neural networks and formulate it as a pixel-level classification problem without explicit shape modeling. This thesis aims to explore whether sparse shape encoding can yield benefits for instance segmentation. Based on the quantitative and qualitative analysis of the proposed method (SparseShape) on two representative instance segmentation datasets (MS COCO and Cityscapes), it can be concluded that the sparse shape encoding observed in the primate cortex has computational advantages that can benefit instance segmentation neural networks.

This proposed method, termed SparseShape, falls in the class of one-stage instance segmentation network and most previous one-stage frameworks have demonstrated both good performance (inference speed) and segmentation quality. Specifically, I encode 2D object shapes using a Distance Transform Map (DTM) and learn a sparse shape basis for this representation. To make use of this encoding, I design and train an instance segmentation head to estimate the sparse coefficients representing the shape of each object and then recover the estimated shape from the zero-crossing level set of the corresponding DTM. The sparse coefficients and shape basis are jointly learned from shape DTMs before training the neural networks, and the coefficients are used to compute the loss.

I find that the DTM representation improves the segmentation qual-

ity over the traditional mask representation, and sparse coding outperforms PCA encoding method. SparseShape outperforms prior contour-based methods and coefficient-based methods, but fall shorts of some region-based methods when high resolution global semantic maps are jointly used, which potentially improves the mask AP. In addition, I find that SparseShape encoding approach produces fewer topological errors than these region-based methods and exhibits superior generalization performance on the Cityscapes traffic instance segmentation task.

SparseShape encodes diverse instance shapes into a set of basis functions, which effectively and efficiently regularizes the shape due to occlusion and complex appearance in the image (the basis functions are learned from masks without appearance features), and captures the variation of shapes by a regressing a continuous set of coefficients. The appearance features of objects tend to vary from image to image, but their topologies tend to stay the same, which explains why SparseShape generates fewer topological errors than the state-of-the-art region-based methods that tend to rely more on the appearance features. In the meantime, SparseShape exhibits better generalization performance on rare classes also due to the fact that, although the distribution of certain classes are underestimated in the target dataset given unbalanced fine-tuning examples, the dictionary is acquired from a larger and more balance data corpus, stays intact during the whole fine-tuning process, and thus mitigates the unbalanced distribution and improves the ability of representing rare shape classes with limited fine-tuning examples.

To the best of my knowledge, this is the first work that incorporates the

distance transform, a direct and explicit shape representation, into a one-stage instance segmentation framework. The major contributions of this thesis include a distance transform encoding that improves neural network ability to estimate shape by encoding richer shape information at the local pixel level, which encodes each shape as a linear combination of basis shapes, thus simultaneously reducing redundancy. Leveraging on one-stage frameworks, I employ a novel feature fusion strategy to estimate the sparse coefficients of this DTM representation. I also implement a novel shape reconstruction pipeline that uses these coefficients to estimate the distance transform as a weighted combination of sparse basis functions, and identifies the shape as the union of the zero-crossing level set of the DTM, and then refines the result using a light-weight residual network. Mask quality scores are jointly estimated with the shape to not only promote training on hard segmentation examples, but also calibrate detection confidence scores to penalize poor mask quality at inference.

Throughout the experiments, I find a few cases where SparseShape does not work as expected compared to the state-of-the-art methods. These cases include detection errors generated from the detection head of the one-stage framework and get propagated to the final masks, oversmoothing boundaries on shapes that are supposed to have significant concavities, and failures on instances with thin and elongated parts. Therefore, I suggest some future directions in the next section.

7 Future Work

There are a number of opportunities for improving on this work.

7.1 Leverage High Resolution Semantic Features

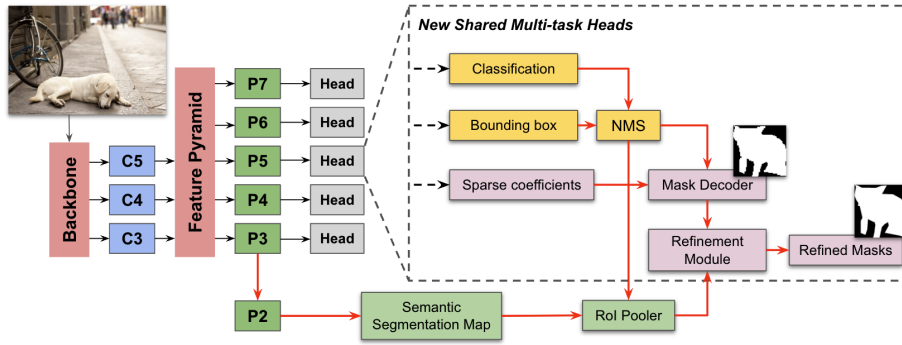


Figure 7.1: A possible instance segmentation head that leverages higher resolution semantic features for mask refinement.

Given the failure modes discussed in the previous sections, a more powerful mask refinement module may be required to mitigate errors coming from thin and elongated parts, over-smoothing problems observed from concave regions and complex shapes missing sharp convexities. Here we propose a possible way of improving the current mask refinement module in Figure 7.1, which can be used as a guide for future research.

The red arrows in Figure 7.1 indicates a series of possible modifications based on the basic segmentation head. First, we add 1 upsampling layer to the $P3$ feature pyramid level to obtain a higher resolution feature map $P2$ ($1/4$ of the input dimension), and process this feature map to output a semantic segmentation map with shape (C, H_{P2}, W_{P2}) , where C is the

number of categories in the dataset ($C = 80$ for MS COCO [28]). Then, in the shared multi-task heads, the classification and bounding box regression outputs will be fed into a non-maximum suppression (NMS) operator to pre-select the top k most promising objects according to their confidence scores (or optionally calibrated by the mask quality score as well). These output objects will be sent to a Mask Decoder and an RoI Pooler in parallel. Next, the Mask Decoder will only decode DTMs and recover their masks for the set of selected objects to reduce computational cost for subsequent modules. The RoI Pooler (e.g. RoIAlign [17]) takes inputs from both the NMS operator and the semantic segmentation map and pools a feature map of the same size for each mask. Finally, the decoded masks and features pooled by the RoI Pooler will be fused and processed to generate refined masks.

This mask refinement module has a greater chance to recover missing shape details mentioned in previous sections with the aid of higher-resolution P2 features.

7.2 Amodal Completion

Humans can often infer information that is not directly visible in an image [39, 5]. A particular example of this is amodal completion, which is the phenomenon of perceiving the complete physical structure or shape of an instance when only a portion of it is visible. Although this powerful ability of human perception has not yet been widely incorporated in deep convolutional networks for computer vision systems, the potential benefits are many. For example, knowing the full extent of motor vehicles in complex

traffic scenes could help detection of risks and near-misses in autonomous driving and traffic surveillance systems. Hence, a robust and reliable instance segmentation system should be able to infer the segmentation masks for occluded portions of the objects as well as their visible parts. Due in part to the limited shape perception capabilities of deep convolutional networks [14], and the lack of proper annotated datasets [55, 37], the problem of amodal instance segmentation has not yet been emphasized in the computer vision community.

In the failure mode section, we discussed the problem of incorrect separation among adjacent objects, for example, Figure 5.3 (example 4 and 5) show cases where estimated instance segmentations overlap. These examples look like partial attempts to perform amodal completion although no explicit amodal annotations are provided during training, which makes sense, since our method embodies a model of global shape. After all, fragmentation resulting from partial occlusion is an artifact of object positioning, not an inherent property of the shape of an object.

We have demonstrated that incorporation of the statistics of shapes through our shape dictionary has the potential to serve as an inductive prior, which can lead to better generalization (Table 4.13) on the Cityscapes dataset [10]. This shape dictionary also has the potential to regularize accidental shapes resulted from object positioning, poses and occlusion, making it easier for the network to estimate amodally-completed shapes based on the visible cues from the modal segmentation.

References

- [1] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2858–2866, 2017.
- [2] Nicholas Baker, Hongjing Lu, Gennady Erlikhman, and Philip J Kellman. Deep convolutional networks do not classify based on global object shape. *PLOS Computational Biology*, 14(12):e1006613, 2018.
- [3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. on Imaging Sciences*, 2:183–202, 2009.
- [4] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. YOLACT: real-time instance segmentation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 9156–9165, 2019.
- [5] Brittany N. Bushnell, Philip J. Harding, Yoshito Kosai, and Anitha Pasupathy. Partial occlusion modulates contour-based shape encoding in primate area V4. *Journal of Neuroscience*, 31(11):4012–4024, 2011.
- [6] E.T. Carlson, R.J. Rasquinha, K. Zhang, and C.E. Connor. A sparse object coding scheme in Area V4. *Current Biology*, 21:288–293, 2011.
- [7] Hao Chen, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan. BlendMask: Top-down meets bottom-up for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8570–8578, 2020.

- [8] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4013–4022, 2018.
- [9] C.E. Connor, S.L. Brincat, and A. Pasupathy. Transformation of shape information in the ventral pathway. *Current Opinion in Neurobiology*, 17(140-147), 2007.
- [10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.
- [11] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. Instance-sensitive fully convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 534–549, 2016.
- [12] Shusil Dangi, Cristian A. Linte, and Ziv Yaniv. A distance map regularized CNN for cardiac cine MR image segmentation. *Medical Physics*, 46(12), 2019.
- [13] J.H Elder. Shape from contour: Computation and representation. *Annual Review of Vision Science*, 4:423–450, 2018.
- [14] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. ImageNet-trained CNNs are

- biased towards texture; increasing shape bias improves accuracy and robustness. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.
- [15] Adam Harley, Konstantinos Derpanis, and Iasonas Kokkinos. Segmentation-aware convolutional networks using local attention masks. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 5048–5057, 2017.
- [16] 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 (CVPR)*, pages 770–778, 2016.
- [17] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.
- [18] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring R-CNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6402–6411, 2019.
- [19] Davood Karimi and Septimiu E. Salcudean. Reducing the hausdorff distance in medical image segmentation with convolutional neural networks. *IEEE Transactions on Medical Imaging*, 39(2):499–513, 2020.
- [20] Hoel Kervadec, Jihene Bouchtiba, Christian Desrosiers, Eric Granger, Jose Dolz, and Ismail Ben Ayed. Boundary loss for highly unbalanced

- p style="text-align: center;">segmentation.
- International Conference on Medical Imaging with Deep Learning*
- , 102:285–296, 2019.
- [21] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: From edges to instances with multicut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7322–7331, 2017.
 - [22] N. Kriegeskorte. Deep neural networks: A new framework for modeling biological vision and brain information processing. *Annual Review of Vision Science*, 1(15), 2015.
 - [23] Jonas Kubilius, Stefania Bracci, and Hans P Op de Beeck. Deep neural networks as a computational model for human shape sensitivity. *PLoS Computational Biology*, 12(4), 2016.
 - [24] Weicheng Kuo, Anelia Angelova, Jitendra Malik, and Tsung-Yi Lin. ShapeMask: Learning to segment novel objects by refining shape priors. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 9206–9215, 2019.
 - [25] Youngwan Lee and Jongyoul Park. Centermask : Real-time anchor-free instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13903–13912, 2020.
 - [26] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detec-

- tion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.
- [27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.
- [28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C Lawrence Zitnick. Microsoft coco: Common objects in context. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.
- [29] Shu Liu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. Sgn: Sequential grouping networks for instance segmentation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 3516–3524, 2017.
- [30] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8759–8768, 2018.
- [31] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 689–696, 2009.
- [32] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image seg-

- mentation. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 565–571, 2016.
- [33] Vida Movahedi and James H. Elder. Design and perceptual validation of performance measures for salient object segmentation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 49–56, 2010.
- [34] Fernando Navarro, Suprosanna Shit, Ivan Ezhov, Johannes Paetzold, Andrei Gafita, Jan C. Peeken, Stephanie E. Combs, and Bjoern H. Menze. Shape-aware complementary-task learning for multi-organ segmentation. *Machine Learning in Medical Imaging*, pages 620–627, 2019.
- [35] A. Pasupathy and C.E. Connor. Shape representation in area V4: Position-specific boundary conformation. *Journal of Neurophysiology*, 86:2505–2519, 2001.
- [36] Sida Peng, Wen Jiang, Huaijin Pi, Xiuli Li, Hujun Bao, and Xiaowei Zhou. Deep snake for real-time instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8530–8539, 2020.
- [37] Lu Qi, Li Jiang, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Amodal instance segmentation with kins dataset. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3009–3018, 2019.
- [38] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal net-

- works. *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 1:91–99, 2015.
- [39] Ronald A. Rensinka and James T. Ennsb. Early completion of occluded objects. *Vision Research*, 38:489–2505, 1998.
- [40] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, 2019.
- [41] Samuel Ritter, David GT Barrett, Adam Santoro, and Matt M Botvinick. Cognitive psychology for deep neural networks: A shape bias case study. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2940–2949, 2017.
- [42] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 282–298, 2020.
- [43] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 9626–9635, 2019.
- [44] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. SOLOv2: Dynamic, faster and stronger. (1487):17721–17732, 2020.

- [45] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. SOLO: segmenting objects by locations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 649–665, 2020.
- [46] Yuqing Wang, Zhaoliang Xu, Hao Shen, Baoshen Cheng, and Lirong Yang. CenterMask: Single shot instance segmentation with point representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9313–9321, 2020.
- [47] Enze Xie, Peize Sun, Xiaogang Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. PolarMask: Single shot instance segmentation with polar representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12190–12199, 2020.
- [48] Wenqiang Xu, Haiyang Wang, Fubo Qi, and Cewu Lu. Explicit shape encoding for real-time instance segmentation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 5167–5176, 2019.
- [49] Yuan Xue, Hui Tang, Zhi Qiao, Guanzhong Gong, Yong Yin, Zhen Qian, Chao Huang, Wei Fan, and Xiaolei Huang. Shape-aware organ segmentation by predicting signed distance maps. volume 34, pages 12565–12572, 2020.
- [50] Ze Yang, Yinghao Xu, Han Xue, Zheng Zhang, Raquel Urtasun, Liwei Wang, Stephen Lin, and Han Hu. Dense RepPoints: Representing

- visual objects with dense point sets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [51] Rufeng Zhang, Zhi Tian, Chunhua Shen, Mingyu You, and Youliang Yan. Mask encoding for single shot instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10223–10232, 2020.
- [52] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9756–9765, 2020.
- [53] Xingyi Zhou, Dequan Wang, and Philipp Krahenbuhl. Objects as points. *arXiv 1904.07850*, 2019.
- [54] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 850–859, 2019.
- [55] Yan Zhu, Yuandong Tian, Dimitris Mexatas, and Piotr Dollár. Semantic amodal segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3001–3009, 2017.