Rethinking Semantic Segmentation: A Prototype View

Tianfei Zhou\textsuperscript{1}, Wenguan Wang\textsuperscript{2,1*}, Ender Konukoglu\textsuperscript{1}, Luc Van Gool\textsuperscript{1}
\textsuperscript{1} Computer Vision Lab, ETH Zurich \textsuperscript{2} ReLER, AAII, University of Technology Sydney

Abstract

Prevalent semantic segmentation solutions, despite their different network designs (FCN based or attention based) and mask decoding strategies (parametric softmax based or pixel-query based), can be placed in one category, by considering the softmax weights or query vectors as learnable class prototypes. In light of this prototype view, this study uncovers several limitations of such parametric segmentation regime, and proposes a nonparametric alternative based on non-learnable prototypes. Instead of prior methods learning a single weight/query vector for each class in a fully parametric manner, our model represents each class as a set of non-learnable prototypes, relying solely on the mean features of several training pixels within that class. The dense prediction is thus achieved by nonparametric nearest prototype retrieving. This allows our model to directly shape the pixel embedding space, by optimizing the arrangement between embedded pixels and anchored prototypes. It is able to handle arbitrary number of classes with a constant amount of learnable parameters. We empirically show that, with FCN based and attention based segmentation models (i.e., HR-Net, Swin, SegFormer) and backbones (i.e., ResNet, HRNet, Swin, MiT), our nonparametric framework yields compelling results over several datasets (i.e., ADE20K, Cityscapes, COCO-Stuff), and performs well in the large-vocabulary situation. We expect this work will provoke a rethink of the current de facto semantic segmentation model design.

1. Introduction

With the renaissance of connectionism, rapid progress has been made in semantic segmentation. Till now, most of state-of-the-art segmentation models [15, 34, 49, 135] were built upon Fully Convolutional Networks (FCNs) [79]. Despite their diversified model designs and impressive results, existing FCN based methods commonly apply parametric softmax (\(\odot\)) over pixel-wise features for dense prediction (Fig. 1(a)). Very recently, the vast success of Transformer [105] stimulates the emergence of attention based segmentation solutions. Many of these ‘non-FCN’ models, like [118, 139], directly follow the standard mask decoding regime, i.e., estimate softmax distributions over dense visual embeddings (extracted from patch token sequences). Interestingly, the others [20, 100] follow the good practice of Transformer in other fields [11, 82, 113] and adopt a pixel-query strategy (Fig. 1(b)): utilize a set of learnable vectors (\(\odot\)) to query the dense embeddings for mask prediction. They speculate the learned query vectors can capture class-wise properties, however, lacking in-depth analysis.

Noticing there exist two different mask decoding strategies, the following questions naturally arise: ① What are the relation and difference between them? and ② If the learnable query vectors indeed implicitly capture some intrinsic properties of data, is there any better way to achieve this?

Tackling these two issues can provide insights into modern segmentation model design, and motivate us to rethink the task from a prototype view. The idea of prototype based classification [31] is classical and intuitive (which can date back to the nearest neighbors algorithm [23] and find evi-
dence in cognitive science [60, 91]): data samples are classified based on their proximity to representative prototypes of classes. With this perspective, in §2, we first answer question 1 by pointing out most modern segmentation methods, from softmax based to pixel-query based, from FCN based to attention based, fall into one grand category: parametric models based on learnable prototypes. Consider a segmentation task with \( C \) semantic classes. Most existing efforts seek to directly learn \( C \) class-wise prototypes – softmax weights or query vectors – for parametric, pixel-wise classification. Hence question 2 becomes more fundamental: 2 What are the limitations of this learnable prototype based parametric paradigm? and 3 How to address these limitations?

Driven by question 3, in §3, we find there are three critical limitations: First, usually only one single prototype is learned per class, insufficient to describe rich intra-class variance. The prototypes are simply learned in a fully parametric manner, without considering their representative ability. Second, to map a \( H \times W \times D \) image feature tensor into a \( H \times W \times C \) semantic mask, at least \( D \times C \) parameters are needed for prototype learning. This hurts generalizability [115], especially in the large-vocabulary case; for instance, if there are 800 classes and \( D = 512 \), we need 0.4M learnable prototype parameters alone. Third, with the cross-entropy loss, only the relative relations between intra-class and inter-class distances are optimized [89, 111, 134]; the actual distances between pixels and prototypes, i.e., intra-class compactness, are ignored.

As a response to question 3, in §3, we develop a nonparametric segmentation framework, based on non-learnable prototypes. Specifically, building upon the ideas of prototype learning [116, 133] and metric learning [40, 64], it is fully aware of the limitations of its parametric counterpart. Independent of specific backbone architectures (FCN based or attention based), our method is general and brings insights into segmentation model design and training. For model design, our method explicitly sets sub-class centers, in the pixel embedding space, as the prototypes. Each pixel data is predicted to be in the same class as the nearest prototype, without relying on extra learnable parameters. For training, as the prototypes are representative of the dataset, we can directly pose known inductive biases (e.g., intra-class compactness, inter-class separation) as extra optimization criteria and efficiently shape the whole embedding space, instead of optimizing the prediction accuracy only. Our model has three appealing advantages: First, each class is abstracted by a set of prototypes, well capturing class-wise characteristics and intra-class variance. With the clear meaning of the prototypes, the interpretability is also enhanced – the prediction of each pixel can be intuitively understood as the reference of its closest class center in the embedding space [3, 7]. Second, due to the nonparametric nature, the generalizability is improved. Large-vocabulary semantic segmentation can also be handled efficiently, as the amount of learnable prototype parameters is no longer constrained to the number of classes (i.e., \( 0 \) vs \( D \times C \)). Third, via prototype-anchored metric learning, the pixel embedding space is shaped as well-structured, benefiting segmentation prediction eventually.

By answering questions 1-3, we formalize prior methods within a learnable prototype based, parametric framework, and link this field to prototype learning and metric learning. We provide literature review and related discussions in §4.

In §5.2, we show our method achieves impressive results over famous datasets (i.e., ADE20K [140], Cityscapes [22], COCO-Stuff [10]) with top-leading FCN based and attention based segmentation models (i.e., HRNet [108], Swin [78], SegFormer [118]) and backbones (i.e., ResNet [45], HRNet [108], Swin [78], MiT [118]). Compared with the parametric counterparts, our method does not cause any extra computational overhead during testing while reduces the amount of learnable parameters. In §5.3, we demonstrate our method consistently performs well when increasing the number of semantic classes from 150 to 847. Accompanied with a set of ablative studies in §5.4, our extensive experiments verify the power of our idea and the efficacy of our algorithm.

Finally, we draw conclusions in §6. This work is expected to open a new venue for future research in this field.

### 2. Existing Semantic Segmentation Models as Parametric Prototype Learning

Next we first formalize the existing two mask decoding strategies mentioned in §1, and then answer question 1 from a unified view of parametric prototype learning.

**Parametric Softmax Projection.** Almost all FCN-like and many attention-based segmentation models adopt this strategy. Their models comprise two learnable parts: i) an encoder \( φ \) for dense visual feature extraction, and ii) a classifier \( ρ \) (i.e., projection head) that projects pixel features into the semantic label space. For each pixel example \( i \), its embedding \( \mathbf{i} \in \mathbb{R}^D \), extracted from \( φ \), is fed into \( ρ \) for \( C \)-way classification:

\[
p(c|i) = \frac{\exp(\mathbf{w}_c^T \mathbf{i})}{\sum_{c'=1}^{C} \exp(\mathbf{w}_{c'}^T \mathbf{i})},
\]

where \( p(c|i) \in [0, 1] \) is the probability that \( i \) being assigned to class \( c \). \( ρ \) is a pixel-wise linear layer, parameterized by \( W = [\mathbf{w}_1, \ldots, \mathbf{w}_C] \in \mathbb{R}^{C \times D} \); \( \mathbf{w}_c \in \mathbb{R}^D \) is a learnable projection vector for \( c \)-th class; the bias term is omitted for brevity.

**Parametric Pixel-Query.** A few attention-based segmentation networks [118, 139] work in a more ‘Transformer-like’ manner: given the pixel embedding \( \mathbf{i} \in \mathbb{R}^D \), a set of \( C \) query vectors, i.e., \( E = [e_1, \ldots, e_C] \in \mathbb{R}^{C \times D} \), are learned to generate a probability distribution over the \( C \) classes:

\[
p(c|i) = \frac{\exp(e_c \times \mathbf{i})}{\sum_{c'=1}^{C} \exp(e_{c'} \times \mathbf{i})},
\]

where ‘\( \times \)’ is inner product between \( ℓ_2 \)-normalized inputs.
Prototype-based Classification. Prototype-based classification [31, 33] has been studied for a long time, dating back to the nearest neighbors algorithm [23] in machine learning and prototype theory [60, 91] in cognitive science. Its prevalence stems from its intuitive idea: represent classes by prototypes, and refer to prototypes for classification. Let \( \{p_m\}_{m=1}^{M} \) be a set of prototypes that are representative of their corresponding classes \( \{c_p, \in \{1, \cdots, C\}\}_{m=1}^{M} \). For a data sample \( i \), prediction is made by comparing \( i \) with \( \{p_m\}_{m=1}^{M} \), and taking the class of the winning prototype as response:

\[
\hat{c}_i = c_{p_m^*}, \quad \text{with} \quad m^* = \arg\min_m \{i, p_m\}_{m=1}^{M}, \tag{3}
\]

where \( i \) and \( \{p_m\}_{m=1}^{M} \) are embeddings of the data sample and prototypes in a feature space, and \( \langle \cdot, \cdot \rangle \) stands for the distance measure, which is typically set as \( \ell_2 \) distance (i.e., \( ||i - p_m|| \)) [123], yet other proximities can be applied.

Further, Eqs. 1-2 can be formulated in a unified form:

\[
p(c| i) = \frac{\exp(-\langle i, g_c \rangle)}{\sum_{c'}^C \exp(-\langle i, g_{c'} \rangle)}, \tag{4}
\]

where \( g_c \in \mathbb{R}^D \) can be either \( w_c \) in Eq. 1 or \( e_c \) in Eq. 2.

With Eqs. 3-4, we are ready to answer questions \( 1. \) Both the two types of methods are based on learnable prototypes; they are parametric models in the sense that they learn one prototype \( g_c \), i.e., linear weight \( w_c \) or query vector \( e_c \), for each class \( c \) (i.e., \( M = C \)). Thus one can consider softmax projection based methods ‘secretly’ learn the query vectors. As for the difference, in addition to different distance measures (i.e., inner product vs cosine similarity), pixel-query based methods [118, 139] can feed the queries into cross-attention decoder layers for cross-context class exchanging, rather than softmax projection based counterparts only leveraging the learned class weights within the softmax layer.

With the unified view of parametric prototype learning, a few intrinsic yet long ignored issues in this field unfold:

First, prototype selection [36] is a vital aspect in the design of a prototype based learner – prototypes should be typical for their classes. Nevertheless, existing semantic segmentation algorithms often describe each class by only one prototype, bearing no intra-class variation. Moreover, the prototypes are directly learned in a fully parametric manner, without accounting for their representative ability.

Second, the amount of the learnable prototype parameters, i.e., \( \{g_c \in \mathbb{R}^D\}_{c=1}^{C} \), grows with the number of classes. This may hinder the scalability, especially when a large number of classes are present. For example, if there are 800 classes and the pixel feature dimensionality is 512, at least 0.4M parameters are needed for prototype learning alone, making large-vocabulary segmentation a hard task. Moreover, if we want to represent each class by ten prototypes, instead of only one, we need to learn 4M prototype parameters.

Third, Eq. 3 intuitively shows that prototype based learners make metric comparisons of data [8]. However, existing algorithms often supervise dense segmentation representation by directly optimizing the accuracy of pixel-wise prediction (e.g., cross-entropy loss), ignoring known inductive biases [83, 84], e.g., intra-class compactness, about the feature distribution. This will hinder the discrimination potential of the learned segmentation features, as suggested by many literature in representation learning [76, 95, 114].

After tackling question \( 1. \) in the next section we will detail our non-learnable prototype based nonparametric segmentation method, which serves as a solid response to question \( 1. \).

3. Non-Learnable Prototype based Nonparametric Semantic Segmentation

We build a nonparametric segmentation framework that conducts dense prediction by a set of non-learnable class prototypes, and directly supervises the pixel embedding space via a prototype-anchored metric learning scheme (Fig. 2).

Non-Learnable Prototype based Pixel Classification. As normal, an encoder network (FCN based or attention based), i.e., \( \phi \), is first adopted to map the input image \( I \in \mathbb{R}^{H \times W \times 3} \), to a 3D feature tensor \( \mathbf{F} \in \mathbb{R}^{H \times W \times D} \). For pixel-wise \( C \) -way classification, rather than prior semantic segmentation models that automatically learn \( C \) class weights \( \{w_c \in \mathbb{R}^D\}_{c=1}^{C} \) (cf. Eq. 1) or \( C \) queries vectors \( \{e_c \in \mathbb{R}^D\}_{c=1}^{C} \) (cf. Eq. 2), we refer to a group of \( CK \) non-learnable prototypes, i.e., \( \{p_{c,k} \in \mathbb{R}^D\}_{c,k=1}^{C,K} \), which are based solely on class data subcenters. More specifically, each class \( c \in \{1, \cdots, C\} \) is represented by a total of \( K \) prototypes \( \{p_{c,k}\}_{k=1}^{K} \), and prototype \( p_{c,k} \) is determined as the center of \( k \)-th sub-cluster of training pixel samples belonging to class \( c \) in the embedding space \( \phi \). In this way, the prototypes can comprehen-
sively capture characteristic properties of the corresponding classes, without introducing extra learnable parameters outside \( \phi \). Analogous to Eq. 3, the category prediction of each pixel \( i \in I \) is achieved by a winner-take-all classification:

\[
\hat{c}_i = c^*, \quad \text{with} \quad (c^*, k^*) = \arg\min_{(c, k)} \{i, p, c, k\} \in c, K, \quad (5)
\]

where \( i \in \mathbb{R}^D \) stands for the \( \ell_2 \)-normalized embedding of pixel \( i \), i.e., \( i \in I \), and the distance measure \( \langle \cdot, \cdot \rangle \) is defined as the negative cosine similarity, i.e., \( \langle i, p \rangle = -i^\top p \).

With this exemplar-based reasoning mode, we first define the probability distribution of pixel \( i \) over the \( C \) classes:

\[
p(c|i) = \frac{\exp(-s_{i,c})}{\sum_{c'=1}^C \exp(-s_{i,c'})}, \quad \text{with} \quad s_{i,c} = \min\{i, p, c, k\} \in K, \quad (6)
\]

where the pixel-class distance \( s_{i,c} \in [-1, 1] \) is computed as the distance to the closest prototype of class \( c \). Given the groundtruth class of each pixel \( i \), i.e., \( c_i \in \{1, \cdots, C\} \), the cross-entropy loss can be therefore used for training:

\[
L_{CE} = -\log p(c_i|i) = -\log \frac{\exp(-s_{i,c_i})}{\sum_{c'=1}^C \exp(-s_{i,c'})} \quad (7)
\]

In our case, Eq. 7 can be viewed as pushing pixel \( i \) closer to the nearest prototype of its corresponding class, i.e., \( c_i \), and further from other close prototypes of irrelevant classes, i.e., \( c' \neq c_i \). However, only adopting such training objective is not enough, due to two reasons. First, Eq. 7 only considers pixel-class distances, e.g., \( s_{i,c} \), without addressing within-class pixel-prototype relations, e.g., \( \langle i, p_c, k \rangle \). For example, for discriminative representation learning, pixel \( i \) is expected to be pushed further close to a certain prototype (i.e., a particularly suitable pattern) of class \( c_i \) and, distant from other prototypes (i.e., other irrelevant but within-class patterns) of class \( c_i \). Eq. 7 cannot capture this nature. Second, as the pixel-class distances are normalized across all classes (cf. Eq. 6), Eq. 7 only optimizes the relative relations between intra-class (i.e., \( s_{i,c} \)) and inter-class (i.e., \( s_{i,c'} \)) distances, instead of directly regularizing the cosine distances between pixels and classes. For example, when the intra-class distance \( s_{i,c} \) of pixel \( i \) is relatively smaller than other inter-class distances \( s_{i,c'} \), the penalty from Eq. 7 will be small, but the intra-class distance \( s_{i,c} \) might still be large [39, 134]. Next we first elaborate on our within-class online clustering strategy and then detail our two extra training objectives which rely on prototype assignments (i.e., clustering results) and address the above two issues respectively.

**Within-Class Online Clustering.** We approach online clustering for prototype selection and assignment: pixel samples within the same class are assigned to the prototypes belonging to that class, and the prototypes are then updated according to the assignments. Clustering imposes a natural bottleneck [55] that forces the model to discover intra-class discriminative patterns yet discard instance-specific details. Thus the prototypes, selected as the sub-cluster centers, are typical of the corresponding classes. Conducting clustering online makes our method scalable to large amounts of data, instead of offline clustering requiring multiple passes over the entire dataset for feature computation [13].

Formally, given pixels \( I^c = \{i_n\}_{n=1}^N \) in a training batch that belong to class \( c \) (i.e., \( c_i = c \)), our goal is to map the pixels \( I^c \) to the \( K \) prototypes \( \{p_{c,k}\}_{k=1}^K \) of class \( c \). We denote this pixel-to-prototype mapping as \( L^c = \{l_{i_n, k}\}_{n=1}^N \in [0, 1]^{K \times N} \), where \( l_{i_n, k} = \{l_{i_n, k}\}_{k=1}^K \in \{0, 1\}^K \) is the one-hot assignment vector of pixel \( i_n \) over the \( K \) prototypes. The optimization of \( L^c \) is achieved by maximizing the similarity between pixel embeddings, i.e., \( X^c = \{x_{n, k}\}_{n=1}^N \in \mathbb{R}^{D \times N} \), and the prototypes, i.e., \( P^c = \{p_{c,k}\}_{k=1}^K \in \mathbb{R}^{D \times K} \):

\[
\max_{L^c} \text{Tr}(L^c P^c X^c),
\]

s.t. \( L^c \in \{0, 1\}^{K \times N}, L^c 1^K = 1^N, L^c 1^N = N 1^K \), \quad (8)

where \( 1^K \) denotes the vector of all ones of \( K \) dimensions. The unique assignment constraint, i.e., \( L^c 1^K = 1^N \), ensures that each pixel is assigned to one and only one prototype. The equipartition constraint, i.e., \( L^c 1^N = N 1^K \), enforces that on average each prototype is selected at least \( N/K \) times in the batch [13]. This prevents the trivial solution: all pixel samples are assigned to a single prototype, and eventually benefits the representative ability of the prototypes. To solve Eq. 8, one can relax \( L^c \) to be an element of the transportation polytope [2, 24]:

\[
\max_{L^c} \text{Tr}(L^c P^c X) + \kappa h(L^c),
\]

s.t. \( L^c \in \mathbb{R}^{K \times N}, L^c 1^K = 1^N, L^c 1^N = N 1^K \), \quad (9)

where \( h(L^c) = \sum_{n,k} -l_{n,k} \log l_{n,k} \) is an entropy, and \( \kappa > 0 \) is a parameter that controls the smoothness of distribution. With the soft assignment relaxation and the extra regularization term \( h(L^c) \), the solver of Eq. 9 can be given as [24]:

\[
L^c = \text{diag}(u) \exp \left( \frac{P^c X^c}{\kappa} \right) \text{diag}(v),
\]

where \( u \in \mathbb{R}^K \) and \( v \in \mathbb{R}^N \) are renormalization vectors, computed by few steps of Sinkhorn-Knopp iteration [24]. Our online clustering is highly efficient on GPU, as it only involves a couple of matrix multiplications; in practice, clustering 10K pixels into 10 prototypes takes only 2.5 ms.

**Pixel-Prototype Contrastive Learning.** With the assignment probability matrix \( L^c = \{l_{n,k}\}_{n=1}^N \in [0, 1]^{K \times N} \), we online group the training pixels \( I^c = \{i_n\}_{n=1}^N \) into \( K \) prototypes \( \{p_{c,k}\}_{k=1}^K \) within class \( c \). After all the samples in current batch are processed, each pixel \( i \) is assigned to \( k_i \)-th prototype of class \( c_i \), where \( k_i = \arg\max_k \{l_{i,k}\}_{k=1}^K \) and \( l_{i,k} \in l_i \). It is natural to derive a training objective for prototype assignment prediction, i.e., maximize the prototype assignment posterior probability. This can be viewed as a pixel-prototype contrastive learning strategy, and addresses
the first limitation of Eq. 7:
\[
\mathcal{L}_{	ext{PPC}} = -\log \frac{\exp (i^\top p_{i,c,k}/\tau)}{\sum_{p \in \mathcal{P}} \exp (i^\top p/\tau)},
\]
where \(\mathcal{P} = \{p_{c,k}\}_{c,k=1}^{C,K}\) and the temperature \(\tau\) controls the concentration level of representations. Intuitively, Eq. 11 enforces each pixel embedding \(i\) to be similar with its assigned (‘positive’) prototype \(p_{i,c,k}\), and dissimilar with other \(C(K-1)\) irrelevant (‘negative’) prototypes \(\mathcal{P}^\sim\). Compared with prior pixel-wise metric learning based segmentation models [111], which consume numerous negative pixel samples, our method only needs \(C(K-1)\) prototypes for pixel-prototype contrast computation, neither causing large memory cost nor requiring heavy pixel pair-wise comparison.

**Pixel-Prototype Distance Optimization.** Building upon the relative comparison over pixel-class/prototype distances, Eq. 7 and Eq. 11 inspire inter-class/-cluster discriminativeness, but less consider reducing the intra-cluster variation, *i.e.*, making pixel features of the same prototype compact. Thus a compactness-aware loss is used for further regularizing representations by directly minimizing the distance between each embedded pixel and its assigned prototype:
\[
\mathcal{L}_{\text{PPD}} = (1 - i^\top p_{i,c,k})^2.
\]
Note that both \(i\) and \(p_{i,c,k}\) are \(\ell_2\)-normalized. This training objective minimizes intra-cluster variations while maintaining separation between features with different prototype assignments, making our model more robust against outliers.

**Network Learning and Prototype Update.** Our model is a nonparametric approach that learns semantic segmentation by directly optimizing the pixel embedding space \(\phi\). It is called nonparametric because it constructs prototype hypotheses directly from the training pixel samples themselves. Thus the parameters of the feature extractor \(\phi\) are learned through stochastic gradient descent, by minimizing the combinatorial loss over all the training pixel samples:
\[
\mathcal{L}_{\text{SEG}} = \mathcal{L}_{\text{CE}} + \lambda_1 \mathcal{L}_{\text{PPC}} + \lambda_2 \mathcal{L}_{\text{PPD}}.
\]
Meanwhile, the non-learnable prototypes \(\{p_{c,k}\}_{c,k=1}^{C,K}\) are not learned by stochastic gradient descent, but are computed as the centers of the corresponding embedded pixel samples. To do so, we let the prototypes evolve continuously by accounting for the online clustering results. Particularly, after each training iteration, each prototype is updated as:
\[
p_{c,k} \leftarrow \mu p_{c,k} + (1 - \mu)\bar{i}_{c,k},
\]
where \(\mu \in [0, 1]\) is a momentum coefficient, and \(\bar{i}_{c,k}\) indicates the \(\ell_2\)-normalized, mean vector of the embedded training pixels, which are assigned to prototype \(p_{c,k}\) by online clustering. With the clear meaning of the prototypes, our segmentation procedure can be intuitively understood as retrieving the most similar prototypes (sub-class centers). Fig. 3 provides prototype retrieval results for person and car with \(K = 3\) prototypes for each. The prototypes are associated with different colors (*i.e.*, red, green, and blue). For each pixel, its distance to the closest prototype is visualized using the corresponding prototype color. As can be seen, the prototypes well correspond to meaningful patterns within classes, validating their representativeness.

**4. Related Work**

In this section, we review representative work in semantic segmentation, prototype learning and metric learning.

**Semantic Segmentation.** Recent years have witnessed remarkable progress in semantic segmentation, due to the fast evolution of backbone architectures—from CNN-based (*e.g.*, VGG [97], ResNet [45]) to Transformer-like [105] (*e.g.*, ViT [30], Swin [78]), and segmentation models—from FCNs [79] to attention networks (*e.g.*, SegFormer [118]). Specifically, FCN [79] is a milestone; it learns dense prediction efficiently. Since it was proposed, numerous efforts have been devoted to improving FCN, by, for example, enlarging the receptive field [15, 16, 25, 124, 128, 135]; strengthening context cues [4, 43, 47, 48, 56, 57, 70, 75, 77, 81, 90, 126, 128, 129, 132, 138, 141]; leveraging boundary information [6, 14, 27, 66, 127, 131, 137]; incorporating neural attention [34, 41, 42, 49, 50, 63, 68, 101, 110, 112, 136]; or automating network engineering [18, 69, 72, 85]. Lately, Transformer-based solutions [20, 100, 118, 139] attained growing attention; enjoying the flexibility in long-range dependency modeling, fully attentive solutions yield impressive results.

Different from current approaches that are typically built upon learnable prototypes, in pre-deep era, many segmentation systems are nonparametric [32, 73, 74, 80, 102, 103]. By absorbing their case-based reasoning ideas, we build a nonparametric segmentation network, which explicitly derives prototypes from sample clusters and hence directly optimizes the embedding space with distance metric constraints. In [62, 111], while cluster-/pixel-level metric loss is adopted to regularize representation, the pixel class is still inferred via parametric softmax [53] purely relies on class embeddings, which, however, are fully trainable. Thus [53, 62, 111] are all parametric methods. As far as we know, [52] is the only non-learnable prototype, deep learning based semantic segmentation model. But [52] treats image regions as prototypes, incurring huge memory and computational demand. Besides, [52] only considers the relative difference between inter-and intra-class sample-prototype distances like the parametric counterparts. Our method is more principled with fewer heuristic designs. Unlike [52], we rep-
resent prototypes as sub-cluster centers and obtain online assignments, allowing our method to scale gracefully to any dataset size. We encourage a sparse distance distribution with compactness-awareness, reinforcing the embedding discrimination. With a broader view, a few embedding based instance segmentation approaches \[26, 86\] can be viewed as nonparametric, i.e., treat instance centroids as prototypes.

**Prototype Learning.** Cognitive psychological studies evidence that people use past cases as models when learning to solve problems\[1, 87, 125\]. Among various machine learning algorithms, ranging from classical statistics based methods to Support Vector Machine to Multilayer Perceptrons\[9, 31, 33, 96\], prototype based classification gains particular interest, due to its exemplar-driven nature and intuitive interpretation: observations are directly compared with representative examples. Based on the nearest neighbors rule – the earliest prototype learning method\[23\], many famous, nonparametric classifiers are proposed\[36\], such as Learning Vector Quantization (LVQ)\[61\], generalized LVQ\[94\], and Neighborhood Component Analysis\[37, 93\]. There has been a recent surge of interest to integrate deep learning into prototype learning, showing good potential in few-shot\[98\], zero-shot\[54\], and unsupervised learning\[116, 120\], as well as supervised classification\[38, 83, 115, 123\] and interpretable networks\[65\]. Remarkably, as many few-shot segmentation models can be viewed as prototype-based networks\[29, 106, 109\], our work sheds light on the possibility of closer collaboration between the two segmentation fields.

**Metric Learning.** The selection of proper distance measure impacts the success of prototype based learners\[8\]; metric learning and prototype learning are naturally related. As the literature on metric learning is vast\[58\], only the most relevant ones are discussed. The goal of metric learning is to learn a distance metric/embedding such that similar samples are pulled together and dissimilar samples are pushed away. It has shown a significant benefit by learning deep representation using metric loss functions (e.g., contrastive loss\[40\], triplet loss\[95\], n-pair loss\[99\]) for applications (e.g., image retrieval\[107\], face recognition\[95\]). Recently, metric learning showed good potential in unsupervised representation learning. Specifically, many instance-based approaches use the contrastive loss\[39, 88\] to explicitly compare pairs of image representations, so as to push away features from different images while pulling together those from transformations of the same image\[17, 19, 44, 46, 88\]. Since computing all the pairwise comparisons on a large dataset is challenging, some clustering-based methods turn to discriminate between groups of images with similar features instead of individual images\[2, 5, 12, 13, 64, 104, 119, 121, 122\]. Our prototype-anchored metric learning strategy shares a similar spirit of posing metric constraints over prototype (cluster) assignments, but it is to reshape the pixel segmentation embedding space with explicit supervision.

5. Experiment

5.1. Experimental Setup

**Datasets.** Our experiments are conducted on three datasets:
- **ADE20K**\[140\] is a large-scale scene parsing benchmark that covers 150 stuff/object categories. The dataset is divided into 20k/2k/3k images for train/val/test.
- **Cityscapes**\[22\] has 5k finely annotated urban scene images, with 2,975/500/1,524 for train/val/test. The segmentation performance is evaluated over 19 challenging categories, such as rider, bicycle, and traffic light.
- **COCO-Stuff**\[10\] has 10k images gathered from COCO\[71\], with 9k and 1k for train and test, respectively. There are 172 semantic categories in total, including 80 objects, 91 stuffs and 1 unlabeled.

**Training.** Our method is implemented on MMSegmentation\[21\], following default training settings. In particular, all backbones are initialized using corresponding weights pre-trained on ImageNet-1K\[92\], while remaining layers are randomly initialized. We use standard data augmentation techniques, including random scale jittering with a factor in [0.5, 2], random horizontal flipping, random cropping as well as random color jittering. We train models using SGD/AdamW for FCN-/attention-based models, respectively. The learning rate is scheduled following the polynomial annealing policy. In addition, for Cityscapes, we use a batch size of 8, and a training crop size of 768×768. For ADE20K and COCO-Stuff, we use a crop size of 512×512 and train the models with batch size 16. The models are trained for 160k, 160k, and 80k iterations on Cityscapes, ADE20K and COCO-Stuff, respectively. Exceptionally, for ablation study, we train models for 40K iterations. The hyper-parameters are empirically set to: \(K = 10, m = 0.999, \tau = 0.1, \kappa = 0.05, \lambda_1 = 0.01, \lambda_2 = 0.01\).

**Testing.** For ADE20K and COCO-Stuff, we rescale the short scale of the image to training crop size, with the aspect ratio kept unchanged. For Cityscapes, we adopt sliding window inference with the window size 768×768. For simplicity, we do not apply any test-time data augmentation. Our model is implemented in PyTorch and trained on eight Tesla V100 GPUs with a 32GB memory per-card. Testing is conducted on the same machine.

**Baselines.** We mainly compare with four widely recognized segmentation models, i.e., two FCN based (i.e., FCN \[79\], HRNet \[108\]) and two attention based (i.e., Swin \[78\] and SegFormer \[118\]). For fair comparison, all the models are based on our reproduction, following the hyper-parameter and augmentation recipes used in MMSegmentation \[21\].

**Evaluation Metric.** Following conventions \[15, 79\], mean intersection-over-union (mIoU) is adopted for evaluation.

5.2. Comparison to State-of-the-Arts

ADE20K\[140\] val. Table 1 reports comparisons with representative models on ADE20K val. Our nonparametric
scheme obtains consistent improvements over the baselines, with fewer learnable parameters. In particular, it yields \(1.2\%\) and \(1.0\%\) mIoU improvements over the FCN-based counterparts, i.e., FCN [79] and HRNet [108]. Similar performance gains (0.6\% and 0.8\%) are obtained over recent attention-based models, i.e., Swin [78] and SegFormer [118], manifesting the high versatility of our approach.

Cityscapes [22] Table 2 shows again our compelling performance on Cityscapes val. Specifically, our approach surpasses all the competitors, i.e., 1.0\% over FCN, 0.7\% over HRNet, 0.8\% over Swin, and 0.6\% over SegFormer. COCO-Stuff [10] test. As listed in Table 3, our approach also demonstrates promising performance on COCO-Stuff test. It outperforms all the baselines. Notably, with MiT-B4 [118] as the network backbone, our approach earns an mIoU score of 43.3\%, establishing a new state-of-the-art.

5.3 Scalability to Large-Vocabulary Semantic Segmentation

Today, rigorous evaluation of semantic segmentation models is mostly performed in a few category regime (e.g., 19/150/172 classes for Cityscapes/ADE20K/COCO-Stuff), while the generalization to more natural large-vocabulary setting is ignored. In this section, we demonstrate the remarkable superiority of our method in large-vocabulary setting. We start with the default setting in ADE20K [140] which includes 150 semantic concepts. Then, we gradually increase the number of concepts based on their visibility frequency, and train/test models on the selected number of classes. In this experiment, we use MiT-V2 [118] as the backbone and train models for 40k iterations.

The results are summarized in Table 4, from which we find that: i) For the parametric scheme, the amount of prototype parameters increases with vocabulary size. For the extreme case of 10 prototypes and 847 classes, the number of prototype parameters is 6.5 M, accounting for \(\sim 20\%\) of total parameters (i.e., 33.96 M). In sharp contrast, our scheme requires no any learnable prototype parameters. ii) Our
method achieves consistent performance elevations against the parametric counterpart under all settings. These results well demonstrate the utility of our nonparametric scheme for unrestricted open-vocabulary semantic segmentation.

### 5.4. Diagnostic Experiment

To investigate the effect of our core designs, we conduct ablative studies on ADE20K [140] val. We use MiT-B2 [118] as the backbone and train models for 40K iterations.

#### Training Objective

To first investigate our overall training objective (cf. Eq. 13). As shown in Table 5a, the model with \( \mathcal{L}_{CE} \) alone achieves an mIoU score of 45.0%. Adding \( \mathcal{L}_{PPC} \) or \( \mathcal{L}_{PPD} \) individually brings gains (i.e., 0.9%/0.4%), revealing the value to explicitly learn pixel-prototype relations. Combing all the losses together leads to the best performance, yielding an mIoU score of 46.4%.

#### Prototype Number Per Class \( K \)

Table 5b reports the performance of our approach with regard to the number of prototype per class. For \( K = 1 \), we directly represent each class as the mean embedding of its pixel samples. The pixel assignment is based simply on ground-truth labels, without using online clustering (Eqs. 8-9). This baseline obtains a score of 45.5%. Further, when using more prototypes (i.e., \( K = 3 \)), we see a clear performance boost (i.e., 45.5% → 46.0%). The score further improves when allowing 5 or 10 prototypes; however, increasing \( K \) beyond 10 gives marginal returns in performance. As a result, we set \( K = 10 \) for a better trade-off between accuracy and computation cost. This study confirms our motivation to use multiple prototypes for capturing intra-class variations.

#### Coefficient \( \mu \)

Table 5c quantifies the effect of momentum coefficient (\( \mu \) in Eq. 14) which controls the speed of prototype updating. The model performs reasonably well using a relatively large coefficient (i.e., \( \mu \in [0.999, 0.9999] \)), showing that a slow updating is beneficial. When \( \mu \) is 0.9 or 0.99, the performance decreases, and drops considerably at the extreme case of \( \mu = 0 \).

#### Distance Measure

By default, we use cosine distance (referred to as ‘Cosine’) to measure pixel-prototype similarity as denoted in Eq. 6, Eq. 11 and Eq. 12. However, other choices are also applicable. Here we study two alternatives. The first is the standard Euclidean distance (i.e., ‘Standard’), i.e., \( \langle y, y' \rangle = \|y - y'\|_2 \). In contrast to ‘Cosine’, here \( x \) and \( y \) are un-normalized real-valued vectors. To handle the non-differentiability in ‘Standard’, we further study an approximated Huber-like function [51] (‘Huberized’), i.e., \( \langle x, y \rangle = \delta(\sqrt{\|x - y\|_2^2}/\delta^2 + 1) \). The hyper-parameter \( \delta \) is empirically set to 0.1. As we find from Table 5d that ‘Cosine’ performs much better than other un-normalized Euclidean measurements. The Huberized norm does not show any advantage over ‘Standard’.

### 6. Conclusion and Discussion

The vast majority of recent effort in this field seek to learn parametric class representations for pixel-wise recognition. In contrast, this paper explores an exemplar-based regime. This leads to a nonparametric segmentation framework, where several typical points in the embedding space are selected as class prototypical representation, and distance to the prototypes determines how a pixel sample is classified. It enjoys several advantages: i) explicit prototypical representation for class-level statistics modeling; ii) better generalization with nonparametric pixel-category prediction; and iii) direct optimization of the feature embedding space. Our framework is elegant, general, and yields outstanding performance. It also comes with some intriguing questions. For example, to pursue better interpretability, one can optimize the prototypes to directly resemble pixel- or region-level observations [52, 65]. Overall, we feel the results in this paper warrant further exploration in this direction.

### Acknowledgements

This work was supported by CCF-Baidu Open Fund and ARC DECRA DE220101390.
References


[34] Salvador Garcia, Joaquin Derrac, Jose Cano, and Francisco Herrera. Prototype selection for nearest neighbor classi-


Xia Li, Yibo Yang, Qijie Zhao, Tiancheng Shen, Zhouchen Lin, and Hong Liu. Spatial pyramid based graph reasoning for semantic segmentation. In CVPR, 2020.


Ce Liu, Jenny Yuen, and Antonio Torralba. Nonparametric scene parsing via label transfer. IEEE TPAMI,


