HYPERGRAPH CONVOLUTIONAL NETWORKS FOR WEAKLY-SUPERVISED SEMANTIC SEGMENTATION

Jhony H. Giraldo[⋆] *Vincenzo Scarrica*[†] *Antonino Staiano*[†] *Francesco Camastra*[†] *Thierry Bouwmans*[⋆]

[⋆] Laboratoire Mathématiques, Image et Applications (MIA), La Rochelle Université, France
[†] Dipartimento di Scienze e Tecnologie, Università di Napoli Parthenope, Italy

2022 IEEE International Conference on Image Processing (ICIP) | 978-1-6654-9620-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICIP46576.2022.9897774

ABSTRACT

Semantic segmentation is a fundamental topic in computer vision. Several deep learning methods have been proposed for semantic segmentation with outstanding results. However, these models require a lot of densely annotated images. To address this problem, we propose a new algorithm that uses HyperGraph Convolutional Networks for Weakly-supervised Semantic Segmentation (HyperGCN-WSS). Our algorithm constructs spatial and k-Nearest Neighbor (k-NN) graphs from the images in the dataset to generate the hypergraphs. Then, we train a specialized HyperGraph Convolutional Network (HyperGCN) architecture using some weak signals. The outputs of the HyperGCN are denominated pseudo-labels, which are later used to train a DeepLab model for semantic segmentation. HyperGCN-WSS is evaluated on the PASCAL VOC 2012 dataset for semantic segmentation, using scribbles or clicks as weak signals. Our algorithm shows competitive performance against previous methods.

Index Terms— Semantic segmentation, weakly supervised learning, hypergraph convolutional networks

1. INTRODUCTION

Semantic segmentation is an important task in computer vision with multiple applications in image, 3D, and video processing [1–3]. The main objective of semantic segmentation is to classify all the pixels in the images into some predefined classes. Deep learning models have dominated the study of semantic segmentation in recent years [4–6]. However, these deep learning methods are usually very complex models containing millions of learnable parameters, and thus they require a lot of densely annotated images to perform well [3, 7].

Currently, there is an increasing interest in weakly supervised learning [8], where the predictions are obtained with a limited amount of labels. As a result, several studies have proposed Weakly-supervised Semantic Segmentation (WSS) methods [9–12], where graphical models have played a central role. Particularly, Graph Convolutional Networks (GCNs) have been widely explored in WSS, reaching state-of-the-art performances [11, 12]. However, these methods have focused on constructing graphs from individual images using spatial information. Thus, these models waste crucial information that can be obtained from other images in the dataset.

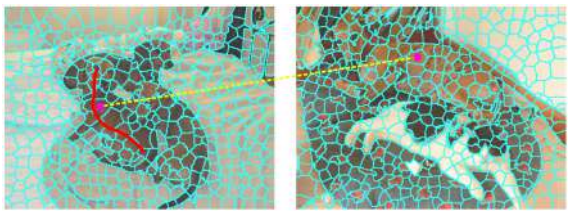

Fig. 1: The idea of HyperGCN-WSS is to rely both on the spatial and structural information in the datasets.

In this work, we propose a new algorithm named HyperGraph Convolutional Networks for Weakly-supervised Semantic Segmentation (HyperGCN-WSS), using scribbles and clicks as weak signals. The key idea of our algorithm is to rely on spatial information as in [11, 12], as well as on structural information that can be captured from other instances in the dataset. Our algorithm uses HyperGCNs [13] to capture such information. HyperGCN-WSS is composed of 1) superpixel segmentation [14] for node representation, 2) VGG-16 for feature extraction [15], 3) spatial and k-NN graph construction, 4) HyperGCN [13] to generate pseudo-labels, and 5) DeepLabV3+ [16] for semantic segmentation using the pseudo-labels as the ground-truth. Fig. 1 shows the motivation of HyperGCN-WSS, where one labeled superpixel (with a scribble) is connected to another non-labeled superpixel in the dataset, allowing the propagation of information between instances in the dataset. HyperGCN-WSS is evaluated in the PASCAL VOC 2012 dataset [17] for semantic segmentation using scribbles and clicks as weak signals. Our algorithm shows competitive performance against previous methods.

The main contributions of this paper are presented as follows: 1) we propose a new algorithm for WSS, 2) we show that using HyperGCNs is better than using GCNs alone for WSS, and 3) we evaluate our algorithm with two types of weak signals, scribbles, and clicks, showing competitive performance against some previous methods.

The rest of the paper is organized as follows. Section 2 explains HyperGCN-WSS. Section 3 introduces the experiments and results. Finally, Section 4 presents the conclusions.

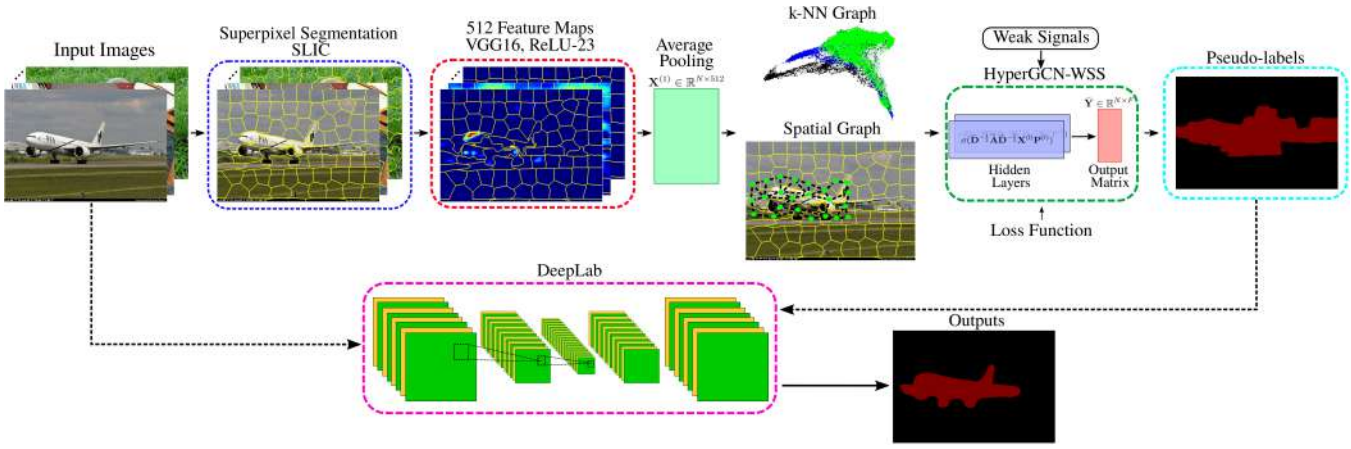

Fig. 2: The pipeline of HyperGCN-WSS. Our algorithm uses SLIC superpixel segmentation, VGG16 feature extraction, average pooling, spatial and k-NN graph construction, a specialized HyperGCN architecture, and a DeepLabV3+ model.

2. PROPOSED METHOD

Fig. 2 shows the pipeline of HyperGCN-WSS, where we have superpixel segmentation, feature extraction, hypergraph construction, HyperGCN, and DeepLab for segmentation.

2.1. Preliminaries

A simple graph $G = (\mathcal{V}, \mathcal{E})$ is a mathematical entity where we have a set of nodes $\mathcal{V} \in \{1, \dots, N\}$ and a set of edges $\mathcal{E} = \{(i,j)\}$. In this paper, we consider undirected and weighted graphs. Let $\mathbf{A} \in \mathbb{R}^{N \times N}$ be the adjacency matrix of G such that $\mathbf{A}(i,j) > 0$ if $(i,j) \in \mathcal{E}$, and 0 otherwise. Let $\mathbf{D} \in \mathbb{R}^{N \times N}$ be the diagonal degree matrix of G such that $\mathbf{D} = \text{diag}(\mathbf{A1})$, where $\mathbf{1}$ is a vector of ones with appropriate dimension, and $\text{diag}(\cdot)$ creates a diagonal matrix from a vector. Notice that $\mathbf{A}$ can only represent edges that connect two nodes. A hypergraph $G_h = (\mathcal{V}, \mathcal{E})$ is a generalization of simple graphs G, where the edges can connect multiple nodes. Let $\mathbf{W} \in \mathbb{R}^{M \times M}$ be the diagonal matrix of hyperedge weights, where $\mathbf{W}(e,e)$ is the weight of the eth hyperedge and $M = |\mathcal{E}|$. Let $\mathbf{H} \in \{0,1\}^{N \times M}$ be the incidence matrix of G_h such that $\mathbf{H}(i,e) = 1$ if the ith node is incident to the edge e, and 0 otherwise, *i.e.*, $\mathbf{H}(i,e) = 1$ if the node i is connected by the edge e. Let $\mathbf{D}_h \in \mathbb{R}^{N \times N}$ be the diagonal matrix of node degree, where $\mathbf{D}_h(i,i) = \sum_{e=1}^{M} \mathbf{W}(e,e)\mathbf{H}(i,e)$. Finally, let $\mathbf{B} \in \mathbb{R}^{M \times M}$ be the diagonal matrix of hyperedge degree, where $\mathbf{B}(e,e) = \sum_{i=1}^{N} \mathbf{H}(i,e)$. In this work, we use the hypergraphs to represent two kinds of relationships: 1) the spatial relationships of the nodes on each image and 2) the relationship of nodes from different images in the dataset.

2.2. Nodes Representation and Graph Construction

We use the SLIC superpixel segmentation [14] method to represent the nodes in the graph G (or G_h). Superpixels obtain homogeneous regions from the images to have a better context for the representation. Furthermore, the input feature description of each node is obtained with some pre-trained Convolutional Neural Network (CNN). In the current work, we use the outputs of the 10th ReLU layer of the VGG16 [15] (the 23th layer of the network), *i.e.*, we use an intermediate layer of the CNN. The feature representation contains 512 features maps. Additionally, an average pooling is performed on the superpixel regions of each feature map to obtain the feature representation, *i.e.*, each node is represented with a 512-dimensional vector.

In the current work, we construct two types of graphs: 1) spatial graphs in the superpixels of each image, and 2) k-NN graphs with k $= 10$ on some embedding space. Let α be the number of images in the dataset, let ξ be the number of superpixels for SLIC, and let μ be the maximum number of nodes we allow for each graph ($\mu = 40000$ in the experiments). Therefore, we construct $\tau = \lfloor \frac{\alpha \times \xi}{\mu} \rceil$ graphs, where we have $\gamma = \lfloor \frac{\alpha}{\tau} \rceil$ images per graph. For the spatial graphs, we connect all the nodes that are in the neighborhood of each superpixel as shown in Fig. 2. Therefore, we create a block diagonal matrix with the γ adjacency matrices of each image, *i.e.*, we have γ unconnected subgraphs for each spatial graph. For the k-NN graph, we use an embedding representation to compute the Euclidean distances. For example, these embeddings can be intermediate outputs of a GCN or a HyperGCN. The weights of the edges for the spatial and k-NN graphs are given by the Gaussian function $\exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / \sigma^2\right)$, where $\mathbf{x}_i$ is the embedding (or feature representation) of the ith node, and σ is the standard deviation given by $\sigma = \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} \|\mathbf{x}_i - \mathbf{x}_j\|_2$.

2.3. Graph and Hypergraph Convolutional Networks

In this paper, we use graph convolutions and hypergraph convolutions in our HyperGCN architecture. For the graph con-

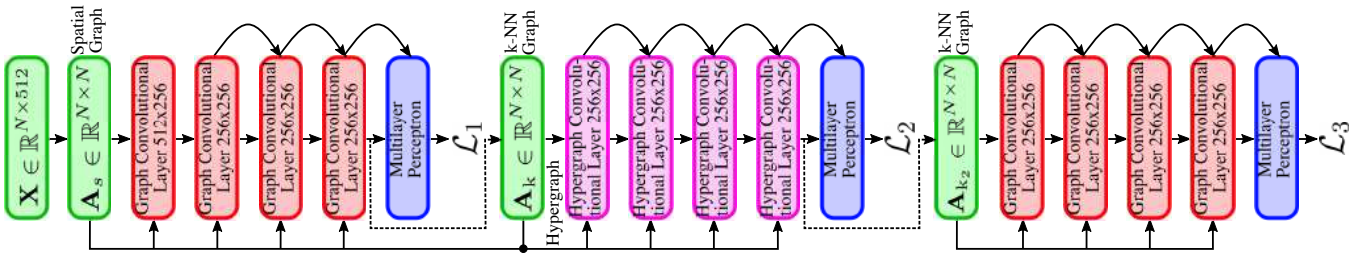

Fig. 3: HyperGCN-WSS architecture with skip connections, as well as several graph and hypergraph convolutional layers. $\mathbf{X}$ is the matrix of features from VGG16. HyperGCN-WSS is trained in three stages, where we have three loss functions $\mathcal{L}_i$.

volutions, we use the model of Kipf and Welling [18]. For the hypergraph convolutions, we use the model of Bai *et al*. [13].

The graph convolution in [18] is given by the following propagation rule:

$$\mathbf{X}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}^{(l)}\mathbf{P}^{(l)}), \tag{1}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$, $\mathbf{X}^{(l)}$ is the matrix of activations in layer l (matrix of features or embeddings), $\mathbf{P}^{(l)}$ is the matrix of trainable weights in layer l, and $\sigma(\cdot)$ is an activation function. Similarly, the hypergraph convolution in [13] is given as follows:

$$\mathbf{X}^{(l+1)} = \sigma(\mathbf{D}_h^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{B}^{-1}\mathbf{H}^{\mathsf{T}}\mathbf{D}_h^{-\frac{1}{2}}\mathbf{X}^{(l)}\mathbf{P}^{(l)}). \tag{2}$$

2.4. HyperGCN Architecture

Fig. 3 shows the architecture of our HyperGCN-WSS. The input $\mathbf{X} \in \mathbb{R}^{N\times 512}$ is the matrix of features from the VGG16 network. Each Graph Convolutional Layer (GCL) contains batch normalization [19], Exponential Linear Unit (ELU) [20] as activation function, and it could have a residual connection [21] as shown in Fig. 3. The GCLs implement the propagarion rule in (1). The Hypergraph Convolutional Layers (HCLs) are similar to the GCLs, but instead of using (1), they implement (2). Our architecture also contains Multilayer Perceptrons that classify the embedding of the GCLs or HCLs. We use intermediate embeddings as shown by the dotted lines in Fig. 3 to construct k-NN graphs. The first k-NN graph $\mathbf{A}_{\mathrm{k}}$ is combined with the spatial graph to create a hypergraph and the second intermediate embeddings are used to construct the k-NN graph $\mathbf{A}_{\mathrm{k}_2}$. We avoid the over-smoothing problem [22] by performing the training procedure in three separate steps with the loss functions $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ [23].

3. EXPERIMENTS AND RESULTS

3.1. Dataset and Evaluation Metrics

HyperGCN-WSS is evaluated on PASCAL VOC 2012 [17] dataset for semantic segmentation. We also use the dataset of scribbles [10] and random clicks as weak signals. PASCAL VOC 2012 has 20 semantic classes and one background category. We use the augmented version of the PASCAL dataset provided by [24], resulting in 10582 images in the training set, and 1449 images in the validation set. In this paper, we use the training set in [24] for training and validation, and we leave the validation set as the test set. We use the mean Interception over Union (mIoU) metric [17] for evaluation.

3.2. Implementation Details

HyperGCN-WSS is implemented using PyTorch with a learning rate of 0.01 and weight decay of $5\mathrm{e}{-4}$. Each GCL or HCL has 256 hidden units and a dropout rate of 50%. Each stage of HyperGCN-WSS is trained for a maximum of 1000 epochs using Adam [25]. For scribbles, we use 5% of the scribbles for validation. For clicks, we use 1% of the clicks for validations. We train HyperGCN-WSS using a learning scheduler that reduces the learning rate when the loss function has stopped improving in the validation set. Our scheduler has a reducing factor of 0.5, patience of 25 epochs, and a minimum learning rate of $1\mathrm{e}{-6}$ (HyperGCN-WSS stops the learning procedure either if we reach the maximum number of epochs or if we reach the minimum learning rate). The final activation of the Multi-layer Perceptrons are logarithmic softmax, and we use the negative log-likelihood as loss functions.

3.3. Experiments

In this work, we perform experiments in 1) the dataset of scribbles [10] and 2) some random clicks that are given by a percentage of N. The percentage of clicks is given by the set $\mathcal{M} = \{\frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}\}$. For example, for $\frac{1}{32} \in \mathcal{M}$ and $\xi = 100$ number of superpixels, we have around 3.125 random clicks per image for training. Similarly, we analyze the impact of the number of superpixels ξ in the set $\mathcal{S} = \{50, 100, 200, 400, 800\}$, for both scribbles and clicks. We report the mIoU of the pseudo-labels in the training set for each loss function $\mathcal{L}_i \; \forall \; i \in \{1, 2, 3\}$ to assess the propagation of information of our algorithm after each stage. We also report the mIoU after the DeepLab training in the validation set. We do not perform an extensive search of semantic segmentation models like in [11, 26] because that is not our scope.

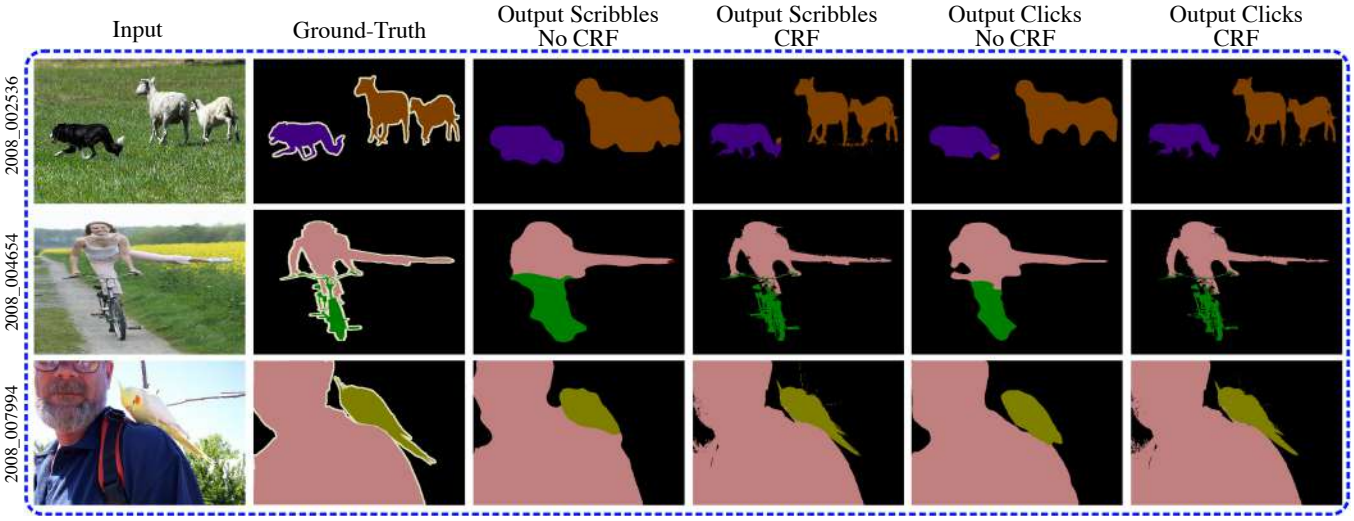

Fig. 4: Some visual results on PASCAL VOC 2012 with our HyperGCN-WSS, using scribbles or clicks as weak signals.

Table 1: Accuracy in mIoU (%) in the train set of PASCAL VOC after each loss function $\mathcal{L}_i \; \forall \; i \in \{1, 2, 3\}$ in our algorithm.

Weak	$\xi = 50$			$\xi = 100$			$\xi = 200$			$\xi = 400$			$\xi = 800$		
Signals	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$
Scribbles	51.85	54.49	54.23	50.04	54.31	**54.51**	46.03	49.72	50.72	39.86	44.65	45.51	32.42	37.99	39.51
Clicks $\frac{1}{32}$	41.35	42.13	42.05	41.43	42.61	42.30	40.46	43.06	42.97	39.36	43.76	**44.28**	32.56	37.91	39.87
Clicks $\frac{1}{16}$	45.59	46.78	46.70	46.06	47.98	47.72	44.81	49.17	**49.19**	39.84	45.84	46.98	33.60	38.77	41.48
Clicks $\frac{1}{8}$	50.04	51.71	51.48	51.08	**54.21**	54.15	47.94	52.98	53.18	41.33	46.48	47.98	33.16	38.76	41.24
Clicks $\frac{1}{4}$	53.44	55.76	55.52	53.63	**57.60**	57.51	49.74	54.51	55.26	41.42	46.92	48.82	34.19	37.49	40.07

Table 2: Accuracy of HyperGCN-WSS and other methods in the validation set of PASCAL VOC. S: Scribbles. C: Clicks.

Method	Weak Signal	CRF	mIoU (%)
ScribbleSup [10]	S	✓	63.1
RAWKS [27]	S	✓	61.4
NormalizedCutLoss [26]	S	-	60.5
GraphNet [11]	S	-	63.3
HyperGCN-WSS (ours)	S	-	65.3
HyperGCN-WSS (ours)	C	-	**65.4**

3.4. Results and Discussions

Fig. 4 shows some visual results of HyperGCN-WSS before and after applying Conditional Random Field (CRF) [28] for visualization purposes. Similarly, Table 1 shows the mIoU of the pseudo-labels after each loss function $\mathcal{L}_i \; \forall \; i \in \{1, 2, 3\}$ in the training set of PASCAL VOC with scribbles and clicks, *i.e.*, we have information of different parts of our architecture. Notice that we do not use the full labeled annotation of the training set of PASCAL VOC, so Table 1 shows how well the information is propagated to the other nodes in the graph. We notice that there is a gap in performance between using the GCN alone and the HyperGCN. For example, in scribbles and clicks, there is a gap of around 4% between $\mathcal{L}_1$ and $\mathcal{L}_2$. The best results for each weak signal in Table 1 (in **bold**) correspond to low values of superpixels ξ. The reason is that the dimensions of the features maps of the VGG16 in the 23 layer are around 8 times smaller than the original image. Therefore, having big values of ξ means smaller superpixels, which are hard to adequately represent with deep layers of CNNs. Finally, Table 2 shows the comparison of HyperGCN-WSS with previous methods. We did not report results with CRF post-processing due to space constraints. Our algorithm shows competitive performance against the other methods.

4. CONCLUSIONS

In this work, we introduced a new HyperGCN-WSS algorithm. Our algorithm is composed of SLIC superpixel segmentation, CNN feature extraction, hypergraph construction, a specialized HyperGCN architecture, and the DeepLabV3+ model. This new HyperGCN architecture combines graph and hypergraph convolutions. HyperGCN-WSS used spatial graphs constructed in the neighborhood of the superpixels, and k-NN graphs constructed in some embedding representation. We showed that using hypergraph convolutions is better than using graph convolutions alone. Similarly, our algorithm showed competitive performance against previous methods.

5. REFERENCES

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[2] G. Li, M. Muller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?," in *IEEE ICCV*, 2019.

[3] J. H. Giraldo, S. Javed, and T. Bouwmans, "Graph moving object segmentation," *IEEE T-PAMI*, 2020.

[4] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE CVPR*, 2015.

[5] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE T-PAMI*, vol. 39, no. 12, pp. 2481–2495, 2017.

[6] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE T-PAMI*, vol. 40, no. 4, pp. 834–848, 2017.

[7] J. H. Giraldo and T. Bouwmans, "Semi-supervised background subtraction of unseen videos: Minimization of the total variation of graph signals," in *IEEE ICIP*, 2020.

[8] Z. H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1, pp. 44–53, 2018.

[9] F. Z. Xing, E. Cambria, W. B. Huang, and Y. Xu, "Weakly supervised semantic segmentation with superpixel embedding," in *IEEE ICIP*, 2016.

[10] Di Lin, J. Dai, J. Jia, K. He, and J. Sun, "ScribbleSup: Scribble-supervised convolutional networks for semantic segmentation," in *IEEE CVPR*, 2016.

[11] M. Pu, Y. Huang, Q. Guan, and Q. Zou, "GraphNet: Learning image pseudo annotations for weakly-supervised semantic segmentation," in *ACM Multimedia*, 2018.

[12] B. Zhang, J. Xiao, J. Jiao, Y. Wei, and Y. Zhao, "Affinity attention graph neural network for weakly supervised semantic segmentation," *IEEE T-PAMI*, 2021.

[13] S. Bai, F. Zhang, and P. H. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognition*, vol. 110, pp. 107637, 2021.

[14] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE T-PAMI*, vol. 34, no. 11, pp. 2274–2282, 2012.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[16] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018.

[17] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge: A retrospective," *IJCV*, vol. 111, no. 1, pp. 98–136, 2015.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[20] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *ICLR*, 2016.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016.

[22] Q. Li, Z. Han, and X. M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *AAAI*, 2018.

[23] Y. Chen, L. Wu, and M. Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," in *NeurIPS*, 2020.

[24] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *IEEE ICCV*, 2011.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[26] M. Tang, A. Djelouah, F. Perazzi, Y. Boykov, and C. Schroers, "Normalized cut loss for weakly-supervised CNN segmentation," in *IEEE CVPR*, 2018.

[27] P. Vernaza and M. Chandraker, "Learning random-walk label propagation for weakly-supervised semantic segmentation," in *IEEE CVPR*, 2017.

[28] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected CRFs with Gaussian edge potentials," in *NeurIPS*, 2011.