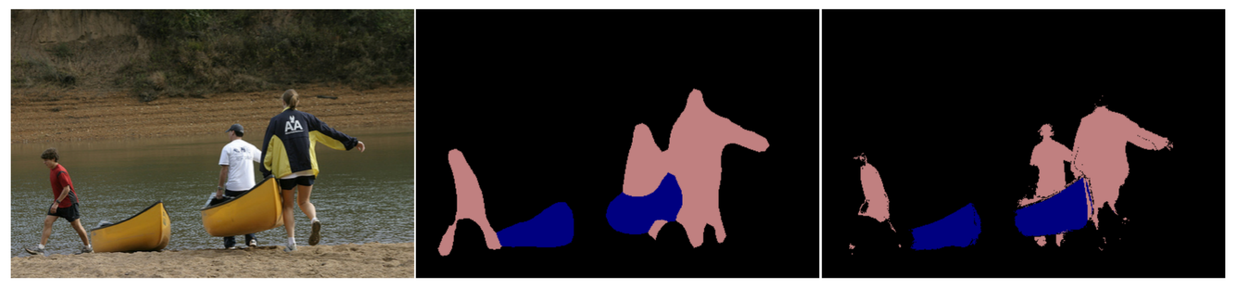
Convolutional Random Walk Networks for Semantic Image Segmentation

Gedas Bertasius, Jianbo Shi — University of Pennsylvania
Lorenzo Torresani — Dartmouth College
Stella X. Yu — UC Berkeley / ICSI

I. Introduction

❑ Goals:

- To address the issues of poor boundary localization and spatially fragmented segmentation predictions.



Input DeepLab [6] DeepLab-CRF [6]

- We want to achieve this goal with an end-to-end trainable model and without increasing the complexity of the model.

	[6]	[5]	[23]	[27]	[19]	RWN
requires post-processing?	✓	✗	✗	✗	✗	✗
uses complex loss?	✗	✗	✗	✓	✓	✗
requires recurrent layers?	✗	✓	✗	✓	✗	✗
model size (in MB)	79	79	961	514	>1000	79

II. Background

❑ Random Graph Walks

- Let $W \in \mathbb{R}^{n \times n}$ be an affinity matrix where $W_{ij} \in [0,1]$ denotes how similar the nodes i and j are.
- In the context of semantic segmentation, each pixel can be viewed as a node and edges can be viewed as a similarity between two given pixels (e.g. color similarity)
- Then let $D \in \mathbb{R}^{n \times n}$ denote a diagonal degree matrix such that $D_{ii} = \sum_{j=1}^{n} W_{ij}$ for all j except j=i.
- Then by definition, the random walk transition matrix is defined as $A = D^{-1}W$.

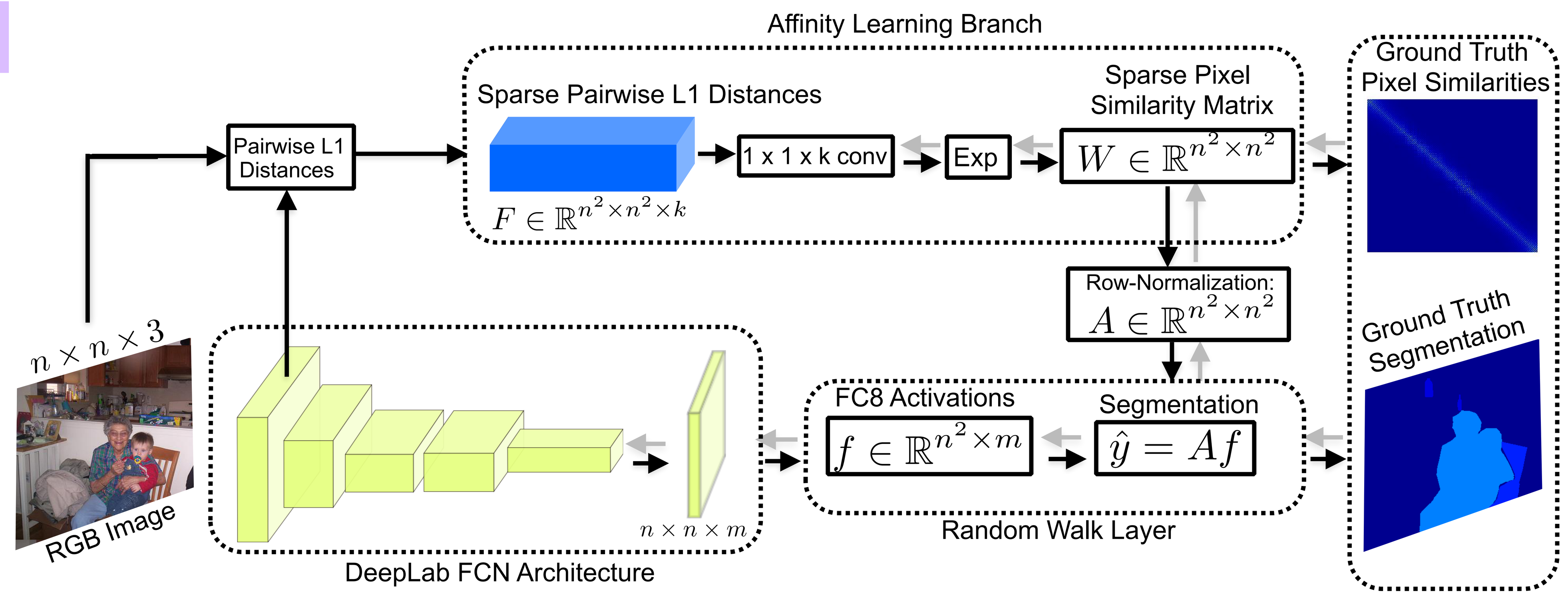
❑ Differences with MRFs/CRFs

- Random walk methods can implement any arbitrary graph structure via an affinity matrix specification while CRFs typically employ graphs with a fixed grid structure.
- Random walk methods can propagate information across the nodes via a standard matrix multiplication without resorting to approximate inference techniques.

III. Convolutional Random Walk Networks

❑ Key Ideas

- A Convolutional Random Walk Network (RWN) is a network composed of 1) semantic segmentation and 2) pixel-level affinity prediction branches.
- The predictions from two branches are merged via a novel random walk layer.
- RWN requires only 131 additional parameters compared to standard FCNs.
- The whole network can be optimized jointly end-to-end.



❑ Semantic Segmentation Branch

- Implemented as a DeepLab FCN

❑ Pixel-Level Affinity Branch

- Uses RGB, conv1_1, conv1_2 features in conjunction to learn the weights predicting how similar two given pixels are.
- The predicted affinities are assembled into an affinity matrix W, which is used to construct a random walk transition matrix A.
- The affinity learning branch uses only 131 parameters.

❑ Random Walk Layer

- Combines the predictions from both branches via a matrix multiplication. $\hat{y} = Af$.
- During training, the gradients $\frac{\partial L}{\partial \hat{y}}$ from the loss layer are propagated back to the affinity learning branch as $\frac{\partial L}{\partial A} = \frac{\partial L}{\partial \hat{y}} f^T$, and as $\frac{\partial L}{\partial f} = A^T \frac{\partial L}{\partial \hat{y}}$ to the segmentation branch.

❑ Prediction

- Applying a random walk until convergence produces the following prediction rule:

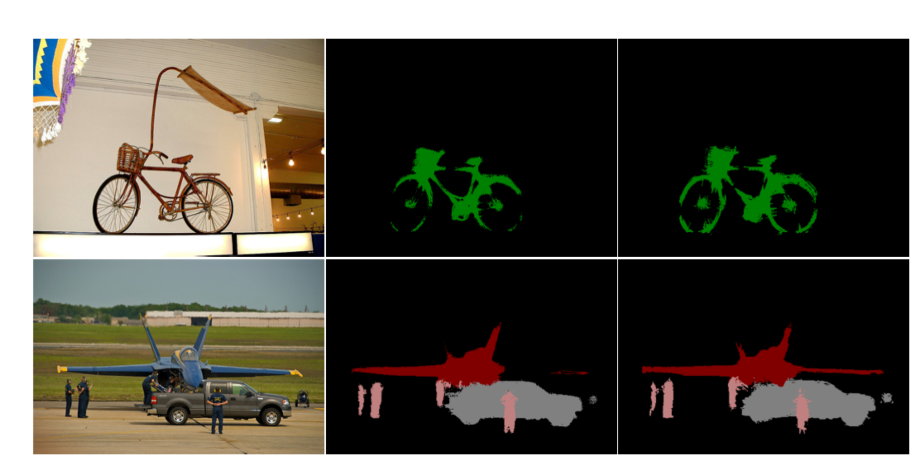
$$\hat{y}^{t+1} = \alpha A \hat{y}^t + (1-\alpha) f \ = (\alpha A)^{t+1} f + (1-\alpha) \sum_{i=0}^{t} (\alpha A)^i f \quad (1)$$

$$\hat{y}^{\infty} = (I - \alpha A)^{-1} f \quad (2)$$

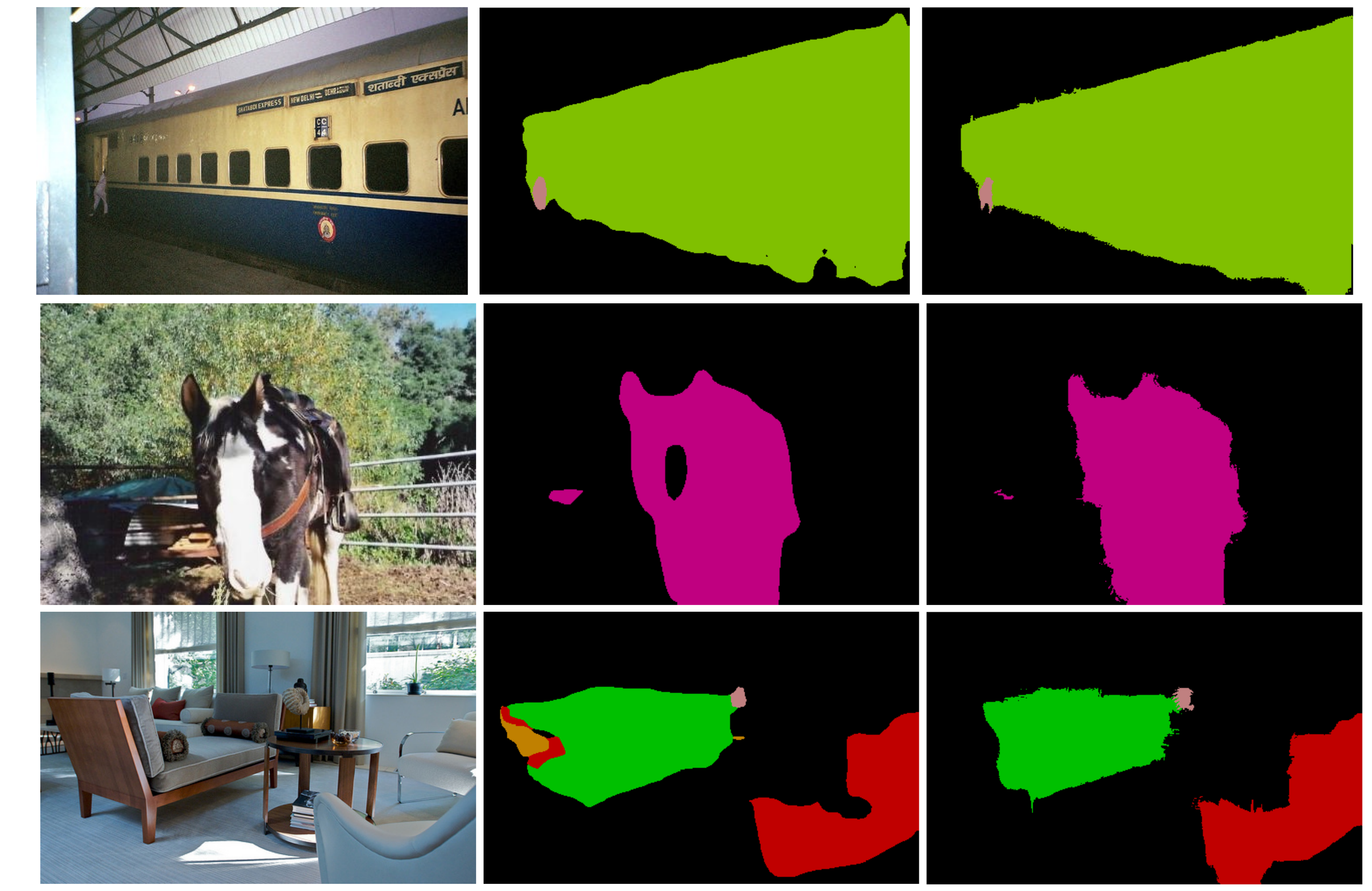
IV. Results

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean	overall
DeepLab-largeFOV	79.8	71.5	78.9	70.9	72.1	87.9	81.2	85.7	46.9	80.9	56.5	82.6	77.9	79.3	80.1	64.4	77.6	52.7	80.3	70.0	73.8	76.0
RWN-largeFOV	**81.6**	**72.1**	**82.3**	**72.0**	**75.4**	**89.1**	**82.5**	**87.4**	**49.1**	**83.6**	**57.9**	**84.8**	**80.7**	**80.2**	**81.2**	**65.7**	**79.7**	**55.5**	**81.5**	**74.0**	**75.8**	**77.9**
DeepLab-attention	83.4	76.0	83.0	**74.2**	77.6	91.6	85.2	89.1	54.4	86.1	62.9	86.7	83.8	84.2	82.4	**70.2**	84.7	61.0	84.8	77.9	79.0	80.5
RWN-attention	**84.7**	**76.6**	**85.5**	74.0	**79.0**	**92.4**	**85.6**	**90.0**	**55.6**	**87.4**	**63.5**	**88.2**	**85.0**	**84.8**	**83.4**	70.1	**85.9**	**62.6**	**85.1**	**79.3**	**79.9**	**81.5**
DeepLab-v2	85.5	**50.6**	86.9	**74.4**	82.7	93.1	88.4	91.9	62.1	89.7	71.5	90.3	86.2	86.3	84.6	**75.1**	87.6	72.2	87.8	81.3	81.4	83.4
RWN-v2	**86.0**	50.0	**88.4**	73.5	**83.9**	**93.4**	**88.6**	**92.5**	**63.9**	**90.9**	**72.6**	**90.9**	**87.3**	**86.9**	**85.7**	75.0	**89.0**	**74.0**	**88.1**	**82.3**	**82.1**	**84.3**

Method	mean IOU	overall IOU
DeepLab-largeFOV-CRF	75.7	77.7
RWN-largeFOV	**75.8**	**77.9**
DeepLab-attention-CRF	**79.9**	**81.6**
RWN-attention	**79.9**	81.5
DeepLab-v2-CRF	81.9	84.2
RWN-v2	**82.1**	**84.3**
DeepLab-DT	76.6	78.7
RWN	**76.7**	**78.8**



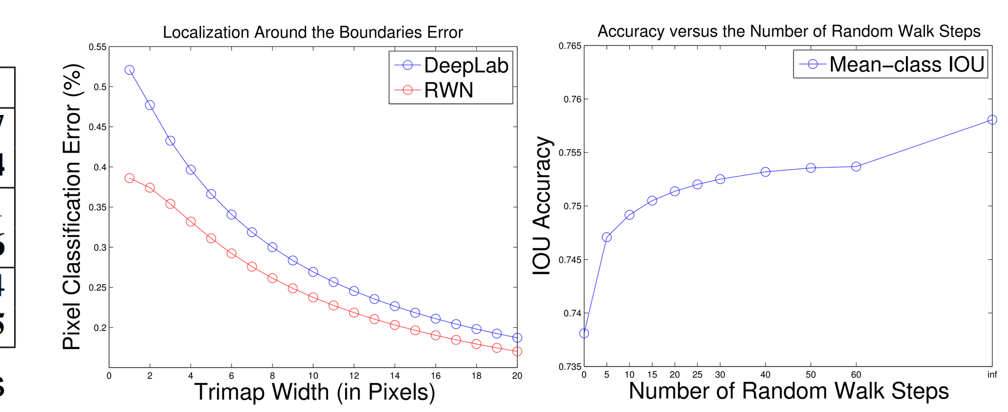
Input DeepLab_v2-CRF RWN_v2

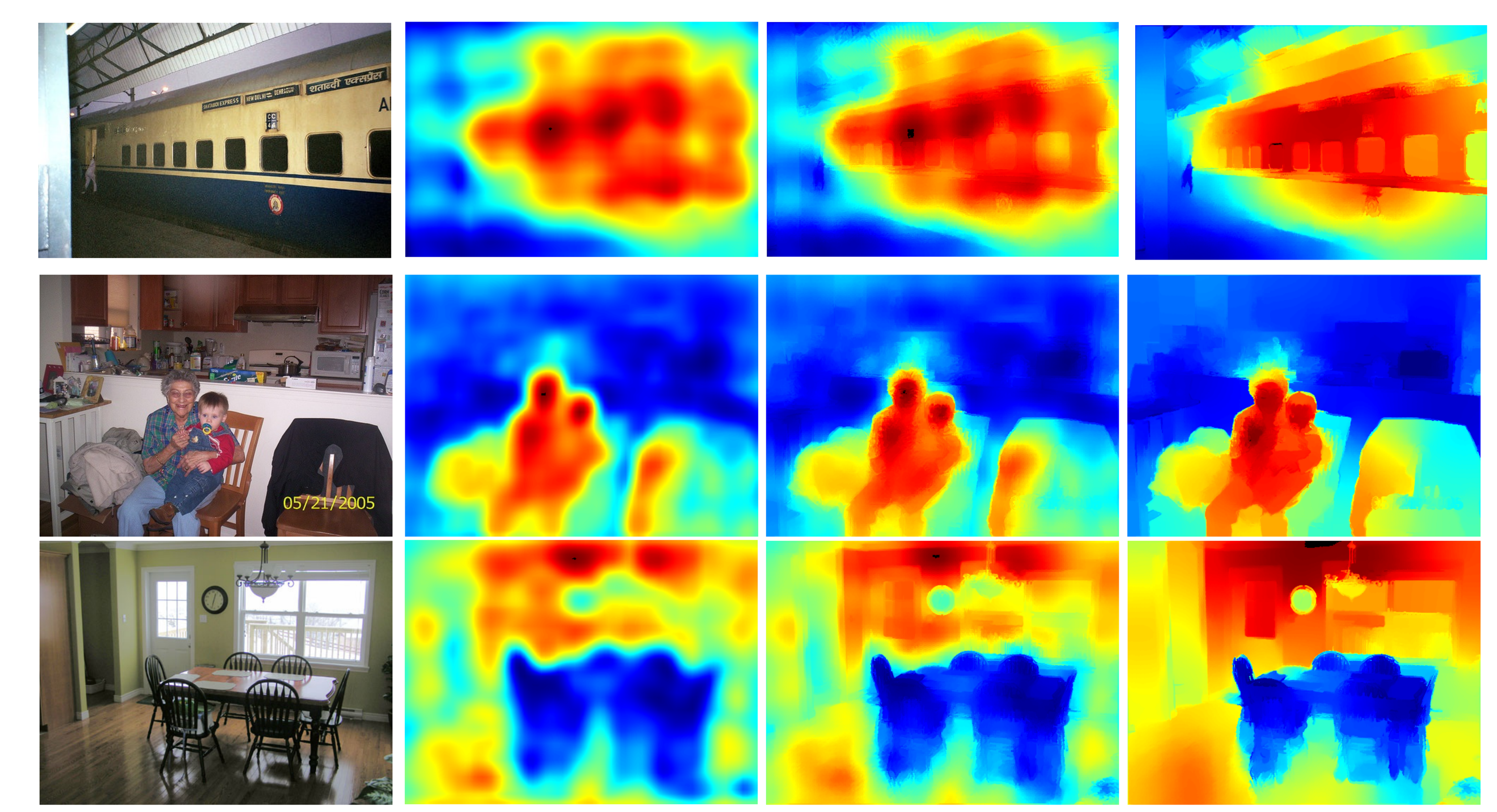


Input Image DeepLab RWN

Method	MF	AP
DeepLab-largeFOV-CRF	0.676	0.457
RWN-largeFOV	**0.703**	**0.494**
DeepLab-attention-CRF	0.722	0.521
RWN-attention	**0.747**	**0.556**
DeepLab-v2-CRF	0.763	0.584
RWN-v2	**0.773**	**0.595**

Spatial Smoothness Results





Input Image Iteration 0 Iteration 10 Iteration 50

❑ Run Time

- denseCRF inference requires ~3.301 seconds per image, and it achieves 81.9% IoU on Pascal SBD dataset.
- A single random walk iteration takes ~0.032 seconds, and it achieves 82.2% IoU (with R=40).

V. Conclusions

- RWN improves upon traditional FCNs by addressing poor localization and spatially disjoint segmentation issues.
- RWN can be easily incorporated into a standard FCN framework, and trained jointly end-to-end.
- Unlike prior methods, RWN achieves these goals without increasing the complexity of the model.