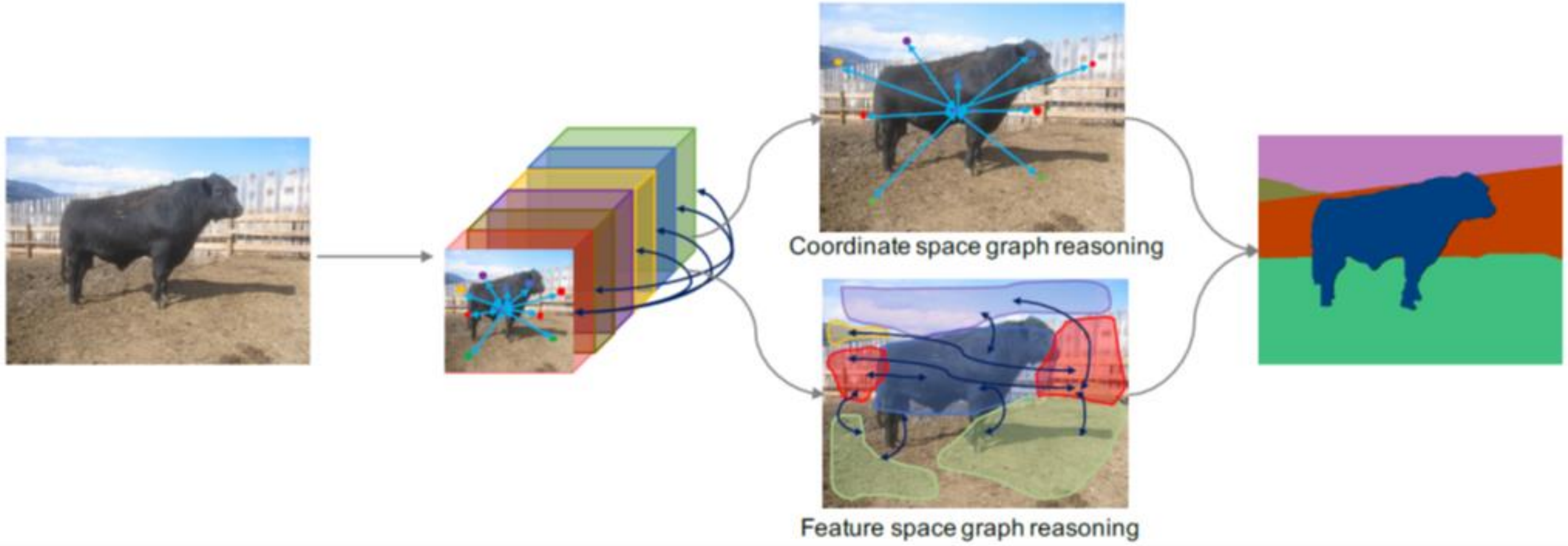
Dual Graph Convolutional Network for Semantic Segmentation

서울대학교 뇌과학 협동과정
2020-27203
김재인



Introduction

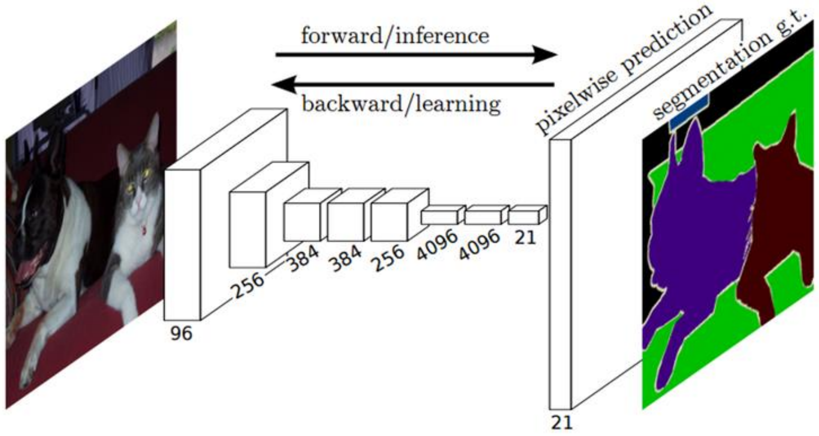
- Semantic segmentation:
  - to assign object class label to each pixel
- Challenge in semantic segmentation task

→ Isolated pixels are difficult to classify.

→ The system should capture contextual info

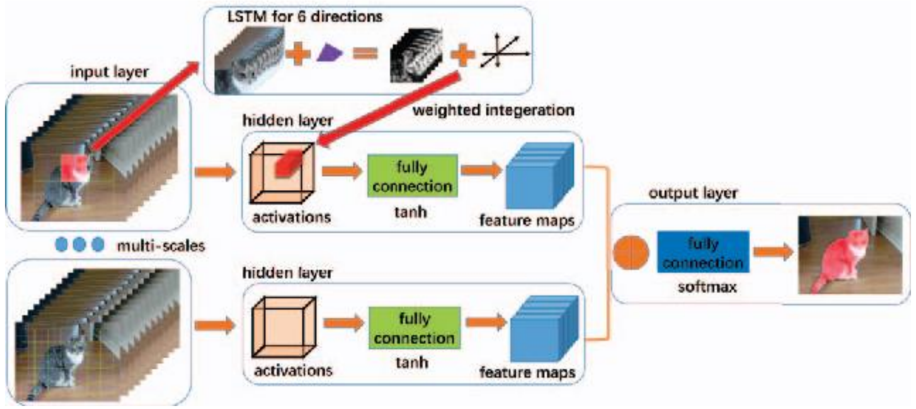
Introduction

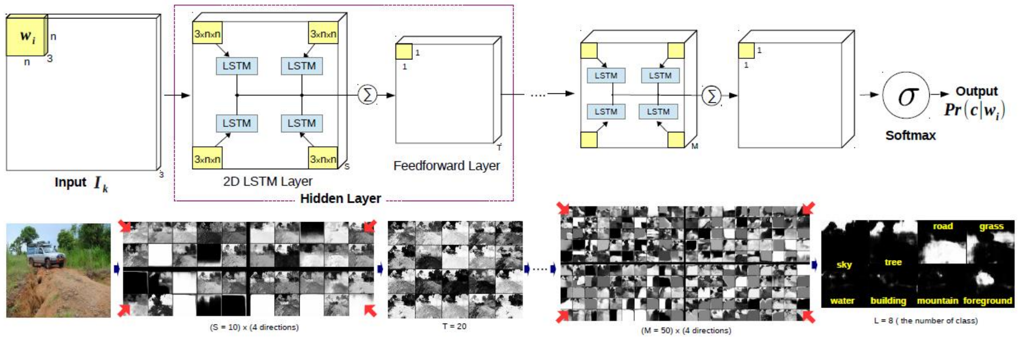
- Convolutional neural network models are commonly used for segmentation problems
- Limitations of FCN
  - The receptive field grows only linearly
  - Not able to capture long-range relationships between pixels
  - Feature representation is dominated by large objects



Introduction

- Other methods?
  - Multiscale feature fusion
  - Using LSTMs
  - Learning an affinity map at each spatial position
- Large memory requirements
- Unsuitable for high resolution image
  - Ex) Cityscapes dataset

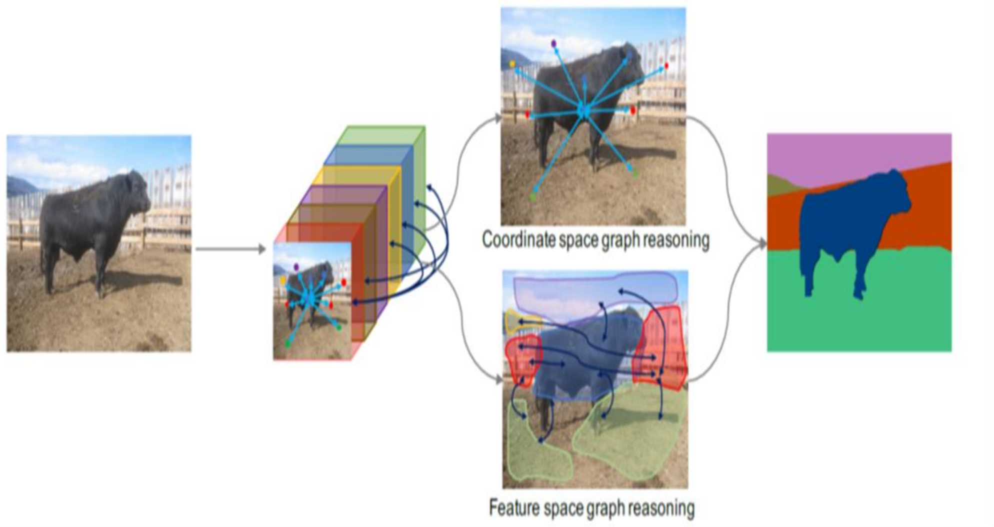




Introduction

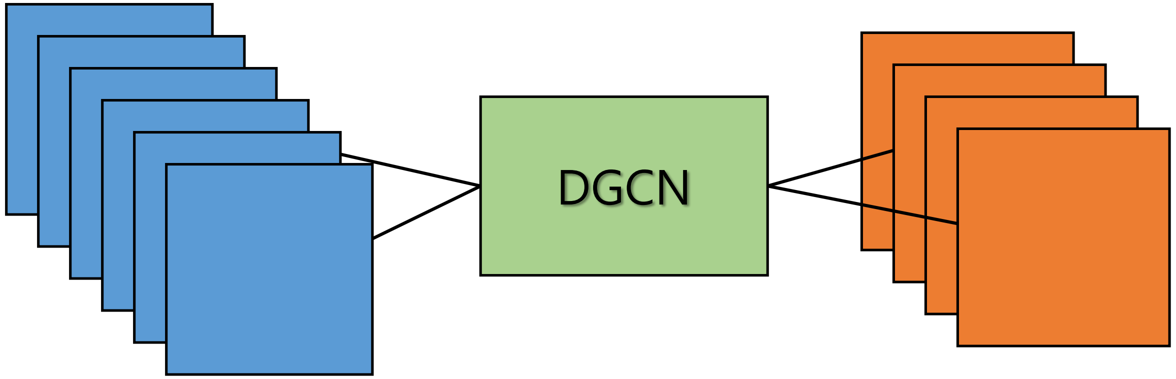
- Use GCN to model contextual information for segmentation efficiently

- Coordinate Space GCN
  - Model spatial relationships between pixels
- Feature Space GCN
  - Model inter-dependencies in feature dimension



Implementation of Dual Graph Convolution Network

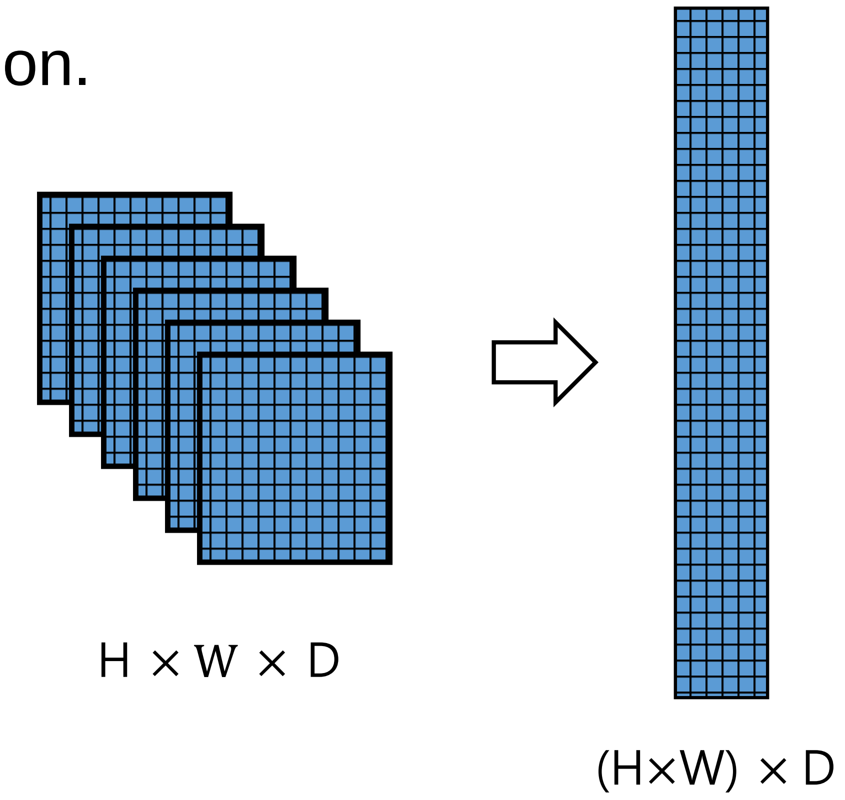
- DGCN refine D feature images from backbone CNN model
- Refined images maintain their shape
- Refined feature images are input to dilated convolution for segmentation



Graph Formulation

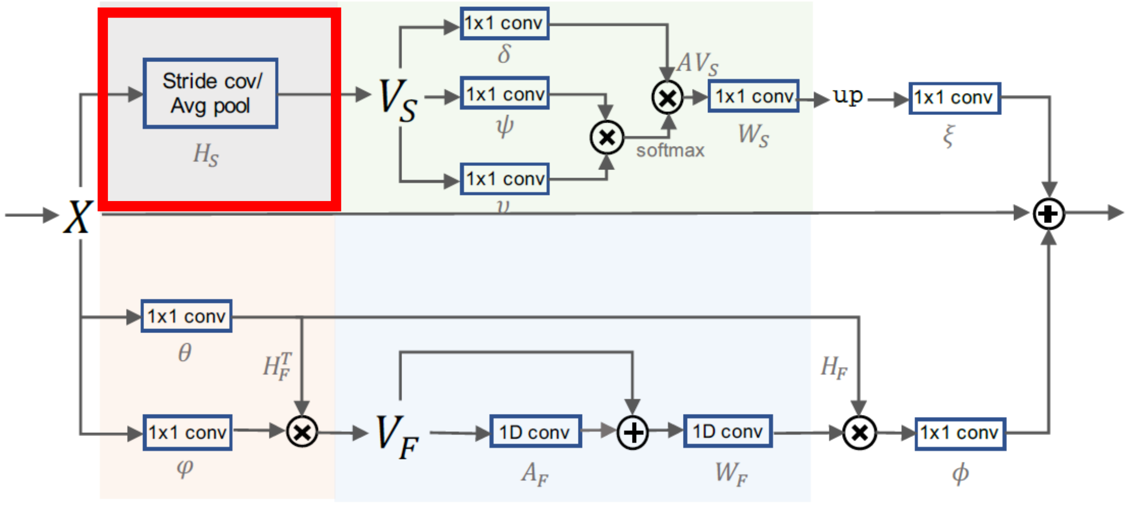
- $X \in R^{N \times D}, N = H \times W$, D is the feature dimension.
- GCN is defined as,

  $\tilde{X} = \sigma(AXW)$
- But graph structure is unknown.



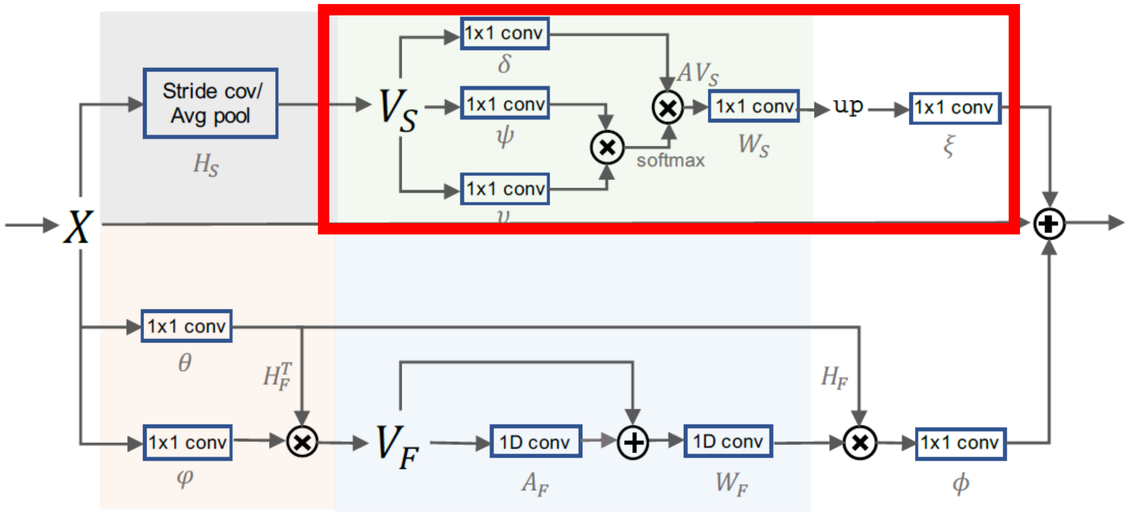
Graph Convolution in Coordinate Space

- Coordinate space projection
- $V_S = H_S X,\ V_S \in R^{\frac{N}{d^2} \times D}$
- Downsampling operator H_S can be,
  1) Parameter-free operation: average pooling
  2) Parametrize to $\log_2(d)$ depth convolution layers with stride=2, kernel size=3
- Nodes of the graph aggregates information from a cluster of pixels



Graph Convolution in Coordinate Space

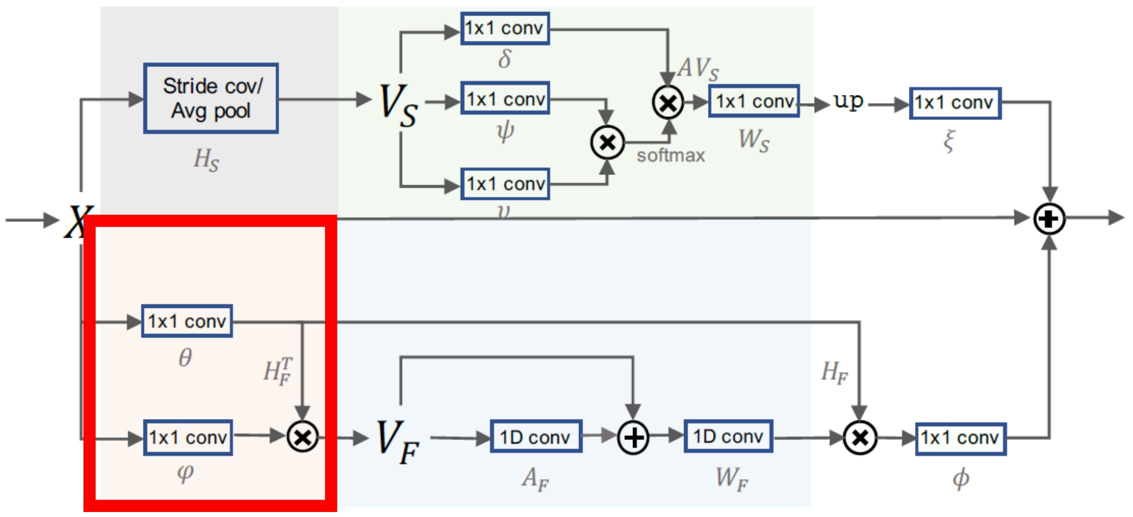
- Coordinate graph convolution
- Build adjacency matrix $A_S \in R^{\left(\frac{H}{d} \times \frac{W}{d}\right)^2}$
- $A_S = \delta(V_S) \cdot \Psi(V_S)^T$
- $M_S = A_S v(V_S) W_S$
- All 1x1 convolution layers δ, Ψ, v change D features to D/2
- Resize M_S to $\tilde{X}_S \in R^{H \times W \times D}$ with upsampling and 1x1 convolution layer
- Computation order in the implementation is different for efficiency
  - O($(HW)^2$) to O(HW)



Graph Convolution in Feature Space

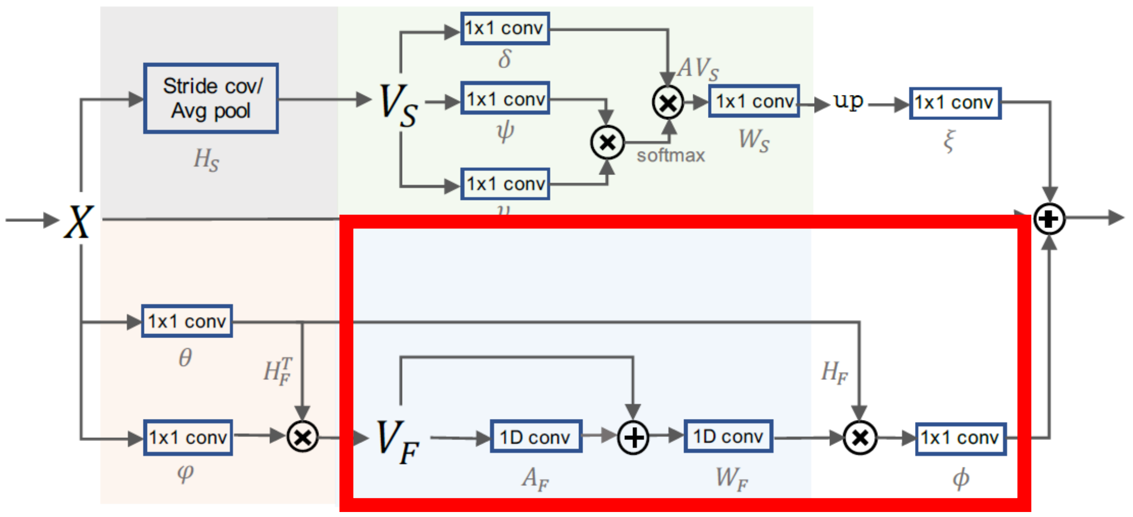
- Feature space projection
- The feature space captures correlation between abstract features in image

  (not a spectral space of graph)
  - Features from later layers are responsive to high level features
- $\theta(\cdot), \varphi(\cdot)$ is 1×1 convolutional layer
- $V_F = H_F^T \theta(X) = \varphi(X)\theta(X), V_F \in R^{D_2 \times D_1}$
- $D_2 = D/4$, $D_1 = D/2$



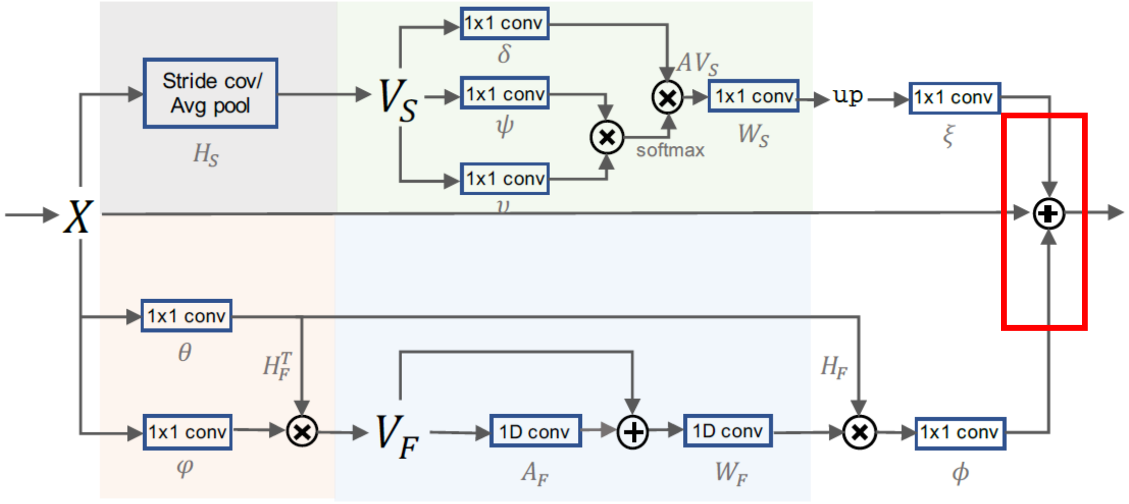
Graph Convolution in Feature Space

- Feature graph convolution
- $M_F = (I - A_F)V_F W_F$
- Laplacian smoothing considered
- $A_F \in R^{D^2 \times D^2}, W_F \in R^{D^1 \times D^1}$ are learnable parameters
- Resize M_F to $\tilde{X}_F \in R^{N \times D}$
- $\tilde{X}_F = \Phi(H_F M_F)$



Graph Convolution in Feature Space

- All extracted features are summed to original input feature
- $\tilde{X} = X + \tilde{X}_S + \tilde{X}_F$
- Proposed module can be easily incorporated to backbone CNN



Other Implementation Details

- Implemented using PyTorch
- FCN ResNet101 is used as backbone model
- Activation functions(ReLU) and batch normalization are not applied in the coordinate graph convolution layers
- Batch size=8, SGD with momentum = 0.9, weight decay = 0.0001, learning rate = $0.01 \times (1 - iter/total\_iter)^{0.9}$
- Cropping (size = 768) and horizontal flipping are randomly applied

Experiment Results

	Backbone	Coord. GCN	Feature GCN	mIoU (%)
Dilated FCN	ResNet-101	✗	✗	75.2
GCN	ResNet-101	✔	✗	78.8
GCN	ResNet-101	✗	✔	79.3
DGCNet	ResNet-101	✔	✔	80.5

- The effect of proposed modules are evaluated on MIoU at the Cityscapes dataset
  - Baseline: 75.2%
  - Only Coordinate Space GCN: 78.8%
  - Only Feature Space GCN: 79.3%
  - Both GCNs: 80.5%

Experiment Results

Downsample rate	d=4	d=8	d=16
Avg. pooling	80.2	80.5	80.5
Stride conv.	80.0	80.5	80.5

- Downsampling influence on the performance
  - Average Pooling achieved similar performance as stride convolution downsampling
  - Both strategies were robust to downsampling rate(d)
  - d=8 is used for other experiment cases

Experiment Results

	OHEM	Multi-grid	MS	mIoU (%)
DGCNet	✘	✘	✘	79.5
DGCNet	✔	✘	✘	79.8
DGCNet	✔	✔	✘	80.5
DGCNet	✔	✔	✔	81.8

- Additional “tricks” applied to improve segmentation performance
  - Online Hard Example Mining: the loss is only computed on the K highest loss pixels in the image
  - Multi-Grid: the last convolutional filters applied with different dilation rates
  - Multi-Scale: averaging segmentation heat maps from diverse scales during inference
- The best performance achieved when all methods above were applied

Experiment Results

Method	Backbone	mIoU (%)
PSPNet [55]	ResNet-101	78.4
PSANet [56]	ResNet-101	78.6
OCNet [50]	ResNet-101	80.1
DGCNet (Ours) †	ResNet-101	**80.9**
SAC [54]	ResNet-101	78.1
AAF [18]	ResNet-101	79.1
BiSeNet [46]	ResNet-101	78.9
PSANet [56]	ResNet-101	80.1
DFN [47]	ResNet-101	79.3
DepthSeg [20]	ResNet-101	78.2
DenseASPP [45]	ResNet-101	80.6
GloRe [8]	ResNet-101	80.9
DANet [14]	ResNet-101	81.5
OCNet [50]	ResNet-101	81.7
DGCNet (Ours) ‡	ResNet-50	80.8
DGCNet (Ours) ‡	ResNet-101	**82.0**

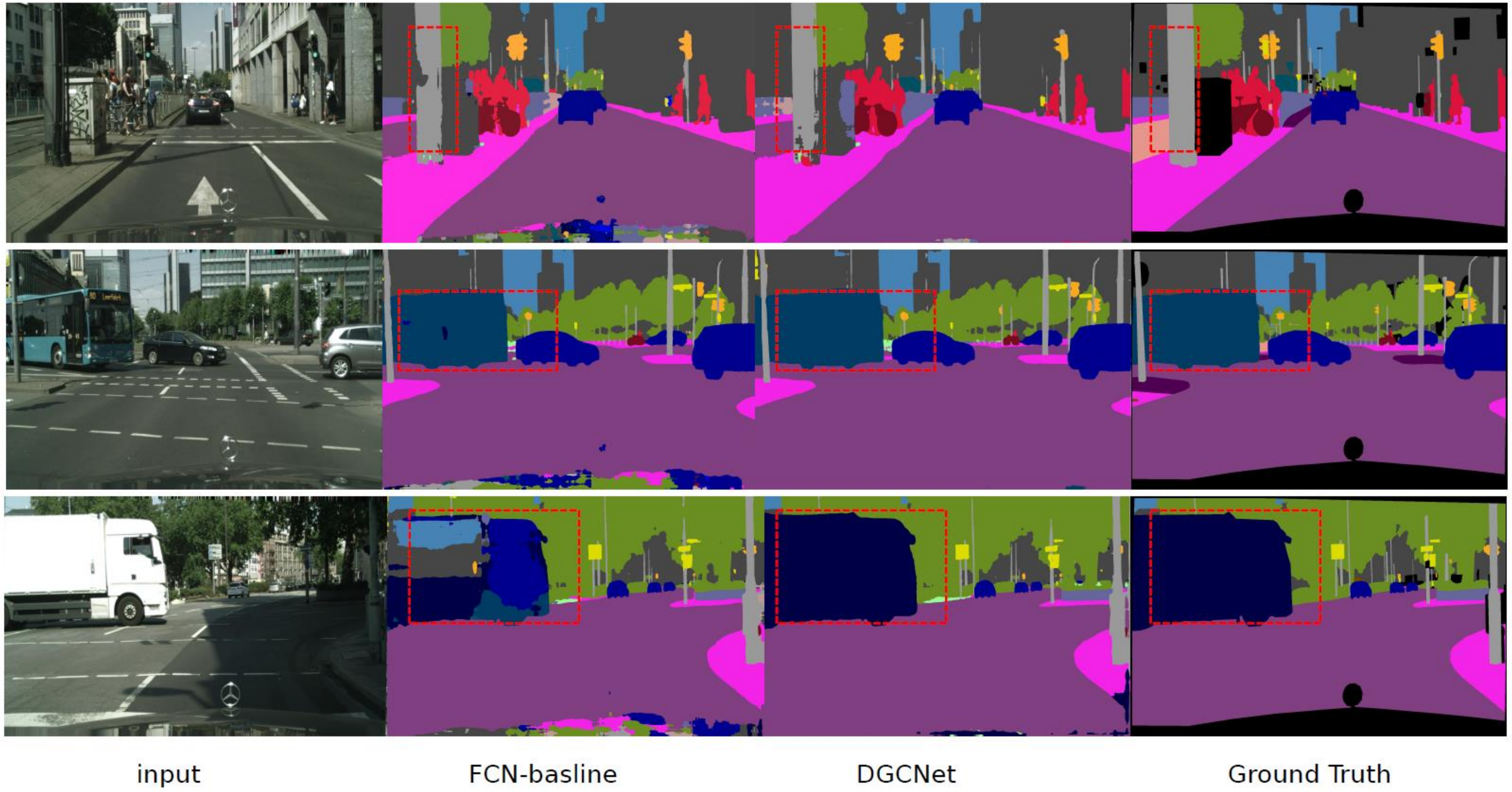
†: trained only on train-fine set.
‡: trained on train-fine and val-fine sets.

- Compared with other SOTA models on the Cityscapes dataset
  - Only trained on training set: 80.9%
  - Trained on both training and validation sets: 82%
  - The proposed model outperformed other SOTA methods

Experiment Results

Methods	Mean IoU	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
DeepLab-v2 [6]	70.4	97.9	81.3	90.3	48.8	47.4	49.6	57.9	67.3	91.9	69.4	94.2	79.8	59.8	93.7	56.5	67.5	57.5	57.7	68.8
RefineNet [29]	73.6	98.2	83.3	91.3	47.8	50.4	56.1	66.9	71.3	92.3	70.3	94.8	80.9	63.3	94.5	64.6	76.1	64.3	62.2	70
GCN [35]	76.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DUC [42]	77.6	98.5	85.5	92.8	58.6	55.5	65	73.5	77.9	93.3	72	95.2	84.8	68.5	95.4	70.9	78.8	68.7	65.9	73.8
ResNet-38 [44]	78.4	98.5	85.7	93.1	55.5	59.1	67.1	74.8	78.7	93.7	72.6	95.5	86.6	69.2	95.7	64.5	78.8	74.1	69	76.7
PSPNet [55]	78.4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
BiSeNet [48]	78.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PSANet [56]	80.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DenseASPP [45]	80.6	**98.7**	87.1	93.4	60.7	62.7	65.6	74.6	78.5	93.6	72.5	95.4	86.2	71.9	96.0	**78.0**	90.3	80.7	69.7	76.8
GloRe [8]	80.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DANet [14]	81.5	98.6	86.1	**93.5**	56.1	63.3	69.7	77.3	**81.3**	93.9	72.9	95.7	87.3	72.9	96.2	76.8	89.4	**86.5**	**72.2**	**78.2**
Ours	**82.0**	**98.7**	**87.4**	**93.9**	**62.4**	**63.4**	**70.8**	**78.7**	**81.3**	**94.0**	**73.3**	**95.8**	**87.8**	**73.7**	**96.4**	76.0	**91.6**	81.6	71.5	**78.2**

Experiment Results



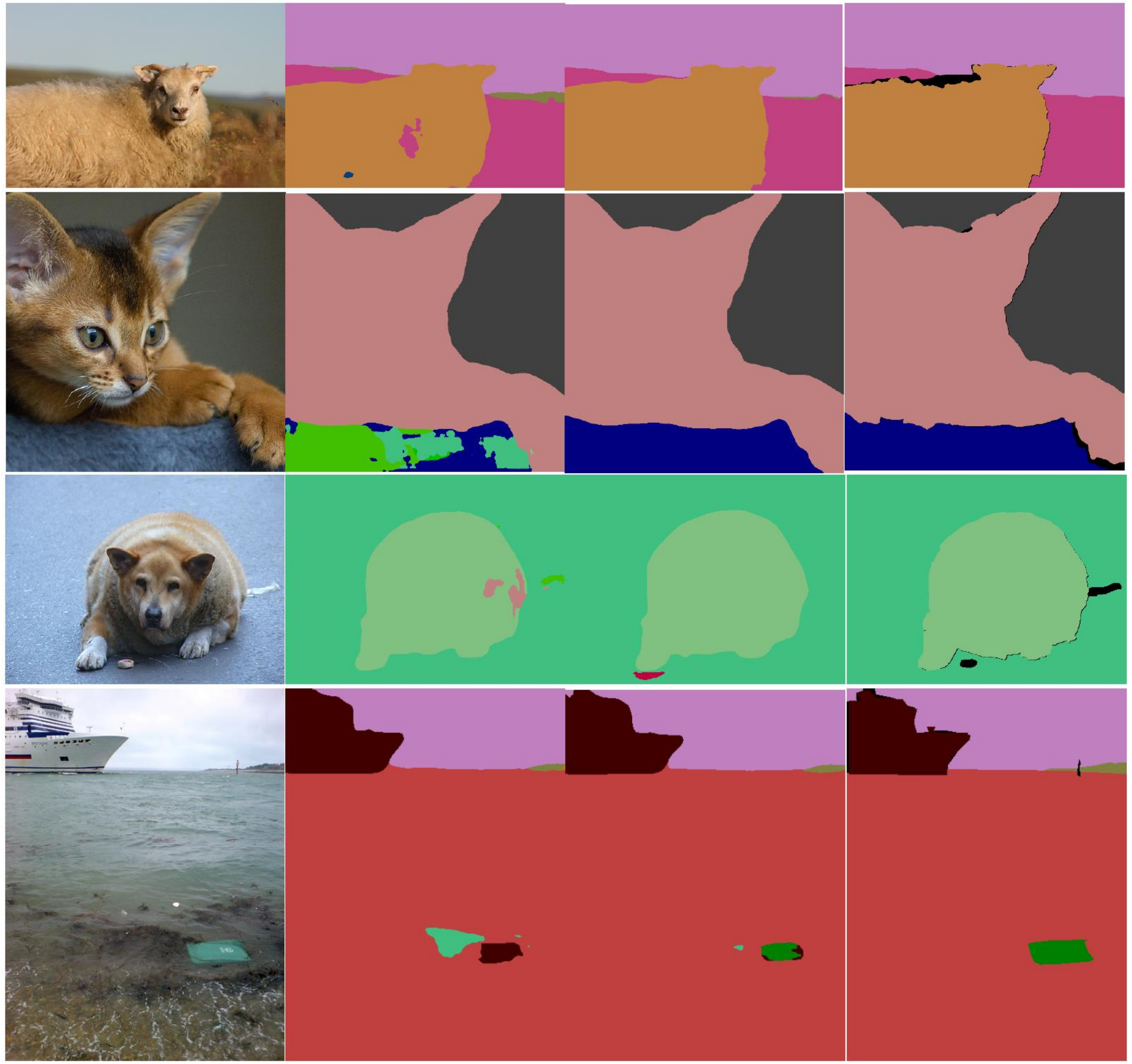
Experiment Results

Method	Backbone	mIoU (%)
FCN8-s [32]	VGG-16	37.8
HO CRF [1]	VGG-16	41.3
Piecewise [28]	VGG-16	43.3
DeepLab-v2 (COCO) [6]	ResNet-101	45.7
RefineNet [29]	ResNet-101	47.3
PSPNet [55]	ResNet-101	47.8
Ding *et al.* [11]	ResNet-101	51.6
EncNet [52] (SS)	ResNet-50	49.0
EncNet [52] (MS)	ResNet-101	51.7
SGR [26]	ResNet-101	52.5
DANet [14]	ResNet-50	50.1
DANet [14]	ResNet-101	52.6
Dilated FCN baseline	ResNet-50	44.3
DGCNet (SS)	ResNet-50	50.1
DGCNet (SS)	ResNet-101	**53.0**
DGCNet (MS)	ResNet-101	**53.7**

SS: Single scale. MS: Multi scale

- Compared with other SOTA models on the Pascal Context dataset
  - The proposed model outperformed again
  - The proposed model with Multi-Scale inference performed the best

Experiment Results



Input EncNet DGCNet Ground Truth

Final Report: Model Reproduction

- Reproduction model specification difference with paper’s
  - ResNet-101 backbone was pretrained by ImageNet according to the paper, but COCO 2017 was used for the reproduction
  - Total epoch = 180, batch size = 1 in reproduction
  - No ‘tricks’ mentioned on the paper were applied. Classifying convolutional layers of FCN ResNet-101 model offered by Torchvision were randomly initialized before training

Final Report: Model Reproduction

- Reproduction workstation specification
  - Ubuntu 18.04
  - Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
  - GeForce RTX 2080 TI
  - CUDA 10.1, cuDNN 7
  - Python 3.6.9
  - Pytorch==1.5.0+cu101, Torchvision==0.6.0+cu101

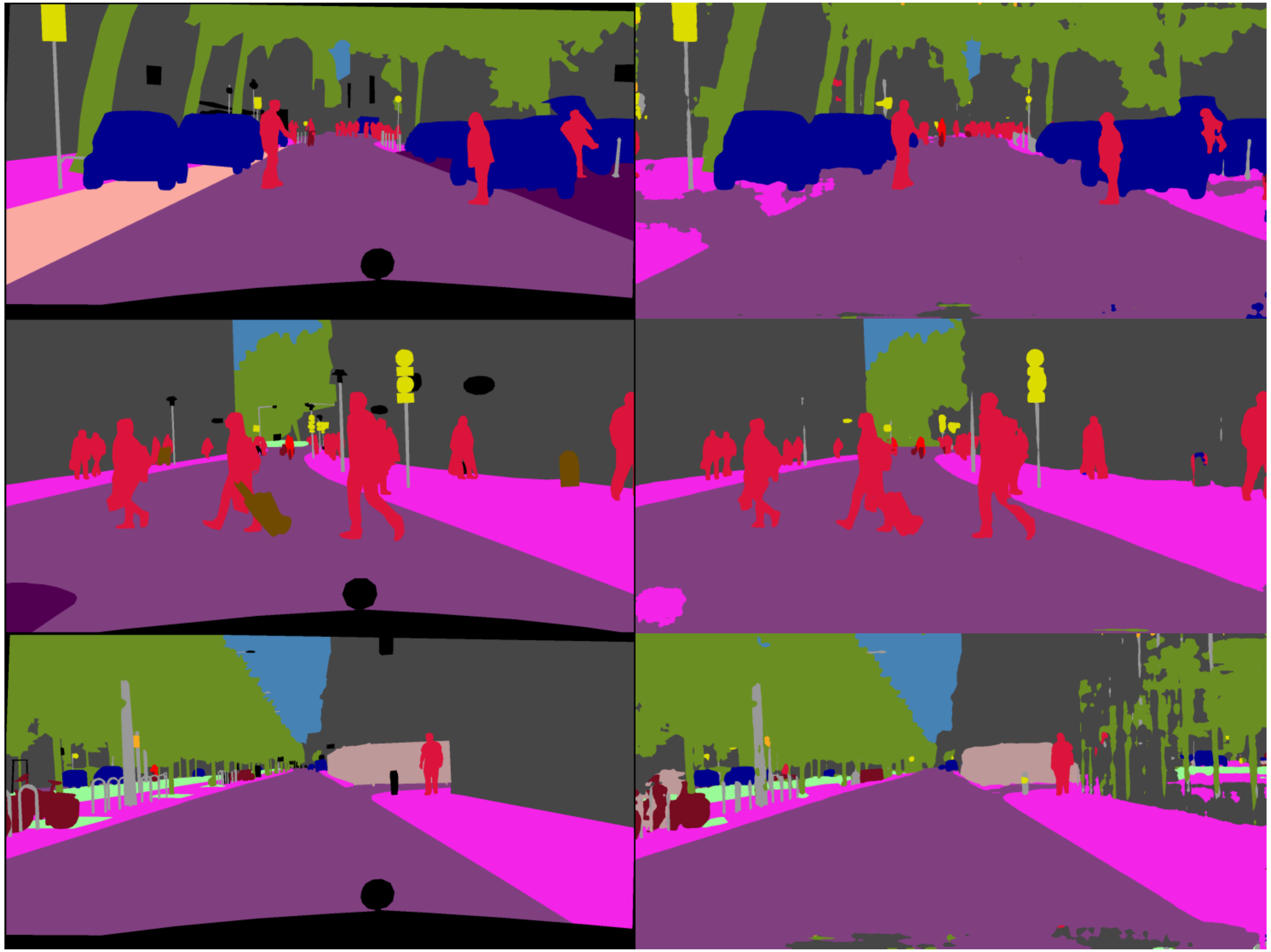
Final Report: Model Reproduction

- Reproduction result
  - Tested on the Cityscapes dataset
  - Object classes are defined according to paper’s classification
  - Model was tested on the validation set because test set of Cityscapes are only offered to who officially apply to challenges
  - MIoU = 0.6440

Class	IoU
Road	0.9742
Sidewalk	0.8045
Building	0.8945
Wall	0.3453
Fence	0.3361
Pole	0.5725
Traffic light	0.6521
Traffic sign	0.7270
Vegetation	0.9124
Terrain	0.6017
Sky	0.9357
Person	0.7683
Rider	0.4470
Car	0.9234
Truck	0.5737
bus	0.5648
Train	0.0997
Motorcycle	0.3918
Bicycle	0.7123
MIoU	**0.6440**

Final Report: Model Reproduction

Ground Truth | Reproduced Image



Final Report: Model Reproduction

- Github url: https://github.com/qpwodlsqp/dgcn
- Pretrained weight download link
- Manual
  - Cityscapes datasets should be downloaded, and a user need to sign up to access the dataset
  - Create */cityscapes* directory on the repository, and unzip datasets in that directory
    - Raw images and segmentation labels are separated in different zip files.
  - Create /model directory to locate the pretrained weight or store weights during training
  - Training
    - foo@bar:~dgcn$ python3 train.py
  - Inference
    - You should input toponym and number parts of dataset, which is common parts of raw images' name and labels' name
    - foo@bar:~dgcn$ python3 infer.py aachen_000000_000019
  - Test
    - foo@bar:~dgcn$ python3 test.py