Lecture 12: CNNs – Case Studies

Xuming He
SIST, ShanghaiTech
Fall, 2019

Outline

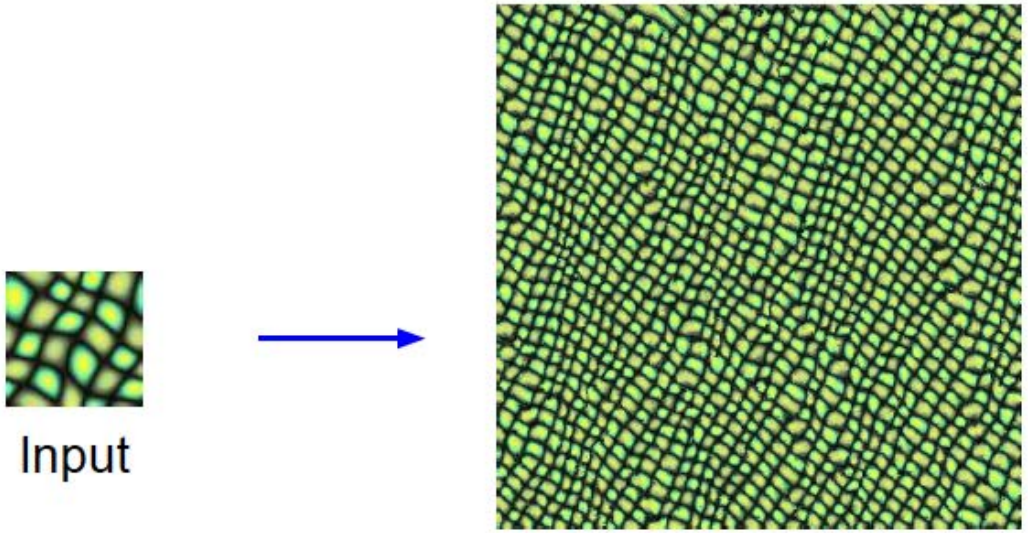
- Image synthesis
  - Neural texture synthesis and style transfer
  - Novel view synthesis [CVPR 2018]
- Visual relations
  - Human-object interaction detection [ICCV 2019]
  - Semantic correspondence estimation [ICCV 2019]
- Semantic segmentation
  - Boundary-aware object instance segmentation [CVPR 2017]
  - Latent graph neural network for visual recognition [ICML 2019]

Acknowledgement: Roger Grosse's CSC231 and Feifei Li et al's cs231n notes

Texture Synthesis

- Problem setup

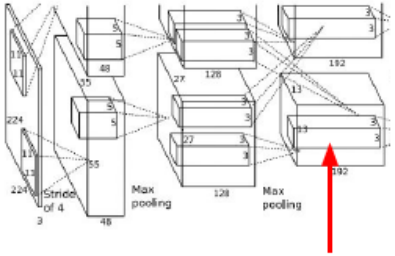
Given a sample patch of some texture, can we generate a bigger image of the same texture?

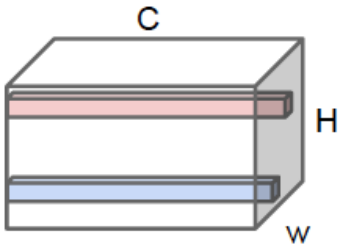


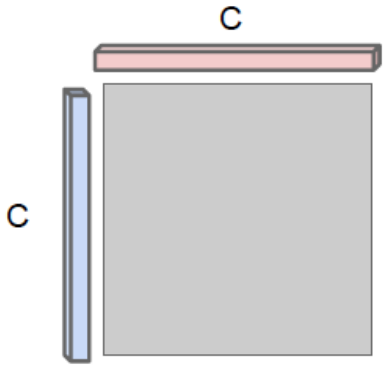
Texture Synthesis

- CNN-based modeling of image statistics

This image is in the public domain.







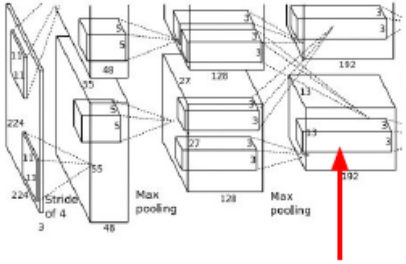
Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

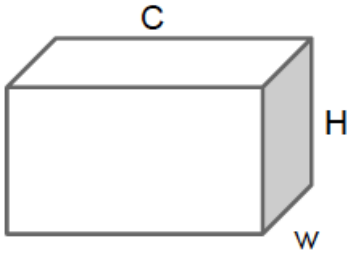
Outer product of two C-dimensional vectors gives C x C matrix measuring co-occurrence

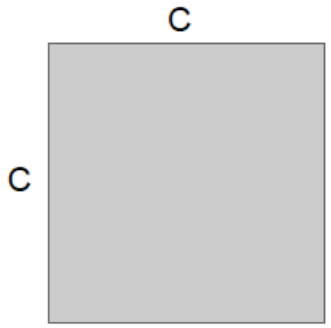
Texture Synthesis

- CNN-based modeling of image statistics

This image is in the public domain.







Gram Matrix

Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

Outer product of two C-dimensional vectors gives C x C matrix measuring co-occurrence

Average over all HW pairs of vectors, giving **Gram matrix** of shape C x C

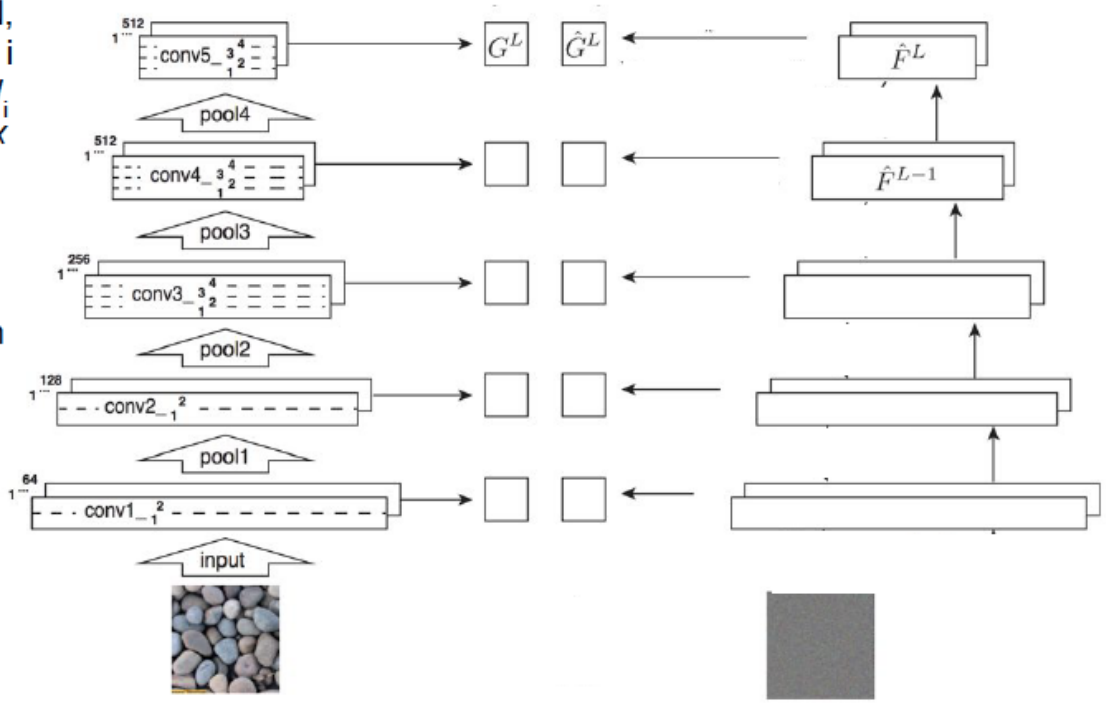
Texture Synthesis

- Neural texture synthesis

1. Pretrain a CNN on ImageNet (VGG-19)
2. Run input texture forward through CNN, record activations on every layer; layer i gives feature map of shape $C_i \times H_i \times W_i$
3. At each layer compute the *Gram matrix* giving outer product of features:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \text{ (shape } C_i \times C_i)$$

4. Initialize generated image from random noise
5. Pass generated image through CNN, compute Gram matrix on each layer



Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015

Texture Synthesis

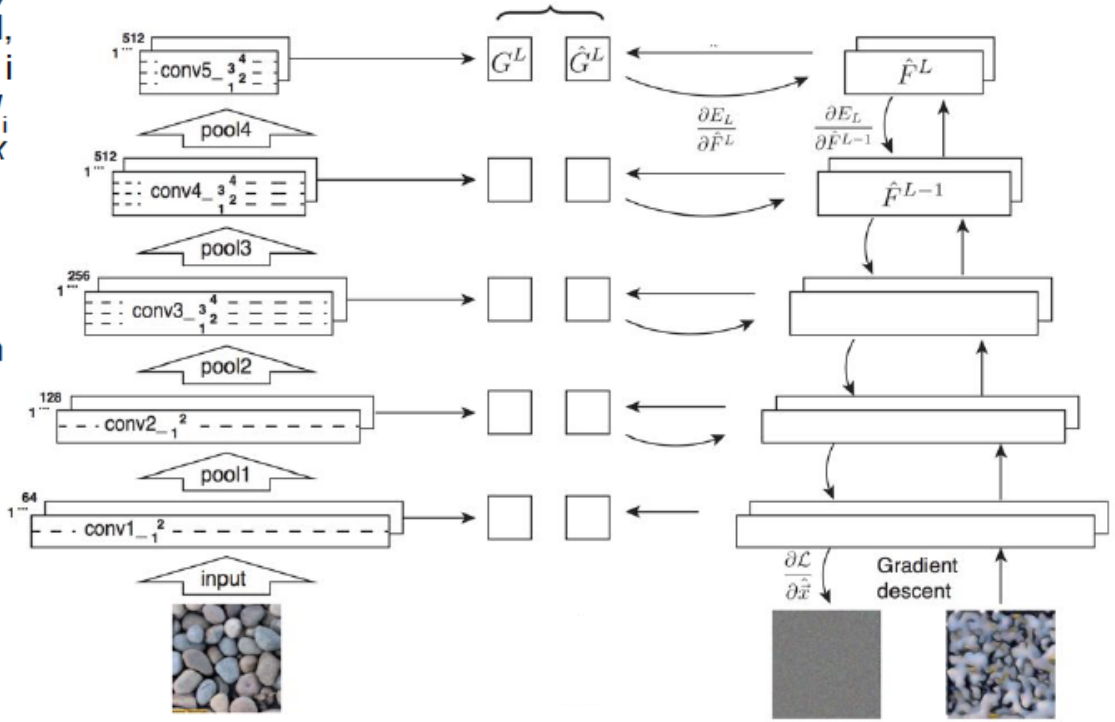
- Neural texture synthesis

1. Pretrain a CNN on ImageNet (VGG-19)
2. Run input texture forward through CNN, record activations on every layer; layer i gives feature map of shape $C_i \times H_i \times W_i$
3. At each layer compute the *Gram matrix* giving outer product of features:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \text{ (shape } C_i \times C_i)$$

4. Initialize generated image from random noise
5. Pass generated image through CNN, compute Gram matrix on each layer
6. Compute loss: weighted sum of L2 distance between Gram matrices
7. Backprop to get gradient on image
8. Make gradient step on image
9. GOTO 5

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - \hat{G}^l_{ij}\right)^2 \qquad \mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$$



Neural Style Transfer

- Problem setup

Content Image

This image is licensed under CC-BY 3.0

+

Style Image

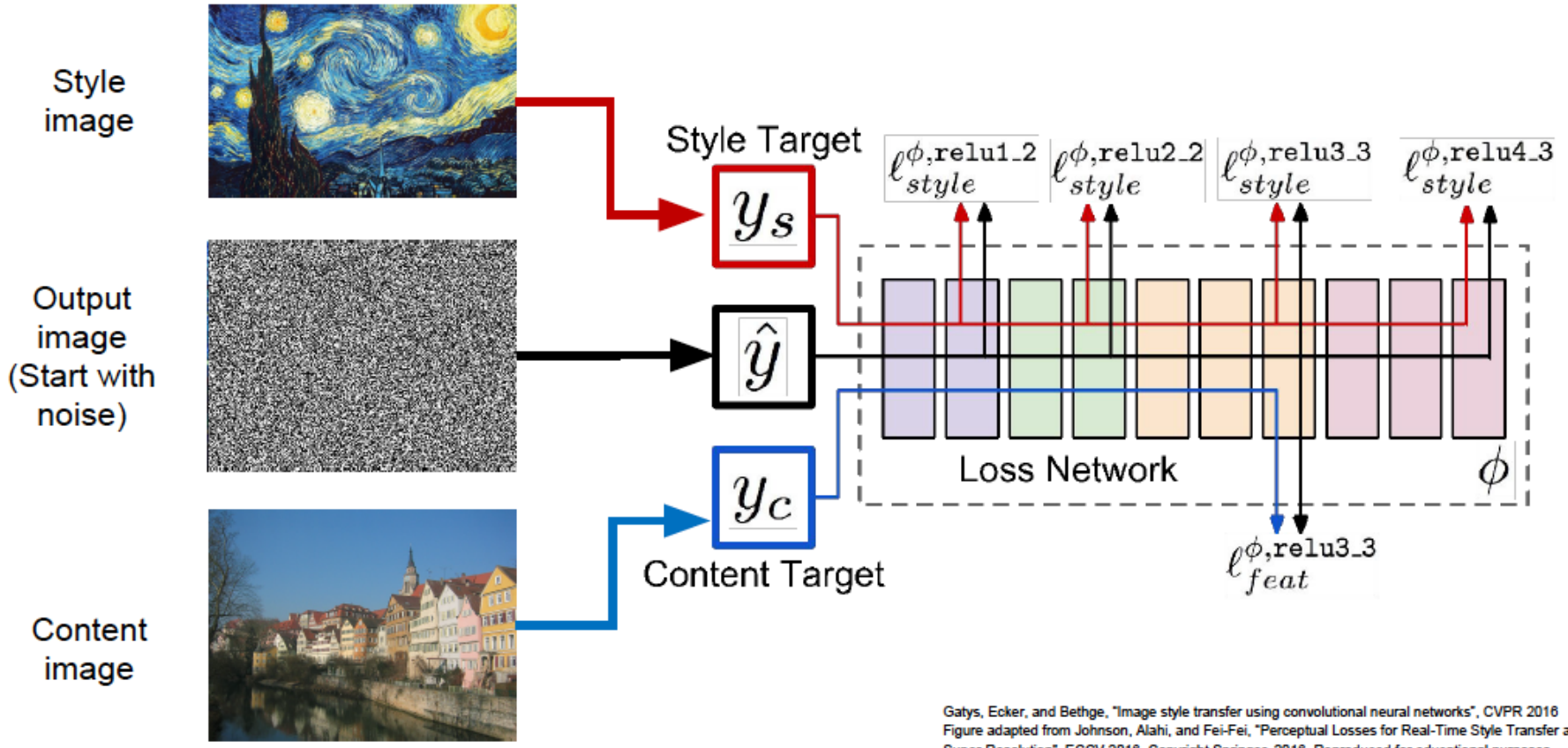
Starry Night by Van Gogh is in the public domain

=

Style Transfer!

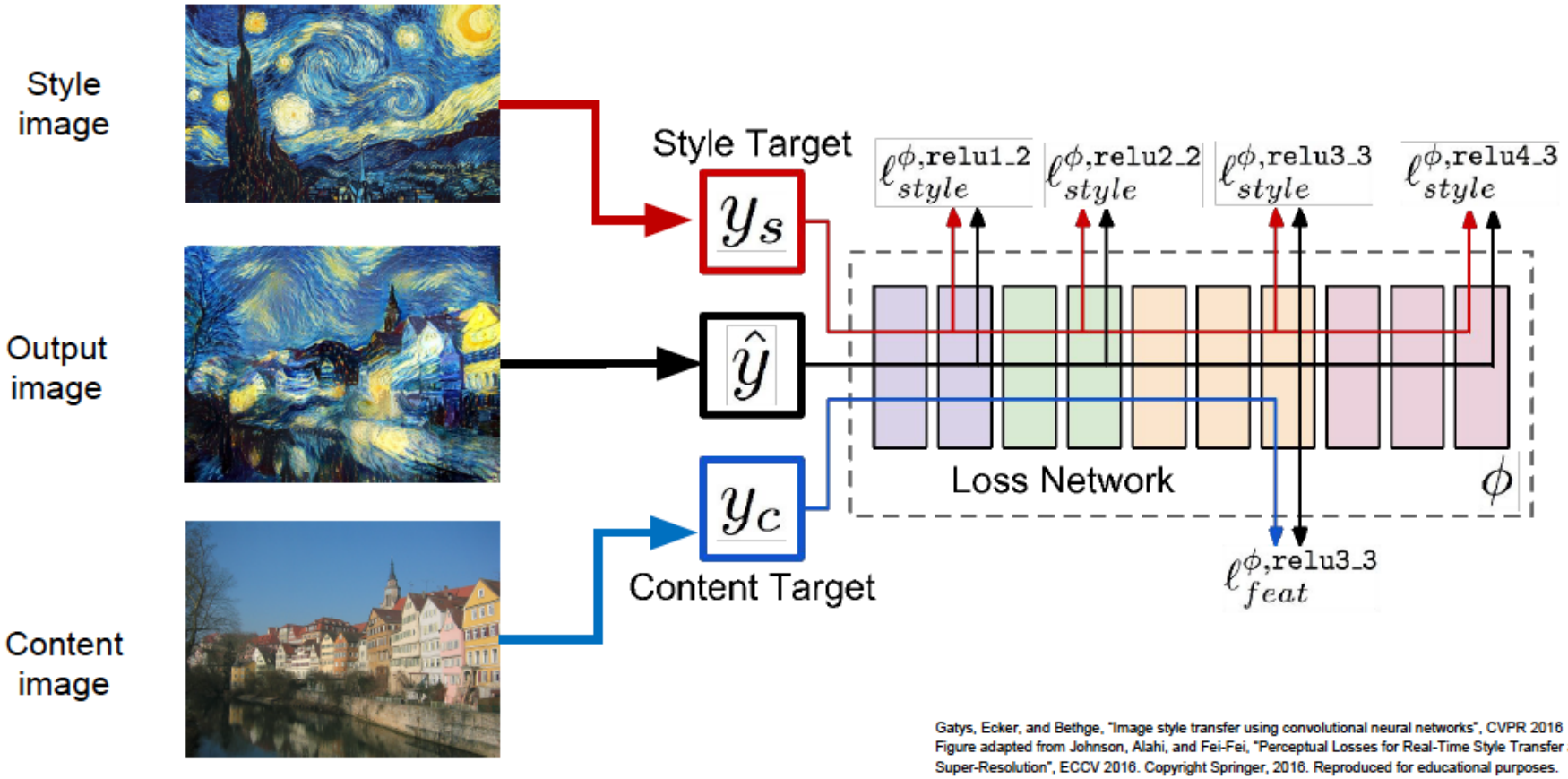
This image copyright Justin Johnson, 2015. Reproduced with permission.

Neural Style Transfer



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016
Figure adapted from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016. Copyright Springer, 2016. Reproduced for educational purposes.

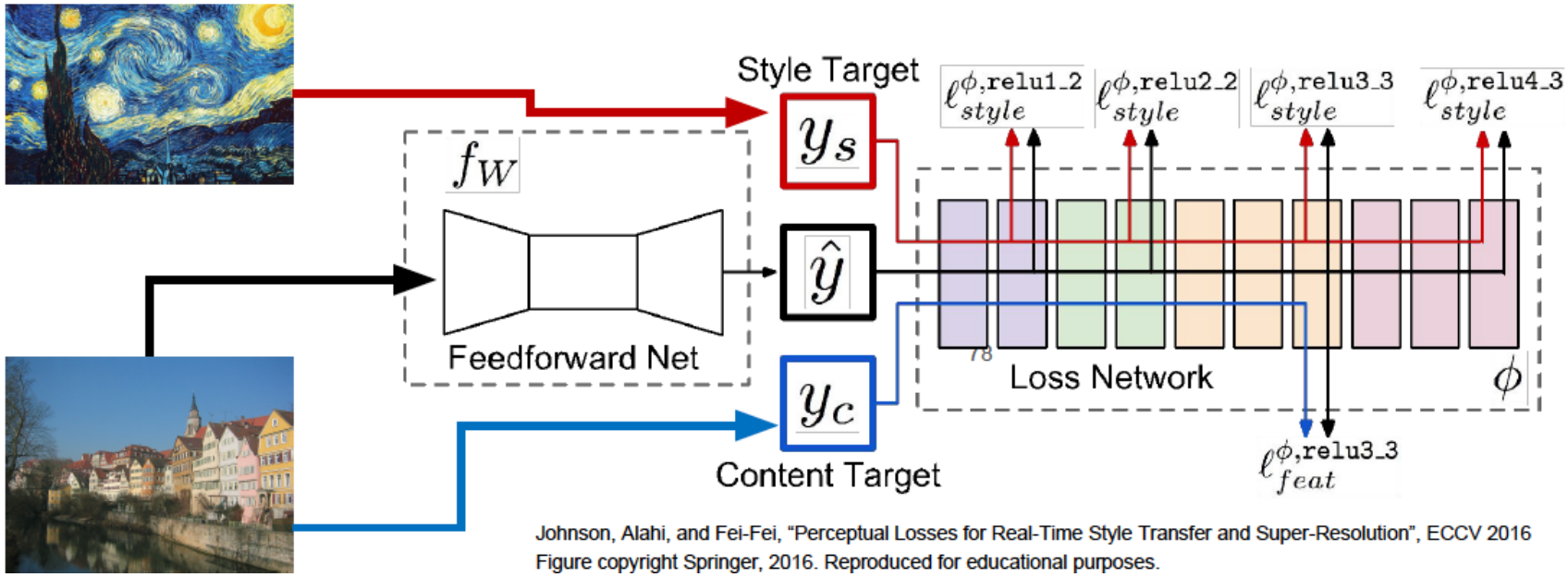
Neural Style Transfer



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016
Figure adapted from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016. Copyright Springer, 2016. Reproduced for educational purposes.

Fast Style Transfer

(1) Train a feedforward network for each style
(2) Use pretrained CNN to compute same losses as before
(3) After training, stylize images using a single forward pass



Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016
Figure copyright Springer, 2016. Reproduced for educational purposes.

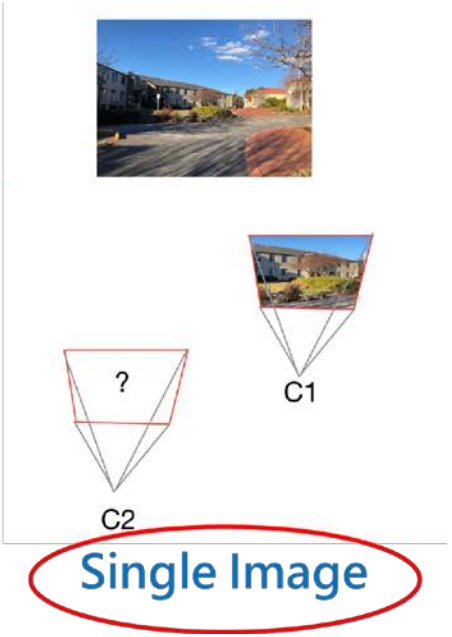
Outline

- Image synthesis
  - Neural texture synthesis and style transfer
  - Novel view synthesis [CVPR 2018]
- Visual relations
  - Human-object interaction detection [ICCV 2019]
  - Semantic correspondence estimation [ICCV 2019]
- Semantic segmentation
  - Boundary-aware object instance segmentation [CVPR 2017]
  - Latent graph neural network for visual recognition [ICML 2019]

Acknowledgement: Roger Grosse's CSC231 and Feifei Li et al's cs231n notes

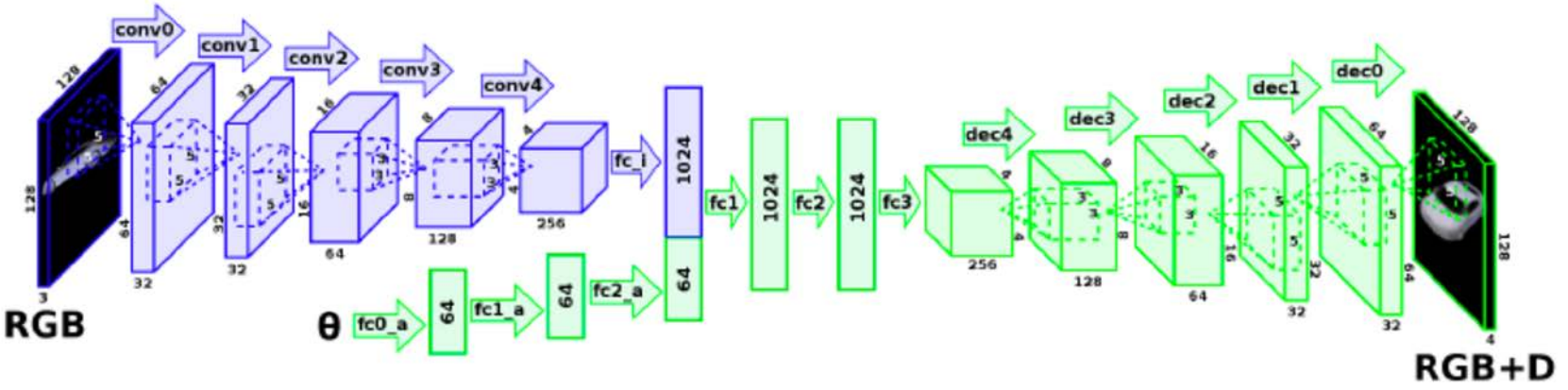
Problem Setting

- What is view synthesis?
  - Generate an image from a novel (virtual) view point



Previous Methods

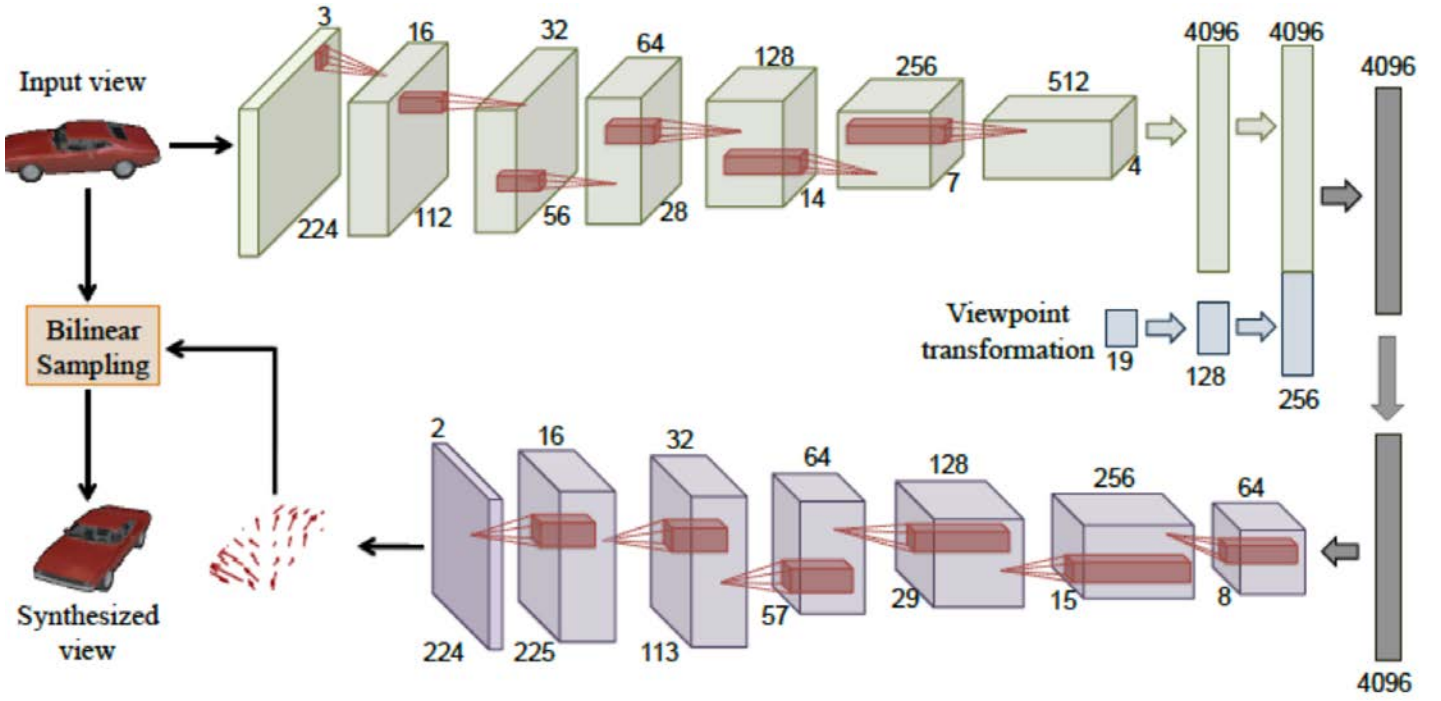
- Previous approaches
  - [1] Multi-view 3D Models from Single Images with a Convolutional Network



[1]

Previous Methods

- Previous approaches
  - [2] View Synthesis by Appearance Flow



[2]

Motivations

- Limitations
  - Working not so well for complex scenes due to diverse 3D geometric structures
  - Generating blurry effects and/or distortions

Source [1] [2]

Our Objective

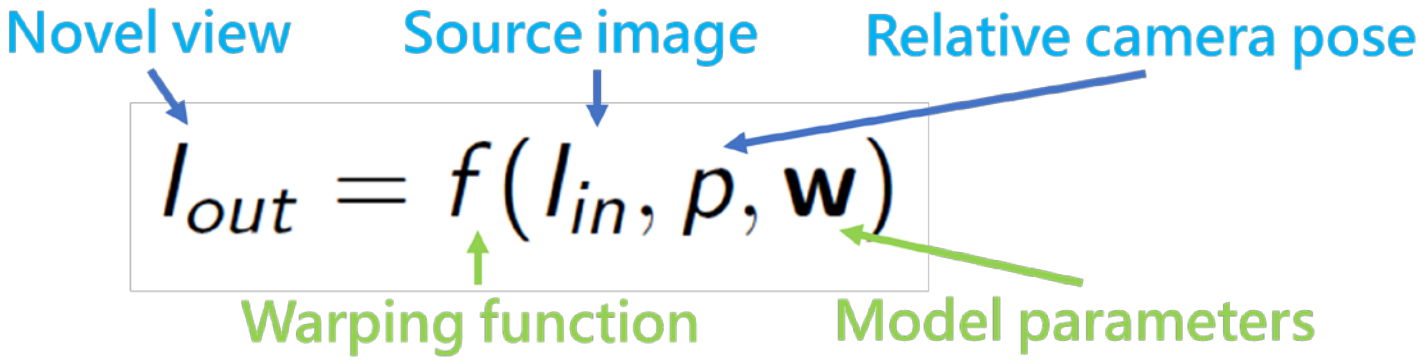
- To generate novel-view images that
  - Consistent 3D geometric structures
  - Much less blurry effects and/or distortions

Source

Our result

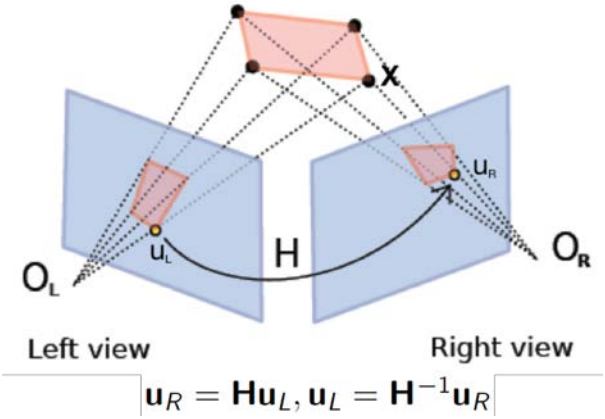
Background

- Problem Formulation



Main Idea

- A complex scene can be approximated by a finite number of planes
- Transformation of planes: Homography



$$\mathbf{u}_R = \mathbf{H}\mathbf{u}_L, \mathbf{u}_L = \mathbf{H}^{-1}\mathbf{u}_R$$

Main Idea

- We propose a new warping function:

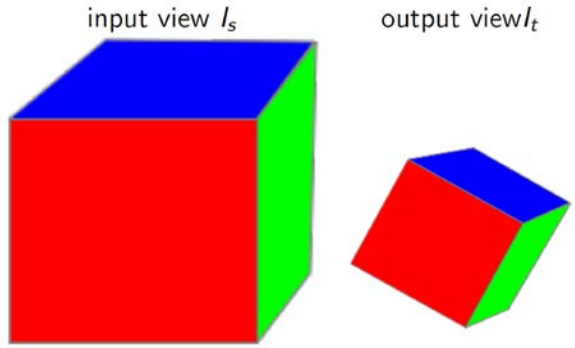
$$I_{out} = \sum_{t=1}^{t=q} M_t(I_{in}, p, \mathbf{W}_s)\; g_t(I_{in}, p, \mathbf{W}_g),$$

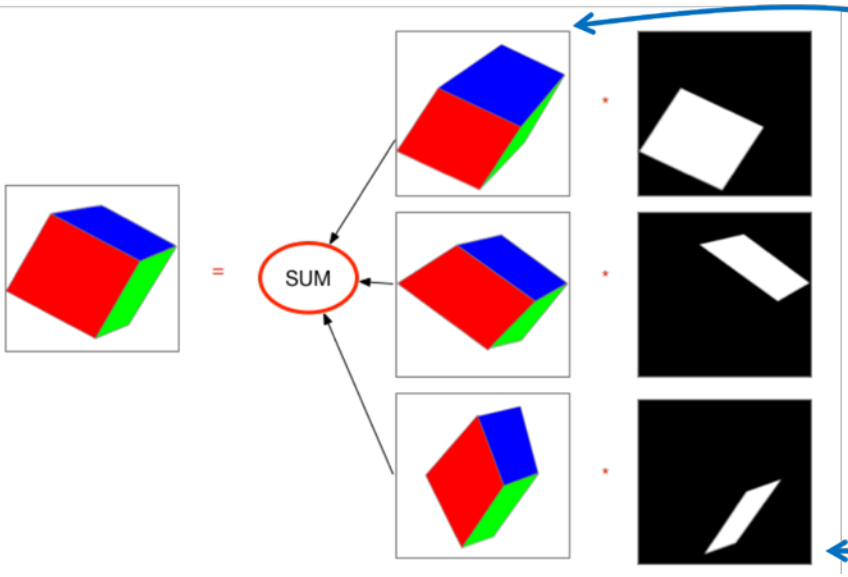
q planes

Homography for plane t

Spatial region for plane t

A Toy Example





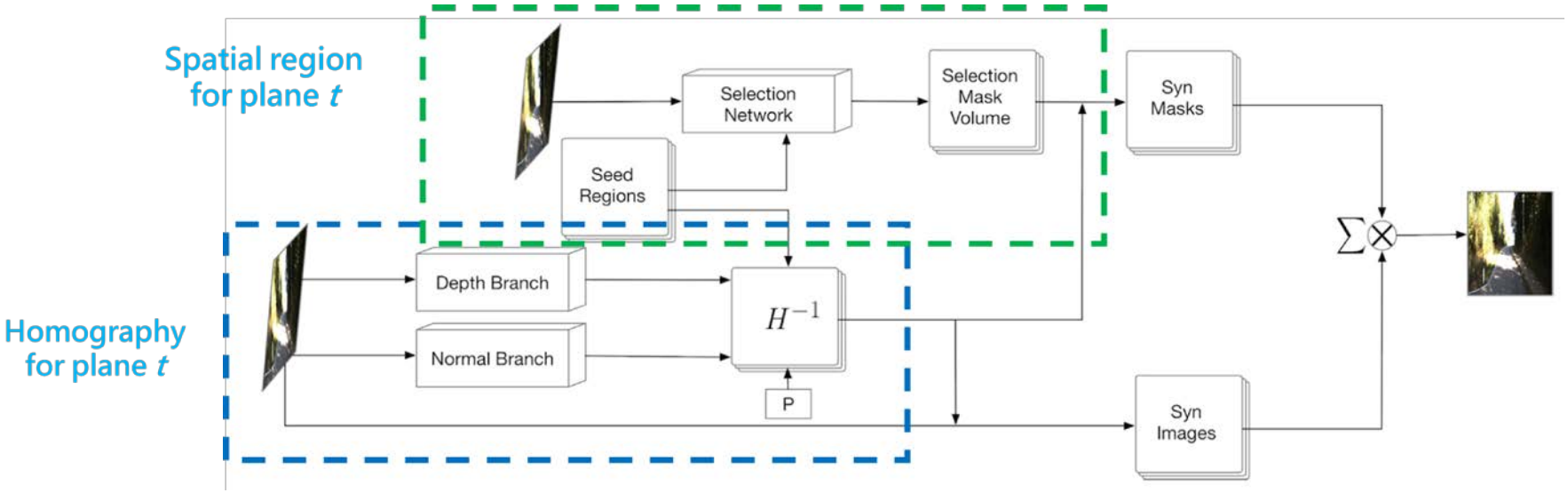
$$I_{out} = \sum_{t=1}^{t=q} M_t(I_{in}, p, \mathbf{W}_s)\, g_t(I_{in}, p, \mathbf{W}_g),$$

Homography for plane t

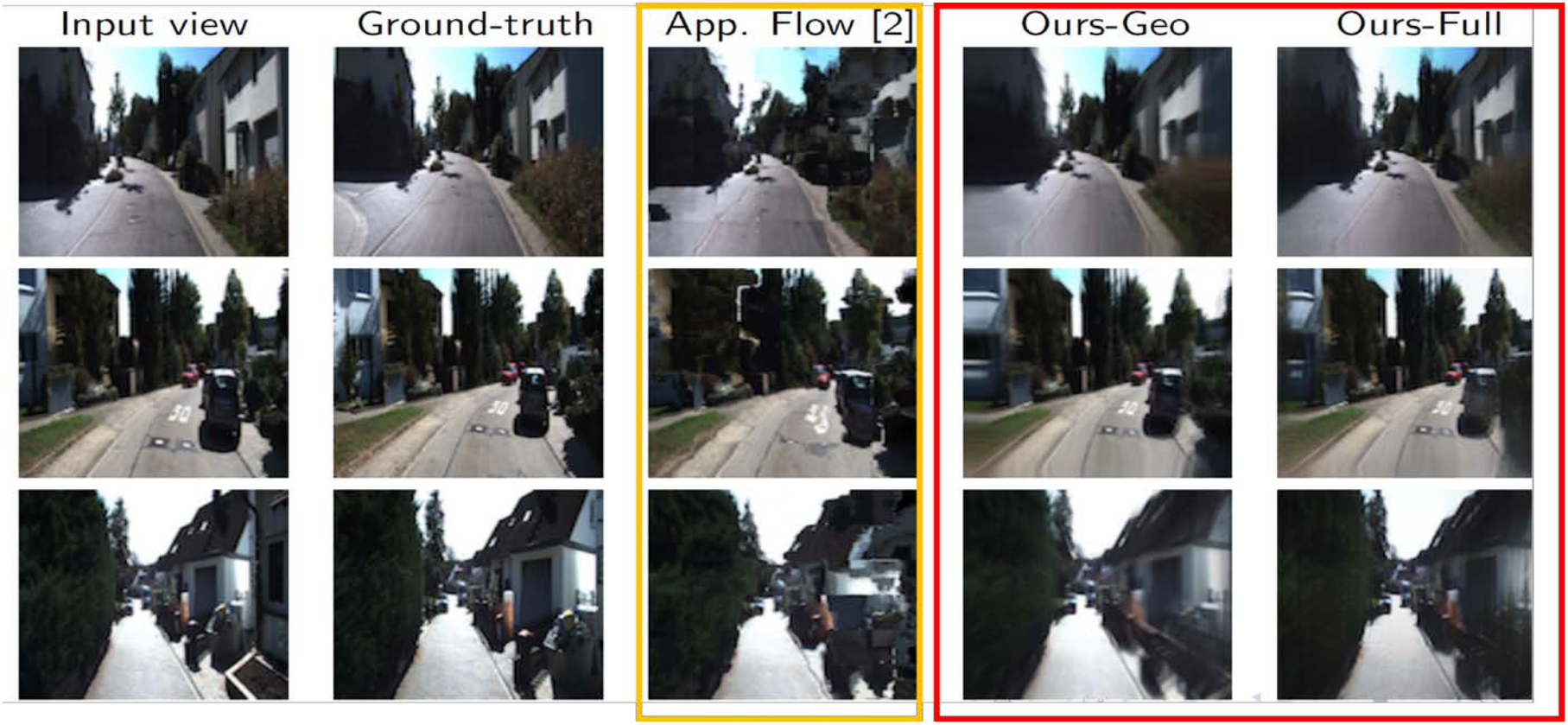
Spatial region for plane t

Our Model

- Region-aware geometric-transform network



Results



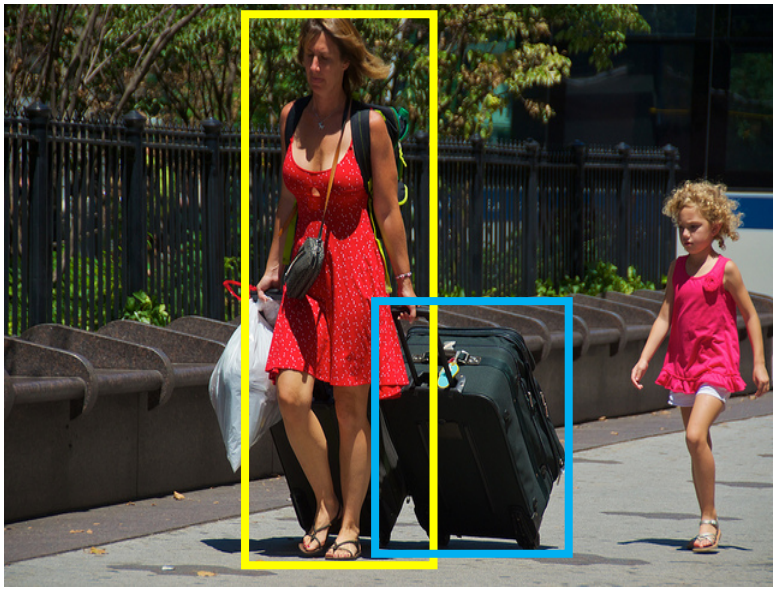
Outline

- Image synthesis
  - Neural texture synthesis and style transfer
  - Novel view synthesis [CVPR 2018]
- Visual relations
  - Human-object interaction detection [ICCV 2019]
  - Semantic correspondence estimation [ICCV 2019]
- Semantic segmentation
  - Boundary-aware object instance segmentation [CVPR 2017]
  - Latent graph neural network for visual recognition [ICML 2019]

Acknowledgement: Roger Grosse's CSC231 and Feifei Li et al's cs231n notes

Problem Definition

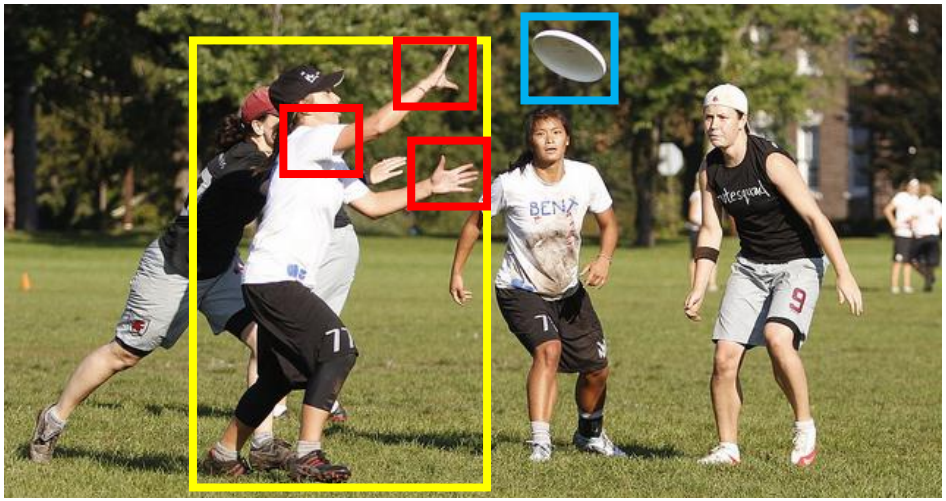
- Human-Object Interaction Detection



carry suitcase

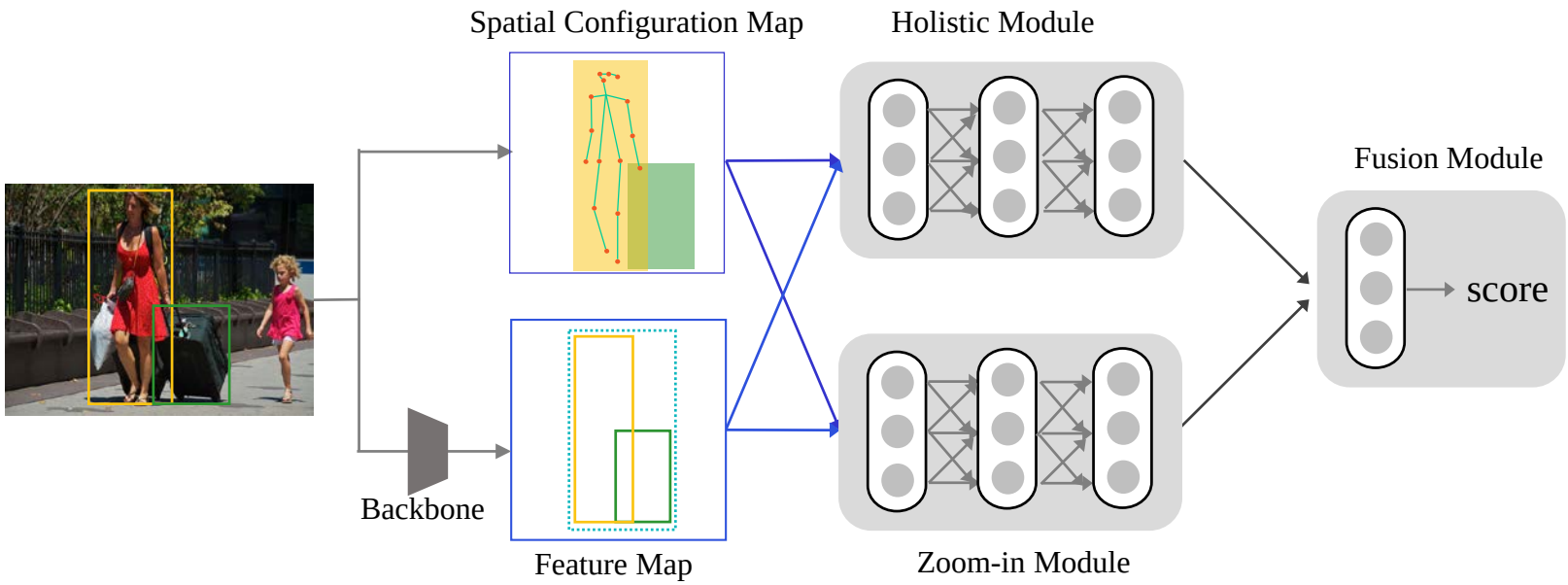
Motivation

- Utilize pose to capture detailed semantic part cues.

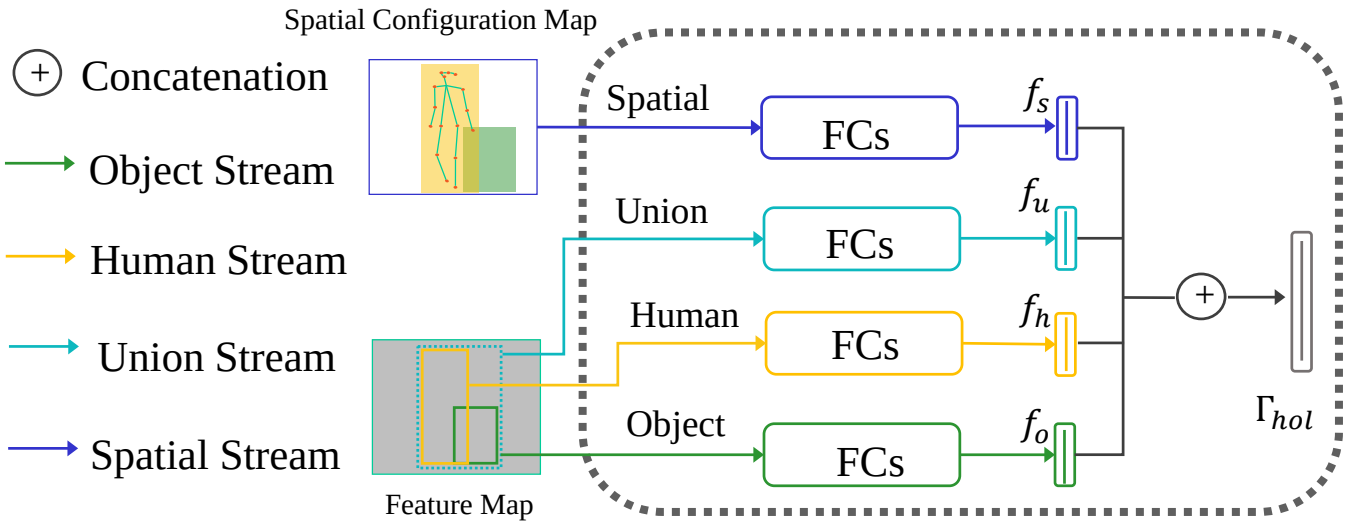


catch or throw??

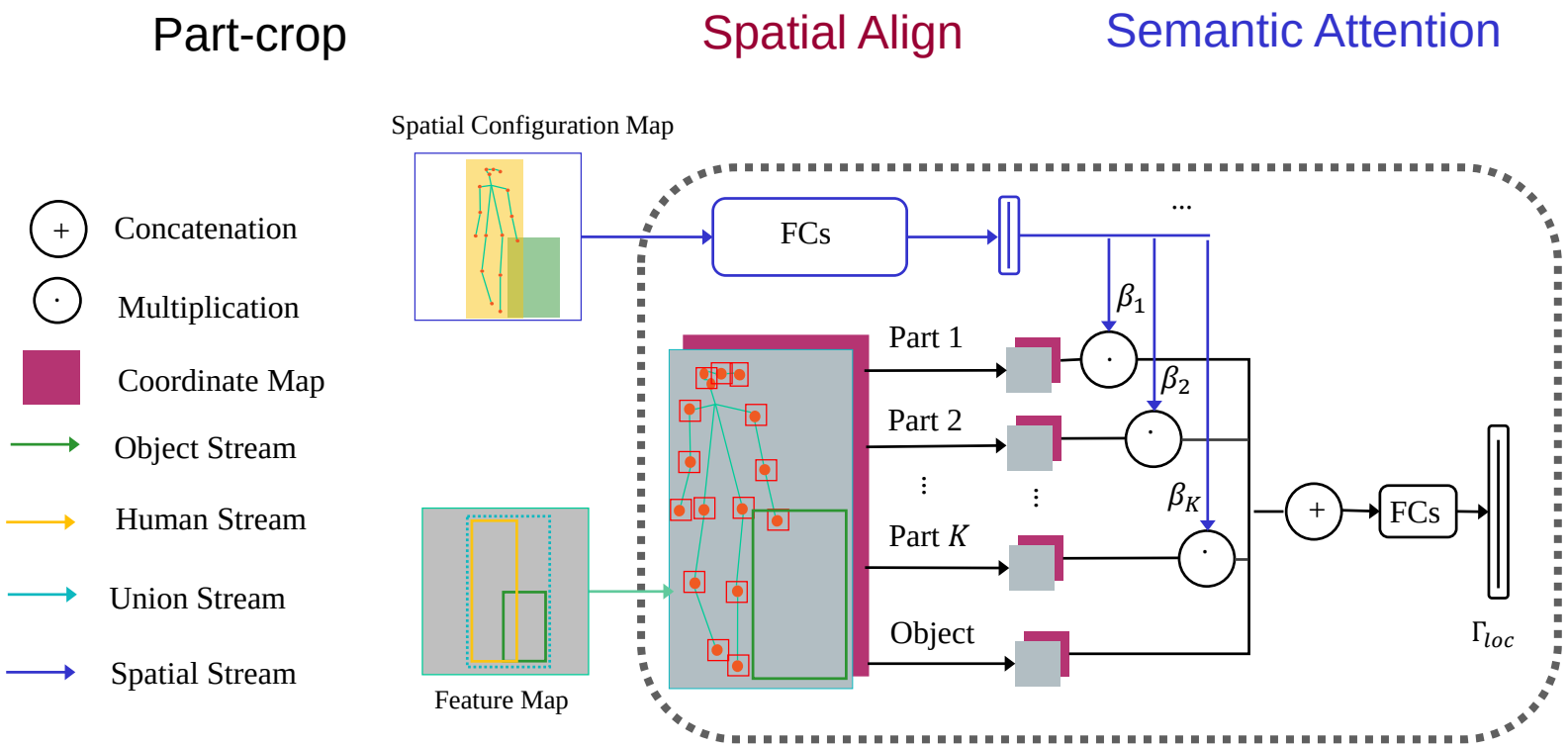
Model overview



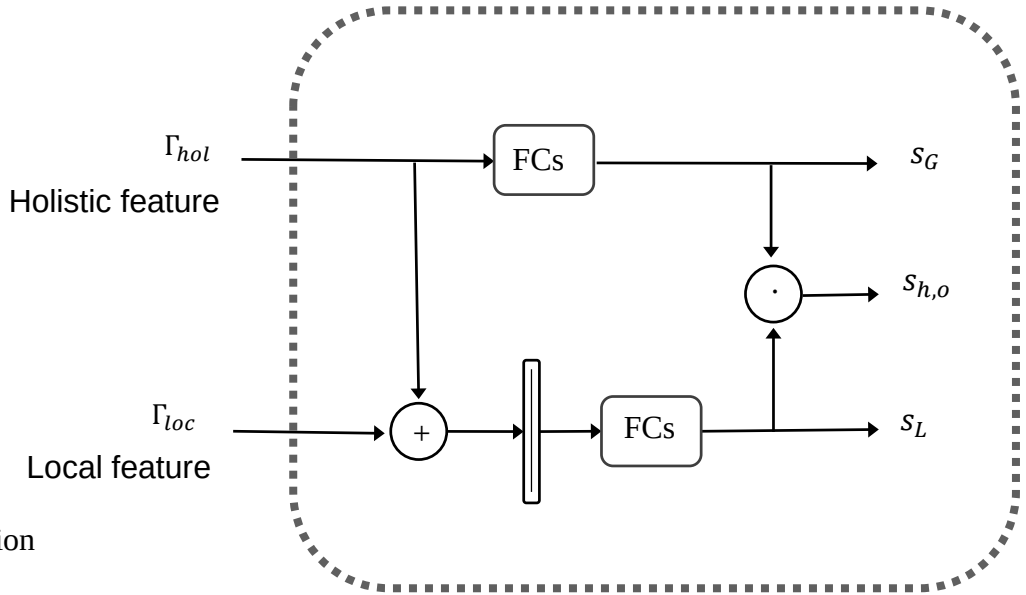
Holistic Module



Zoom-in Module



Fusion Module



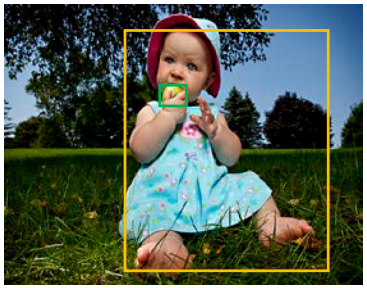
Qualitative Results

cut_instr: knife 0.23 0.85

throw: baseball 0.17 0.78

read: book 0.35 0.87

kick: soccer 0.13 0.73



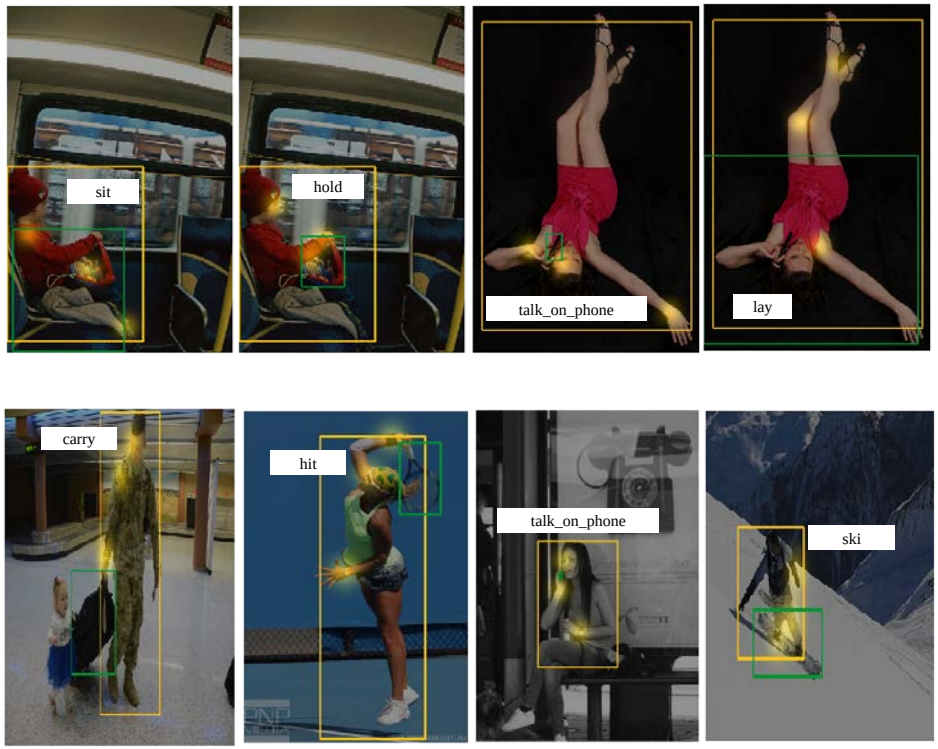
eat_obj: apple 0.01 0.57

talk_on_phone: phone 0.05 0.72

Red: w/o part cues

Green: ours

Visualization of Semantic Attention



Outline

- Image synthesis
  - Neural texture synthesis and style transfer
  - Novel view synthesis [CVPR 2018]
- Visual relations
  - Human-object interaction detection [ICCV 2019]
  - Semantic correspondence estimation [ICCV 2019]
- Semantic segmentation
  - Boundary-aware object instance segmentation [CVPR 2017]
  - Latent graph neural network for visual recognition [ICML 2019]

Acknowledgement: Roger Grosse's CSC231 and Feifei Li et al's cs231n notes

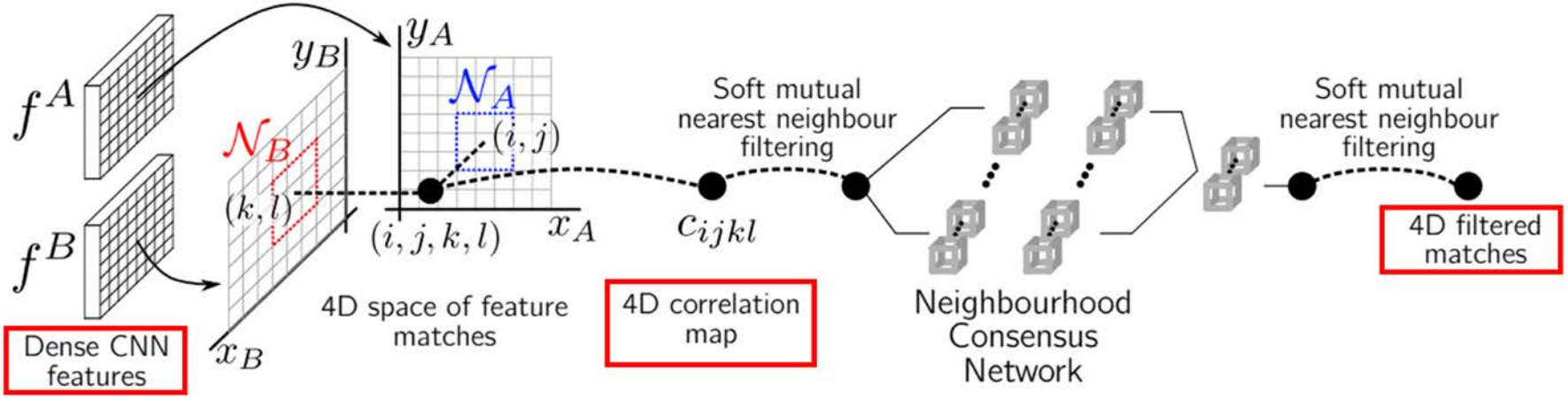
Problem Definition

- Semantic correspondence/alignment
  - Generate dense pixel correspondence across two semantically related images

- Large intra-class variations
- Viewpoint changes
- Background clutters
- Lack of data with annotation

Existing Approaches

- Correspondence network
  - Learn a convolutional feature embedding for matching



NCNet (NeurIPS 2018)

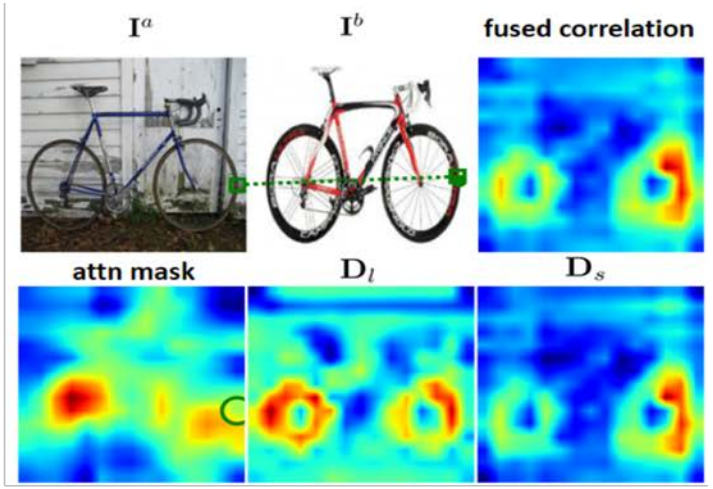
Motivations

- Feature ambiguity due to lack of global context

Main Ideas

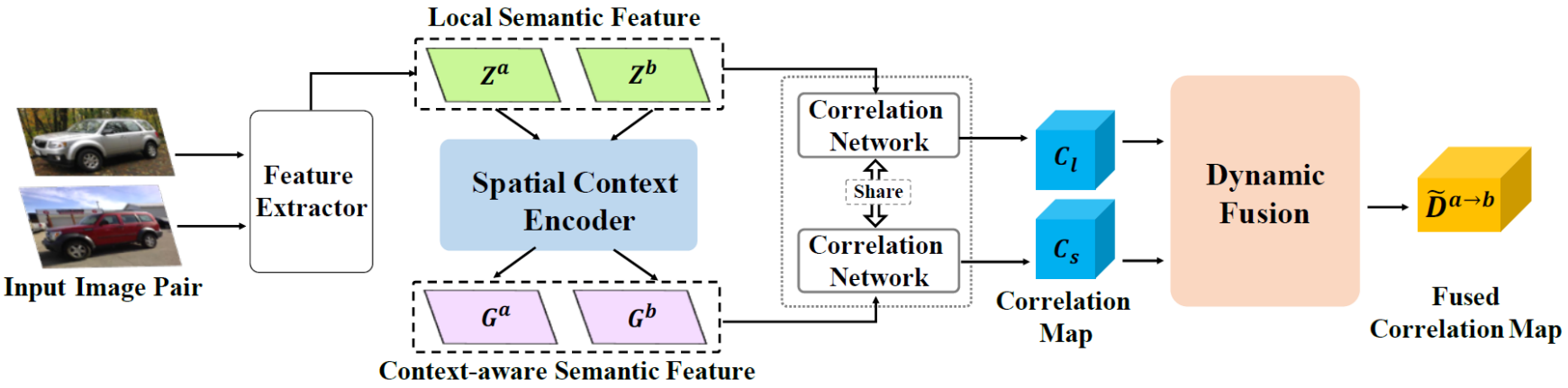
- Incorporating context cues in feature representation
- Fusing global and local features dynamically

Context cues



Dynamic fusion

Model Overview



Results

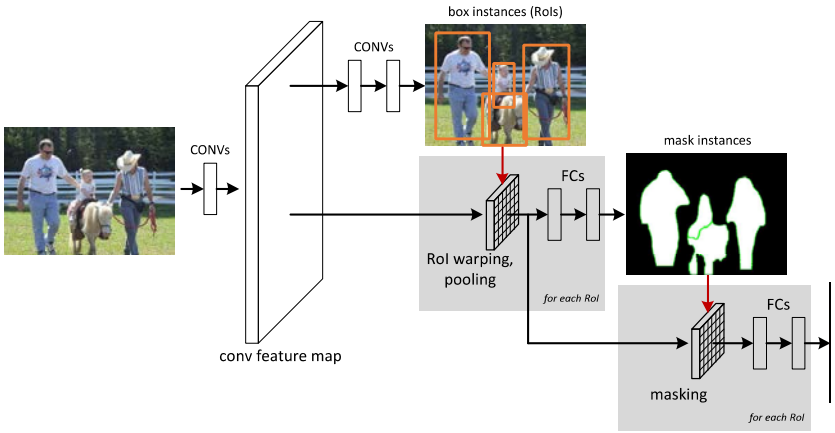
Outline

- Image synthesis
  - Neural texture synthesis and style transfer
  - Novel view synthesis [CVPR 2018]
- Visual relations
  - Human-object interaction detection [ICCV 2019]
  - Semantic correspondence estimation [ICCV 2019]
- Semantic segmentation
  - Boundary-aware object instance segmentation [CVPR 2017]
  - Latent graph neural network for visual recognition [ICML 2019]

Acknowledgement: Roger Grosse's CSC231 and Feifei Li et al's cs231n notes

Problem setting

- Semantic Instance segmentation
- Binary masks:
  - Pros: easy to predict
  - Cons:
    - lacking global representation of shapes
    - Sensitive to the initial object localization

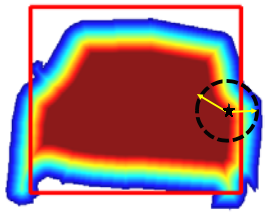


MNC (Dai et al 16)

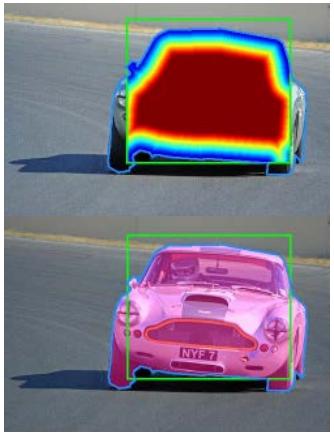
Main ideas

- Re-define the pixel labels
  - Multi-value label space
  - Each pixel's label indicates its relative position
  - Encoding object instance boundaries

- Our instantiation



Truncated
Distance Transform



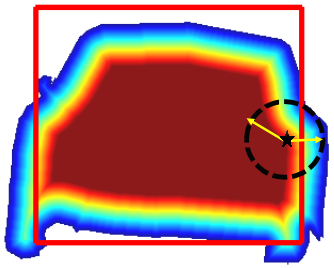
Hayder, He and Salzmann, CVPR 2017

Our mask representation

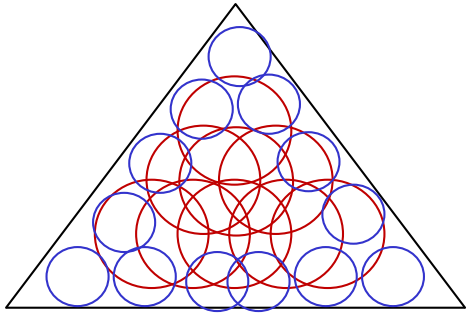
- Truncated DT
  - New value:

$$D(p) = \min\left(\min_{\forall q \in Q} \lceil d(p,q) \rceil, R\right)$$

    - Q is the object boundary



  - Each pixel encodes partial boundary info
  - Redundant representation
- How to get back to object mask?
  - Partial object mask
  - Avoid ad hoc post-processing
  - Approximate Inverse DT



Inverse Distance Transform

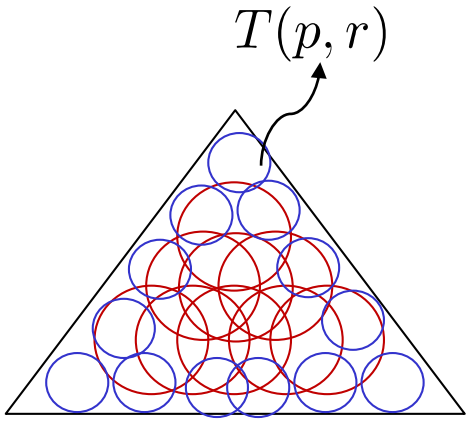
- Alternative encoding method
  - Quantize into K bins
  - One-hot encoding

$$D(p) = \sum_{n=1}^{K} r_n \cdot b_n(p), \quad \sum_{n=1}^{K} b_n(p) = 1$$

- Decoding mask

$$M = \bigcup_p T(p, D(p)) = \bigcup_p T(p, \sum_{n=1}^{K} r_n \cdot b_n(p))$$

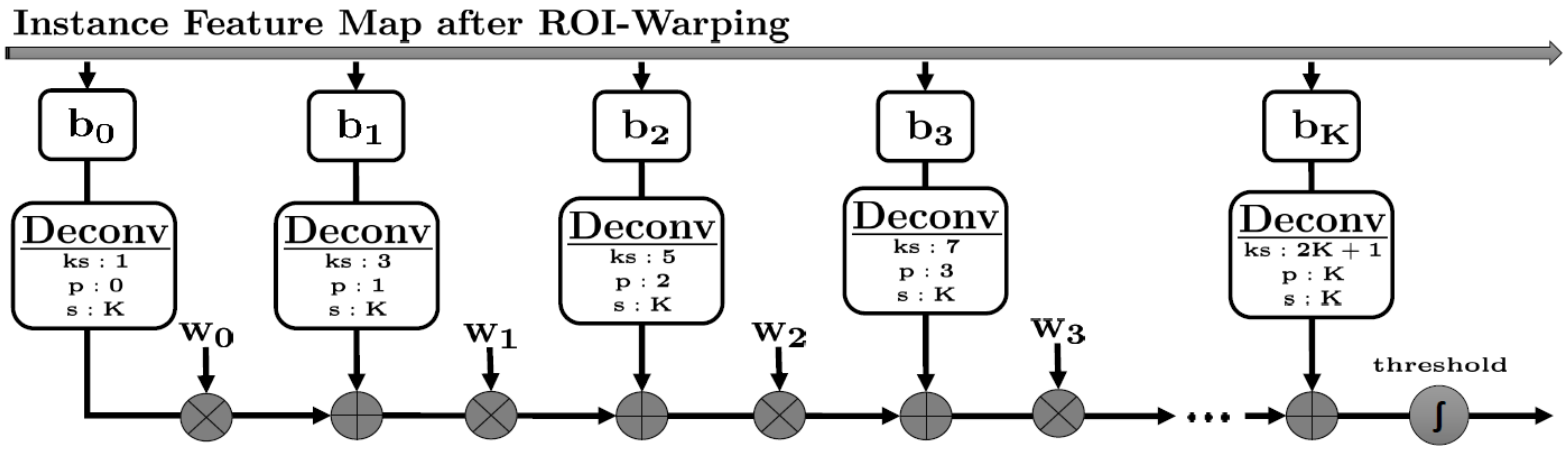
$$= \bigcup_{n=1}^{K} \bigcup_p T(p, r_n \cdot b_n(p)) = \bigcup_{n=1}^{K} T(\cdot, r_n) * B_n$$



B_n Binary mask for n-th bin

Network implementation

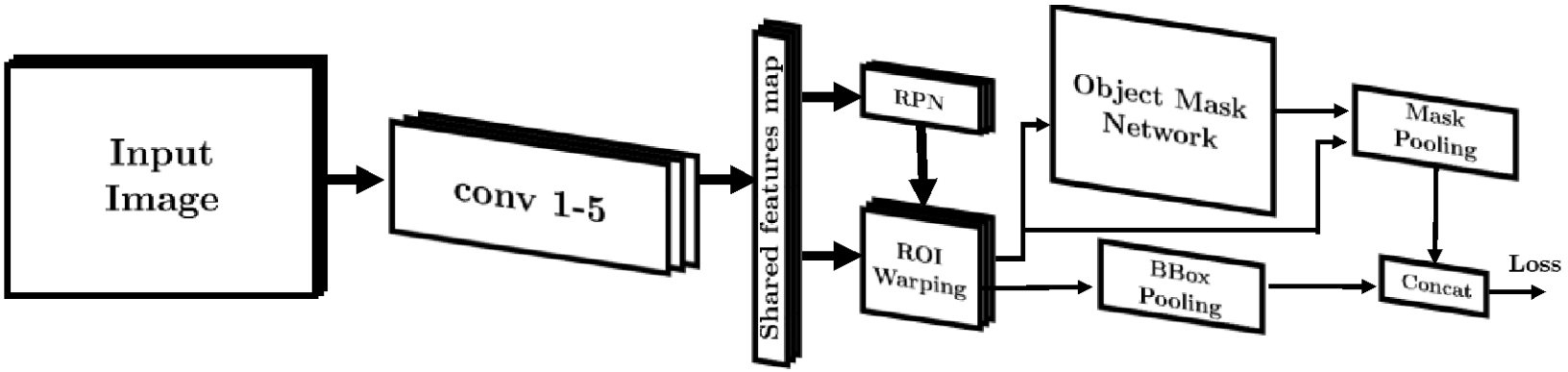
- IDT as union of deconvolutions
  - Each bin is a deconvolution with fixed weights
  - Union is approximated by weighted sum + sigmoid



$$M = \bigcup_{p} T(p, D(p)) = \bigcup_{n=1}^{K} \bigcup_{p} T(p, r_n \cdot b_n(p)) = \bigcup_{n=1}^{K} T(\cdot, r_n) * B_n$$

Combining with object proposals

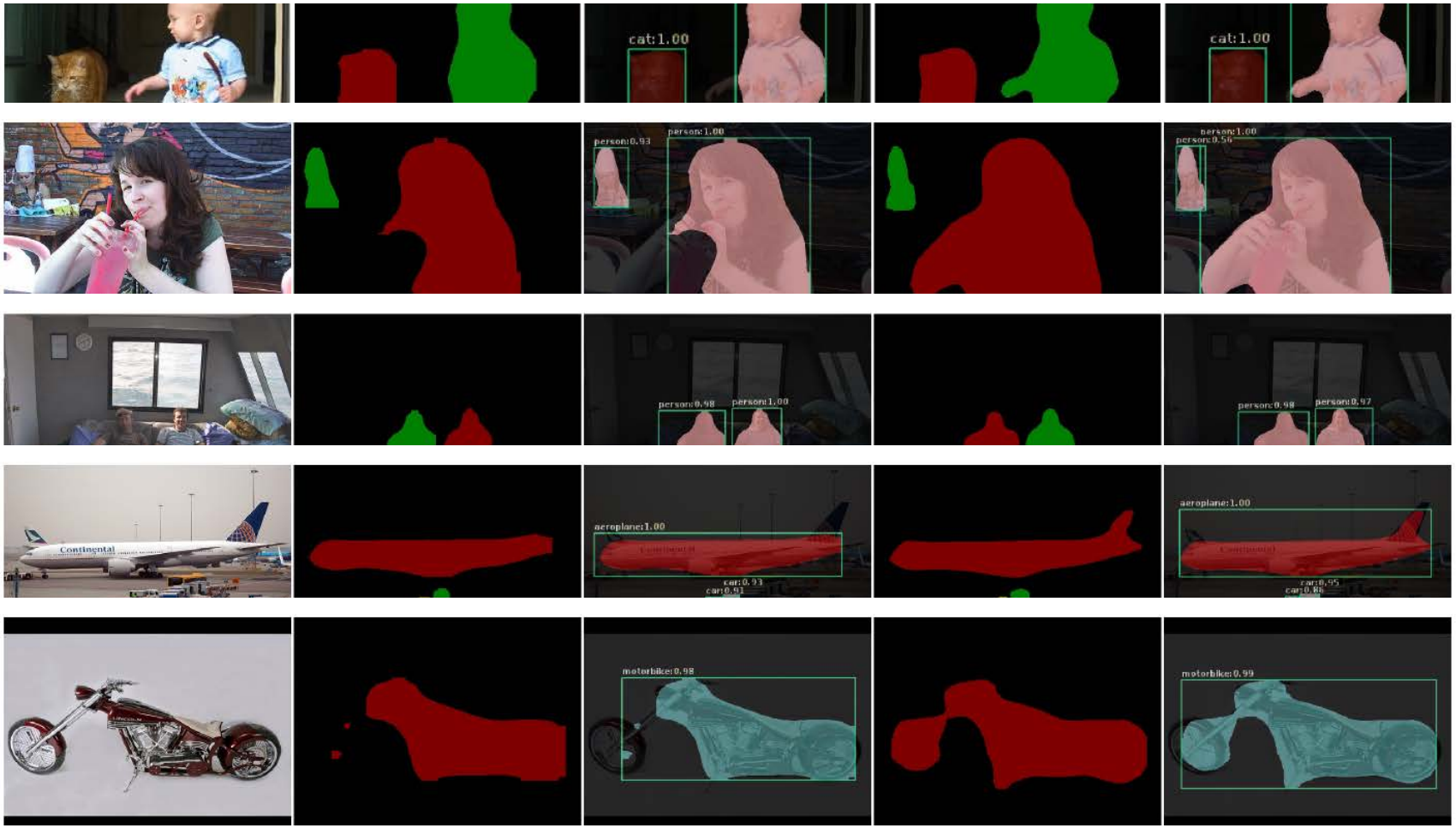
- Object mask network (OMN)
  - Integrating with MNC or other proposal-based methods
- RPN + OMN
  - End-to-end training with cross entropy loss for mask proposals
- MNC + OMN
  - End-to-end training with multi-task loss for instance segmentation



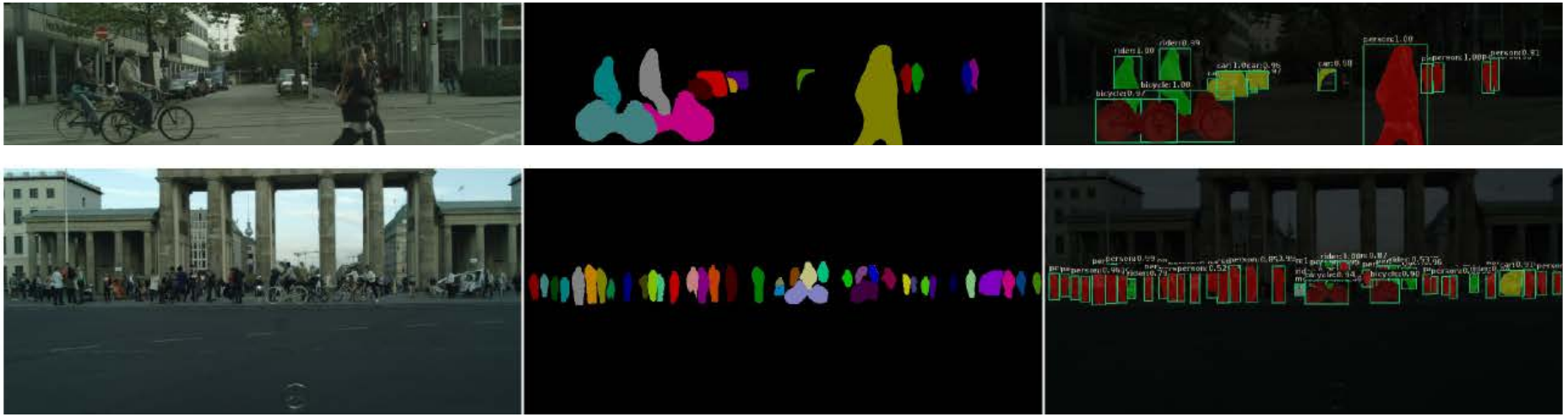
Some examples

MNC Results

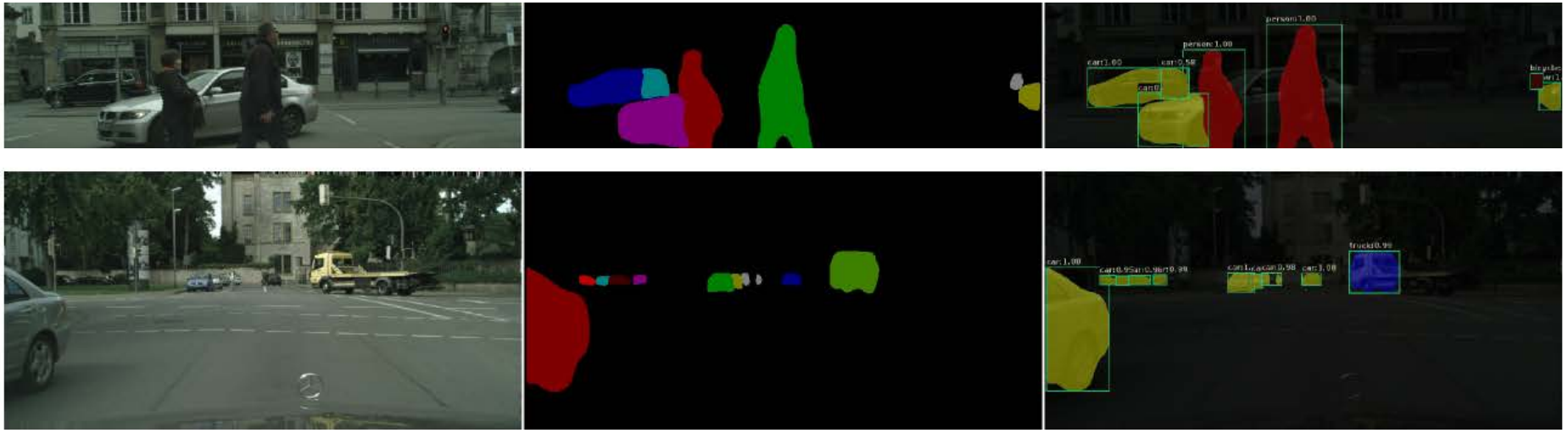
Our Results



Cityscape examples



- Failure cases:



Outline

- Image synthesis
  - Neural texture synthesis and style transfer
  - Novel view synthesis [CVPR 2018]
- Visual relations
  - Human-object interaction detection [ICCV 2019]
  - Semantic correspondence estimation [ICCV 2019]
- Semantic segmentation
  - Boundary-aware object instance segmentation [CVPR 2017]
  - Latent graph neural network for visual recognition [ICML 2019]

Acknowledgement: Roger Grosse's CSC231 and Feifei Li et al's cs231n notes

Problem Setting

- Long-range ***dependency*** is important for scene understanding
  - A combination of objects
    - (TV+sofa+rug+bookshelf = living-room)
  - Spatial relationships between shapes
  - Context effects
    - (A coffee-maker is usually in a kitchen)

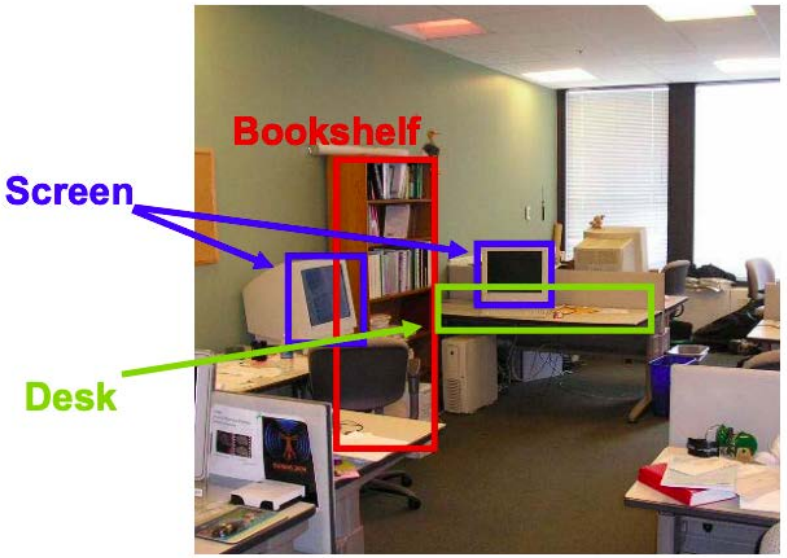
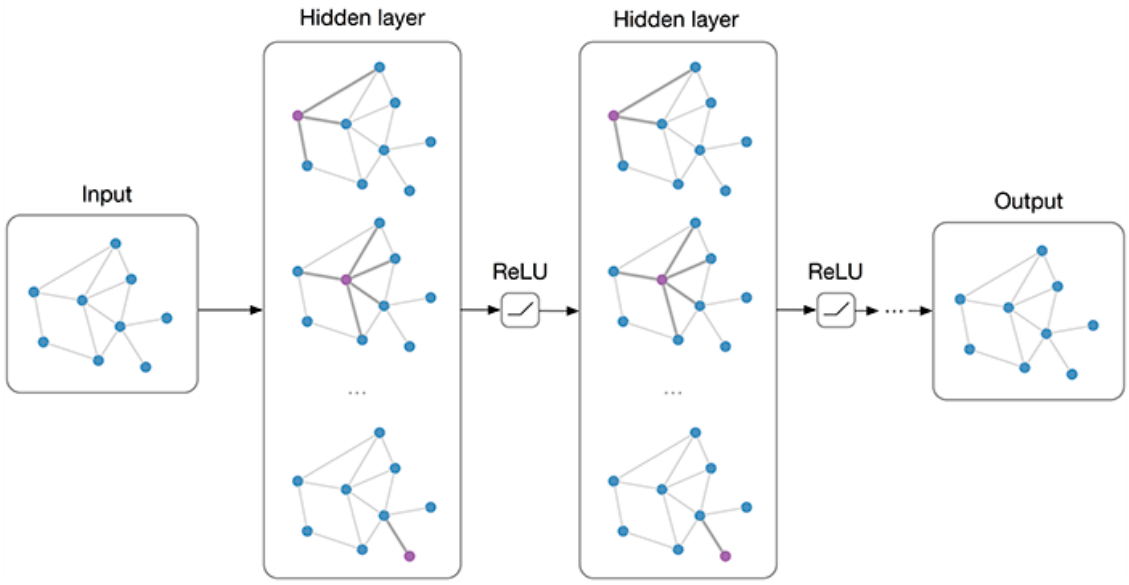


Image Credit: Advance in Computer Vision-MIT

Related work

Graph Convolutional Network



- Kipf & Welling (ICLR 2017), Semi-Supervised Classification with Graph Convolutional Networks
- Defferrard et al. (NIPS 2016), Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

Nonlocal graph network: a closer look

Input: Image Conv-feature, $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]^\top, \mathbf{x}_i \in \mathbb{R}^c, N = h \cdot w$

Output: Context-aware Conv-feature, $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_N]^\top, \tilde{\mathbf{x}}_i \in \mathbb{R}^c, N = h \cdot w$

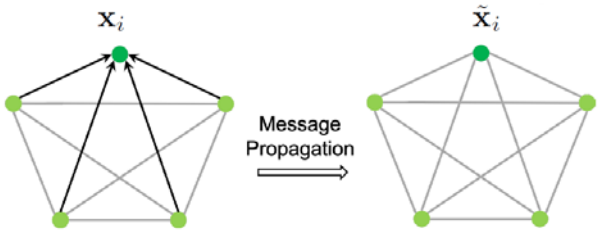
Element-wise activation (ReLU)

Pair-wise relation function

Each Location $\mathbf{x}'_i = h\left(\frac{1}{Z_i(\mathbf{X})} \sum_{j=1}^{N} g(\mathbf{x}_i, \mathbf{x}_j)\, \mathbf{W}^\top \mathbf{x}_j\right)$ $\tilde{\mathbf{x}}_i = \lambda \mathbf{x}'_i + \mathbf{x}_i$

Normalization Factor

Linear Mapping

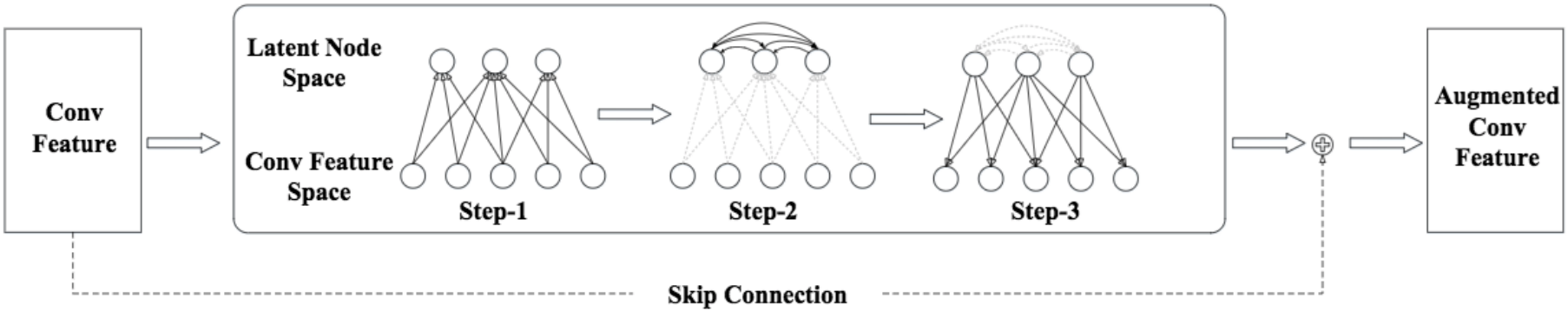


GNN with single-round message passing

Our proposal: Latent Graph Neural Network

Goal: Learning efficient Non-local relations.

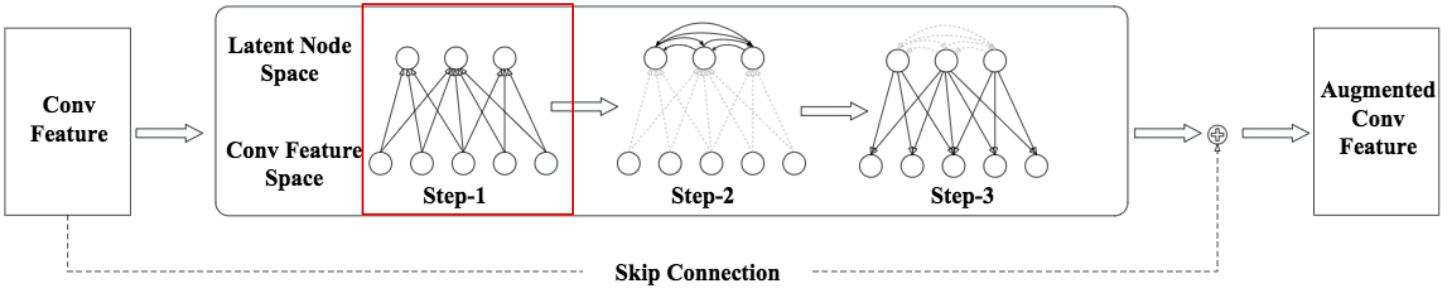
Key Idea: Introduce a latent space for efficient global context encoding



Conv-feature Space: $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]^\mathsf{T}, \mathbf{x}_i \in \mathbb{R}^c, N = h \cdot w$

Latent Space: $\mathbf{Z} = [\mathbf{z}_1, \cdots, \mathbf{z}_d]^\mathsf{T}, \mathbf{z}_i \in \mathbb{R}^c, d \ll N$

Latent Graph Neural Network



Step-1: Visible-to-latent propagation(Bipartite Graph)

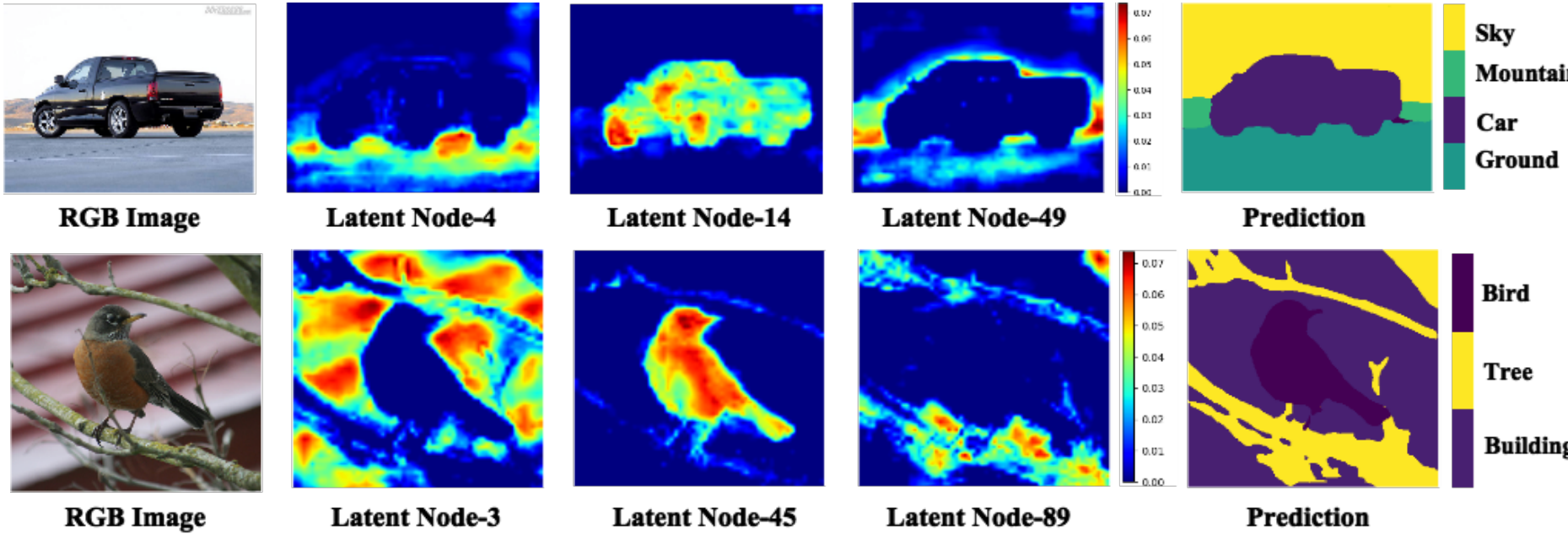
- **Each Latent Node:**

$$\mathbf{z}_k = \sum_{j=1}^{N} \frac{1}{m_k(\mathbf{X})} \psi(\mathbf{x}_j, \theta_k) \mathbf{W}^{\mathsf{T}} \mathbf{x}_j, \quad 1 \leq k \leq d$$

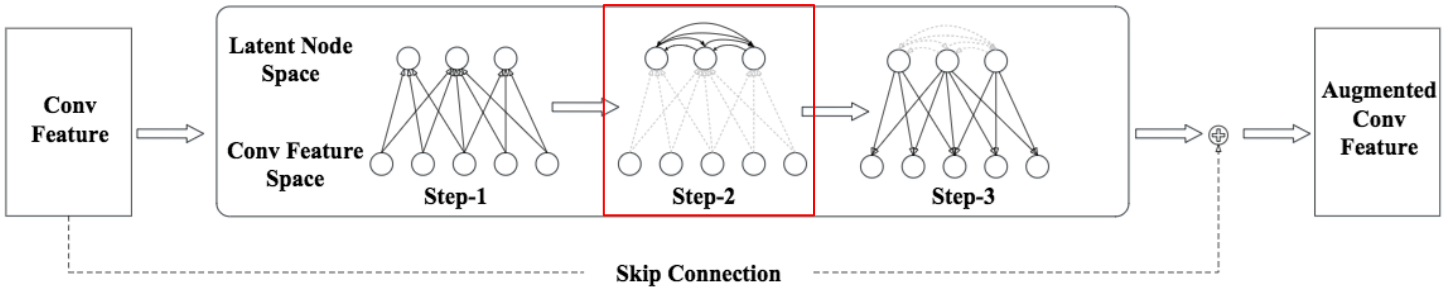
- $\psi(\mathbf{x}_j, \theta_k)$: : encode the normalized affinity between node $\mathbf{x}_j$ and node $\mathbf{z}_k$
- $m_k(\mathbf{X})$: the normalization factor

Understanding latent nodes

- Visualization of activated regions of latent nodes



Latent Graph Neural Network



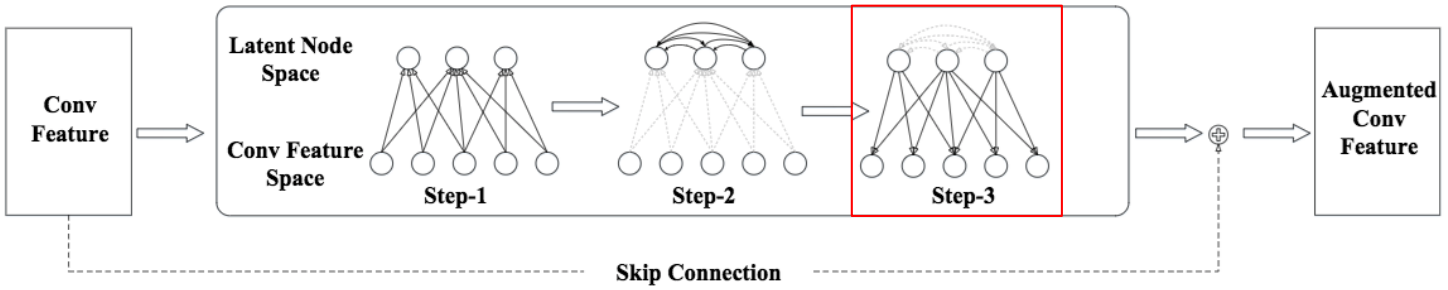
Step-2: Latent-to-latent propagation(Fully-connected Graph)

- **Each Latent Node:**

$$\tilde{\mathbf{z}}_k = \sum_{j=1}^{d} f(\phi_k, \phi_j, \mathbf{X})\mathbf{z}_j, \quad 1 \leq k \leq d$$

- $f(\phi_k, \phi_j, \mathbf{X})$: data-dependent pair-wise relations between two latent nodes

Latent Graph Neural Network



Step-3: Latent-to-visible propagation(Bipartite Graph)

- **Each Latent Node:**

$$\tilde{\mathbf{x}}_i = h\left(\sum_{k=1}^{d} \psi(\mathbf{x}_i, \theta_k)\tilde{\mathbf{z}}_k\right), \quad 1 \le i \le N$$

Experimental results

Grid Data: Object Detection/Instance Segmentation on MSCOCO

- **+NLBlock**: insert the non-local block in the last stage of the backbone.
- **+LatentGNN**: Integrate LatentGNN with the backbone at different stages.

Model	Stage	Kernels	AP_{box}	AP^{50}_{box}	AP^{75}_{box}	AP_{sem}	AP^{50}_{sem}	AP^{75}_{sem}	FLOPS	#Params
ResNet-50[1]	-	-	38.0	59.6	41.5	34.6	56.4	36.5	-	-
+NL Block[1]	Stage4	1	39.0	61.1	41.9	35.5	58.0	37.4	+10.67G	+ 2.09M
ResNet-50(1x)[2]	-	-	37.8	59.1	41.2	34.2	55.8	36.3	-	-
+ NL Block[2]	Stage4	1	38.7	60.2	42.2	35.0	57.0	37.1	+10.67G	+ 2.09M
+ LatentGNN	Stage3	1	38.2	59.7	41.7	34.7	56.3	36.8	+1.48G	**+ 0.06M**
+ LatentGNN	Stage4	1	39.0	60.7	42.6	35.2	57.6	37.4	+1.11G	+ 0.20M
+ LatentGNN	Stage5	1	38.8	61.0	42.0	35.0	57.6	37.0	**+0.97G**	+ 0.81M
+ LatentGNN	Stage345	1	**39.5**	**61.6**	**43.2**	**35.6**	**58.3**	**37.7**	+3.59G	+1.07M
ResNet-101(1x)	-	-	39.9	61.3	43.8	35.9	58.2	38.1	-	-
+ LatentGNN	Stage4	1	41.0	63.2	45.0	36.9	59.6	39.4	**+1.11G**	**+ 0.20M**
+ LatentGNN	Stage345	1	**41.4**	**63.7**	**45.2**	**37.2**	**60.1**	**39.5**	+3.59G	+1.07M
ResNeXt-101(1x)	-	-	42.1	64.1	45.9	37.8	60.3	39.5	-	-
+ LatentGNN	Stage4	1	43.0	65.3	46.9	38.5	61.9	40.9	**+1.11G**	**+ 0.20M**
+ LatentGNN	Stage345	1	**43.2**	**65.6**	**47.2**	**38.8**	**62.1**	**41.0**	+3.59G	+1.07M

Summary

- CNNs in computer vision
  - Many and more applications
  - Still lack of deep understanding

- Next time:
  - No class next week due to ICCV
  - Recurrent Neural Networks