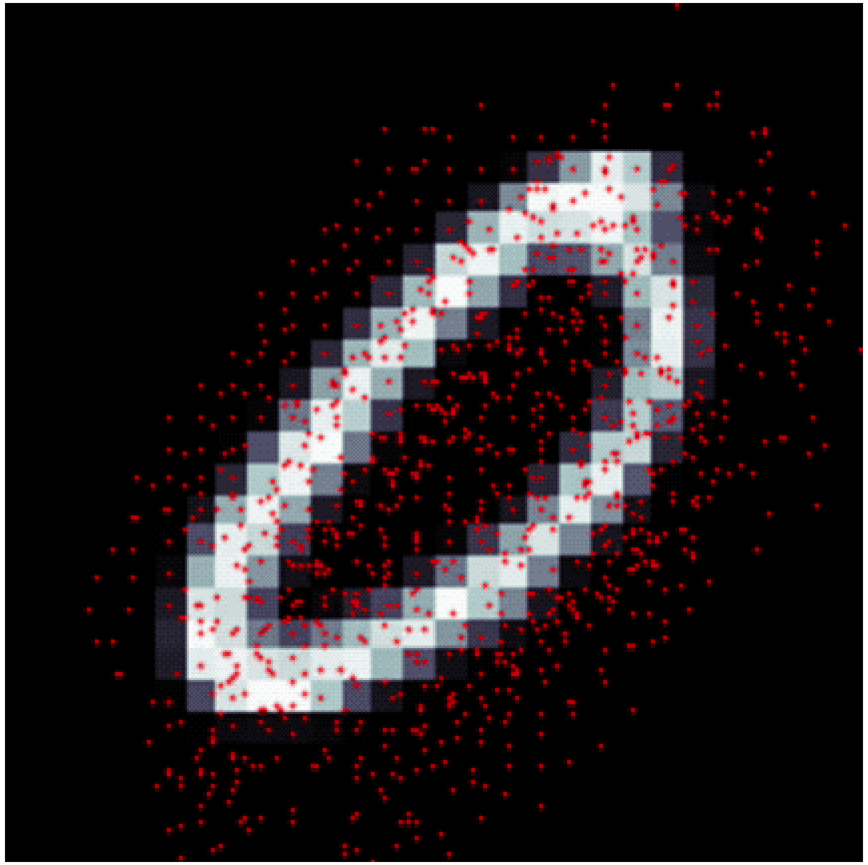
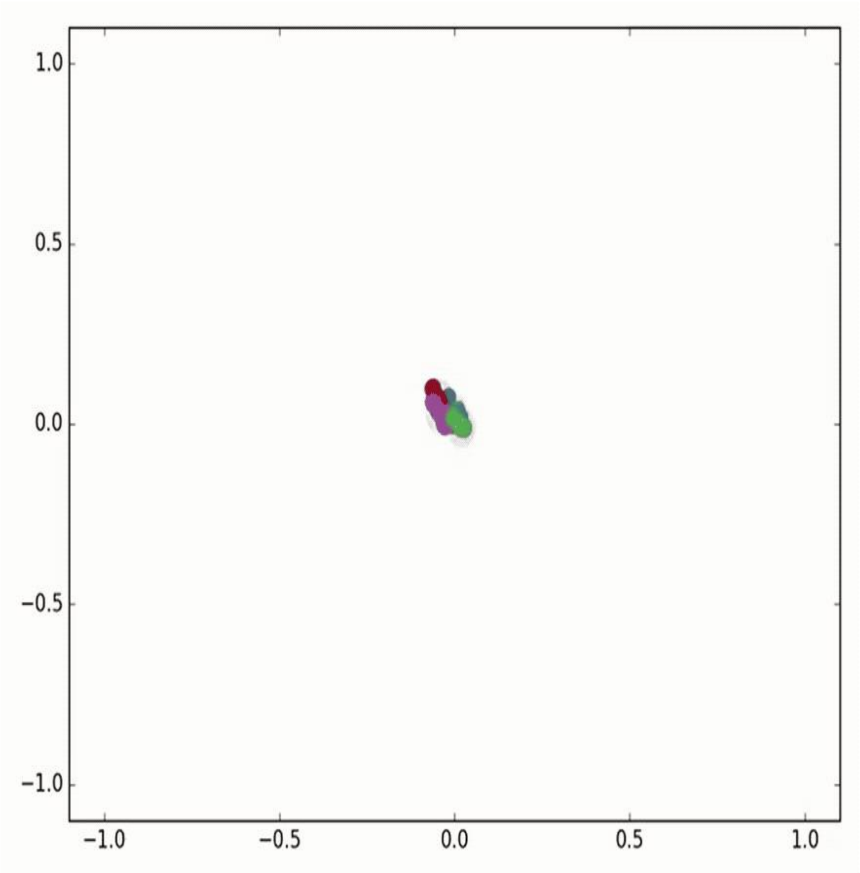


Deformable Convolutional Networks (ICCV'17)



Graph Neural Networks (ICLR'17, ICLR'18)

Geometric Convolutions

Presented by: Anand Bhattad (2nd year of MS, CS)

CS598 BL: Adversarial Machine Learning
Instructor: Prof. Bo Li

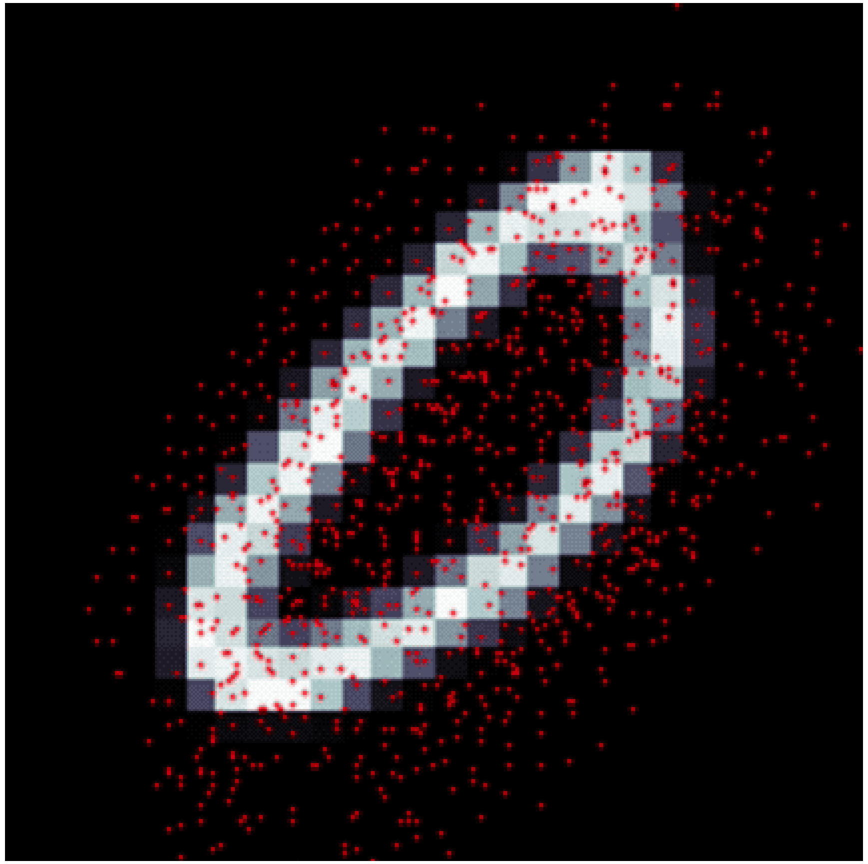
Outline

Deformable Convolutions

- Motivation and Contribution
- How they work?
- Benefits

Graph Neural Networks

- Why graph neural networks?
- How they are different from CNNs?
- Few problems using graph networks



Deformable Convolution Networks

Dai, Jifeng, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. "Deformable Convolutional Networks." ICCV (2017).

Motivation

Fixed Geometric Structures in current CNNs

Motivation

Fixed Geometric Structures in current CNNs

Invariant to Geometric Variations/Spatial Transformations

Motivation

Fixed Geometric Structures in current CNNs

Invariant to Geometric Variations/Spatial Transformations

- Scale
- Pose
- Viewpoint
- Deformation
- Inter-class variation

Motivation

Fixed Geometric Structures in current CNNs

Invariant to Geometric Variations/Spatial Transformations

- Scale
- Pose
- Viewpoint
- Deformation
- Inter-class variation

Generalization to new tasks

Contribution

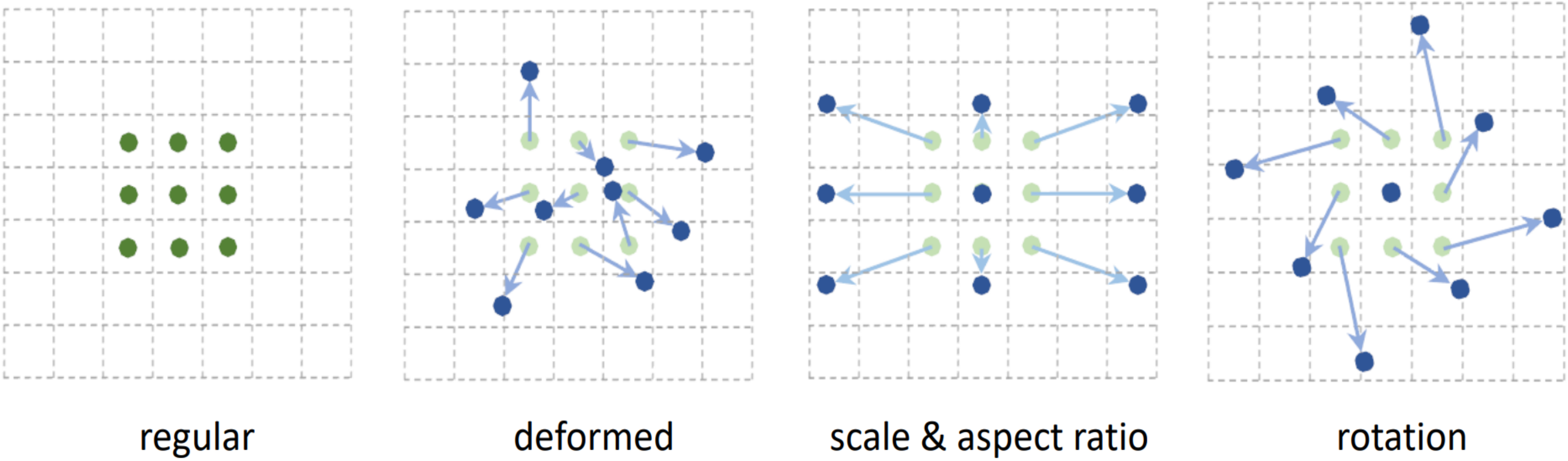
Deformable convolution – Convolution + Learnable offset

Deformable RoI pooling – RoI Polling + Learnable offset

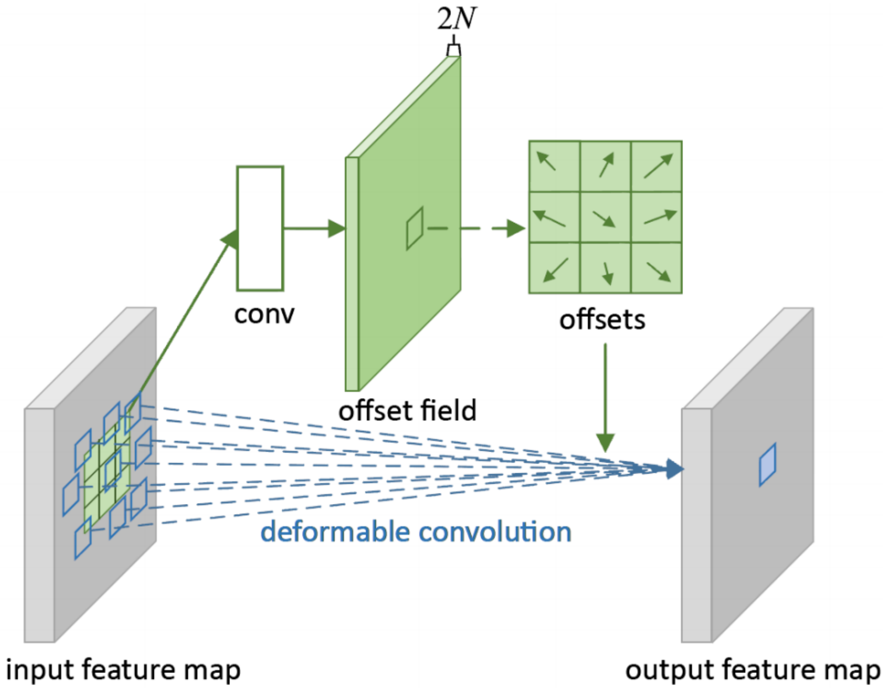
Contribution

Deformable convolution – Convolution + Learnable offset

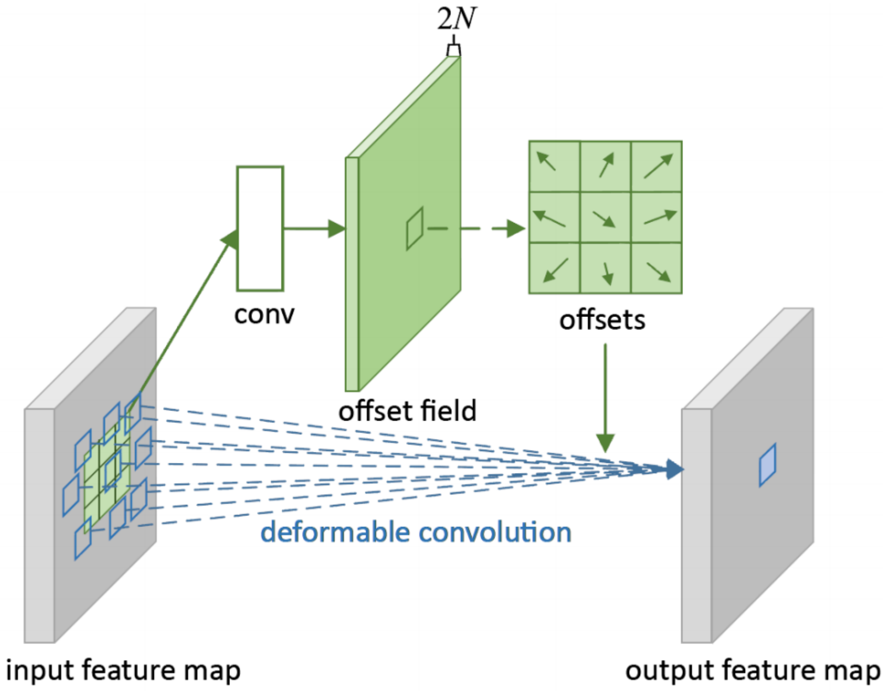
Deformable RoI pooling – RoI Polling + Learnable offset



Deformable Convolutions



Deformable Convolutions

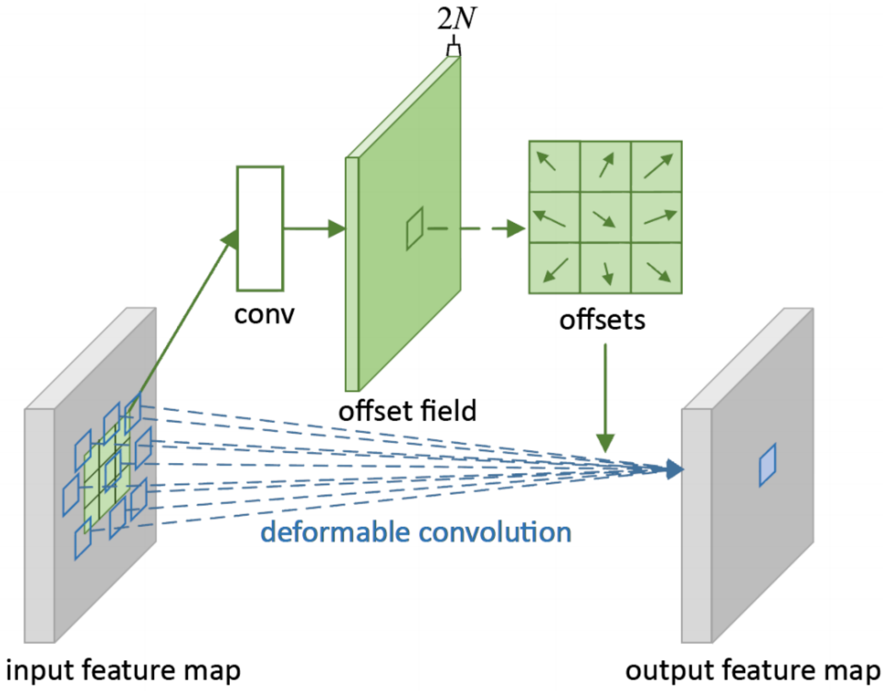


Two Branches

- Regular conv. layer
- another conv. layer to learn 2D offset

Generalized /“learnable” dilated convolution

Deformable Convolutions

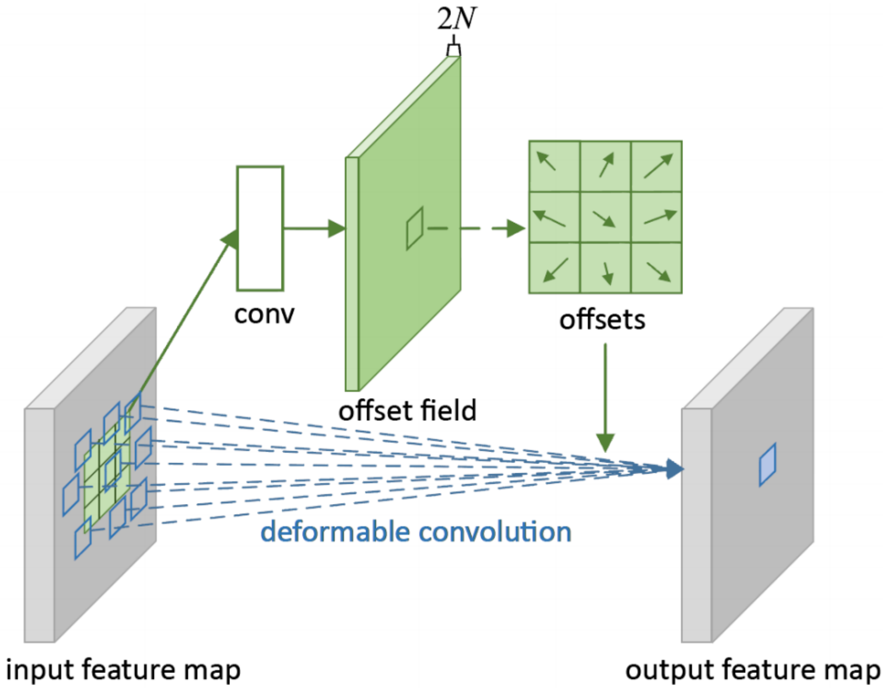


Regular convolution

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n)$$

$$\mathcal{R} = \{(-1,-1), (-1,0), \ldots, (0,1), (1,1)\}$$

Deformable Convolutions



Regular convolution

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n)$$

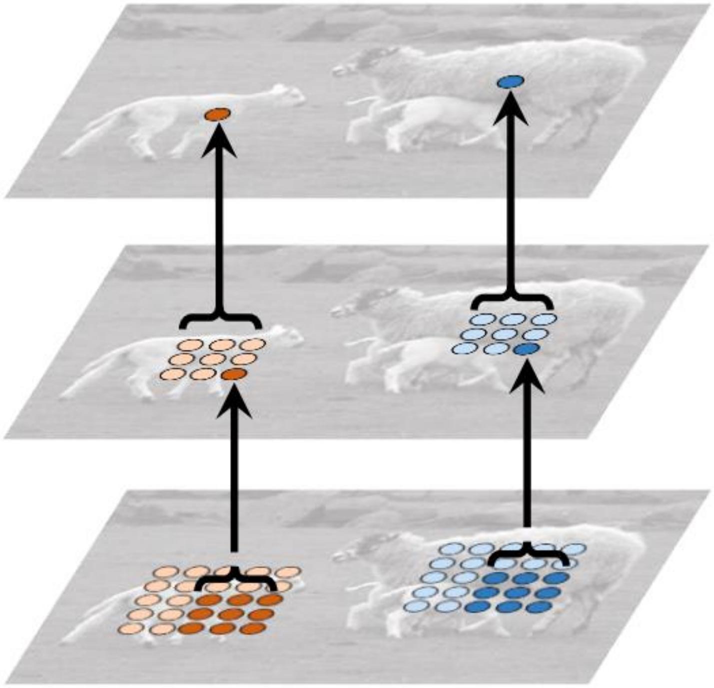
Deformable convolution

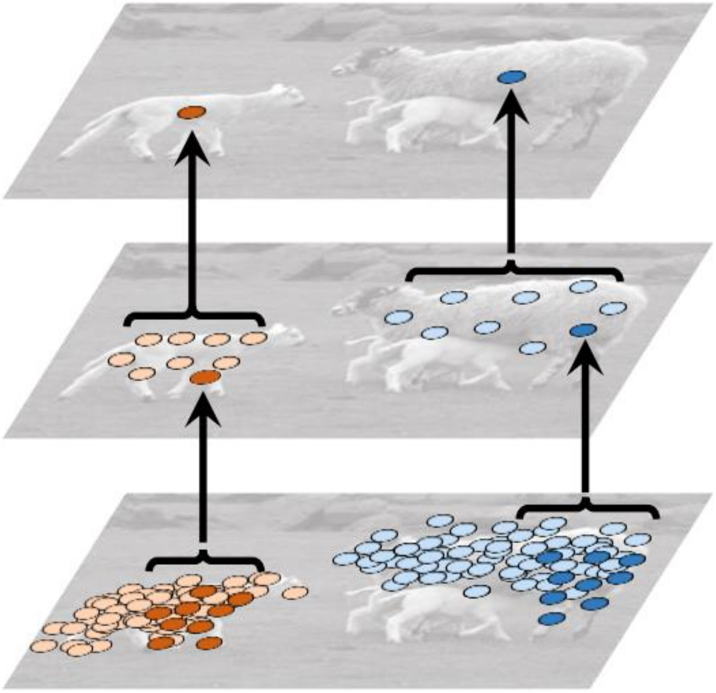
$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n + \Delta \mathbf{p}_n)$$

$$\mathcal{R} = \{(-1,-1), (-1,0), \ldots, (0,1), (1,1)\}$$

Benefits

Local, Dense and Adaptive

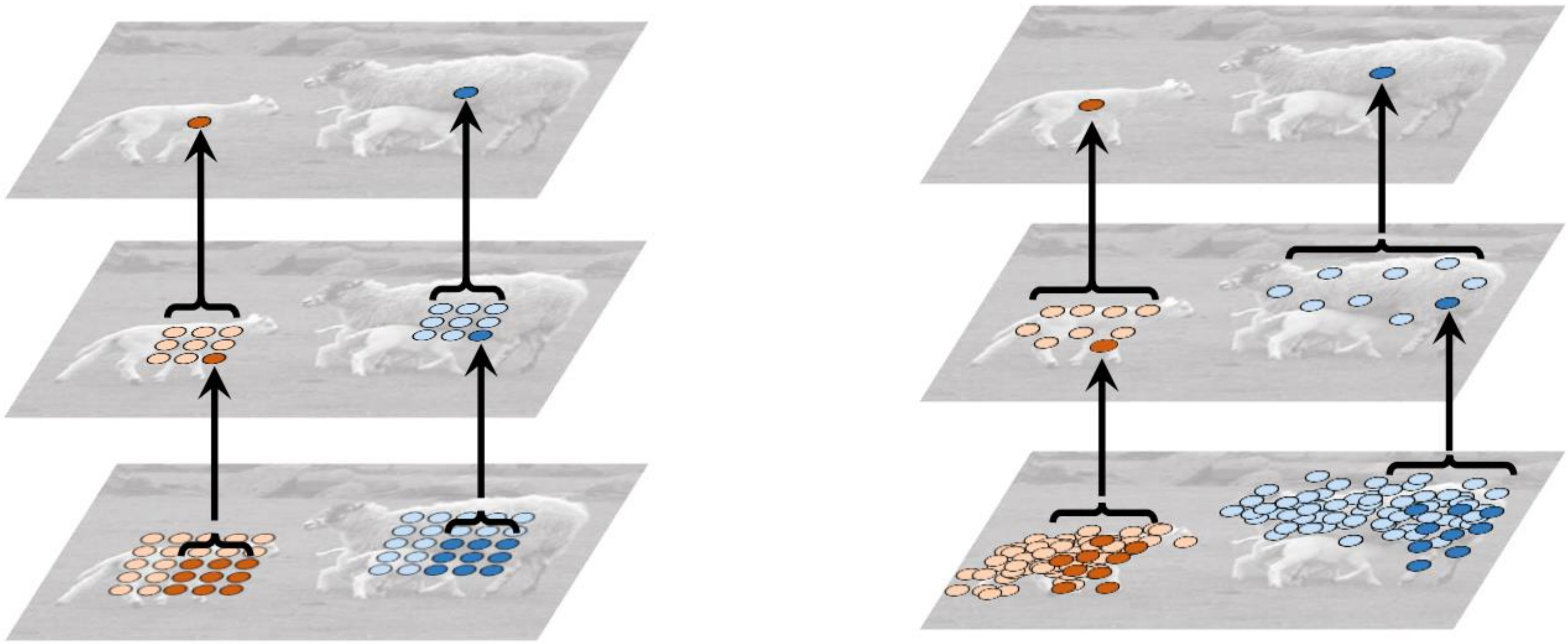




Benefits

Local, Dense and Adaptive

Dynamic and learnable receptive field



Benefits

Local, Dense and Adaptive

Dynamic and learnable receptive field

Benefits

Local, Dense and Adaptive

Dynamic and learnable receptive field

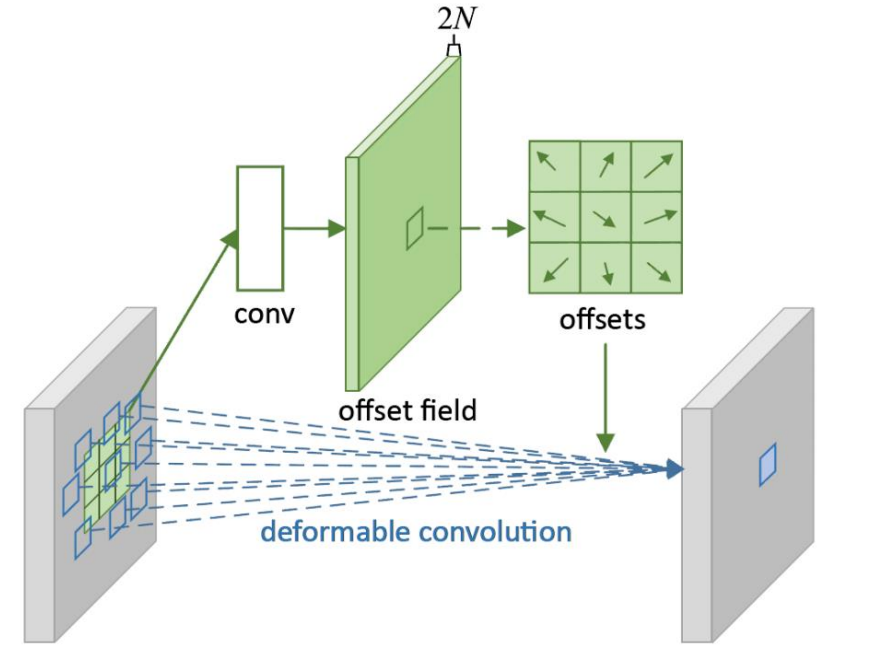
Benefits

Local, Dense and Adaptive

Dynamic and learnable receptive field

Attention embedded in Conv

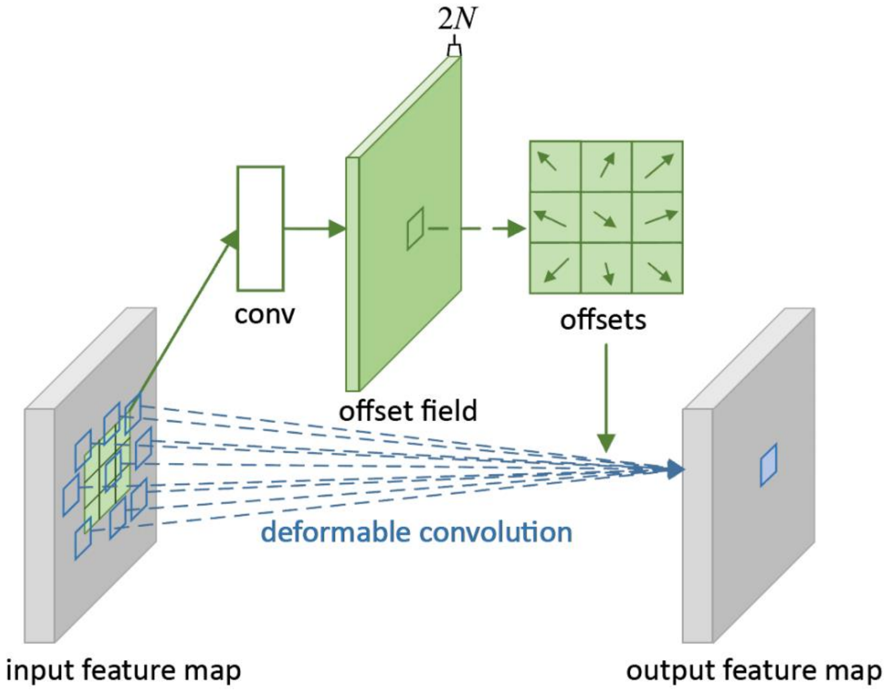
Deformable Convolutions on Segmentation



usage of deformable convolution (# layers)	DeepLab	
	mIoU@V (%)	mIoU@C (%)
none (0, baseline)	69.7	70.4
res5c (1)	73.9	73.5
res5b,c (2)	74.8	74.4
res5a,b,c (3, default)	**75.2**	**75.2**
res5 & res4b22,b21,b20 (6)	74.8	75.1

Results of using deformable convolution in the last 1, 2, 3, and 6 convolutional layers (of 3 × 3 filter) in ResNet-101 (TABLE 1)

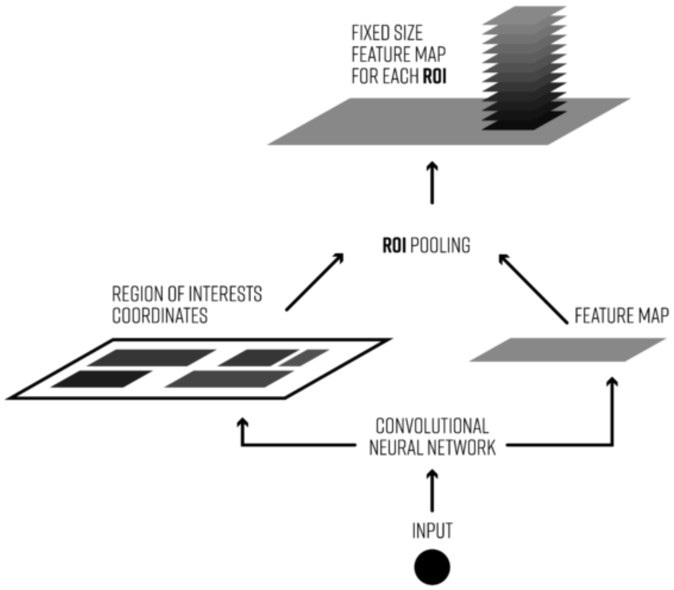
Deformable Convolutions vs Atrous Convolution

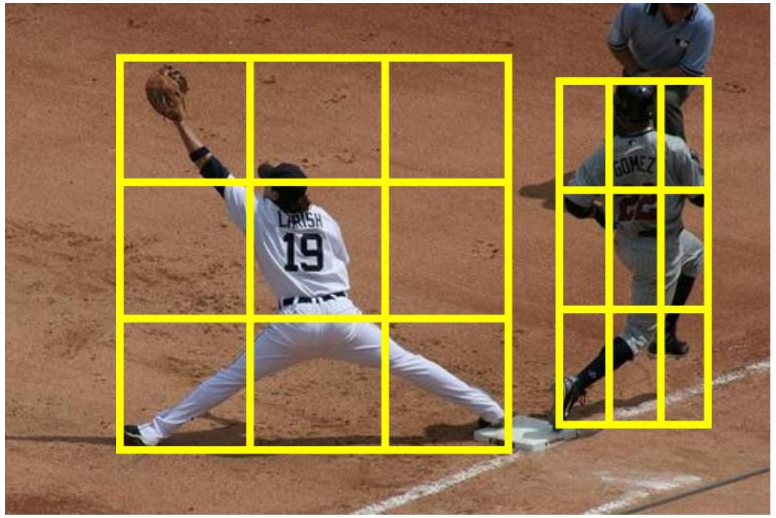


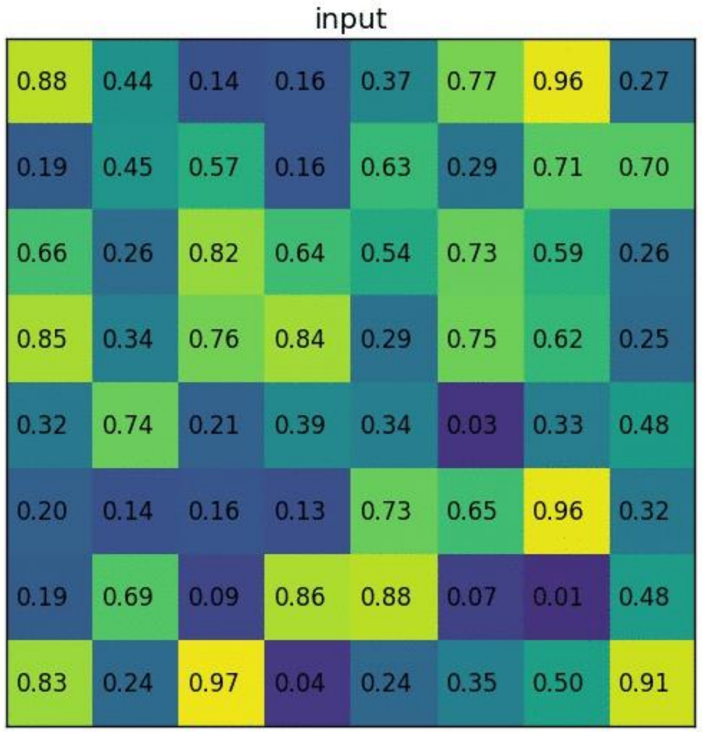
deformation modules	DeepLab mIoU@V / @C
atrous convolution (2,2,2) (default)	69.7 / 70.4
atrous convolution (4,4,4)	73.1 / 71.9
atrous convolution (6,6,6)	73.6 / 72.7
atrous convolution (8,8,8)	73.2 / 72.4
deformable convolution	**75.3 / 75.2**

ResNet-101 (TABLE 3)

RoI Pooling

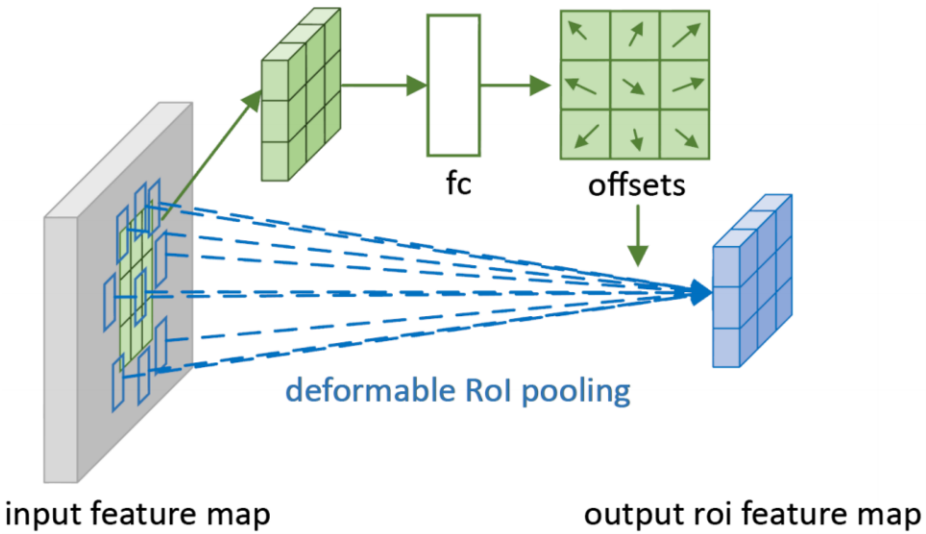






Source: https://blog.deepsense.ai/region-of-interest-pooling-explained/

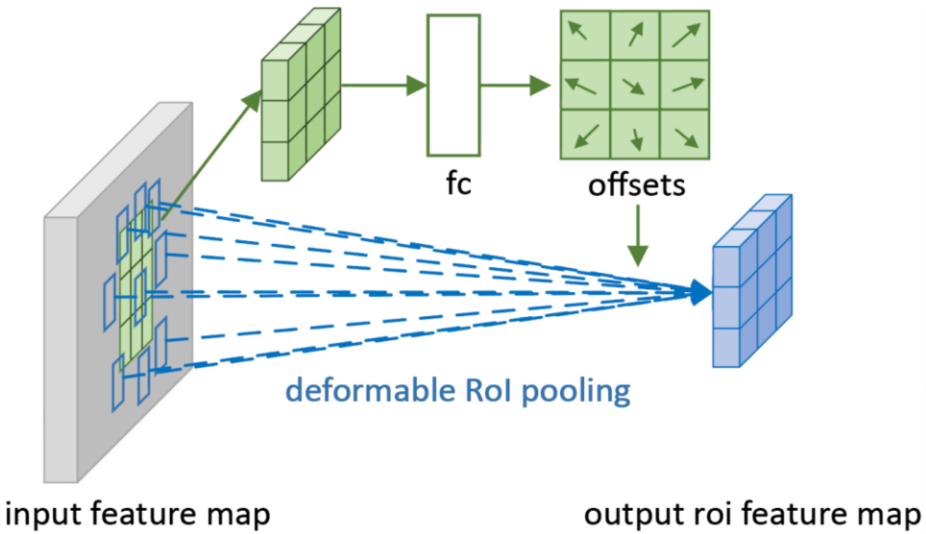
Deformable RoI Pooling



deformable RoI Pooling

Don’t predict the raw offset

Deformable RoI Pooling

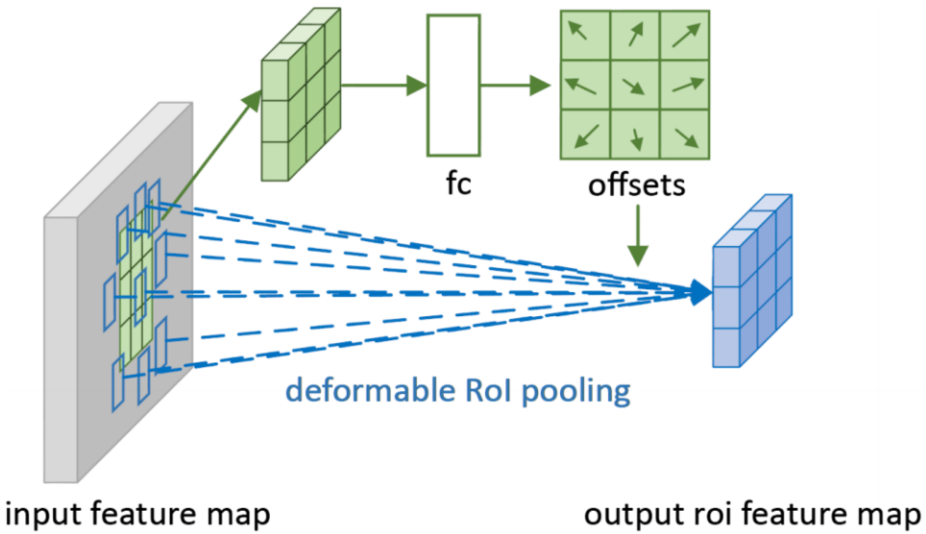


deformable RoI Pooling

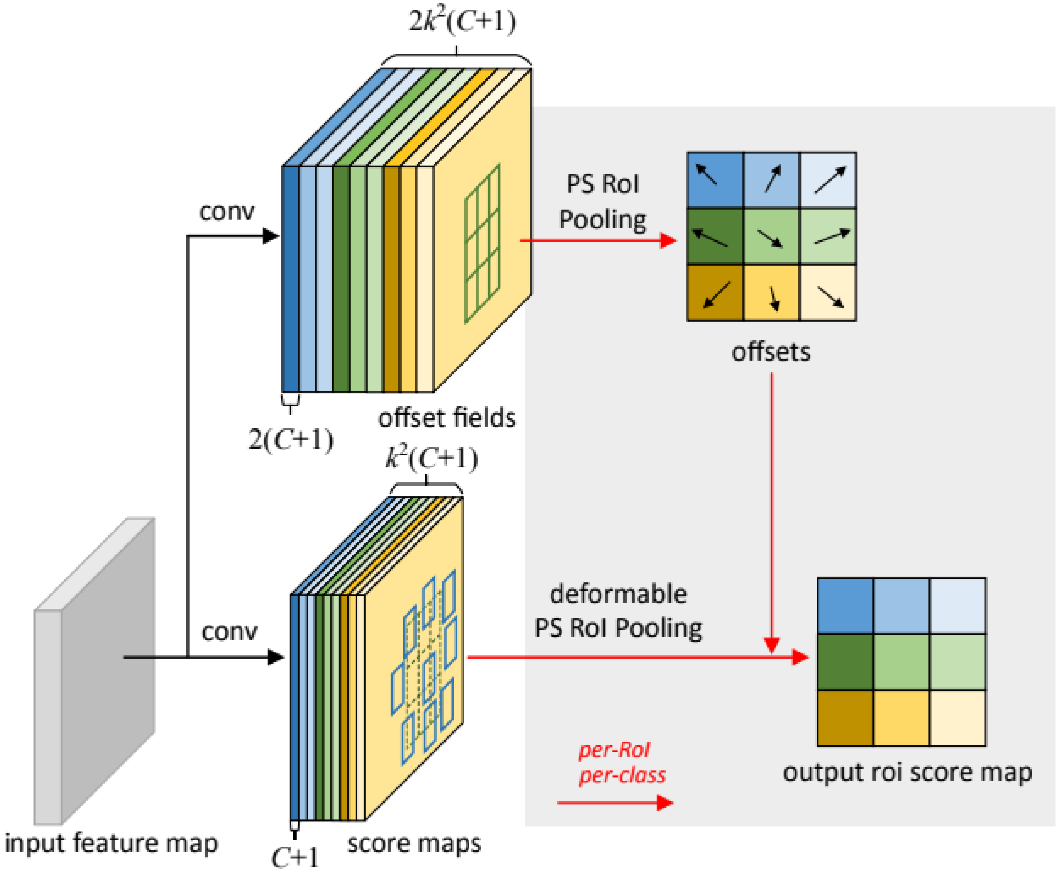
Don’t predict the raw offset

Normalize offsets – invariant to ROI size

Deformable RoI Pooling

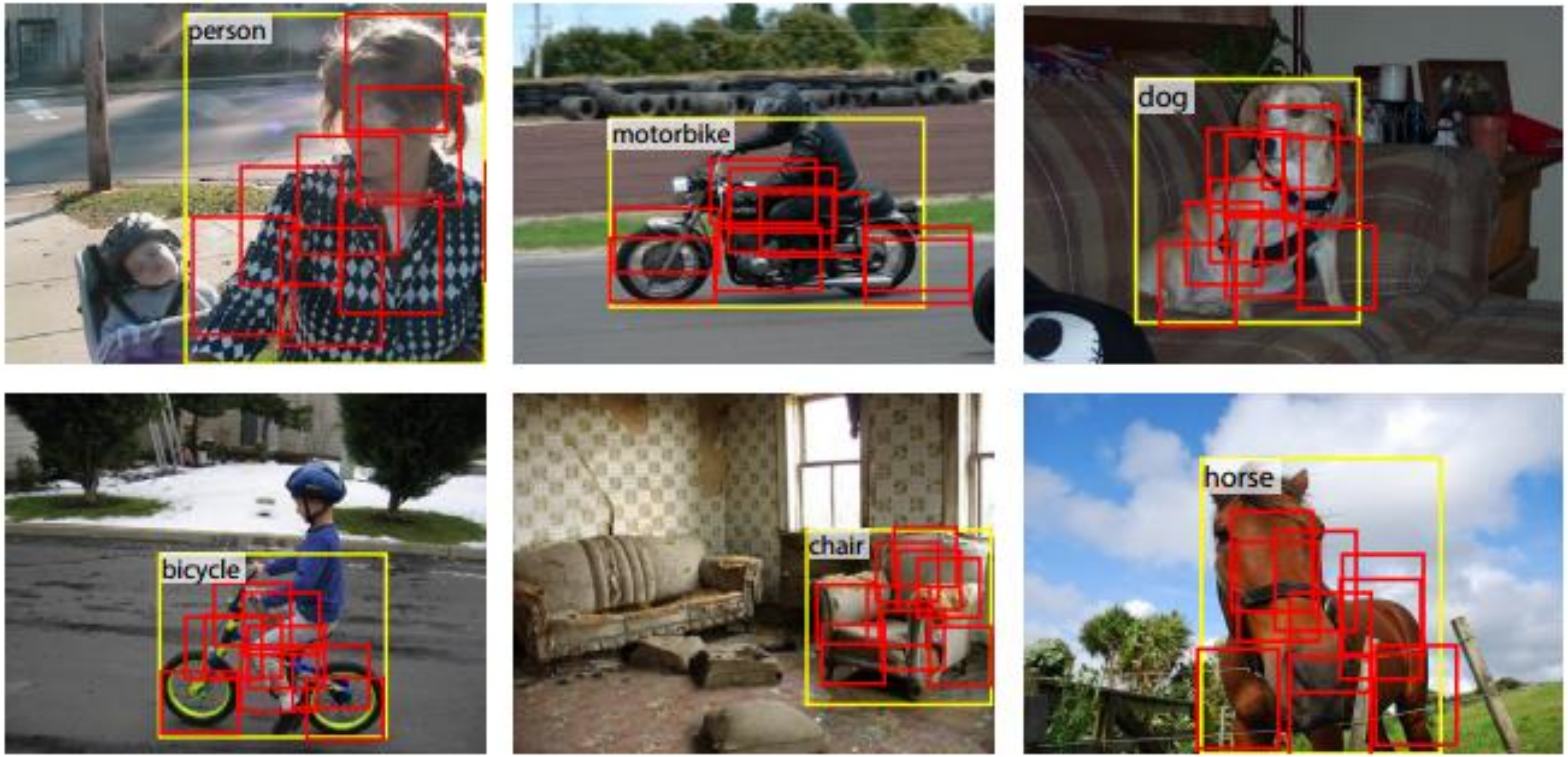


deformable RoI Pooling



PS deformable RoI Pooling

PS Deformable RoI Pooling



Object Detection (PASCAl VOC)

usage of deformable convolution (# layers)	class-aware RPN		Faster R-CNN		R-FCN	
	mAP@0.5 (%)	mAP@0.7 (%)	mAP@0.5 (%)	mAP@0.7 (%)	mAP@0.5 (%)	mAP@0.7 (%)
none (0, baseline)	68.0	44.9	78.1	62.1	80.0	61.8
res5c (1)	73.5	54.4	78.6	63.8	80.6	63.0
res5b,c (2)	74.3	56.3	78.5	63.3	81.0	63.8
res5a,b,c (3, default)	74.5	57.2	78.6	63.3	81.4	64.7
res5 & res4b22,b21,b20 (6)	**74.6**	**57.7**	**78.7**	**64.0**	**81.5**	**65.4**

Model Complexity and Runtime Comparison

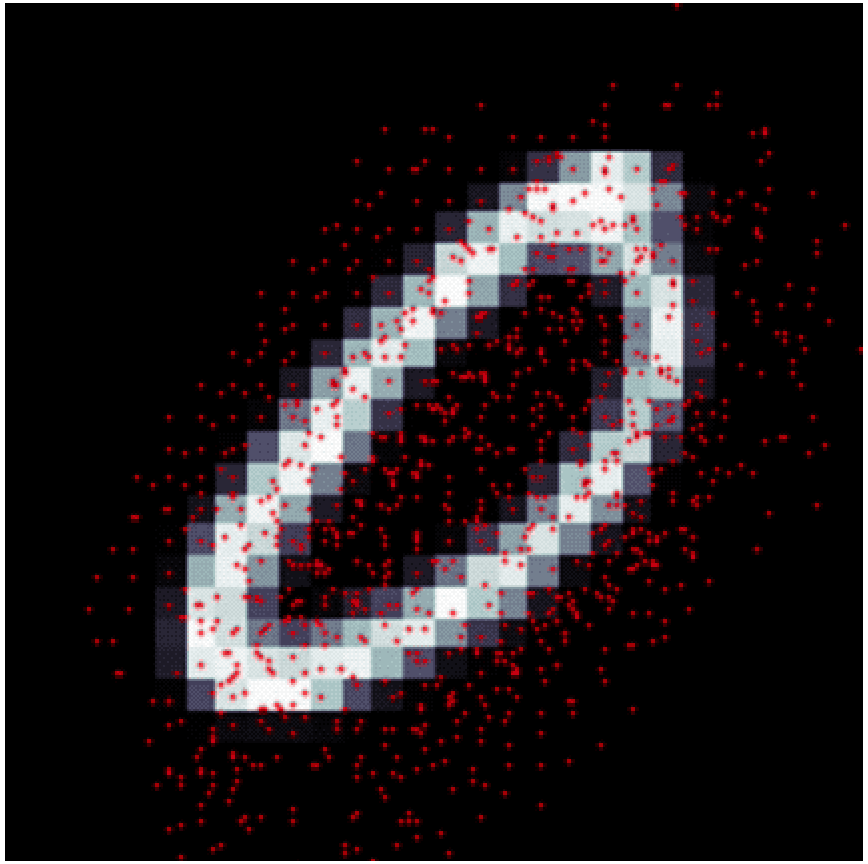
method	# params	net. forward (sec)	runtime (sec)
DeepLab@C	46.0 M	0.610	0.650
Ours	46.1 M	0.656	0.696
DeepLab@V	46.0 M	0.084	0.094
Ours	46.1 M	0.088	0.098
class-aware RPN	46.0 M	0.142	0.323
Ours	46.1 M	0.152	0.334
Faster R-CNN	58.3 M	0.147	0.190
Ours	59.9 M	0.192	0.234
R-FCN	47.1 M	0.143	0.170
Ours	49.5 M	0.169	0.193

Robustness

We found that rotation-equivariant networks are significantly less vulnerable to geometric-based attacks than regular networks on the MNIST, CIFAR-10, and ImageNet datasets.

ROBUSTNESS OF ROTATION-EQUIVARIANT NETWORKS TO ADVERSARIAL PERTURBATIONS (arXiv 2018)

Robustness



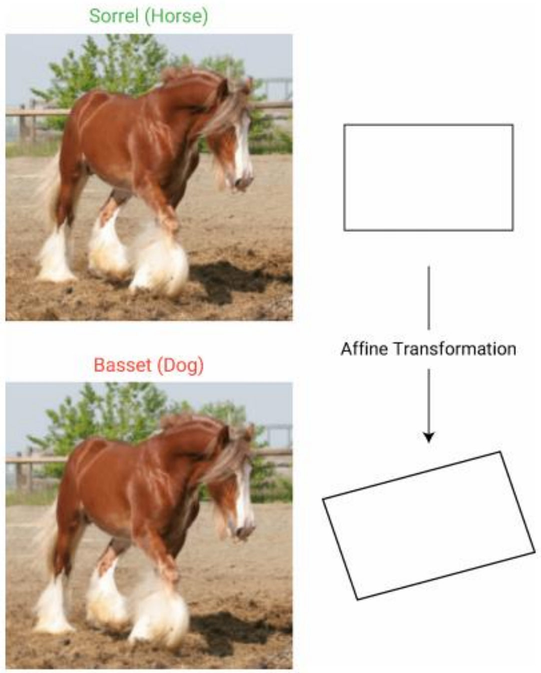
Test Accuracy	Regular CNN	Deformable CNN
Regular MNIST	98.74%	97.27%
Scaled MNIST	57.01%	92.55%

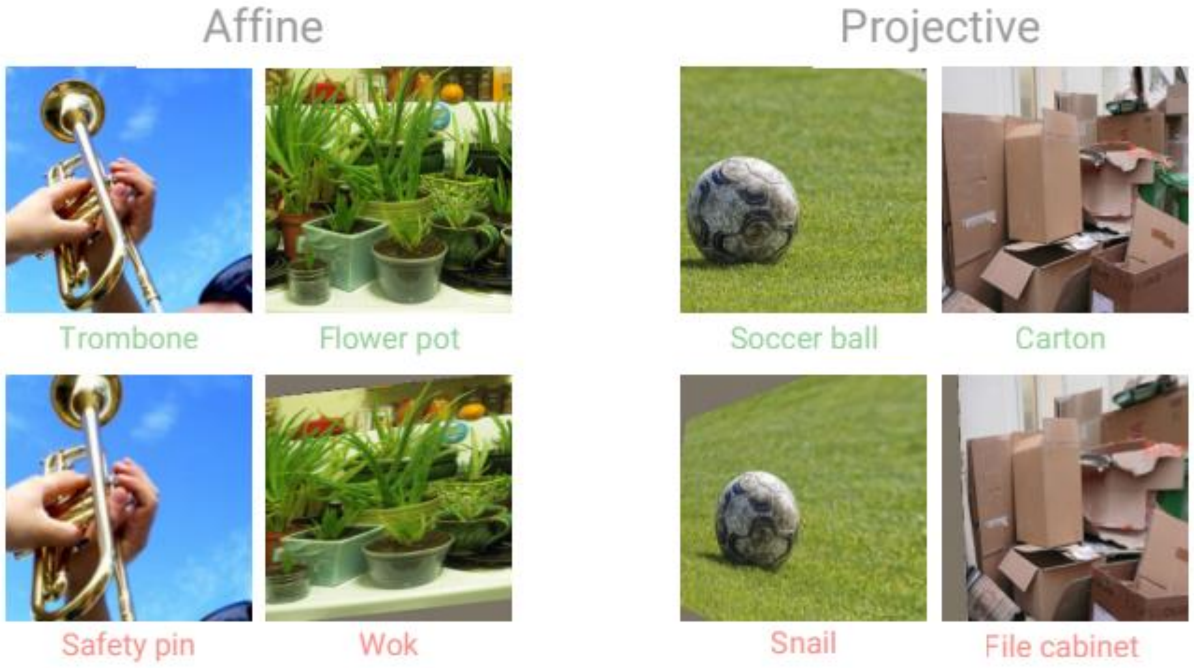
deformable convolution is able to more effectively utilize already learned feature map to represent geometric distortion.

https://github.com/felixlaumon/deform-conv

Questions to think about

Can Deformable Conv add stability to a ‘fooling’ from affine transform? [1]



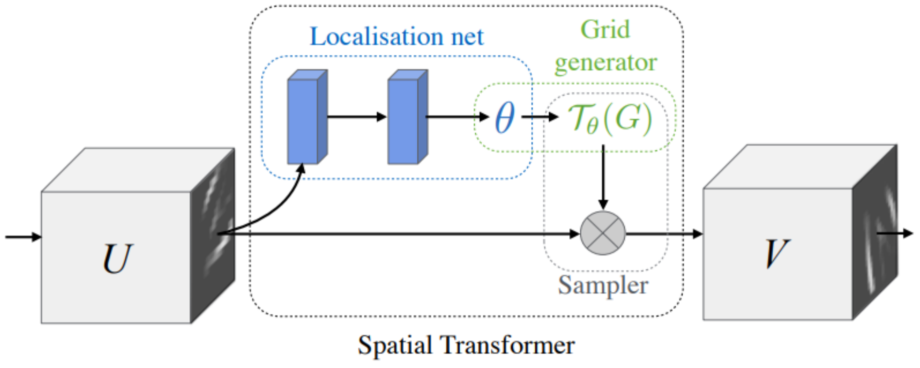


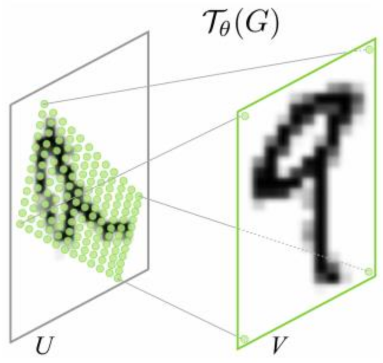
Can spatially transformed adversarial examples be contained? [2]

1. Geometric robustness of deep networks: analysis and improvement (CVPR 2018)
2. Spatially Transformed Adversarial Examples (ICLR 2018)

Related Work

Spatial Transformer Network (NIPS 2015)

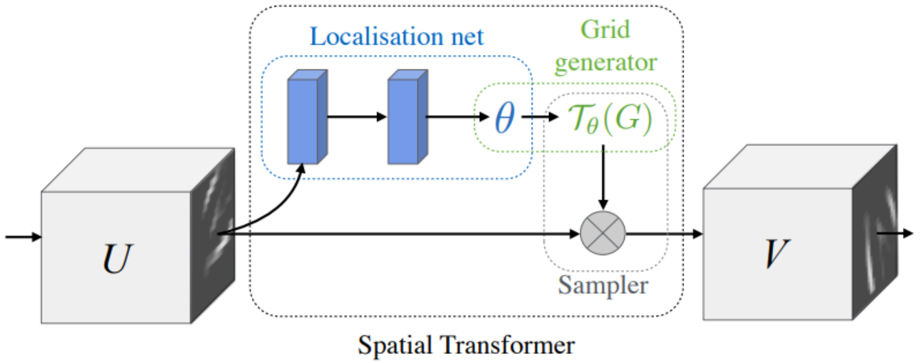


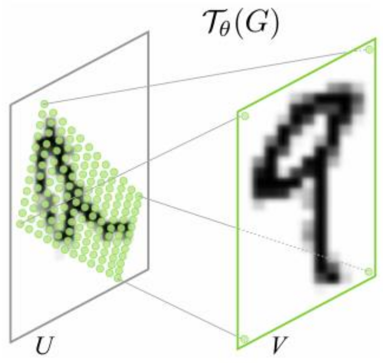


[Jaderberg, Max, Karen Simonyan, and Andrew Zisserman. "Spatial transformer networks." NIPS 2015.]

Related Work

Spatial Transformer Network (NIPS 2015)





Dynamic Filter Networks (NIPS 2016)

- Conditioned on Input features like Deformable Convolutions
- Filters weights are learned and not sampling locations

[Jaderberg, Max, Karen Simonyan, and Andrew Zisserman. "Spatial transformer networks." NIPS 2015.]

[Jia, Xu, Bert De Brabandere, Tinne Tuytelaars, and Luc V. Gool. "Dynamic filter networks.“ NIPS 2016.]

Unanswered Questions

Does this reduce data augmentation?

Unanswered Questions

Does this reduce data augmentation?

Are they equally adaptable to Pooling and multiple conv kernels?

Unanswered Questions

Does this reduce data augmentation?

Are they equally adaptable to Pooling and multiple conv kernels?

Do they reduce model complexity if trained from scratch?

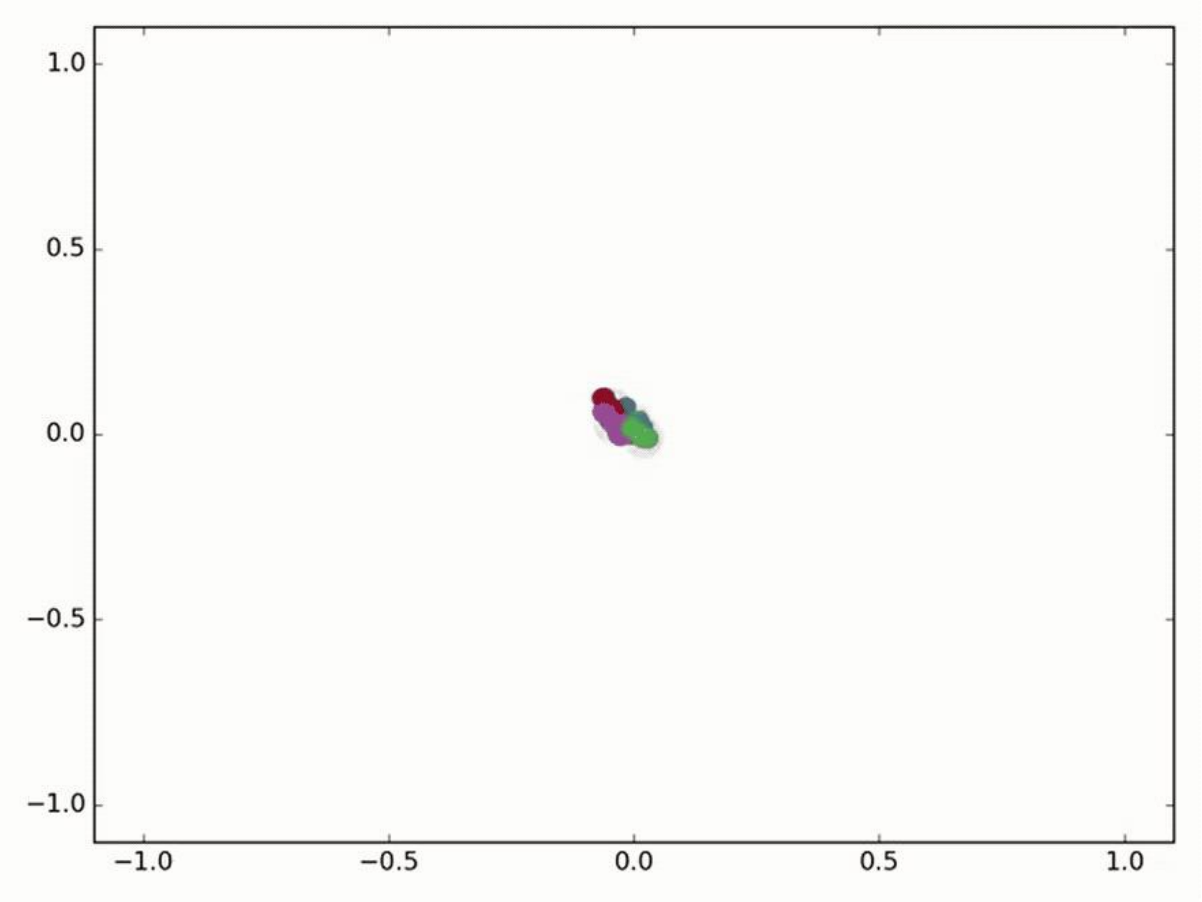
Unanswered Questions

Does this reduce data augmentation?

Are they equally adaptable to Pooling and multiple conv kernels?

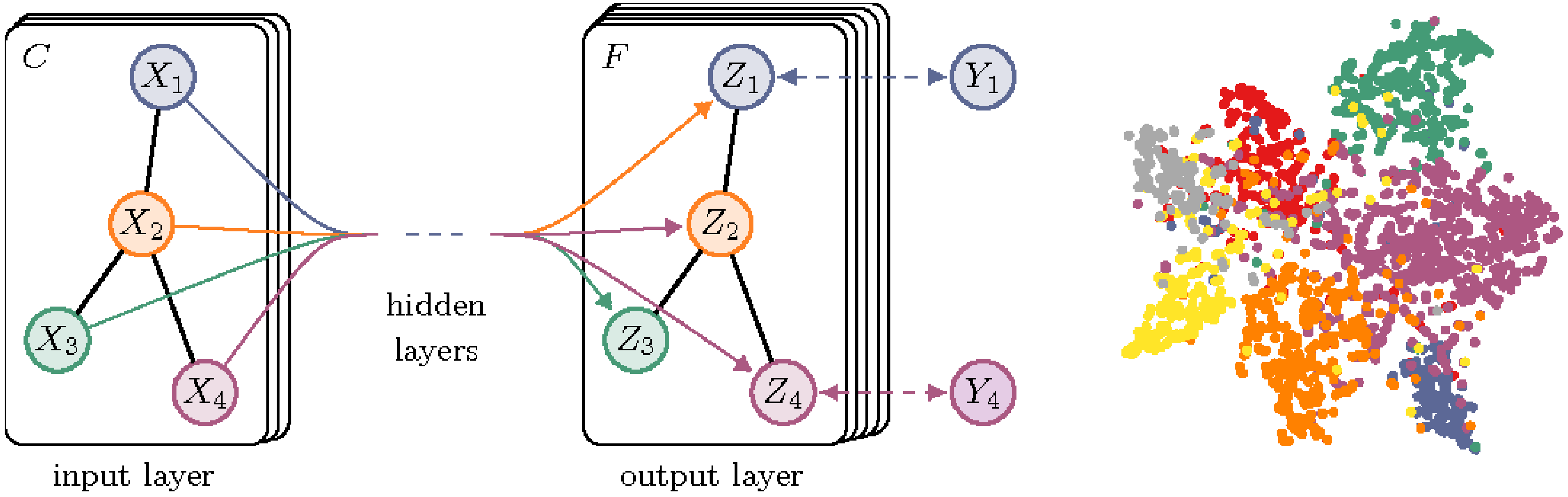
Do they reduce model complexity if trained from scratch?

Vulnerable to attacks?



Graph Neural Networks

Graph Convolutional Network



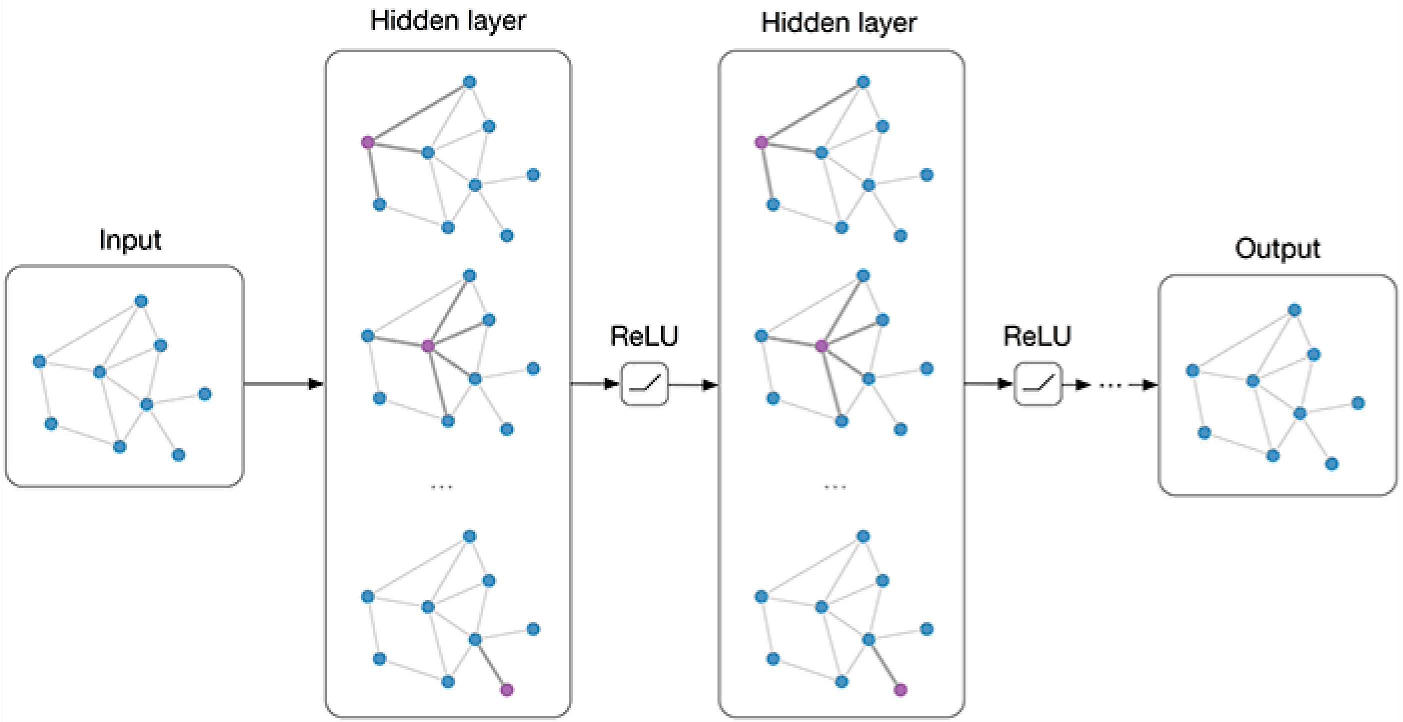
(a) Graph Convolutional Network

(b) Hidden layer activations

Image Credits: Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).

Applications

- Social Networks
- Protein-Protein Interaction
- 3D Meshes
- Clustering
- Scene Graphs



Survey: Bronstein, Michael M., et al. "Geometric deep learning: going beyond euclidean data." *IEEE Signal Processing Magazine*34.4 (2017): 18-42.

CNN vs GCNN

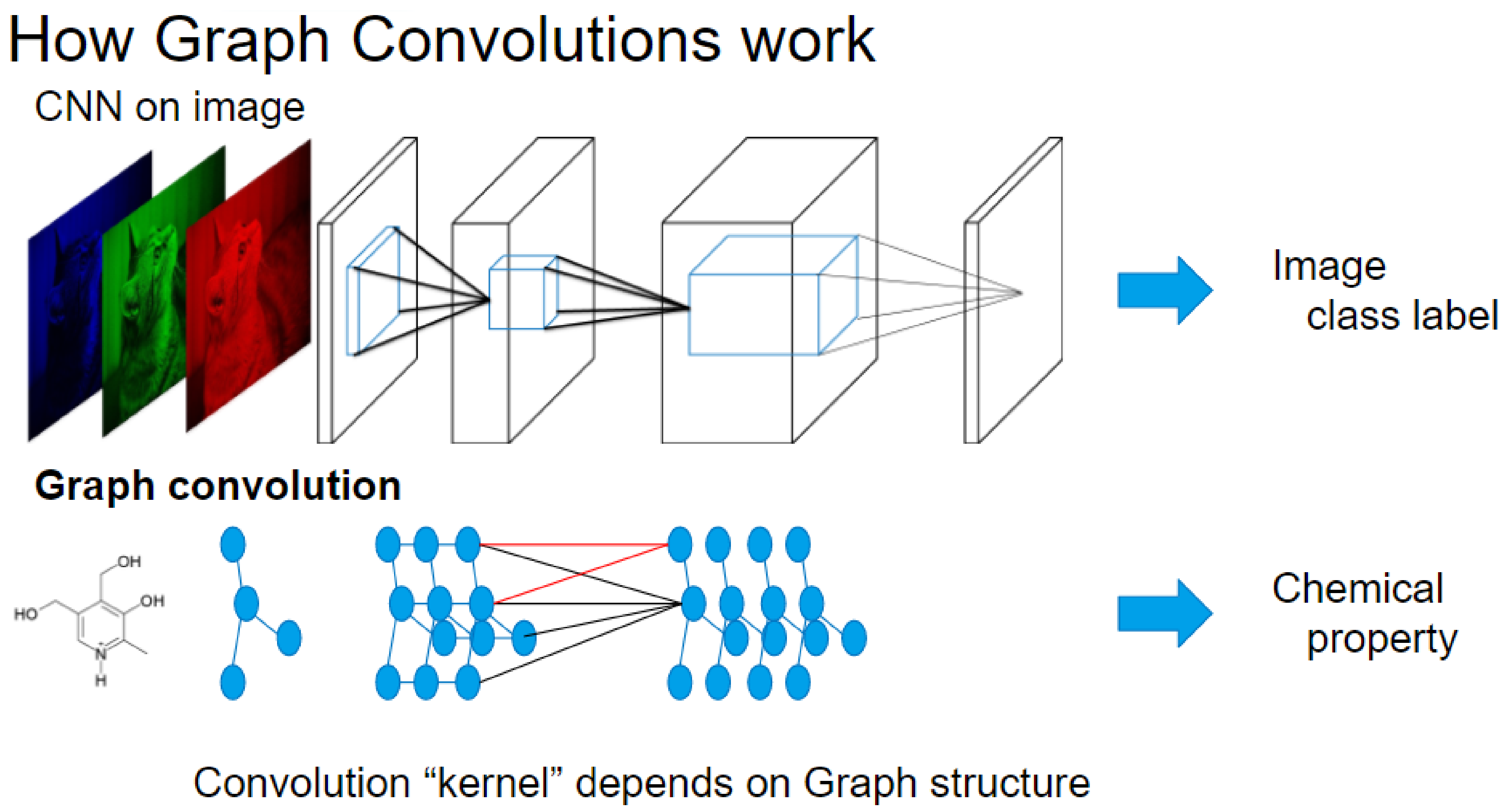
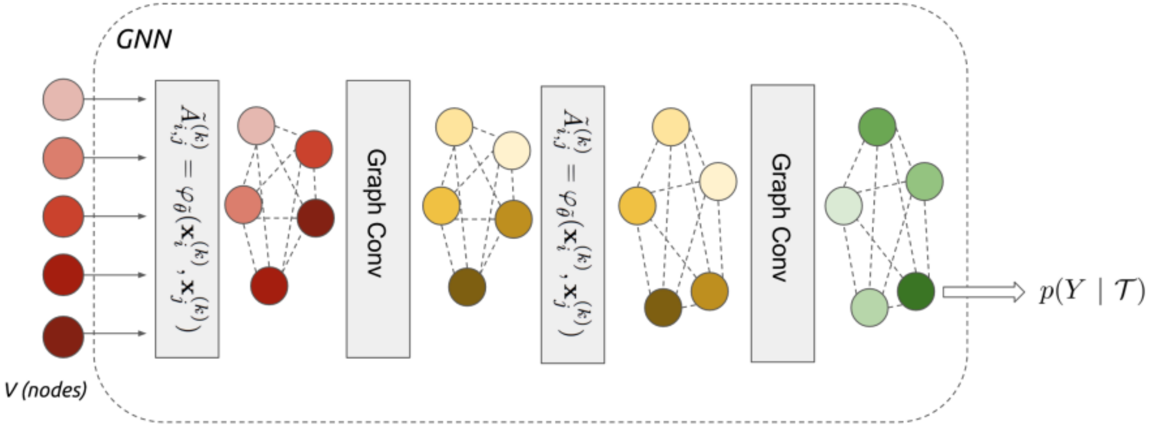


Image Credit: https://preferredresearch.jp/2017/12/18/chainer-chemistry-beta-release/

Graph Learning Problem

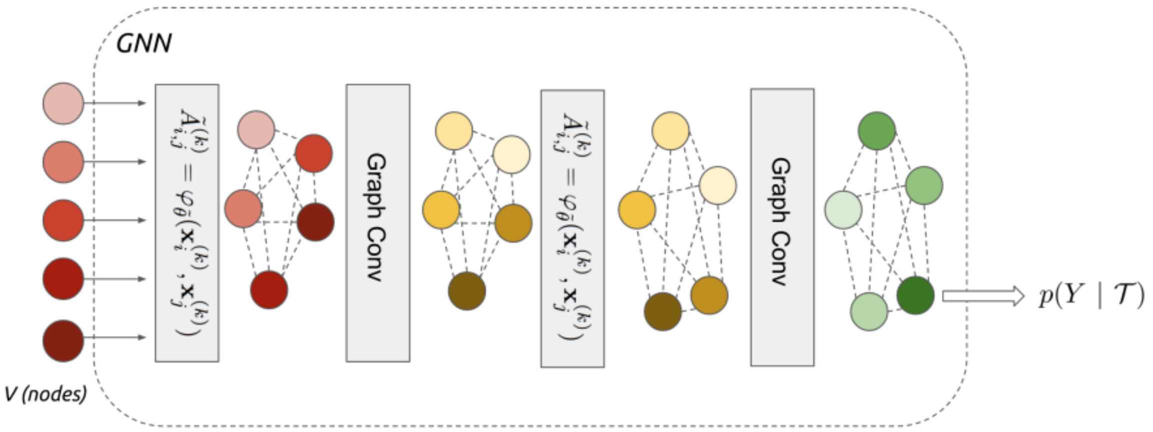
Given a set of nodes, each with some observed numeric attributes $\boldsymbol{x}_i$



Graph Learning Problem

Given a set of nodes, each with some observed numeric attributes $\boldsymbol{x}_i$

For each node, predict an output or label $\boldsymbol{y}_i$

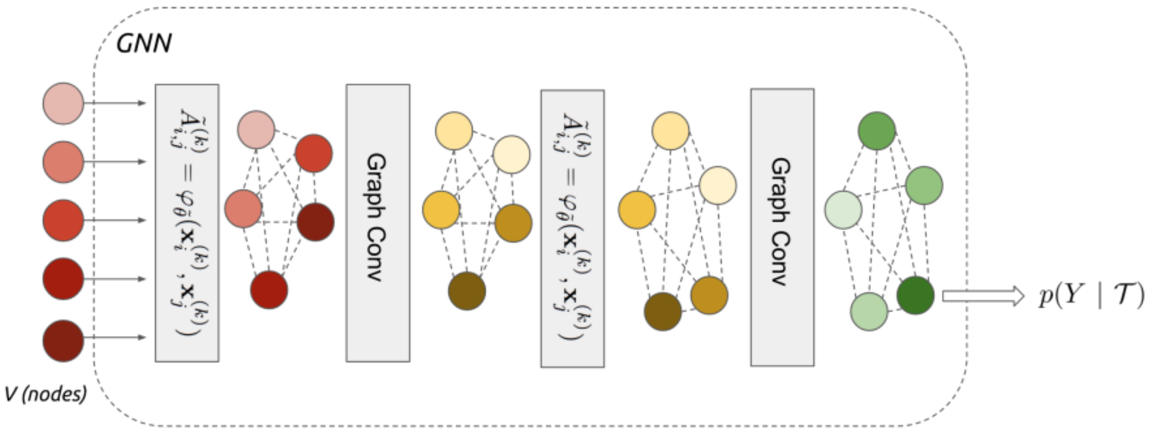


Graph Learning Problem

Given a set of nodes, each with some observed numeric attributes $\boldsymbol{x}_i$

For each node, predict an output or label $\boldsymbol{y}_i$

We observe these labels for some, but not all, of the nodes



Graph Learning Problem

Given a set of nodes, each with some observed numeric attributes $\boldsymbol{x}_i$

For each node, predict an output or label $\boldsymbol{y}_i$

We observe these labels for some, but not all, of the nodes

A set of weighted edges, an adjacency matrix **A**

Graph Learning Problem

Given a set of nodes, each with some observed numeric attributes $\boldsymbol{x}_i$

For each node, predict an output or label $\boldsymbol{y}_i$

We observe these labels for some, but not all, of the nodes

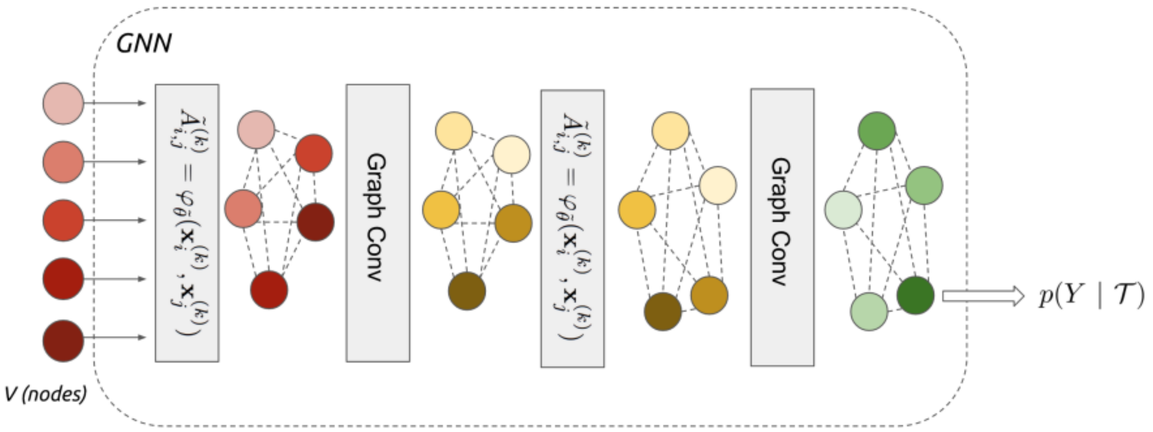
A set of weighted edges, an adjacency matrix $\mathbf{A}$

Message Passing like graphical models

Understanding Graph Neural Networks

Eq 2 of the paper

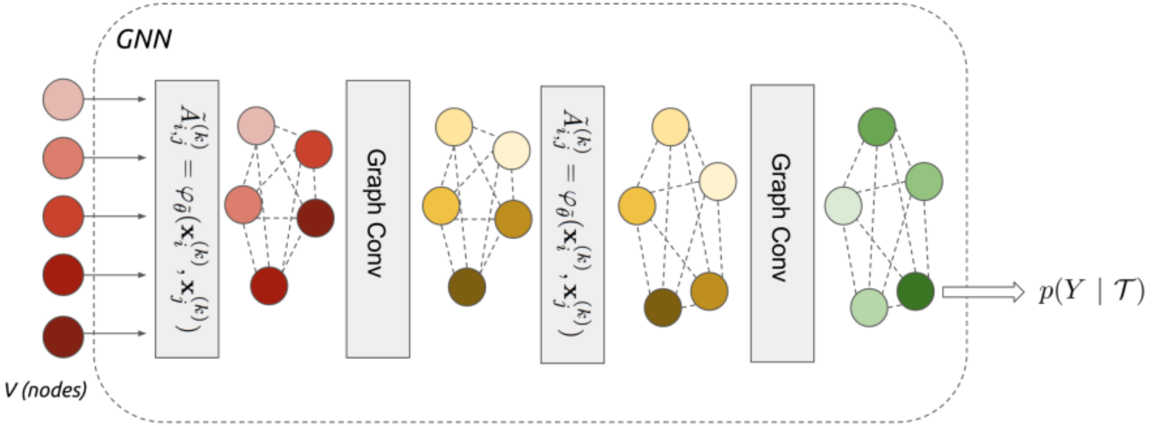
$$\mathbf{x}_l^{(k+1)} = \mathrm{Gc}(\mathbf{x}^{(k)}) = \rho\left(\sum_{B\in\mathcal{A}} B\mathbf{x}^{(k)}\theta_{B,l}^{(k)}\right)\ ,\ l = d_1 \ldots d_{k+1}$$



Understanding Graph Neural Networks

Eq 2 of the paper

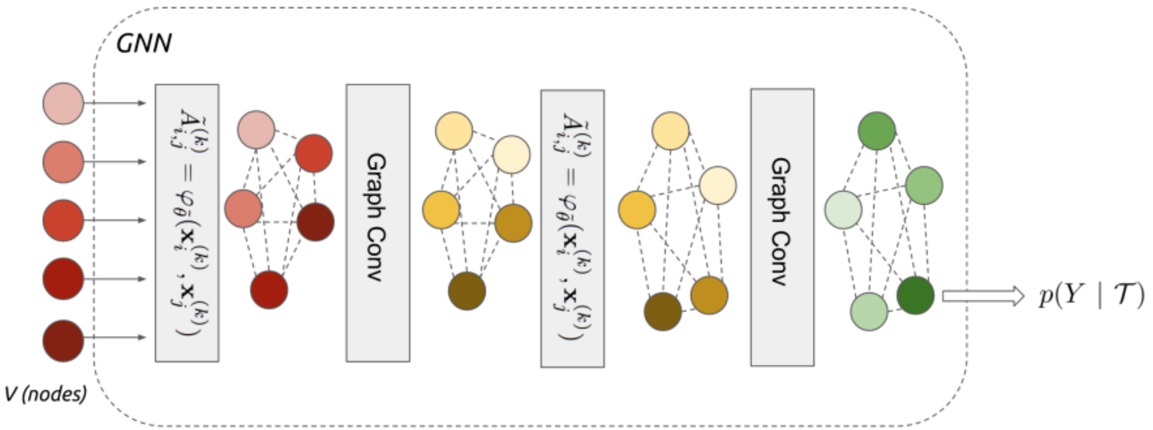
$$\mathbf{x}_l^{(k+1)} = \mathrm{Gc}(\mathbf{x}^{(k)}) = \rho\left(\sum_{B\in\mathcal{A}} B\mathbf{x}^{(k)}\theta_{B,l}^{(k)}\right) \; , \; l = d_1 \ldots d_{k+1}$$



Understanding Graph Neural Networks

Eq 2 of the paper

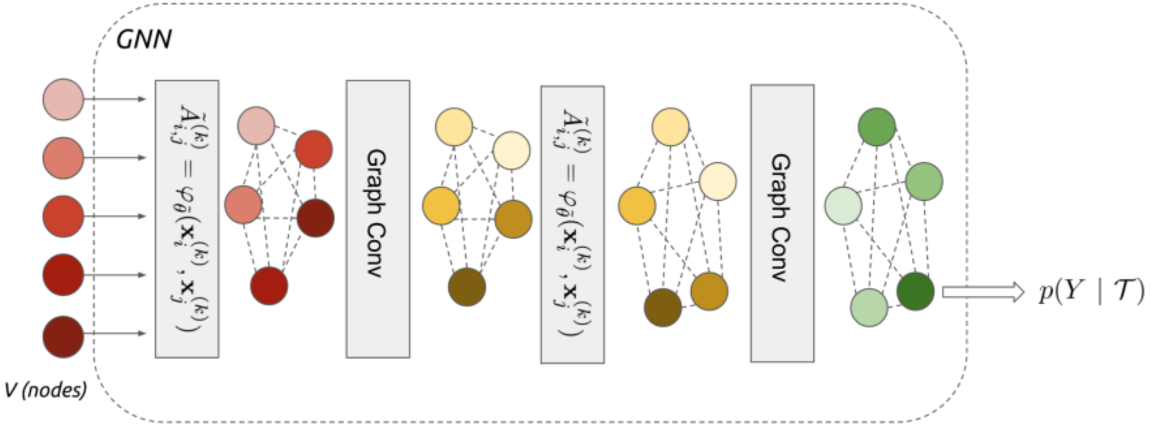
$$\mathbf{x}_l^{(k+1)} = \mathrm{Gc}(\mathbf{x}^{(k)}) = \rho\left(\sum_{B \in \mathcal{A}} B\mathbf{x}^{(k)}\theta_{B,l}^{(k)}\right), \; l = d_1 \ldots d_{k+1}$$



Understanding Graph Neural Networks

Eq 2 of the paper

$$\mathbf{x}_l^{(k+1)} = \mathrm{Gc}(\mathbf{x}^{(k)}) = \rho\left(\sum_{B\in\mathcal{A}} B\mathbf{x}^{(k)}\theta_{B,l}^{(k)}\right) \quad, l = d_1 \ldots d_{k+1}$$



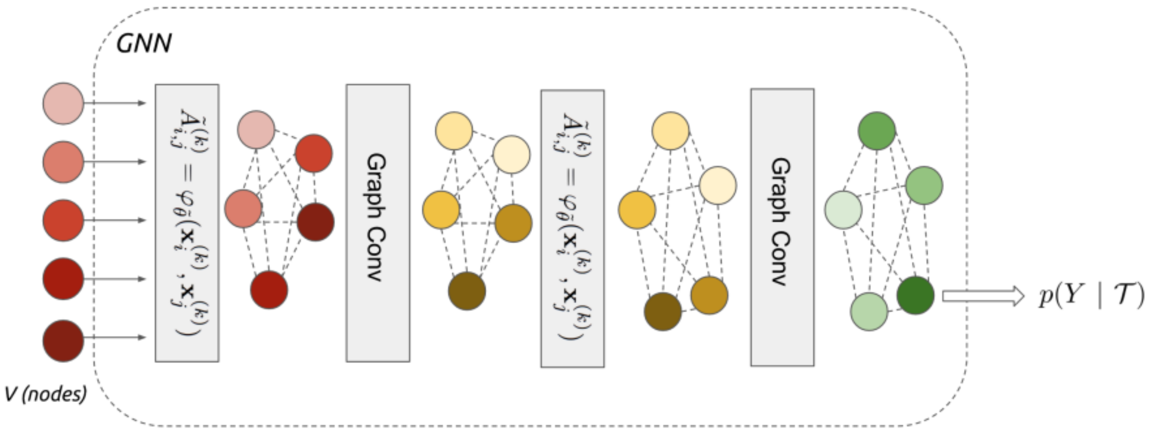
Understanding Graph Neural Networks

Eq 2 of the paper

$$\mathbf{x}_l^{(k+1)} = \mathrm{Gc}(\mathbf{x}^{(k)}) = \rho\left(\sum_{B\in\mathcal{A}} B\mathbf{x}^{(k)}\theta_{B,l}^{(k)}\right) \,,\, l = d_1 \ldots d_{k+1}$$

Adjacency Matrix (Eq 4 of the paper)

$$\varphi_{\tilde{\theta}}(\mathbf{x}_i^{(k)}, \mathbf{x}_j^{(k)}) = \mathrm{MLP}_{\tilde{\theta}}(abs(\mathbf{x}_i^{(k)} - \mathbf{x}_j^{(k)}))$$



Problem Setup

$$\mathcal{T} \quad = \quad \{\{(x_1, l_1), \ldots (x_s, l_s)\}, \{\tilde{x}_1, \ldots, \tilde{x}_r\}, \{\bar{x}_1, \ldots, \bar{x}_t\} \,;\, l_i \in \{1, K\}, x_i, \tilde{x}_j, \bar{x}_j \sim \mathcal{P}_l(\mathbb{R}^N)\}$$

$$Y \quad = \quad (y_1, \ldots, y_t) \in \{1, K\}^t$$

t: number of images to classify

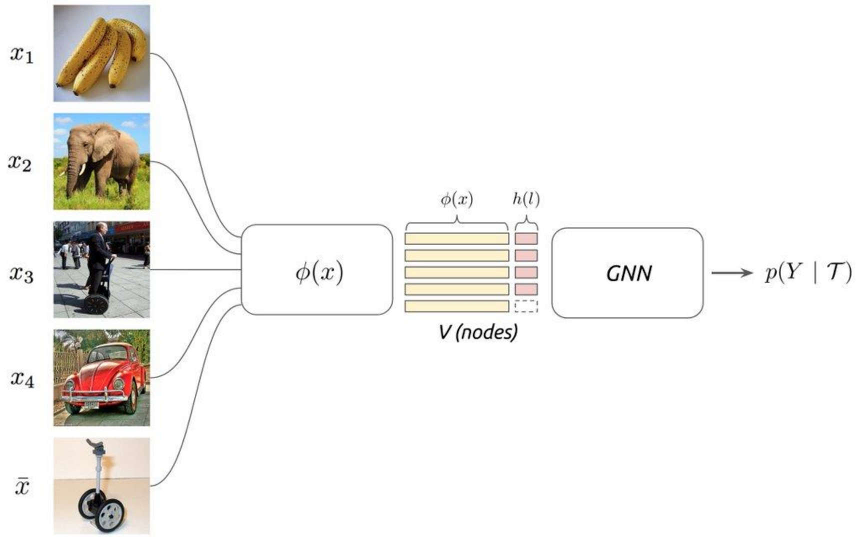
Problem Setup

$$\mathcal{T} = \{\{(x_1, l_1), \ldots (x_s, l_s)\}, \{\tilde{x}_1, \ldots, \tilde{x}_r\}, \{\bar{x}_1, \ldots, \bar{x}_t\} \; ; \; l_i \in \{1, K\}, x_i, \tilde{x}_j, \bar{x}_j \sim \mathcal{P}_l(\mathbb{R}^N)\}$$

$$Y = (y_1, \ldots, y_t) \in \{1, K\}^t$$

t: number of images to classify

Few Shot Learning: r=0, t=1, s=qk (q-shot, k-way learning)



Problem Setup

$$\mathcal{T} = \{\{(x_1, l_1), \ldots (x_s, l_s)\}, \{\tilde{x}_1, \ldots, \tilde{x}_r\}, \{\bar{x}_1, \ldots, \bar{x}_t\} \,;\, l_i \in \{1, K\}, x_i, \tilde{x}_j, \bar{x}_j \sim \mathcal{P}_l(\mathbb{R}^N)\}$$

$$Y = (y_1, \ldots, y_t) \in \{1, K\}^t$$

t: number of images to classify

Few Shot Learning: r=0, t=1, s=qk (q-shot, k-way learning)

Semi-Supervised Learning: r>0 and t =1

Problem Setup

$$\mathcal{T} = \{\{(x_1, l_1), \ldots (x_s, l_s)\}, \{\tilde{x}_1, \ldots, \tilde{x}_r\}, \{\bar{x}_1, \ldots, \bar{x}_t\} \, ; \, l_i \in \{1, K\}, x_i, \tilde{x}_j, \bar{x}_j \sim \mathcal{P}_l(\mathbb{R}^N)\}$$

$$Y = (y_1, \ldots, y_t) \in \{1, K\}^t$$

t: number of images to classify

Few Shot Learning: r=0, t=1, s=qk (q-shot, k-way learning)

Semi-Supervised Learning: r>0 and t =1

Active Learning: Request labels for $\{\tilde{x}_1, \ldots, \tilde{x}_r\}$

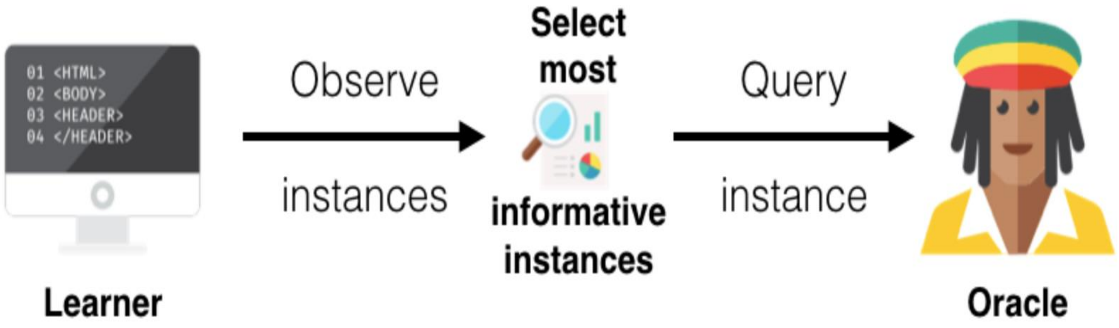


Image Credits: https://www.datacamp.com/community/tutorials/active-learning

Few Shot Learning Experiments

Model	5-Way 1-shot	5-Way 5-shot	20-Way 1-shot	20-Way 5-shot
Pixels Vinyals et al. (2016)	41.7%	63.2%	26.7%	42.6%
Siamese Net Koch et al. (2015)	97.3%	98.4%	88.2%	97.0%
Matching Networks Vinyals et al. (2016)	98.1%	98.9%	93.8%	98.5%
N. Statistician Edwards & Storkey (2016)	98.1%	99.5%	93.2%	98.1%
Res. Pair-Wise Mehrotra & Dukkipati (2017)	-	-	94.8%	-
Prototypical Networks Snell et al. (2017)	97.4%	99.3%	95.4%	98.8%
ConvNet with Memory Kaiser et al. (2017)	98.4%	99.6%	95.0%	98.6%
Agnostic Meta-learner Finn et al. (2017)	98.7 ±0.4%	99.9 ±0.3%	95.8 ±0.3%	98.9 ±0.2%
Meta Networks Munkhdalai & Yu (2017)	98.9%	-	97.0%	-
TCML Mishra et al. (2017)	98.96% ±0.20%	99.75% ±0.11%	97.64% ±0.30%	99.36% ±0.18%
Our GNN	99.2%	99.7%	97.4%	99.0%

Table 1: Few-Shot Learning — Omniglot accuracies. Siamese Net results are extracted from Vinyals et al. (2016) reimplementation.

Few Shot Learning Experiments

	5-Way	
Model	**1-shot**	**5-shot**
Matching Networks Vinyals et al. (2016)	43.6%	55.3%
Prototypical Networks Snell et al. (2017)	46.61% ±0.78%	65.77% ±0.70%
Model Agnostic Meta-learner Finn et al. (2017)	48.70% ±1.84%	63.1% ±0.92%
Meta Networks Munkhdalai & Yu (2017)	49.21% ±0.96	-
Ravi & Larochelle Ravi & Larochelle (2016)	43.4% ±0.77%	60.2% ±0.71%
TCML Mishra et al. (2017)	55.71% ±0.99%	68.88% ±0.92%
Our metric learning + KNN	49.44% ±0.28%	64.02% ±0.51%
Our GNN	50.33% ±0.36%	66.41% ±0.63%

Table 2: Few-shot learning — Mini-Imagenet average accuracies with 95% confidence intervals.

Semi-Supervised Experiments

Model	5-Way 5-shot 20%-labeled	40%-labeled	100%-labeled
GNN - Trained only with labeled	99.18%	99.59%	99.71%
GNN - Semi supervised	99.59%	99.63%	99.71%

Table 3: Semi-Supervised Learning — Omniglot accuracies.

Model	5-Way 5-shot 20%-labeled	40%-labeled	100%-labeled
GNN - Trained only with labeled	50.33% ±0.36%	56.91% ±0.42%	66.41% ±0.63%
GNN - Semi supervised	52.45% ±0.88%	58.76% ±0.86%	66.41% ±0.63%

Table 4: Semi-Supervised Learning — Mini-Imagenet average accuracies with 95% confidence intervals.

Active Learning Experiments

Method	5-Way 5-shot 20%-labeled
GNN - AL	99.62%
GNN - Random	99.59%

Method	5-Way 5-shot 20%-labeled
GNN - AL	55.99% $\pm$1.35%
GNN - Random	52.56% $\pm$1.18%

Table 5: Omniglot (left) and Mini-Imagneet (right), average accuracies are shown at both tables, the GNN-AL is the learned criterion that performs Active Learning by selecting the sample that will maximally reduce the loss of the current classification. The GNN - Random is also selecting one sample, but in this case a random one. Mini-Imagenet results are presented with 95% confidence intervals.

Kipf and Welling (ICLR 2017)

Generalization of convolutions, and easiest to define in spectral domain

Thomas N. Kipf and Max Welling (2016) **Semi-Supervised Classification with Graph Convolutional Networks**
Ferenc Huszar's blog: https://www.inference.vc/how-powerful-are-graph-convolutions-review-of-kipf-welling-2016-2/

Kipf and Welling (ICLR 2017)

Generalization of convolutions, and easiest to define in spectral domain

Fourier transform scales poorly with size of data so we need relaxations

Thomas N. Kipf and Max Welling (2016) **Semi-Supervised Classification with Graph Convolutional Networks**
Ferenc Huszar's blog: https://www.inference.vc/how-powerful-are-graph-convolutions-review-of-kipf-welling-2016-2/

Kipf and Welling (ICLR 2017)

Generalization of convolutions, and easiest to define in spectral domain

Fourier transform scales poorly with size of data so we need relaxations

First order approximation in Fourier-domain to obtain an efficient linear-time graph-CNNs

Thomas N. Kipf and Max Welling (2016) **Semi-Supervised Classification with Graph Convolutional Networks**
Ferenc Huszar's blog: https://www.inference.vc/how-powerful-are-graph-convolutions-review-of-kipf-welling-2016-2/

Kipf and Welling (ICLR 2017)

Generalization of convolutions, and easiest to define in spectral domain

Fourier transform scales poorly with size of data so we need relaxations

First order approximation in Fourier-domain to obtain an efficient linear-time graph-CNNs

Modelling power is **severely impoverished**, due to the first-order and other approximations made.

Thomas N. Kipf and Max Welling (2016) **Semi-Supervised Classification with Graph Convolutional Networks**
Ferenc Huszar's blog: https://www.inference.vc/how-powerful-are-graph-convolutions-review-of-kipf-welling-2016-2/

Limitations

Brittle – Also see, On Computational Hardness with Graph Neural Networks. Joan Bruna [video]

Limitations

Brittle – Also see, On Computational Hardness with Graph Neural Networks. Joan Bruna [video]

How powerful are graph neural networks? [arXiv, 10/01/2018]

GNNs revolutionizing graph representation learning, there is limited understanding of their representational properties and limitations. Here, we present a theoretical framework for analyzing the expressive power of GNNs in capturing different graph structures. Our results characterize the discriminative power of popular GNN variants, such as Graph Convolutional Networks and GraphSAGE, and show that they cannot learn to distinguish certain simple graph structures. We then

Attacks on Language-Vision problems?

Image Captioning [this]

Visual Dialog

Scene Graph Generation

10/25 **Adversarial Attacks on Graphs**

- Adversarial Attack on Graph Structured Data
- Adversarial Attacks on Neural Networks for Graph Data

Related Work

- Bronstein, Michael M., et al. "Geometric deep learning: going beyond euclidean data." *IEEE Signal Processing Magazine*34.4 (2017): 18-42. [paper]
- Schlichtkrull, Michael, et al. "Modeling relational data with graph convolutional networks." *European Semantic Web Conference*. Springer, Cham, 2018. [paper]
- Wang, Nanyang, et al. "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images." ECCV (2018). [paper]
- Yang, Jianwei, et al. "Graph R-CNN for Scene Graph Generation." ECCV (2018). [paper]

Other Questions on g-sheets

- In "Deformable Convolutional Networks", for position-sensitive RoI pooling, why do we need C+1 for C object classes? [Background]
- I don't see how deformable convolution can help against adversarial examples since everything is differentiable too. [Can break classifiers, will make detectors more robust]
- On Deformable Conv, How is this different from using a large kernel with dropout per filter application [Not sure]
- Few-Shot Learning with Graph Neural Networks: Why the particular choice of pointwise abs in Eq 4, as opposed to something else? (Like pointwise squared distance) [features are more representative]

Other Questions on g-sheets

- Deformoable convolution is just an extreme case of self attention. Why don't the authors use self attention? [Yes, but self-attention doesn’t provide invariance properties to CNN]
- In GCN, is there a way to use GCN as an unsupervised feature extractor on graph to learn node embedding. [GCN are inherently clustering nodes – maybe extract mean features of each cluster]
- Have trouble understanding the graph operations. Would like to get an explanation [Hopefully, it’s clear now]
- Deformable Convolutional Networks": Could the deformable convolutional layers be extended to be useful in generative models as well? [Interesting question - I don’t think they would add anything substantial]