Machine Intelligence & Deep Learning Workshop

Raymond Ptucha, Majid Rabbani, Mark Smith

The Kate Gleason COLLEGE OF ENGINEERING

Graph CNN

Raymond Ptucha
June 27-29, 2018
Rochester Institute of Technology
www.rit.edu/kgcoe/cqas/machinelearning

Fair Use Agreement

This agreement covers the use of all slides in this document, please read carefully.

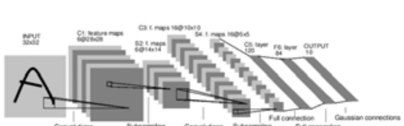
- You may freely use these slides for personal use, if:
  - My name (R. Ptucha) appears on each slide.
- You may freely use these slides externally, if:
  - You send me an email telling me the conference/venue/company name in advance, and which slides you wish to use.
  - You receive a positive confirmation email back from me.
  - My name (R. Ptucha) appears on each slide you use.

(c) Raymond Ptucha, rwpeec@rit.edu

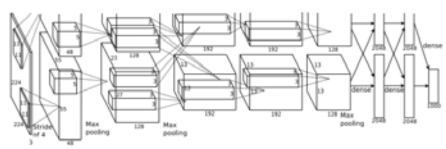
Agenda

- Wed, June 27
  - 9-10:30am Regression and Classification
  - 10:30-10:45pm Break
  - 10:45-12:15pm Boosting and SVM
  - 12:15-1:30pm Lunch
  - 1:30-3:30pm Neural Networks and Dimensionality Reduction
  - 3:30-5pm Hands-on Python and Machine Learning
- Thur, June 28
  - 9-10:30am Introduction to deep learning
  - 10:30-10:45pm Break
  - 10:45-12:15pm Convolutional Neural Networks
  - 12:15-1:30pm Lunch
  - 1:30-3:30pm Region and pixel-level convolutions
  - 3:30-5pm Hands-on CNNs
- Fri, June 29
  - 9-10:30am Recurrent neural networks
  - 10:30-10:45pm Break
  - 10:45-12:15pm Language and Vision
  - 12:15-1:30pm Lunch
  - 1:30-3:30pm **Graph convolutional neural networks**; Generative adversarial networks
  - 3:30-5pm Hands-on regional CNNs, RNNs

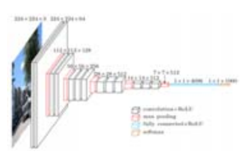
Many Flavors of CNNs…



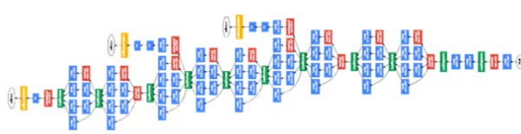
LeNet-5, LeCun 1989



AlexNet, Krizhevsky 2012



VGGNet, Simonyan 2014

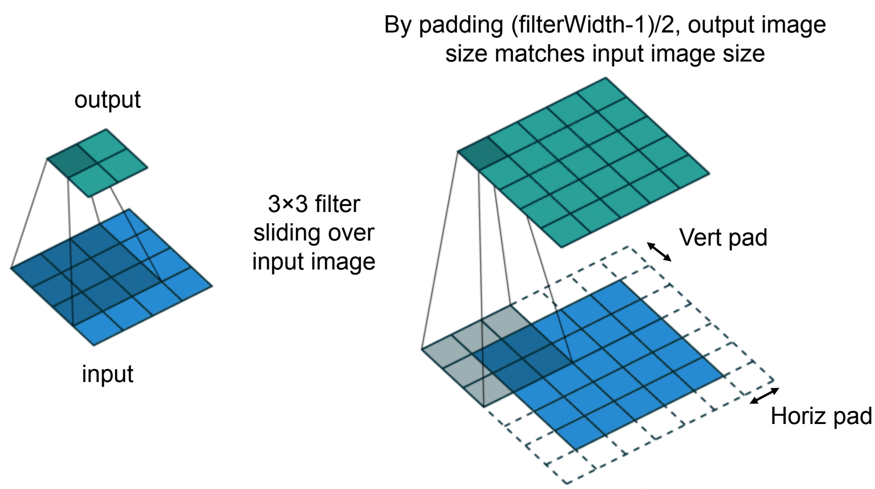


GoogLeNet (Inception), Szegedy 2014

ResNet, He 2015

DenseNet, Huang 2017

Image Convolution



https://github.com/vdumoulin/conv_arithmetic

Max Pooling- Reducing the Size of an Image

Single depth slice

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

x

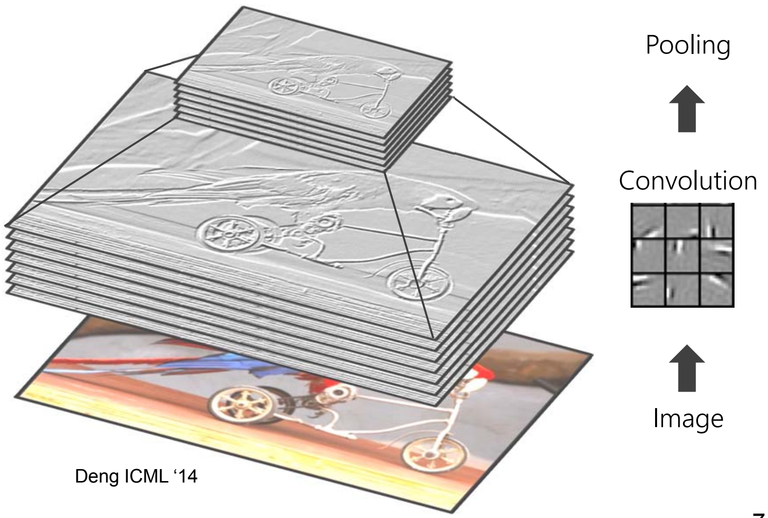
y

max pool with 2x2 filters and stride 2

6	8
3	4

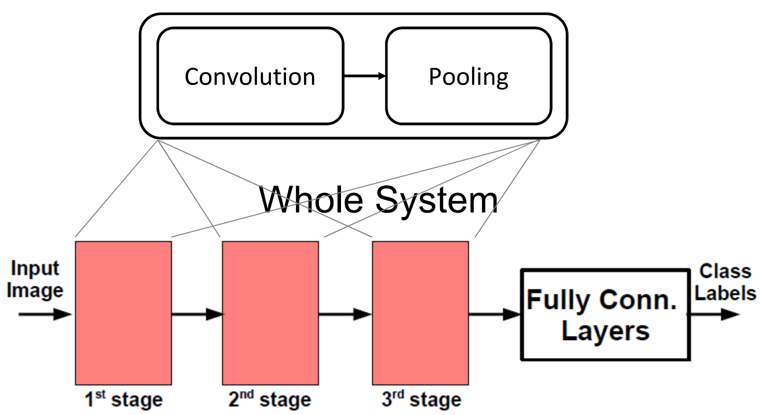
cs321n, Karpathy, Li

Convolution Neural Network (CNN) Building Block

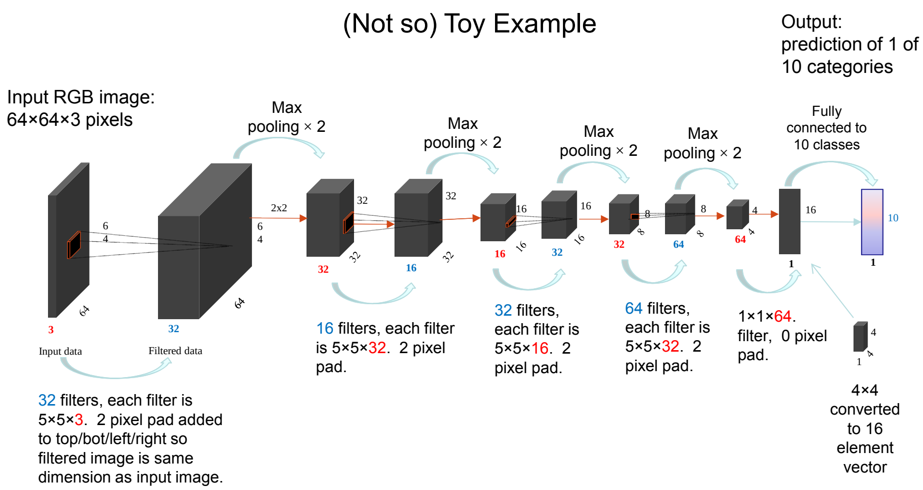


Deng ICML '14

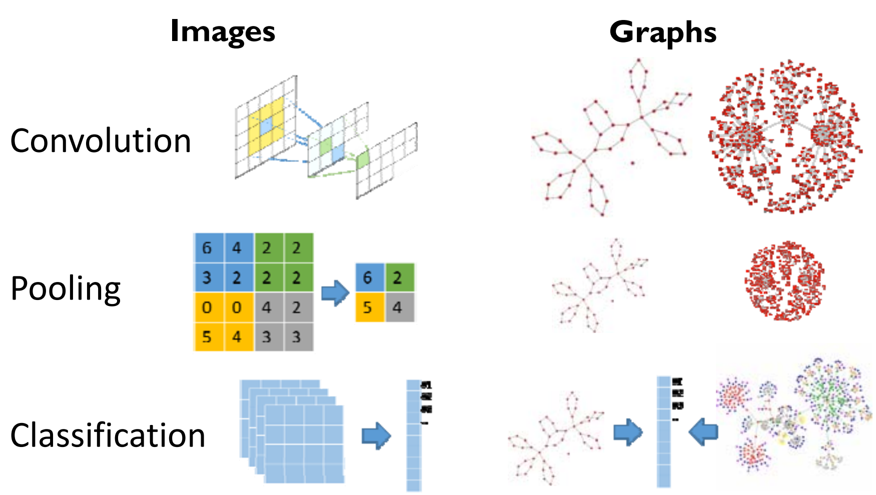
Putting it All Together



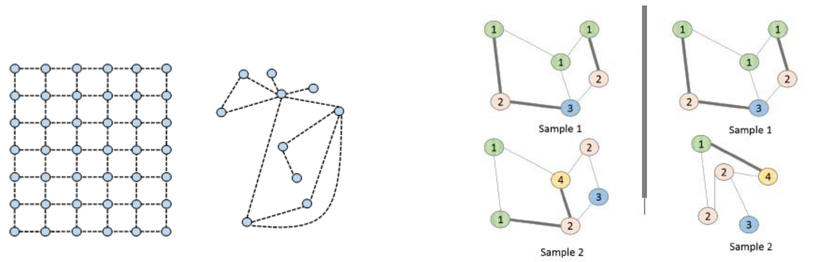
CNN Architecture



But, the vast majority of the world's problems can't be described by gridded structures such as images. Have you ever tried to do a CNN on a graph?



GraphCNN affords the wonderful CNN benefits to non-gridded problems such as trade, security, protein structures, weather, brain scans, etc.

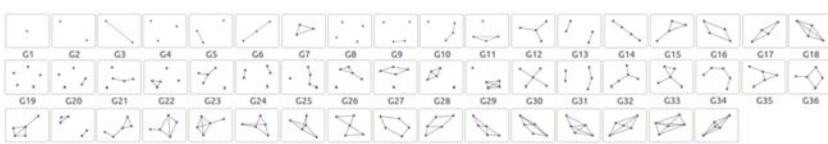


Gridded Non-Gridded Homogeneous Heterogeneous

(Each sample has different number vertices or edges.)

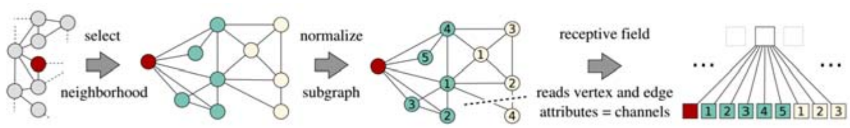
Spatial Approaches

- Graph Kernels- comparison of local neighborhoods



P. Yanardag and S. Vishwanathan, "Deep graph kernels," in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2015, pp. 1365–1374.

- Turn Vertices into Padded Vectors



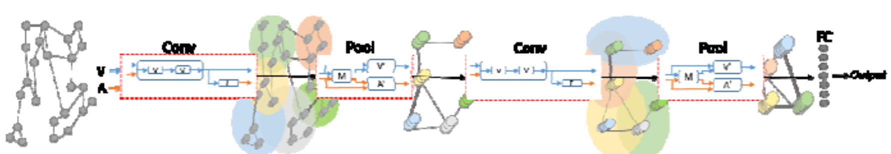
Niepert, Mathias, Mohamed Ahmed, and Konstantin Kutzkov. "Learning Convolutional Neural Networks for Graphs." *arXiv preprint arXiv:1605.05273*(2016).

Spectral Approaches

- Filter in a spectral domain by constructing an analogue to the DFT
  - Eigenvector decomposition of the graph Laplacian
  - Laplacian L = D - A;
    - A is adjacency matrix,
    - D is diagonal matrix of row-wise sums of A
- Apply PCA, to get to spectral space
- Filter in PCA space (convolutions are multiplies)
- Pool in PCA space (trim out high frequencies and resample)

Graph Convolutional Neural Network (Graph CNN)

- Replaces classic layers with graph capable alternatives.
  - Superset of CNNs.
  - CNN techniques can be transferred (e.g. SGD, BatchNorm, Dropout, Regularization).



CNN - Convolution

2 x 2 filter with a stride of 2

$$\begin{bmatrix}6&4&2&2\\3&2&2&2\\0&0&4&2\\5&4&3&3\end{bmatrix} * \begin{bmatrix}1&-1\\1&-1\end{bmatrix} = \begin{bmatrix}3&0\\1&2\end{bmatrix}$$

Vector implementation (CUDA)

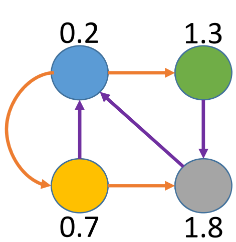
Vectorized using im2col

$$\begin{bmatrix}6&4&2&2\\3&2&2&2\\0&0&4&2\\5&4&3&3\end{bmatrix} \Rightarrow \begin{bmatrix}6&4&3&2\\2&2&2&2\\0&0&5&4\\4&2&3&3\end{bmatrix}$$

Matrix Mult. + Reshape

$$\begin{bmatrix}6&4&3&2\\2&2&2&2\\0&0&5&4\\4&2&3&3\end{bmatrix} \times \begin{bmatrix}1\\-1\\1\\-1\end{bmatrix} = \begin{bmatrix}3\\0\\1\\2\end{bmatrix} \Rightarrow \begin{bmatrix}3&0\\1&2\end{bmatrix}$$

Graph-CNN - Convolution



Adjacency Matrix

Incoming Purple Connections

A

$$\begin{bmatrix}0&0&1&1\\0&0&0&0\\0&0&0&0\\0&1&0&0\end{bmatrix}$$

Incoming Orange Connections

A

$$\begin{bmatrix}0&0&0&0\\1&0&0&0\\1&0&0&0\\0&0&1&0\end{bmatrix}$$

Vertices

V

$$\begin{bmatrix}0.2\\1.3\\0.7\\1.8\end{bmatrix}$$

Can have multiple connection types (A can be a tensor) and can have multiple features (V can be a tensor)

Graph-CNN - Convolution

A

0	0	1	1
0	0	0	0
0	0	0	0
0	1	0	0

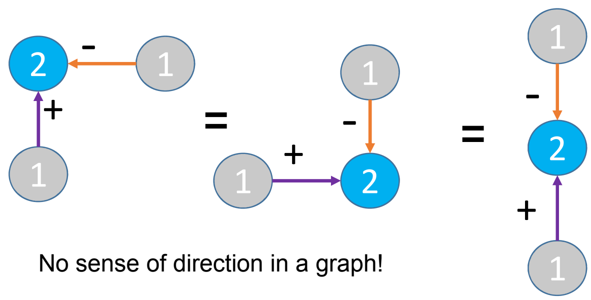
A

0	0	0	0
1	0	0	0
1	0	0	0
0	0	1	0

V

0.2
1.3
0.7
1.8

Replace each vertex with 2× itself + 1× incoming purple - 1× incoming orange



No sense of direction in a graph!

Graph-CNN - Convolution

A

0	0	1	1
0	0	0	0
0	0	0	0
0	1	0	0

A

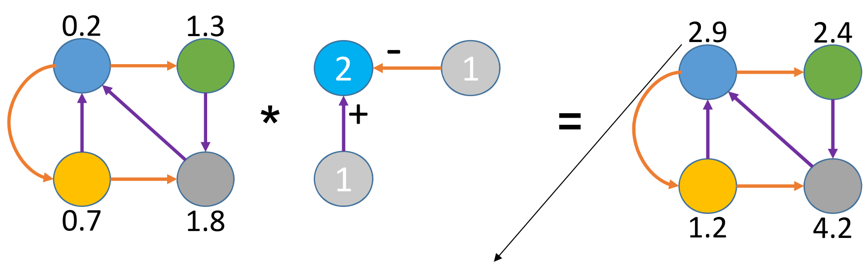
0	0	0	0
1	0	0	0
1	0	0	0
0	0	1	0

V

0.2
1.3
0.7
1.8

In Graph CNN, we will learn many such filters (like the [2 1 -1]) per adjacency matrix.

Replace each vertex with 2× itself + 1× incoming purple - 1× incoming orange



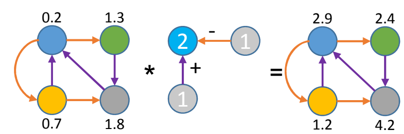
2× (0.2) + 1× (0.7+1.8) - 1× (0)

Graph-CNN - Convolution

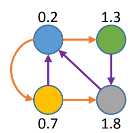
A			
0	0	1	1
0	0	0	0
0	0	0	0
0	1	0	0

A			
0	0	0	0
1	0	0	0
1	0	0	0
0	0	1	0

V
0.2
1.3
0.7
1.8



Update V’s for incoming A’s



V	AV	AV
0.2	2.5	0
1.3	0	0.2
0.7	0	0.2
1.8	1.3	0.7

$N^l_{i,aC+b} = A_{i,a,j}V^l_{j,b}$

Matrix Multiplication

$$\begin{array}{ccc} V & AV & AV \\ \hline 0.2 & 2.5 & 0 \\ 1.3 & 0 & 0.2 \\ 0.7 & 0 & 0.2 \\ 1.8 & 1.3 & 0.7 \end{array} \times \begin{array}{c} 2 \\ 1 \\ -1 \end{array} = \begin{array}{c} 2.9 \\ 2.4 \\ 1.2 \\ 4.2 \end{array}$$

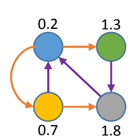
$V^l_{i,k} = f(N^{l-1}_{i,d}W_{d,k})$

20

Graph-CNN - Convolution

- We are learning the weights of the filters.
- We don’t care how many vertices!
- Can learn several sets of weights, one for each filter.

Update V’s for incoming A’s



V	AV	AV
0.2	2.5	0
1.3	0	0.2
0.7	0	0.2
1.8	1.3	0.7

$N^l_{i,aC+b} = A_{i,a,j}V^l_{j,b}$

Matrix Multiplication

$$\begin{array}{ccc} V & AV & AV \\ \hline 0.2 & 2.5 & 0 \\ 1.3 & 0 & 0.2 \\ 0.7 & 0 & 0.2 \\ 1.8 & 1.3 & 0.7 \end{array} \times \begin{array}{c} 2 \\ 1 \\ -1 \end{array} = \begin{array}{c} 2.9 \\ 2.4 \\ 1.2 \\ 4.2 \end{array}$$

$V^l_{i,k} = f(N^{l-1}_{i,d}W_{d,k})$

21

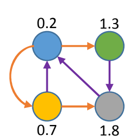
CNN – Fully Connected Layer

$$\begin{bmatrix} 6 & 2 \\ 5 & 4 \end{bmatrix} \circ \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = 3$$

- Matrix multiplication.
  - Parameters are learned.
- Requires fixed input shape, size, and order.
- Obtains representation vector.

Graph-CNN – Graph Representation Vector

Turn arbitrary # vertices into a single vertex



$$\begin{array}{cc} V & M \\ \begin{bmatrix} 0.2 \\ 1.3 \\ 0.7 \\ 1.8 \end{bmatrix} \circ & \begin{bmatrix} 2 \\ 1 \\ -2 \\ 1 \end{bmatrix} \end{array} \overset{V^T M}{=} 2.1$$

$$\begin{array}{cc} V & M^* = softmax(M) \\ \begin{bmatrix} 0.2 \\ 1.3 \\ 0.7 \\ 1.8 \end{bmatrix} \circ & \begin{bmatrix} 0.57 \\ 0.21 \\ 0.01 \\ 0.21 \end{bmatrix} \end{array} \overset{V^T M^*}{=} 0.77$$

```
x = [2 1 -2 1]
M* = exp(x)./sum(exp(x))
[0.2 1.3 0.7 1.8]*M*'
```

- Graphs can have varying vertices, but we often need fixed nodes for say a final classification task.
- Define a soft attention applied to vertices of graph- learn *M*, a linear combination of all vertices.
- This reduces all vertices to a single vertex.
- A softmax is applied to *M* before computing linear combination, this ensures the sum of the weights=1.

CNN - Pooling

$$\begin{bmatrix}6&4&2&2\\3&2&2&2\\0&0&4&2\\5&4&3&3\end{bmatrix} \Rightarrow \begin{bmatrix}6&2\\5&4\end{bmatrix}$$

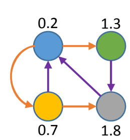
Vectorized using im2col

$$\begin{bmatrix}6&4&2&2\\3&2&2&2\\0&0&4&2\\5&4&3&3\end{bmatrix} \Rightarrow \begin{bmatrix}6&4&3&2\\2&2&2&2\\0&0&5&4\\4&2&3&3\end{bmatrix}$$

Max + Reshape

$$\max\begin{bmatrix}6&4&3&2\\2&2&2&2\\0&0&5&4\\4&2&3&3\end{bmatrix} = \begin{bmatrix}6\\2\\5\\4\end{bmatrix} \Rightarrow \begin{bmatrix}6&2\\5&4\end{bmatrix}$$

Graph-CNN – Embed Pooling



$$\begin{array}{c}V^l\\ \begin{bmatrix}0.2\\1.3\\0.7\\1.8\end{bmatrix}\end{array} \circ \begin{array}{c}M^* = softmax(M)\\ \begin{bmatrix}0.57\\0.21\\0.01\\0.21\end{bmatrix}\end{array} = 0.77$$

$$\begin{array}{c}M^*\\ \begin{bmatrix}0.6&0.2&0.0&0.2\end{bmatrix}\end{array} \times \begin{array}{c}V^l\\ \begin{bmatrix}0.2\\1.3\\0.7\\1.8\end{bmatrix}\end{array} = V$$

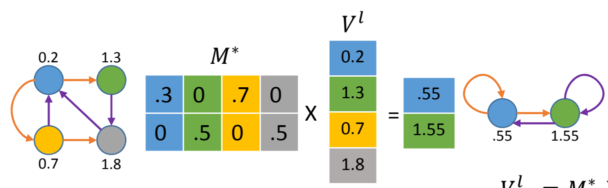
- Instead of generating a single vertex, generate many vertices.
- The number of output vertices is controlled, and can simplify the model.

We now have three *M** row vectors

$$\begin{bmatrix} & & & \\ & & & \\ & & & \end{bmatrix} \times \begin{array}{c}V^l\\ \begin{bmatrix}0.2\\1.3\\0.7\\1.8\end{bmatrix}\end{array} = \begin{bmatrix}V1\\V2\\V3\end{bmatrix}$$

If FC layer has ten connections, learn ten *M** row vectors!

Graph-CNN – Embed Pooling



$$V^l_{i,d} = M^*_{i,j} V^{l-1}_{j,d}$$

$$A^l_{i,a,j} = M^*_{i,x} A^{l-1}_{x,a,y} M^*_{j,y}$$

- Many Graph Representation Vectors combined.
- Output Adjacency matrix calculated accordingly.
- Fixed number of output vertices.
- Independent of number of input vertices.
- Results in fully connected graph, including self connections.

Results - Cora

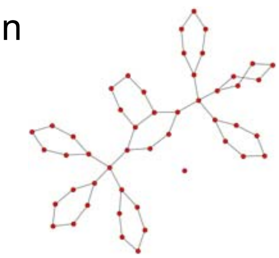
- Document Classification.
  - One large graph (2,708 vertices with 1,433 features).
  - Each vertex has 1 class out of 7.
  - Vertices in the test set are removed during training.

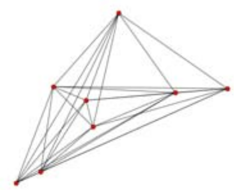
Method	Split	Accuracy
Yang [36]	1000 test	75.7
Kipf [37]	1000 test	81.5
Monti [38]	1000 test	81.69
DCNN [15]	3-fold	86.77
Ours	1000 test	**86.56** $\pm$ 0.68
Ours	3 fold	**87.55** $\pm$ 1.38
Ours	10 fold	**89.18** $\pm$ 1.96

F. Petroski Such*, S. Sah*, M. Dominguez, S. Pillai, Chao Zhang, Andrew Michael, N. Cahill, R. Ptucha "Robust Spatial Filtering with Graph Convolutional Neural Networks," special issue IEEE Journal of Selected Topics in Signal Processing, Volume 11, Issue 6, 2017.

Results – NCI1 / D&D

- Chemical Compound Classification
  - Each sample is a different graph
- Varying number of vertices
  - Each sample has their own A, V



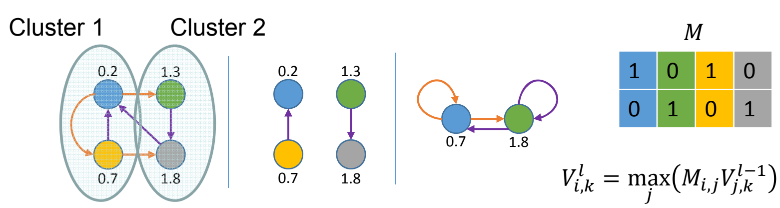


Data set	NCI1	D&D
Maximum graph size	111	5748
Average graph size	29.87	284.32
# Graphs	4110	1178
GK * [34]	62.28 ± 0.29	78.45 ± 0.26
WL * [35]	80.22 ± 0.51	77.95 ± 0.70
PSCN [1]	78.59 ± 1.89	77.12 ± 2.41
Deep GK [32]	80.31 ± 0.46	–
3×16F-3x32F-GFC32	83.69 ± 1.40	–
6×32F-GFC32	83.57 ± 1.99	–
2×64F-Pool32-FC256	84.08 ± 1.45	–
2×64F-Pool32-32F-Pool8-FC256	**84.45** ± 0.94	81.45 ± 2.87
2×64F-Pool32-32F-Pool8-64F-FC256	83.48 ± 1.36	–
2×64F-Pool32-64F-Pool8-FC256	84.35 ± 1.00	**81.88** ± 3.39
5-hop DCNN # [15]	62.61	–
2×64F-Pool32-32F-Pool8-FC256 #	**81.98** ± 0.76	–

- 3-fold experiments

F. Petroski Such*, S. Sah*, M. Dominguez, S. Pillai, Chao Zhang, Andrew Michael, N. Cahill, R. Ptucha "Robust Spatial Filtering with Graph Convolutional Neural Networks," special issue IEEE Journal of Selected Topics in Signal Processing, Volume 11, Issue 6, 2017.

Graph-CNN – Local Max Pooling



$$V_{i,k}^{l} = \max_{j}\left(M_{i,j} V_{j,k}^{l-1}\right)$$

$$A_{i,a,j}^{l} = M_{i,x}^{*} A_{x,a,y}^{l-1} M_{j,y}^{*}$$

- Graph embed pooling replaces each output vertex as a linear combination of all input vertices.
- What if the graph had millions of vertices?
- Enforce graph clusters where output vertices are linear combination of local neighborhood.
- Cluster generated any greedy or exhaustive method.
- Output Adjacency matrix calculated accordingly.
- Variable number of input and output vertices.

Thank you!!

Ray Ptucha
rwpeec@rit.edu

https://www.rit.edu/mil