Directing neural networks to learn graph abstractions

Karthik Gopalakrishnan

2018 Summer Internship

Directing neural networks to learn graph abstractions

1. Classical methods for image classification: limitations and our motivation

2. Directing the neural network to learn graph abstractions

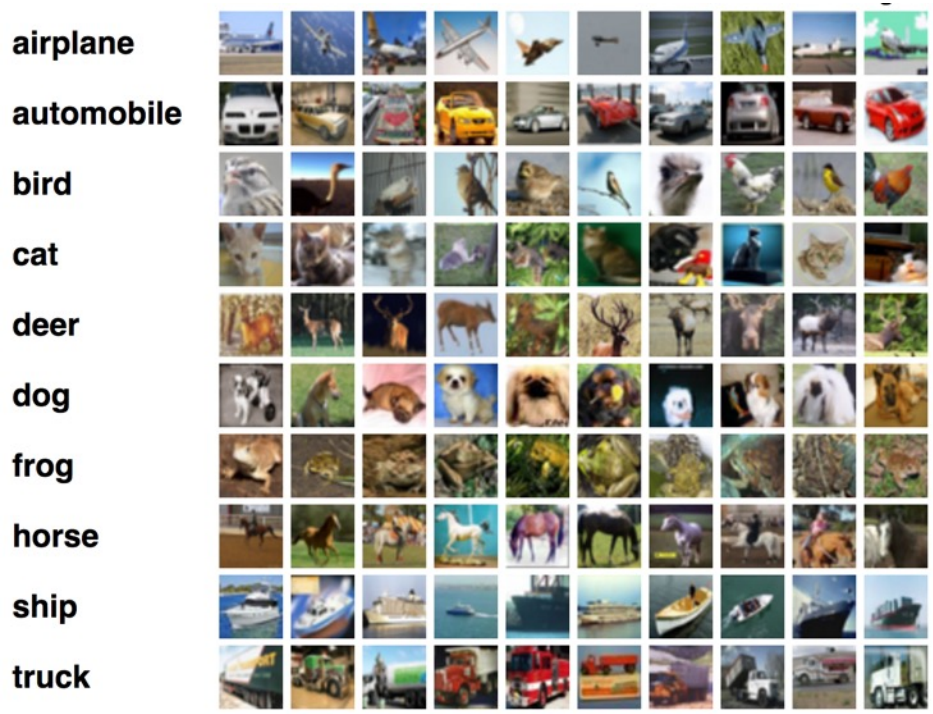
3. Two sub-problems when dealing with graphs

4. Preliminary results

Image Classification

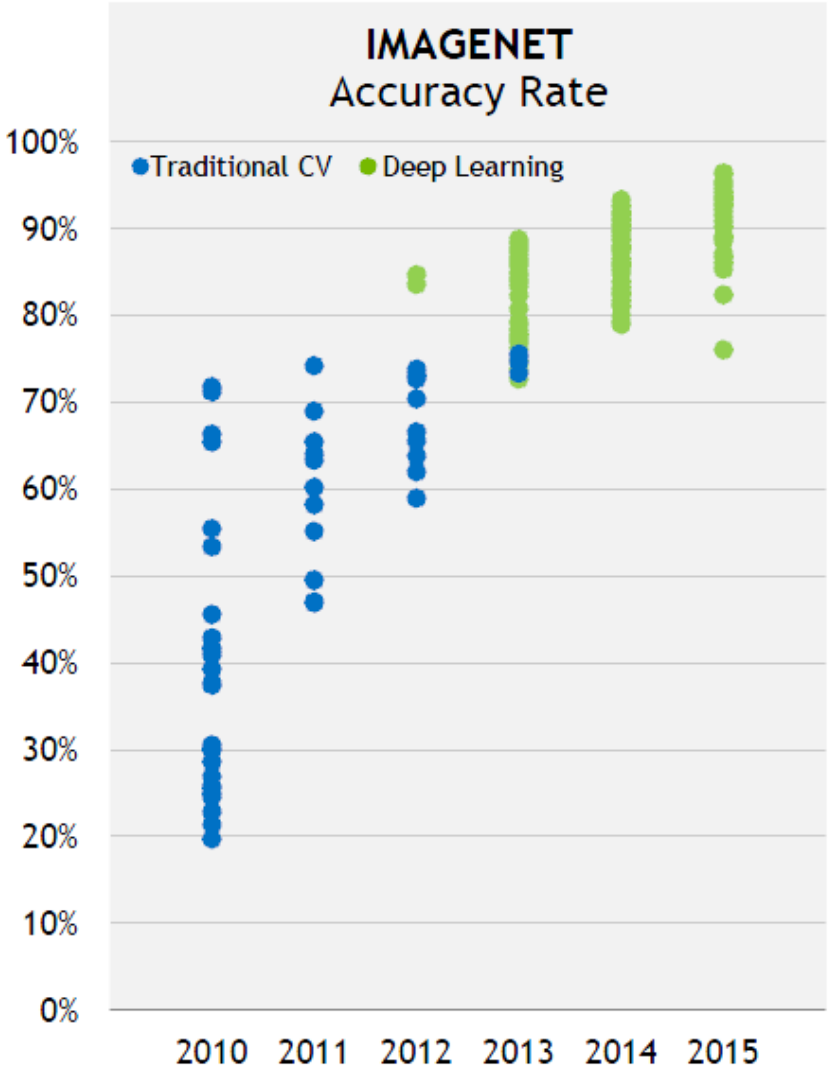
Training images

Test case

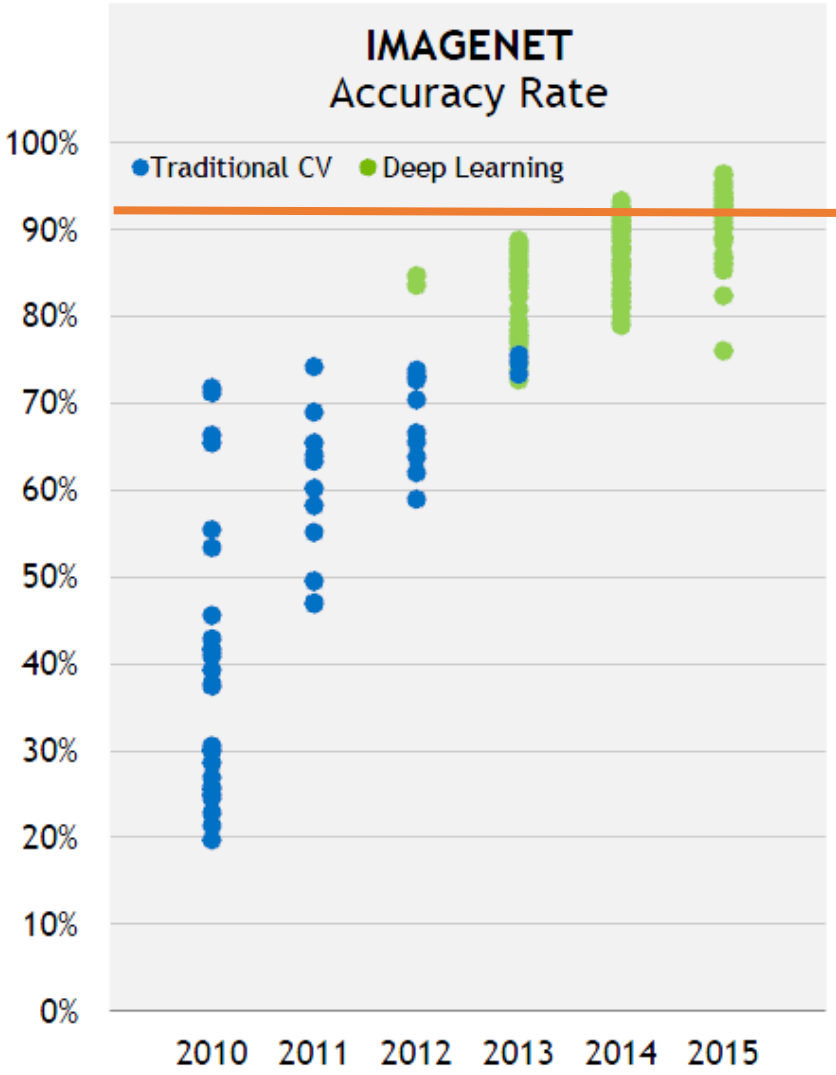


Airplane : 3 %
Automobile : 10 %
…
Truck : 85 %

Success = higher accuracy



Success = higher accuracy



Human performance = 92.9%

Today:
Error of ~2 % and ImageNet problem is considered solved

Convolutional Neural Networks

- The key operation: Convolution

Input Image

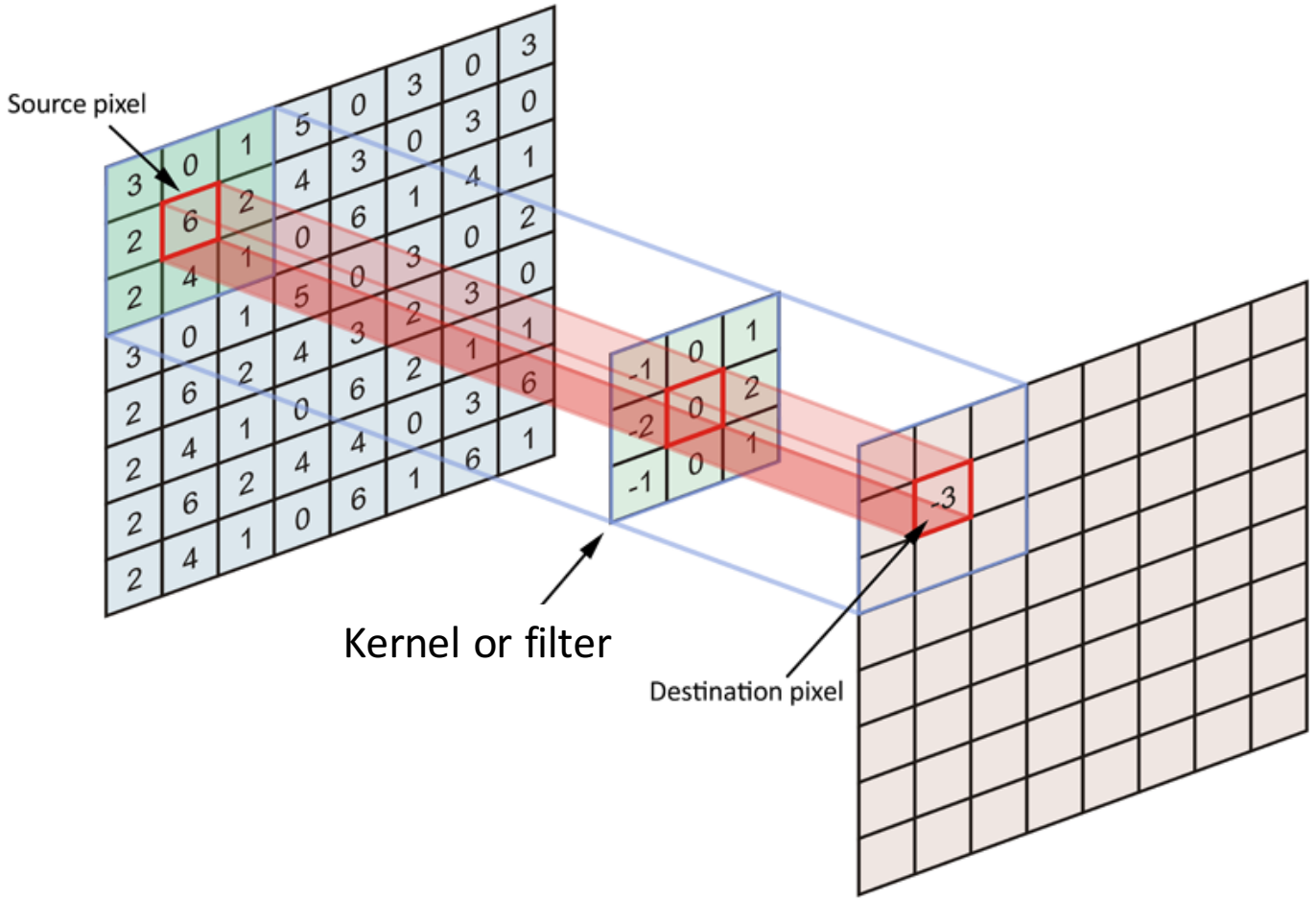
Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

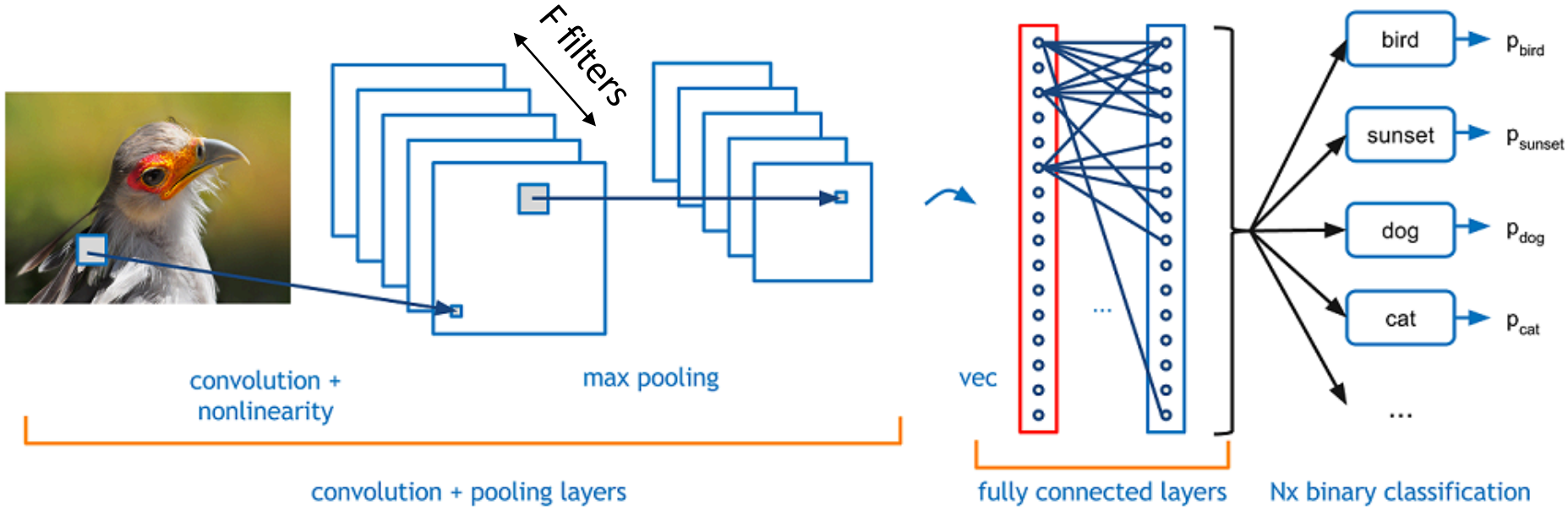
Output Image

Convolutional Neural Networks

- The numbers in the kernel are `learnt'



CNN architecture



So what does a CNN learn?

- A bunch of kernel weights

Input Image

Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Output Image

CNN learns several such kernel matrices

A simplified picture

Image

Convolution Neural Net

Learnt parameters:

- *kernel 1*
- *kernel 2*
- *…*
- *…*
- *Fully connected layer weights*

Classification label

Airplane : 3 %
Automobile : 10 %
…
Truck : 85 %

Emerging concerns with CNNs

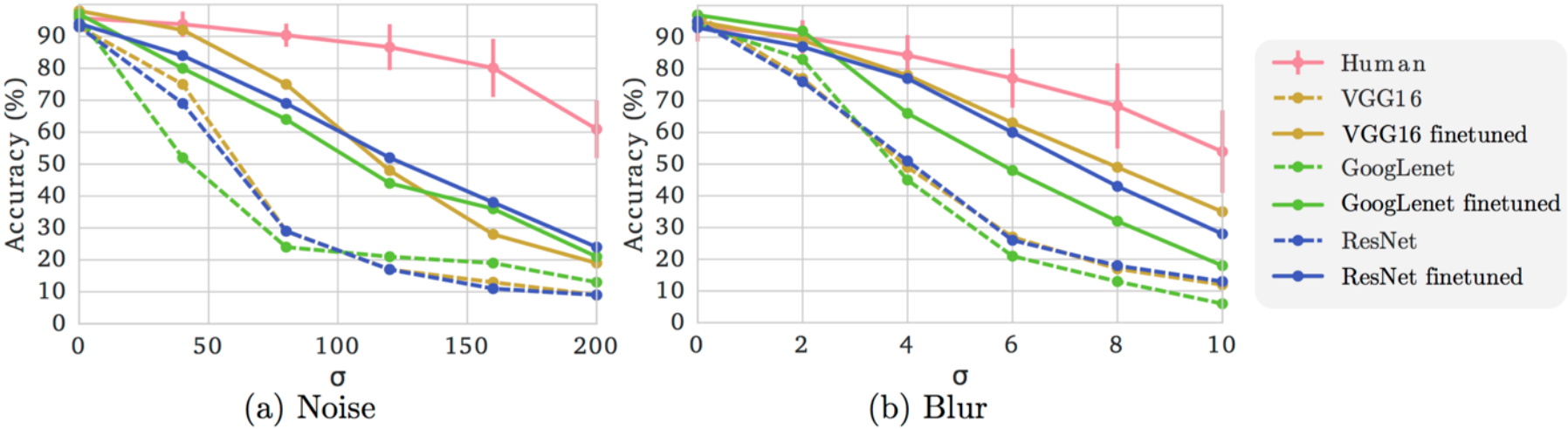
1. Requirement of large training datasets

Emerging concerns with CNNs

1. Requirement of large training datasets
2. Not extremely interpretable

Emerging concerns with CNNs

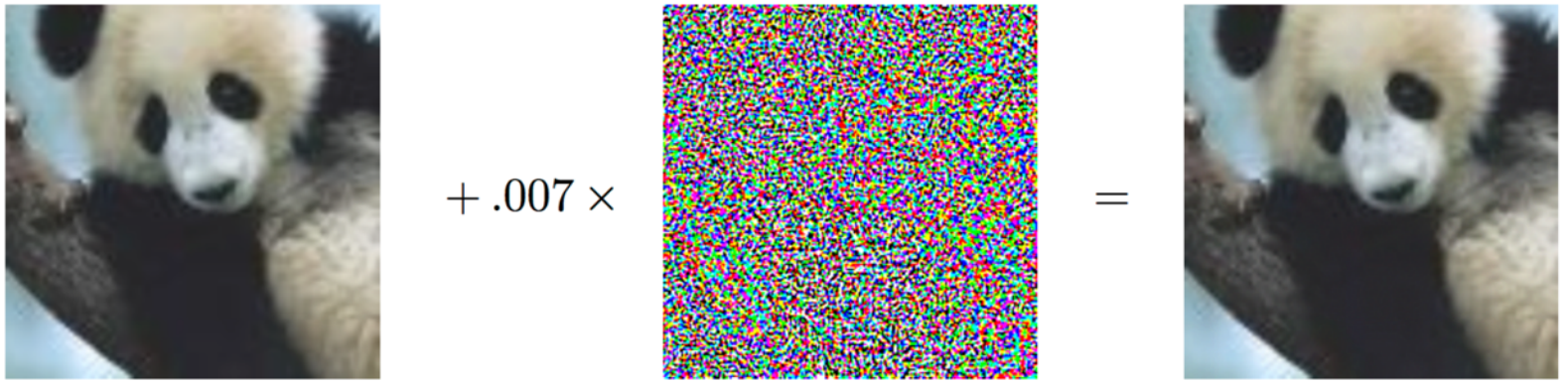
1. Requirement of large training datasets
2. Not extremely interpretable
3. Robustness to perturbations



[Dodge and Karam 2017]

Emerging concerns with CNNs

1. Requirement of large training datasets
2. Not extremely interpretable
3. Robustness to perturbations



[Goodfellow et. al. 2015]

Emerging concerns with CNNs

1. Requirement of large training datasets
2. Not extremely interpretable
3. Robustness to perturbations

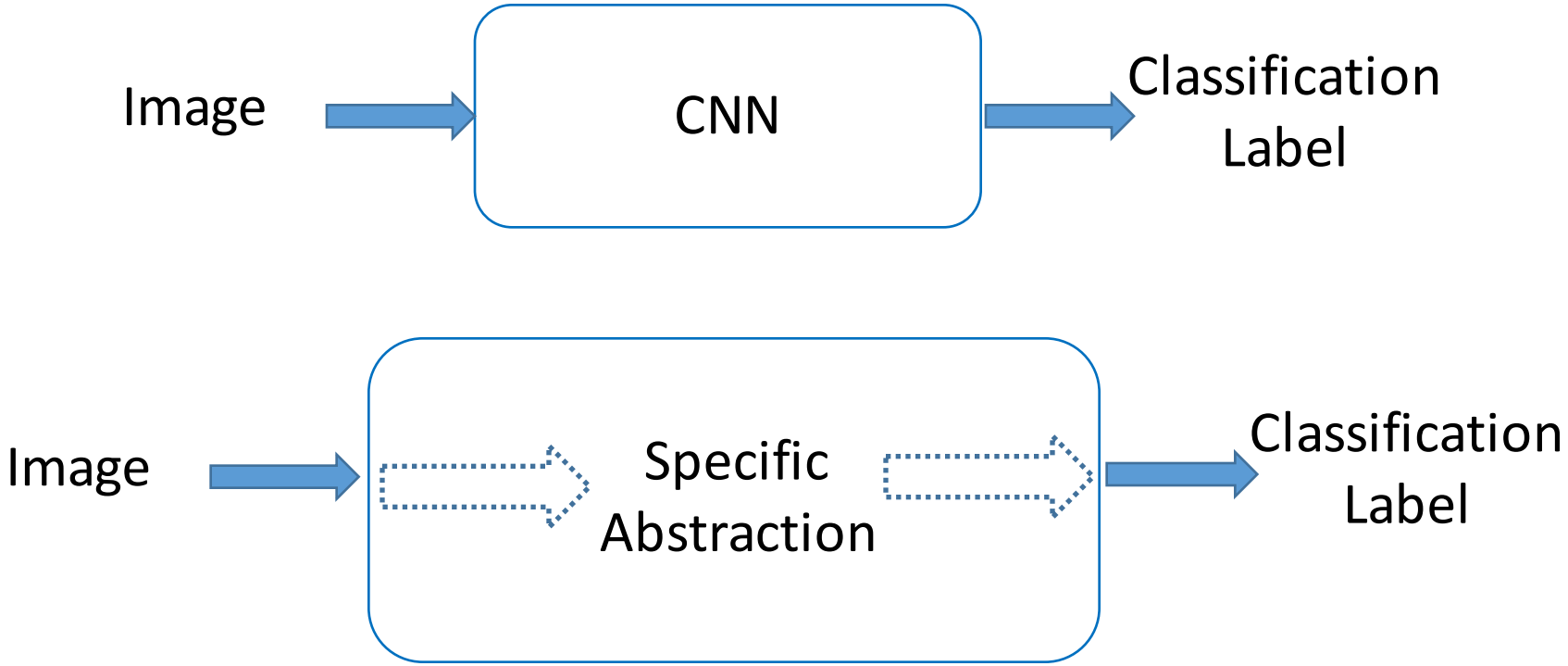
[Eykhlot et. al. 2018]

Key idea 1

Directing the neural network to learn an appropriate abstraction for the problem can help alleviate some of these concerns

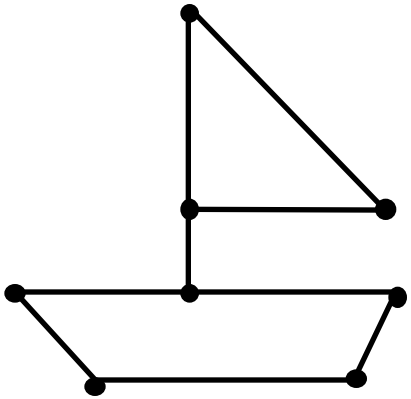
Key idea 1

Directing the neural network to learn an appropriate abstraction for the problem can help alleviate some of these concerns



Key idea 2

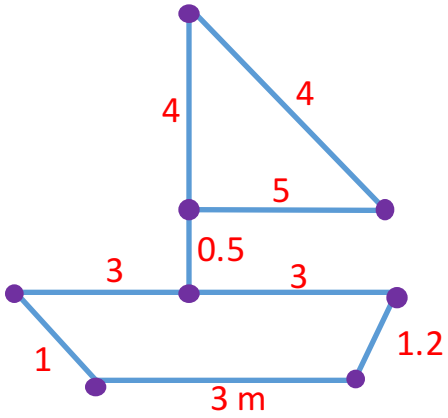
A graph is potentially a good abstraction for many classification problems



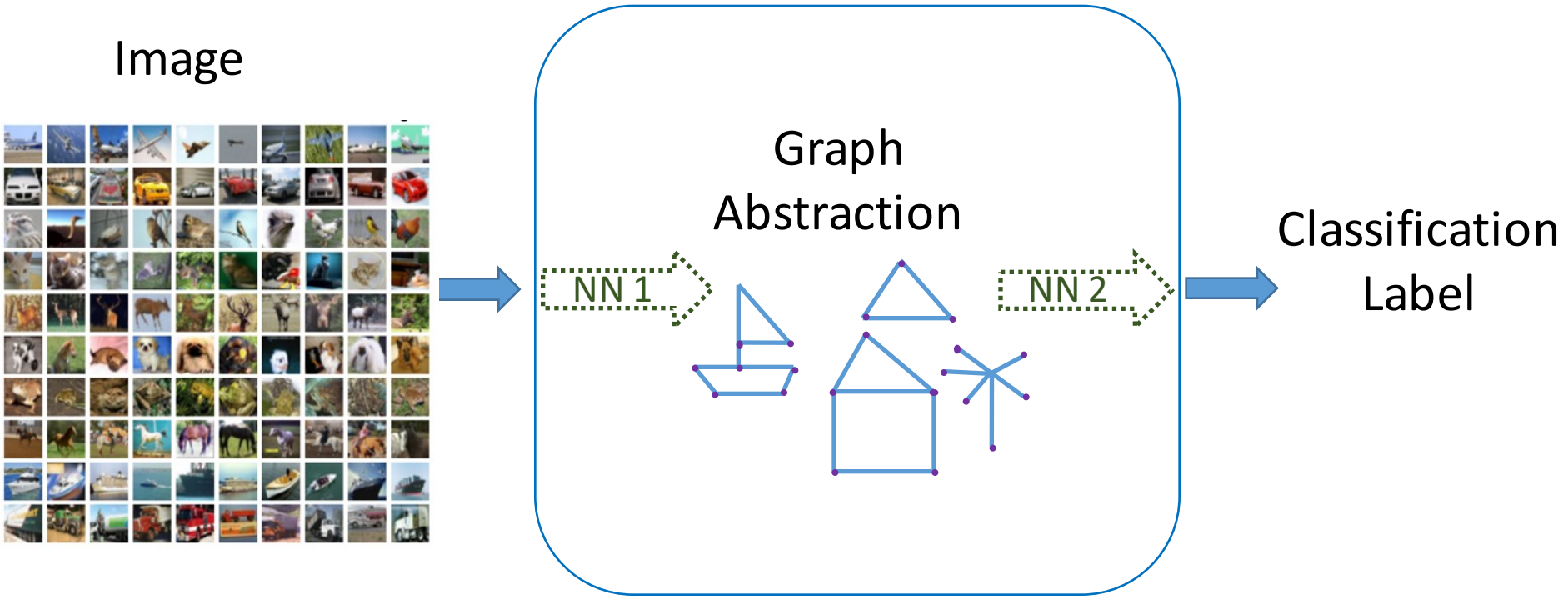
Graph as an abstraction

Can capture

- geometry (relative size) : edge weights
- Topology (shape) : edge connections
- Other features (like color) : nodal values



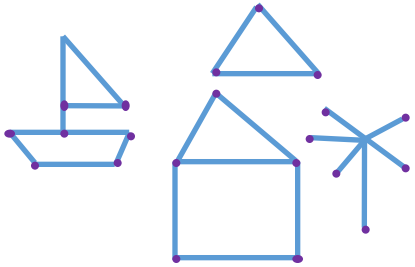
Our proposed architecture



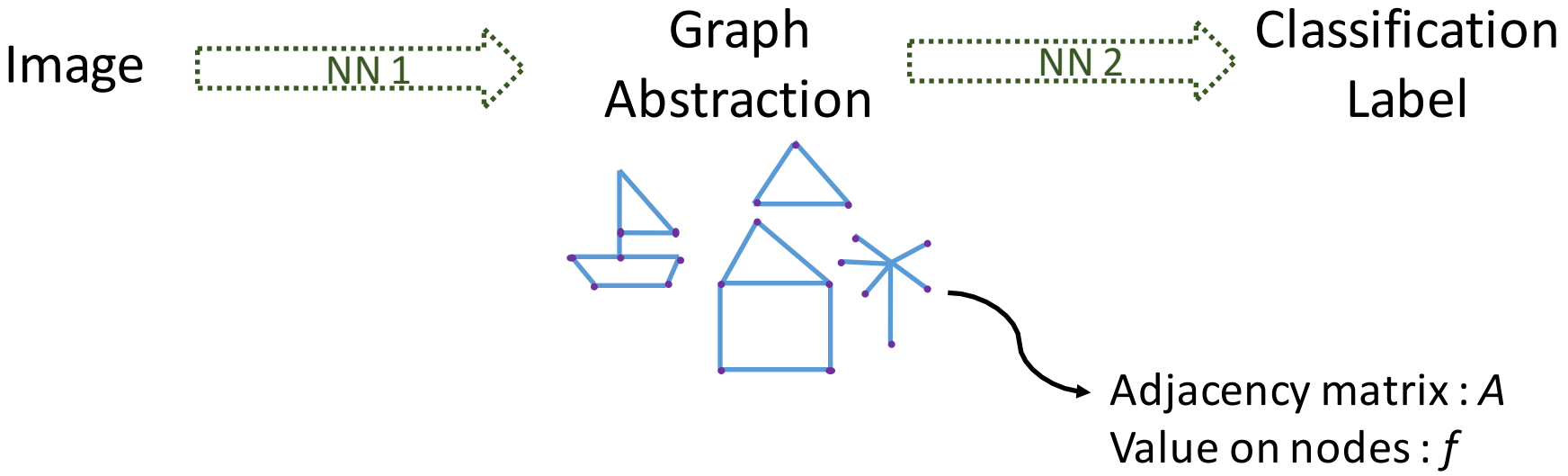
NN1 : Image input, graph output
NN2 : Graph input, label output

The two sub-problems

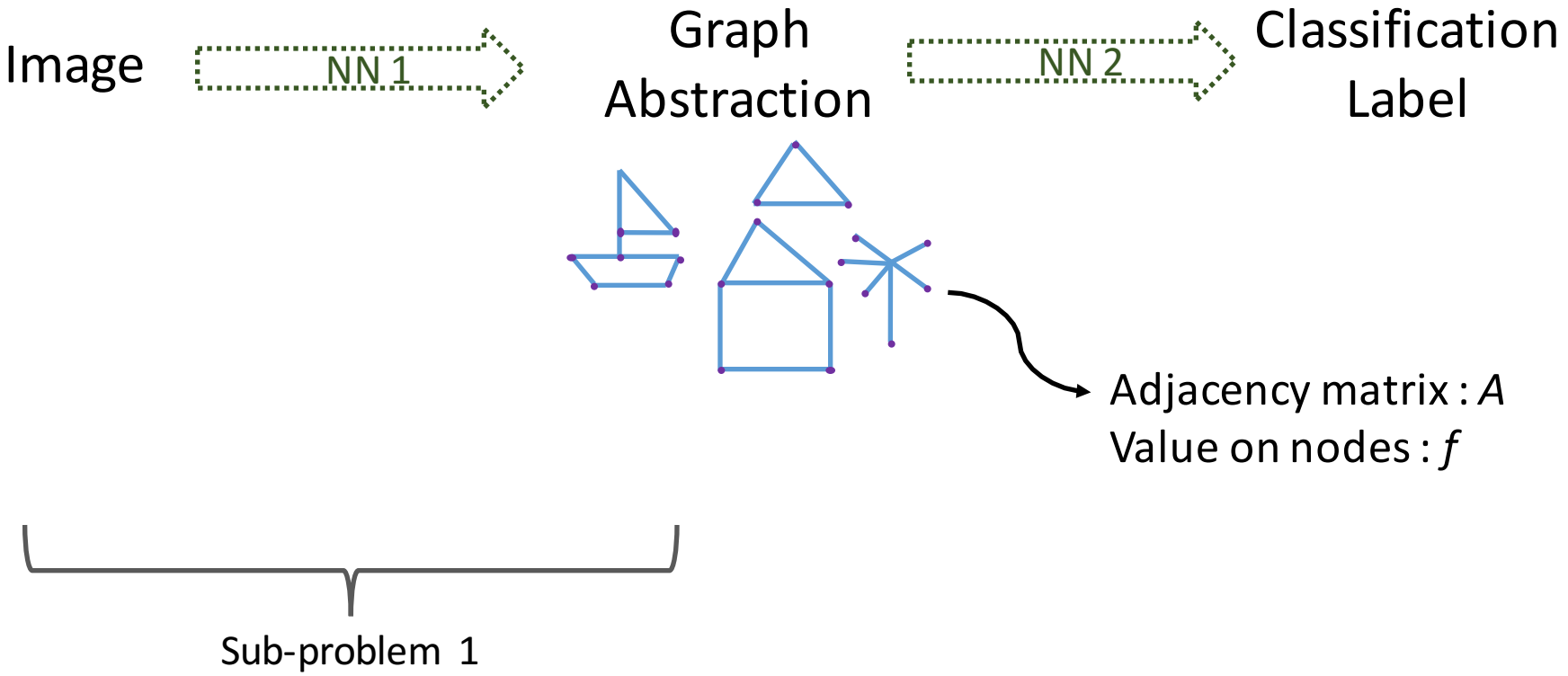
Image → NN 1 → Graph Abstraction → NN 2 → Classification Label



The two sub-problems

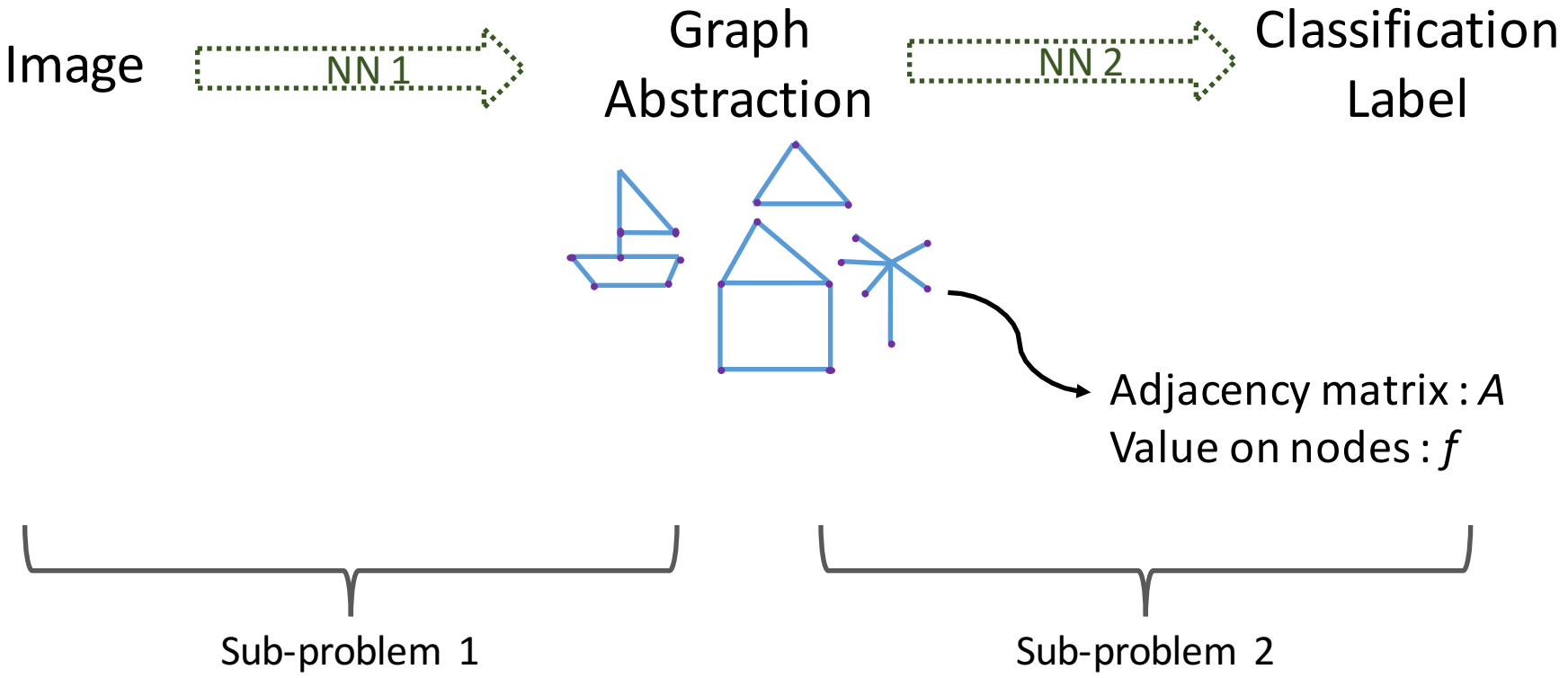


The two sub-problems



- Input : Image, Output : *image specific graph*
- No prior work

The two sub-problems

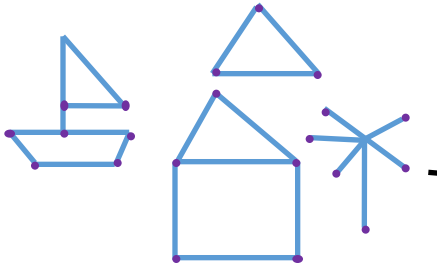


- Input : Image, Output : *image specific graph*
- No prior work

- Input: a *random* graph, Output: label
- Geometric Deep Learning

The two sub-problems

Image → NN 1 → Graph Abstraction → NN 2 → Classification Label



Adjacency matrix : A
Value on nodes : f

Sub-problem 1

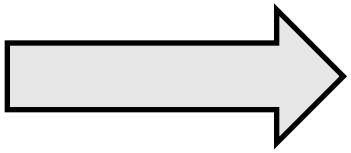
- Input : Image, Output : *image specific graph*
- No prior work

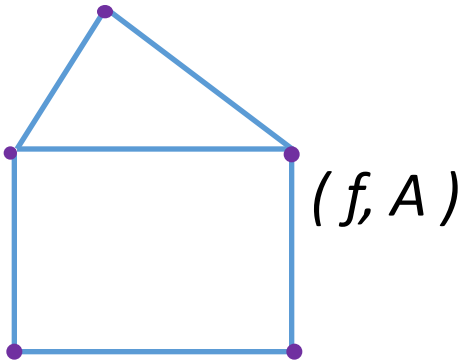
Sub-problem 2

- Input: a *random* graph, Output: label
- Geometric Deep Learning

Need to be compatible for end-to-end learning

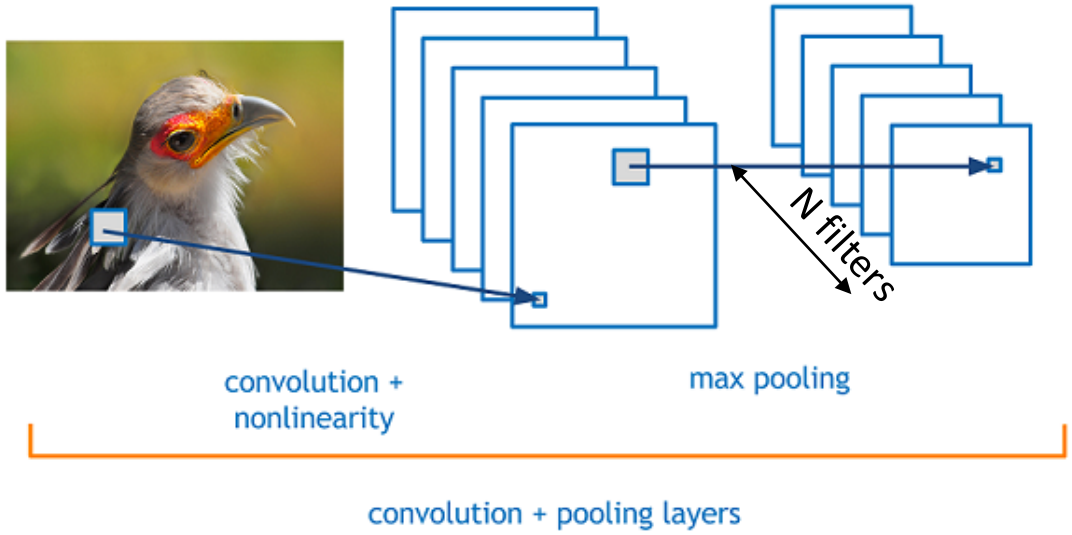
Sub-problem 1:
Image to a Graph





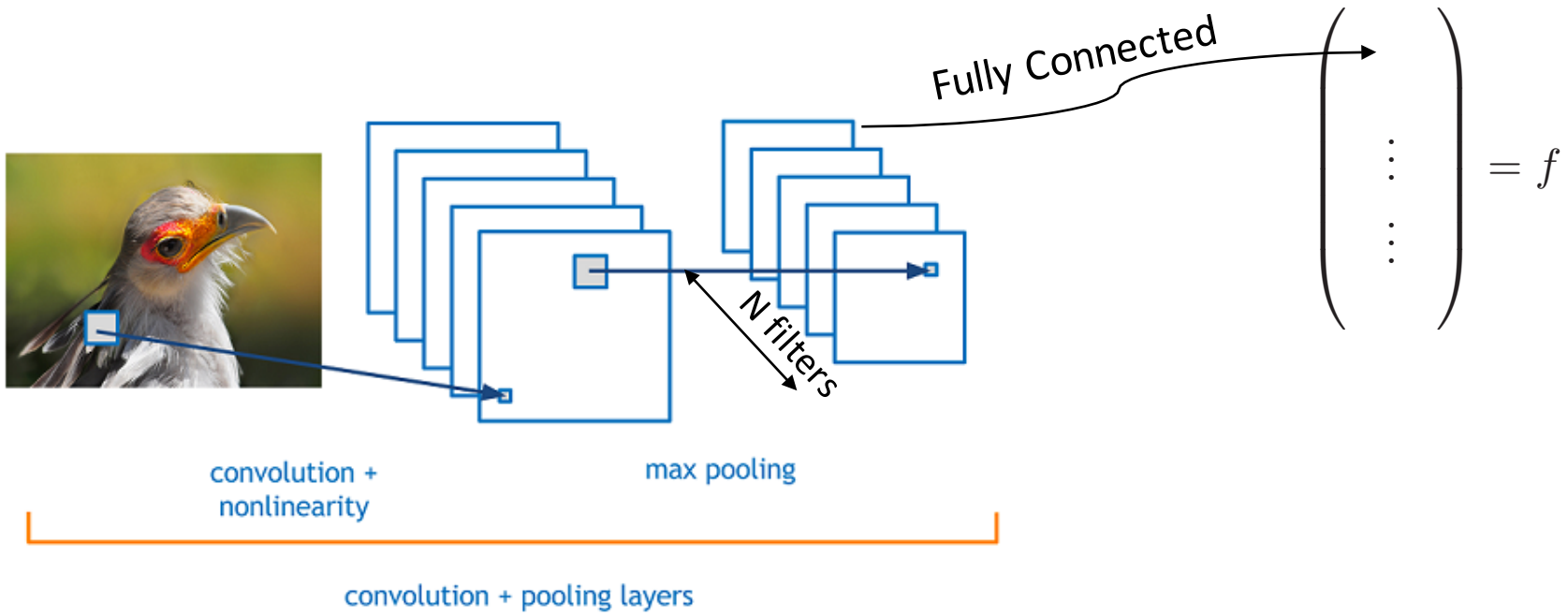
A possible architecture

Objective: Obtain a graph with N nodes



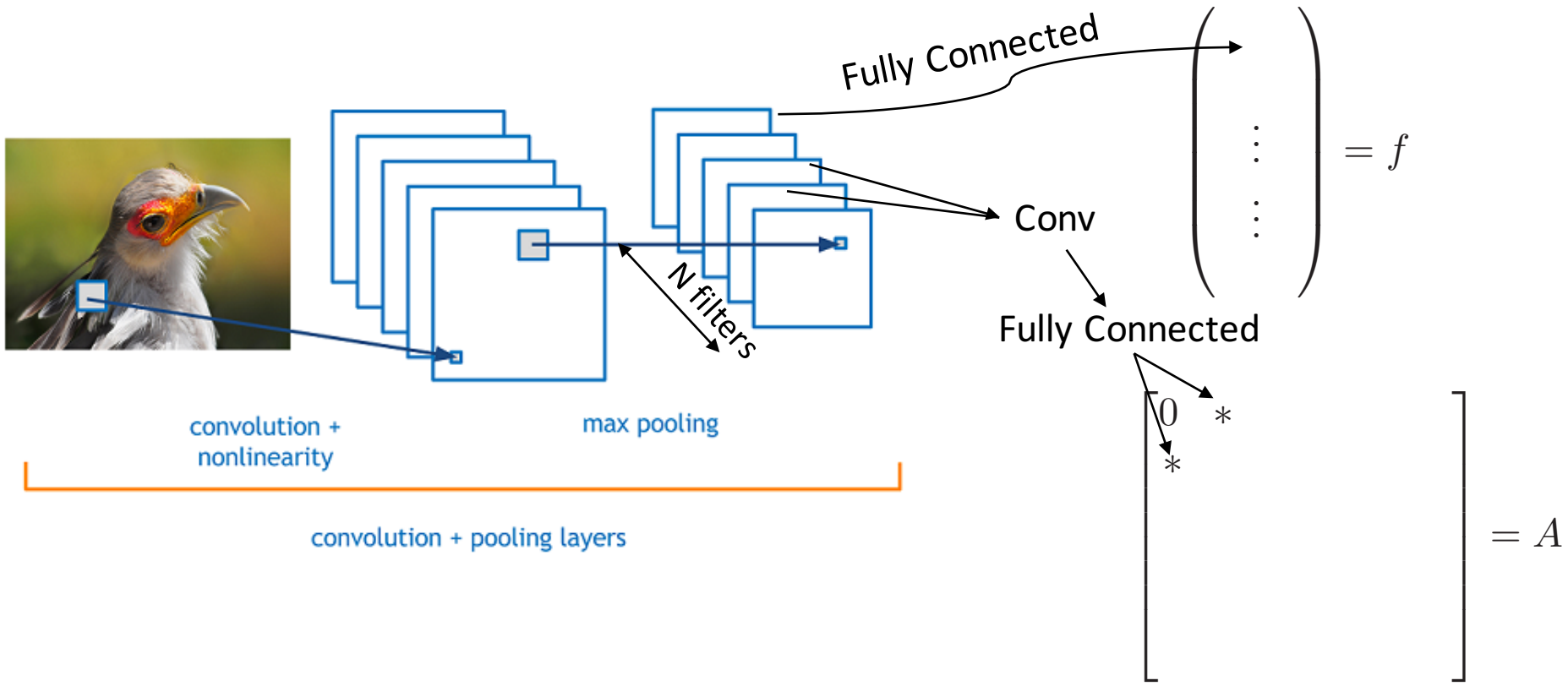
A possible architecture

Objective: Obtain a graph with N nodes

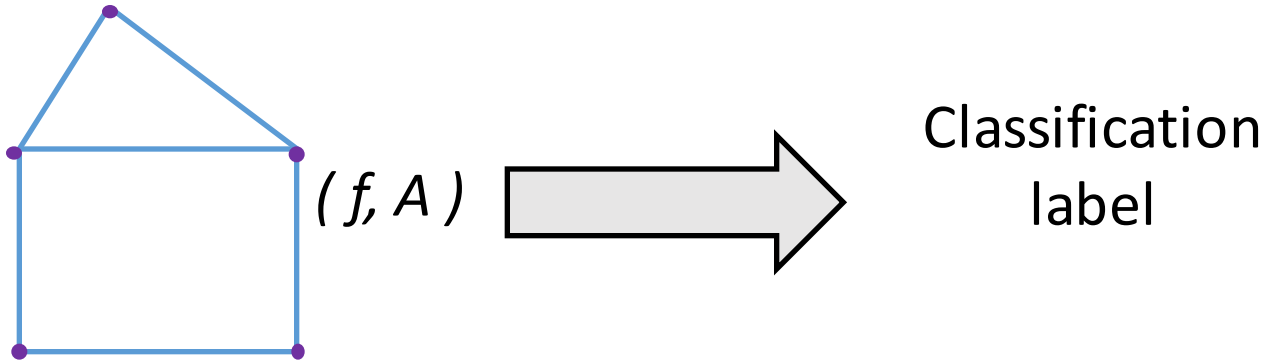


A possible architecture

Objective: Obtain a graph with N nodes

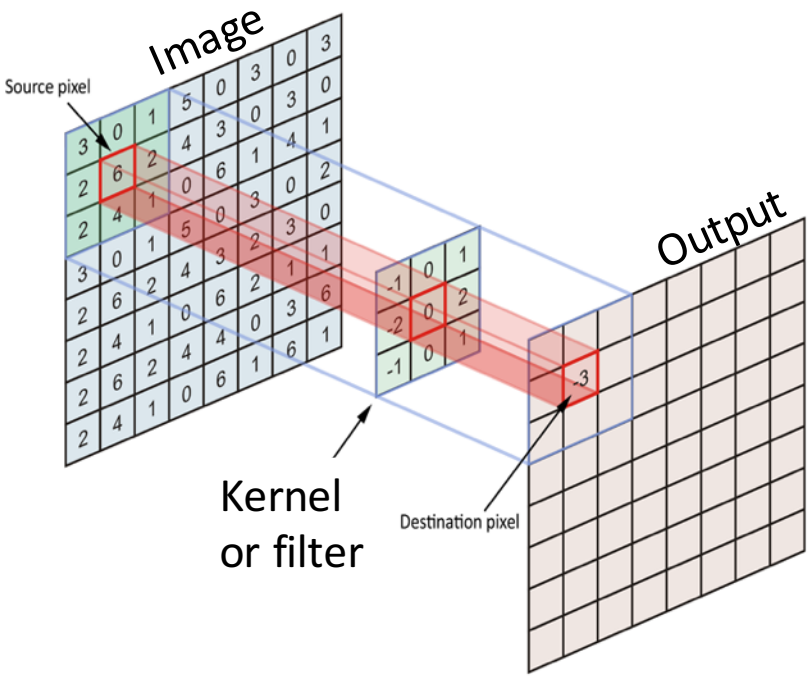


Sub-problem 2:
Graph to a classification label



Extension of CNNs to graphs!

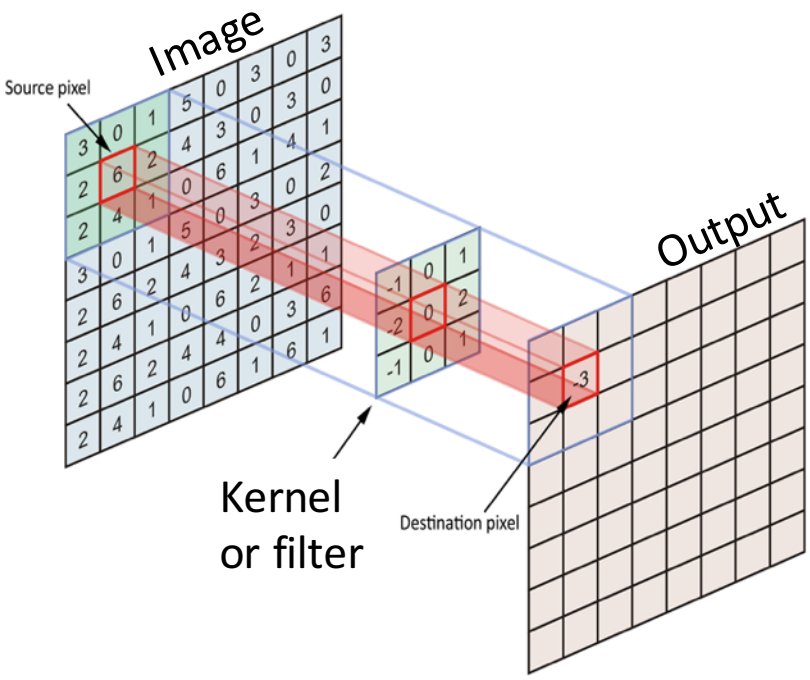
Graph convolution is non-trivial



For Images,

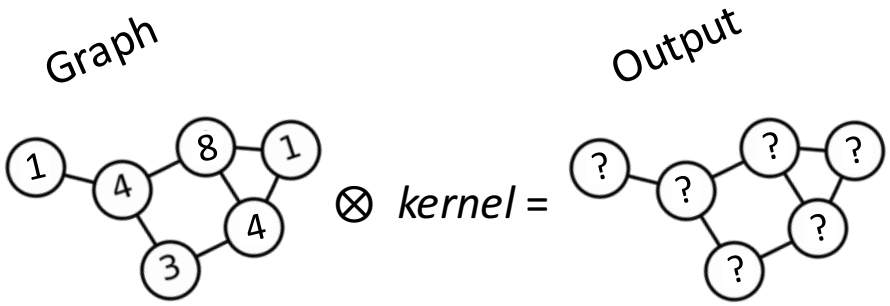
- kernel is fixed
- Can slide it over easily

Graph convolution is non-trivial



For Images,

- kernel is fixed
- Can slide it over easily

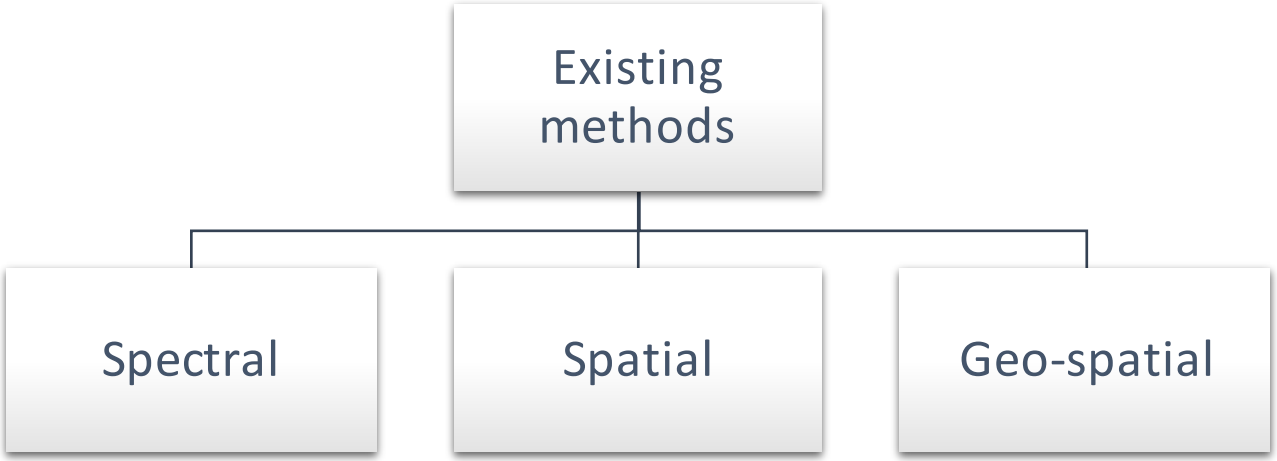


For graphs,

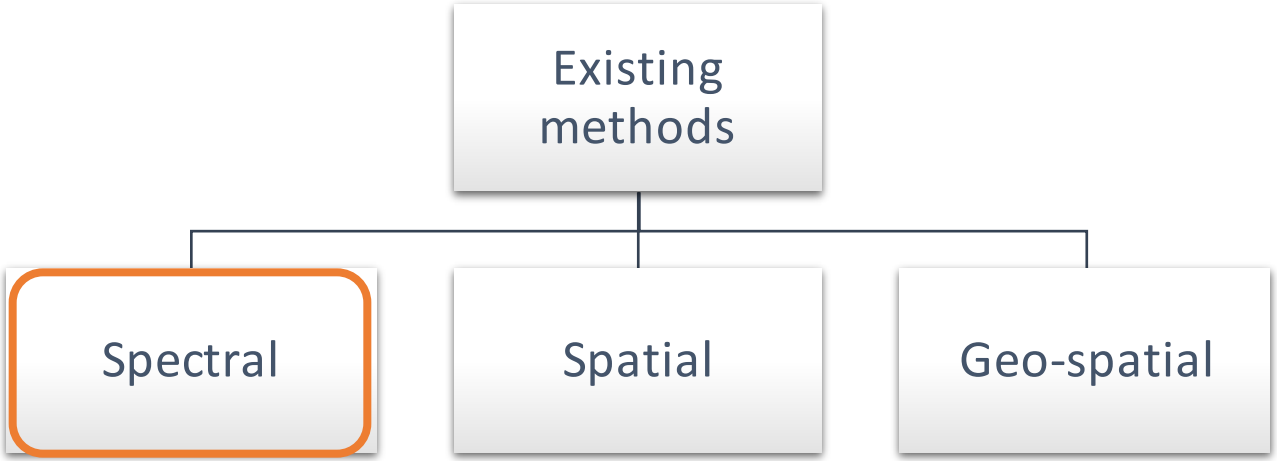
- kernel cant be a fixed template
- Diff num. of edges for each node
- No sense of orientation

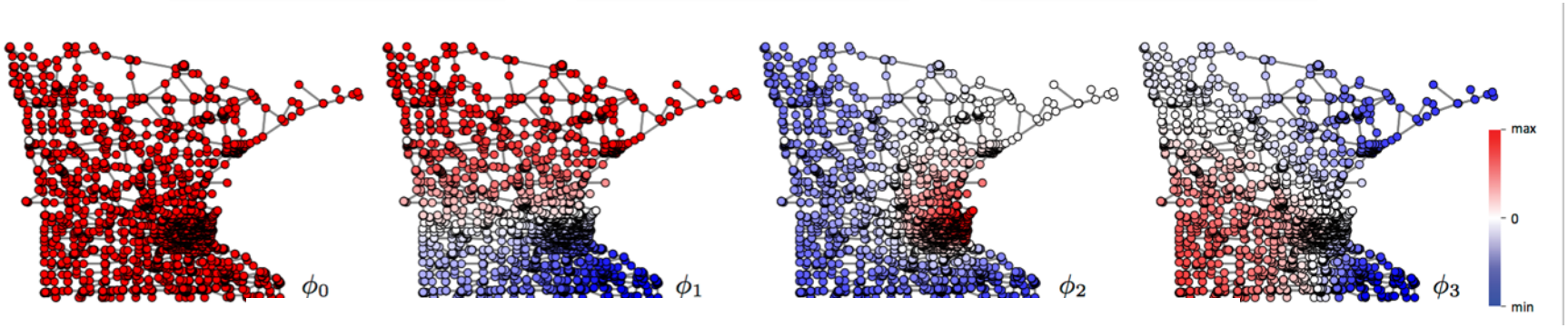
Geometric Deep Learning

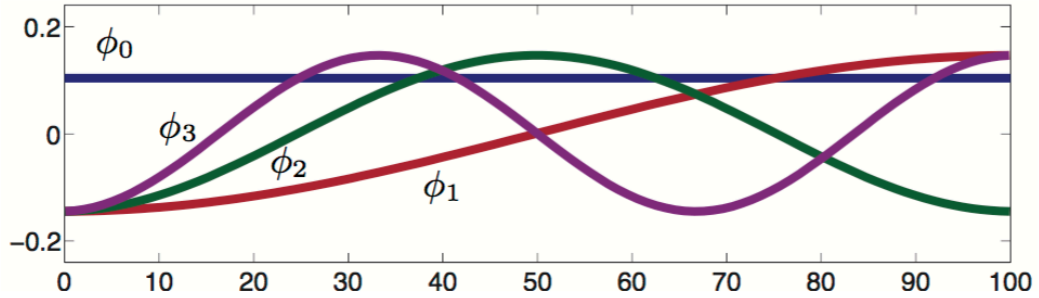
- Maps a graph signal to a classification label
- Different approaches to enable convolution on graphs



Geometric Deep Learning

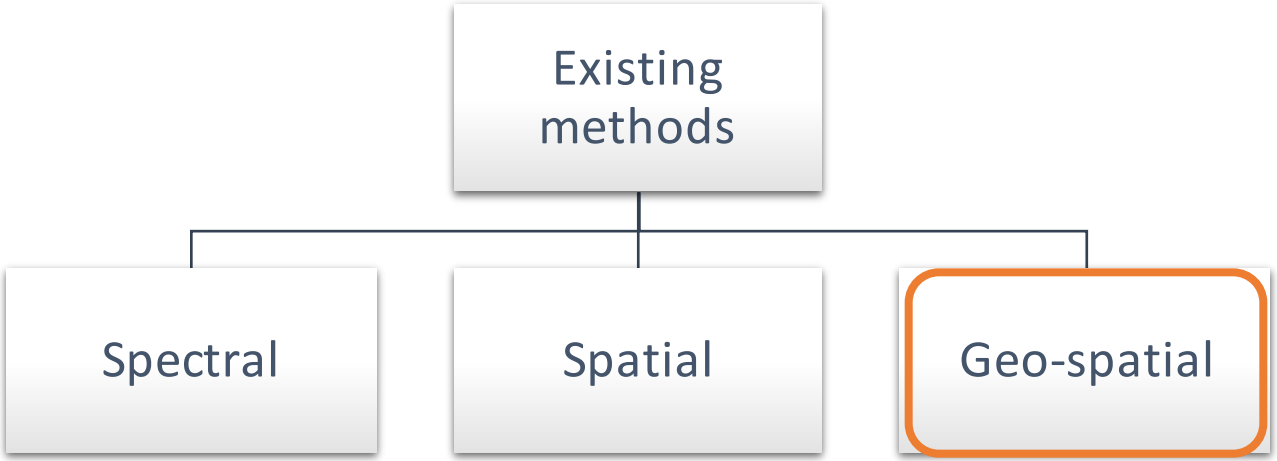


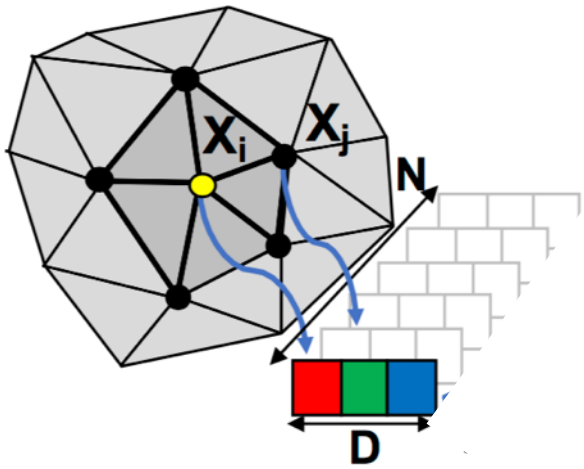




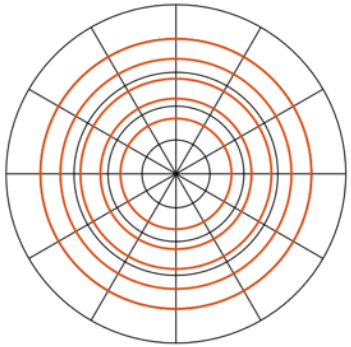
[Bronstein et. al. 2017]

Geometric Deep Learning





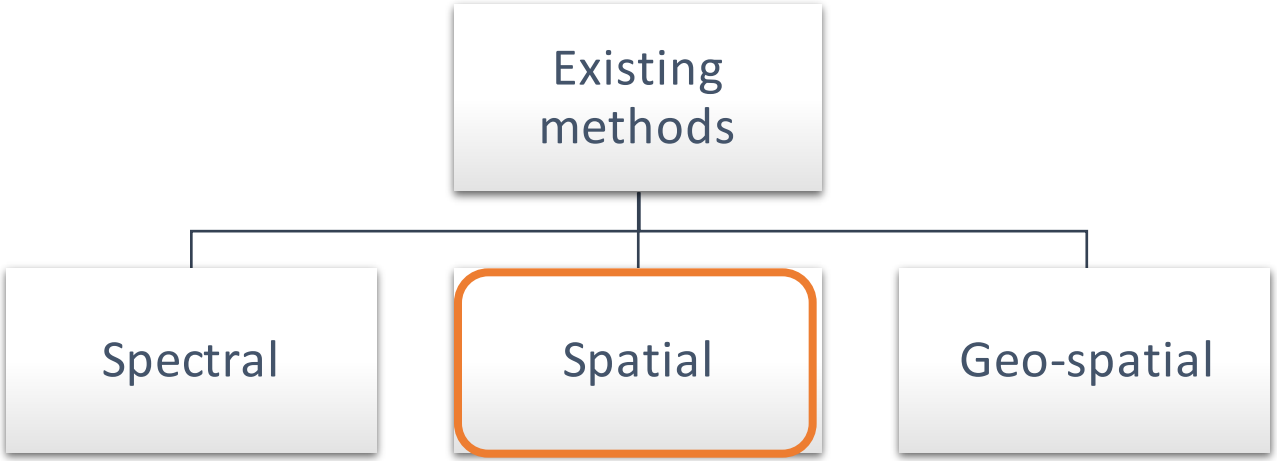
Interpolate function, create a field



Learn and apply a simple kernel at every node

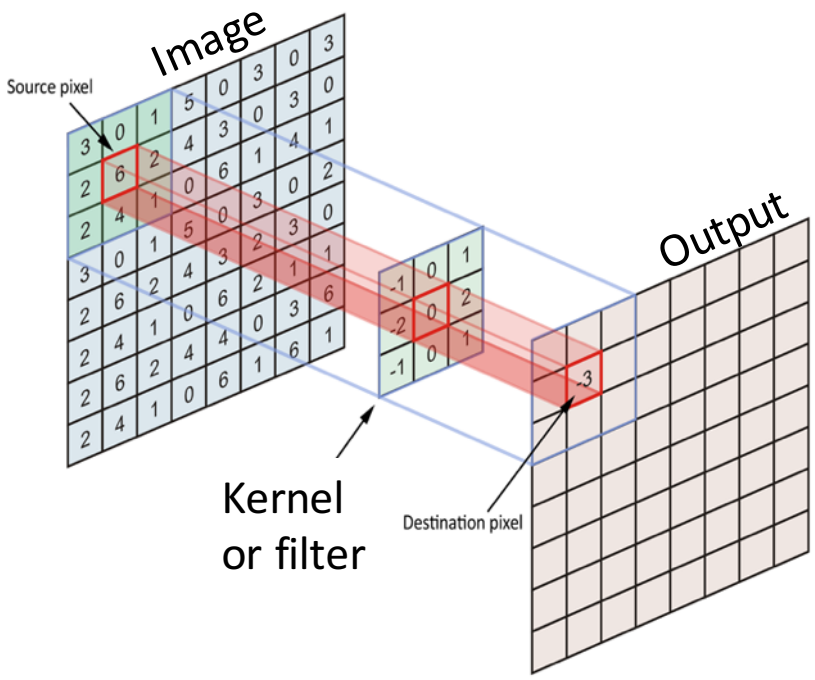
[Verma et. al. 2018, Monti et. al. 2017]

Geometric Deep Learning



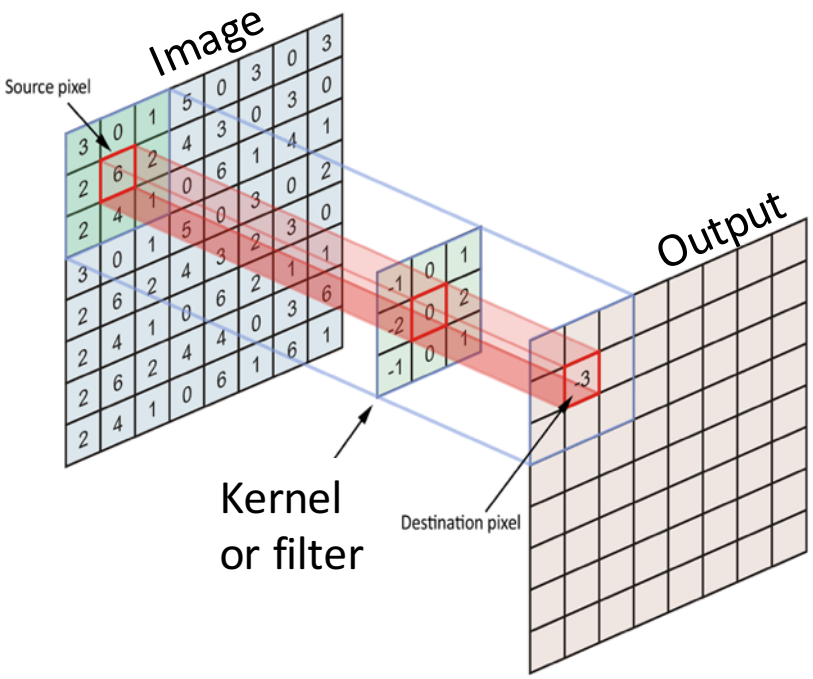
Spatial method: An example

CNN on images

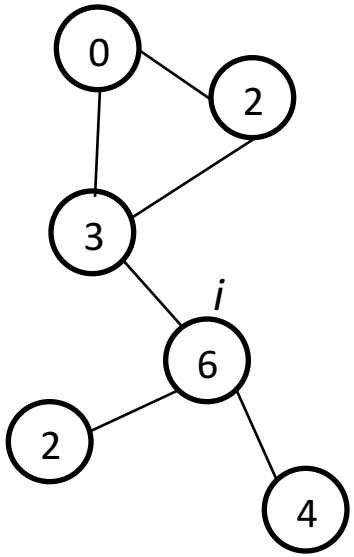


Spatial method: An example

CNN on images

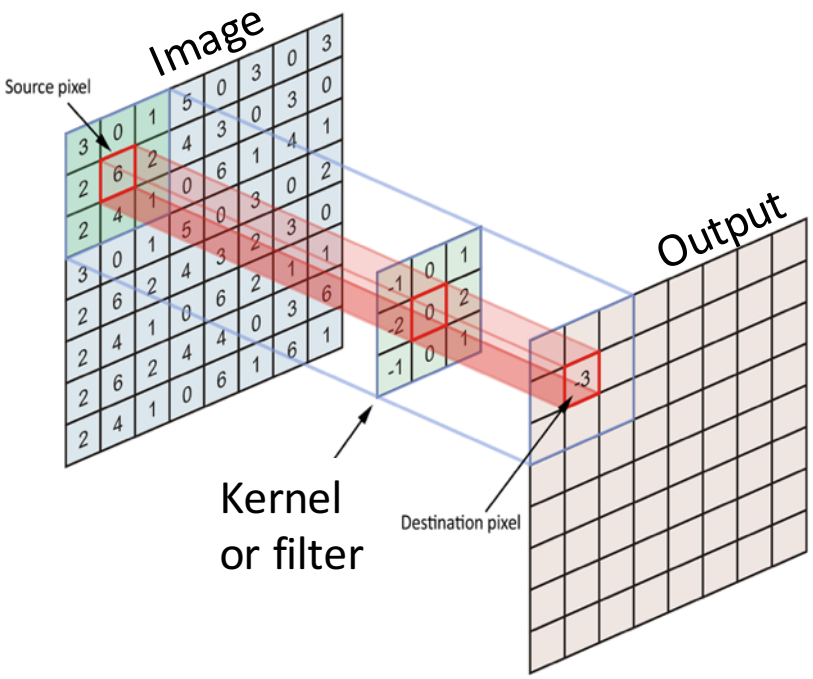


Graph CNN [Such et. al. 2017]

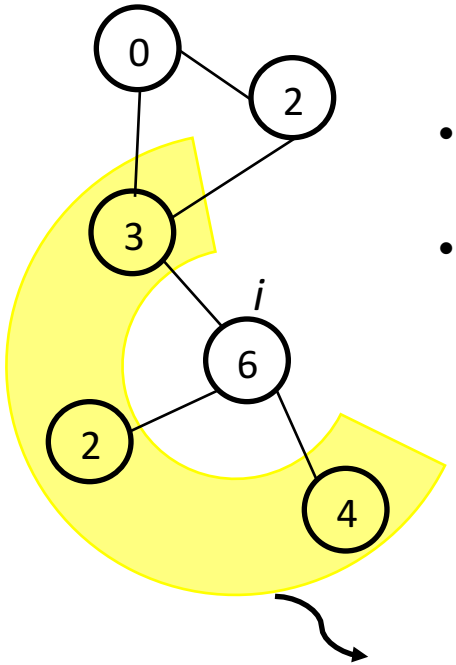


Spatial method: An example

CNN on images

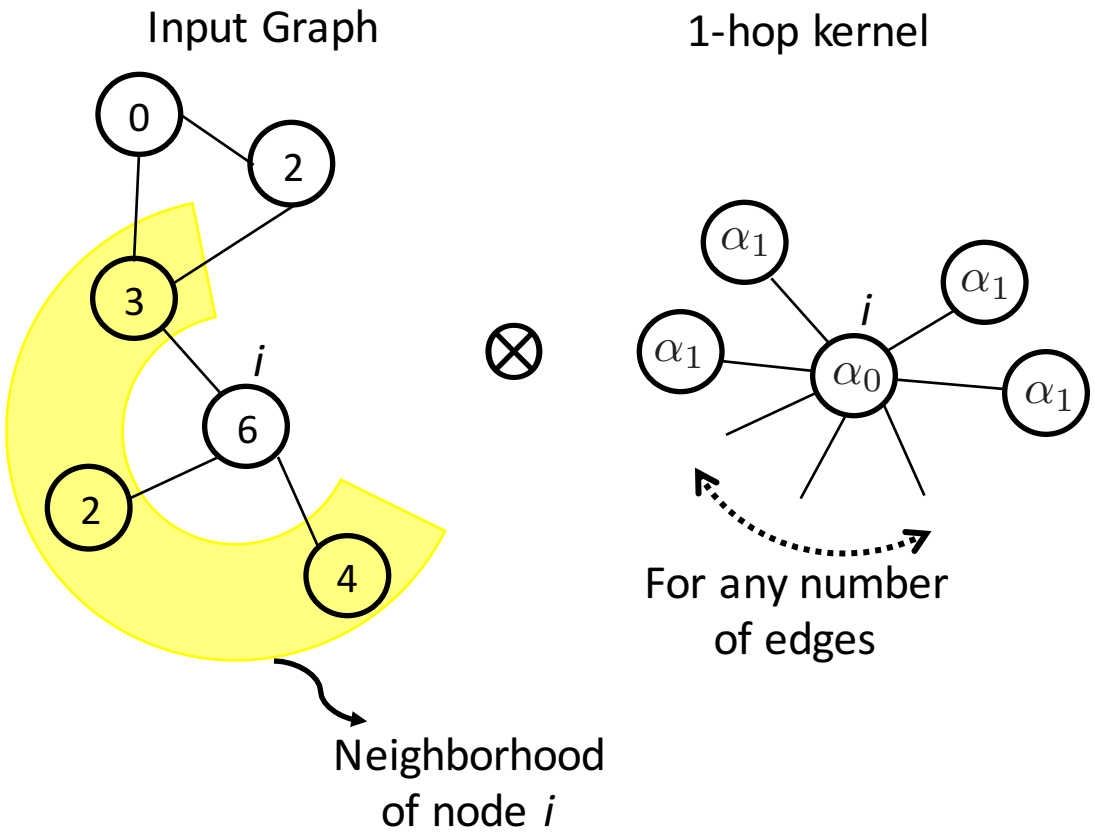


Graph CNN [Such et. al. 2017]

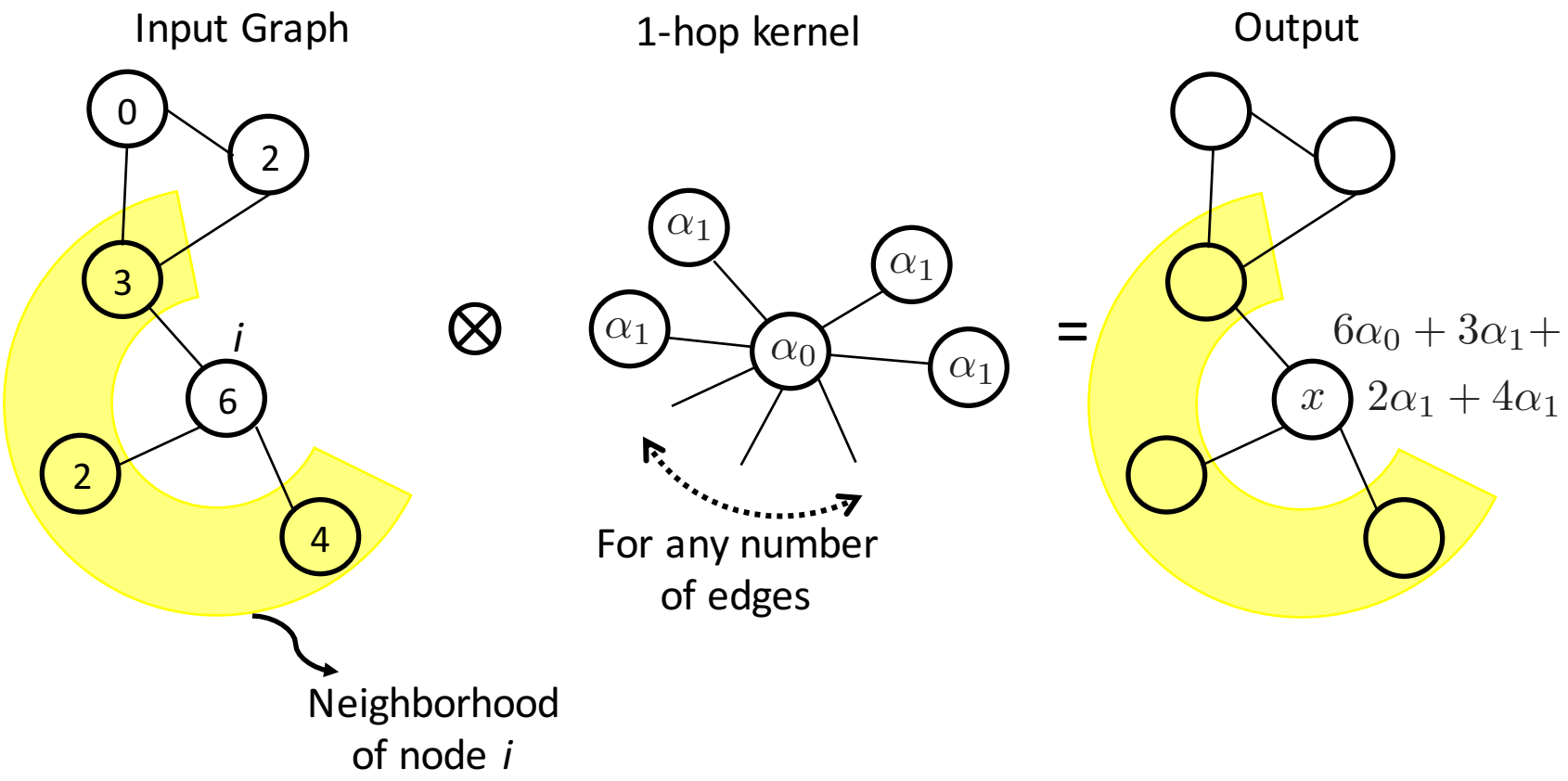


- Want a fixed kernel (template)
- It must 'morph' based on the neighborhood of a node *i*

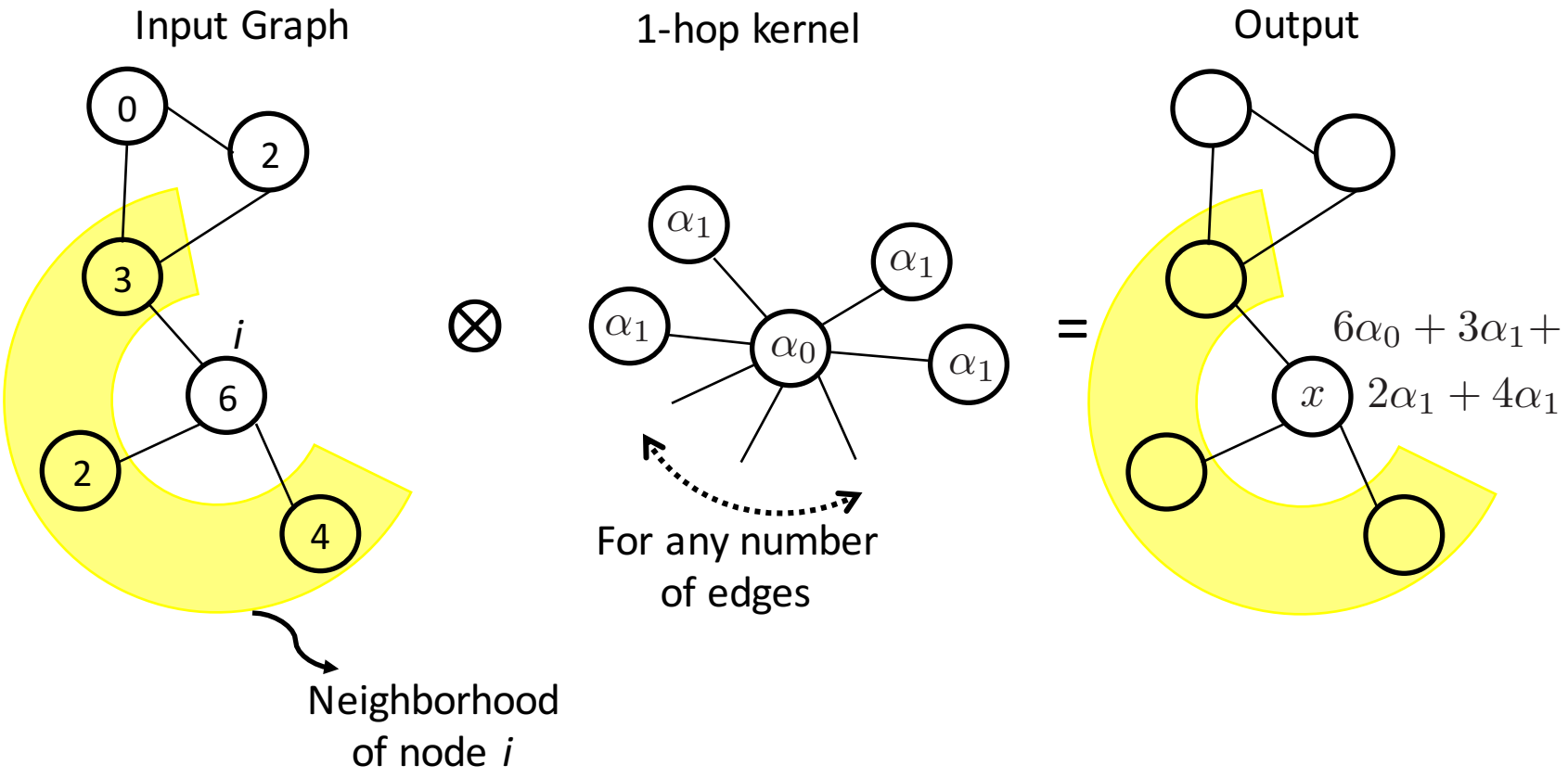
Example: Graph CNN



Example: Graph CNN

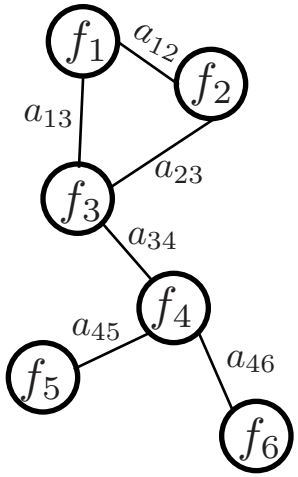


Example: Graph CNN



- Kernel is symmetric, so orientation is not an issue
- Easy way to algebraically deal with variable number of edges

Example: Graph CNN

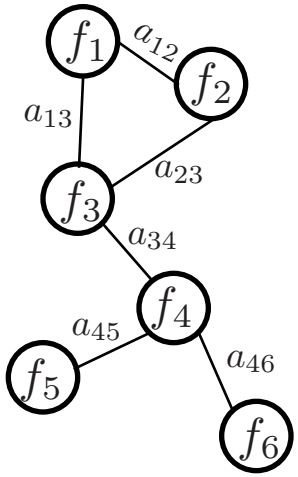


- Signal vector : *f*
- Adjacency matrix : *A*
- The convolution operation :

$$f^{out} = \xi(\alpha_0 I + \alpha_1 A) f$$

- The alphas are learnt

Example: Graph CNN



- Signal vector : *f*
- Adjacency matrix : *A*
- The convolution operation :

$$f^{out} = \xi(\alpha_0 I + \alpha_1 A) f$$

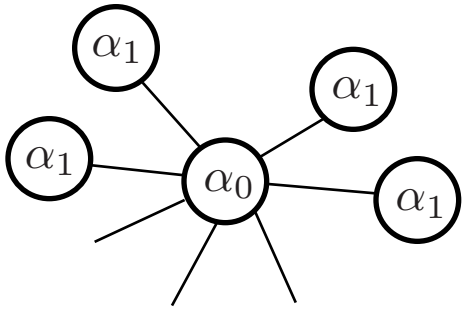
- The alphas are learnt

- In general, this kernel is less expressive than the one for images!

Image kernel

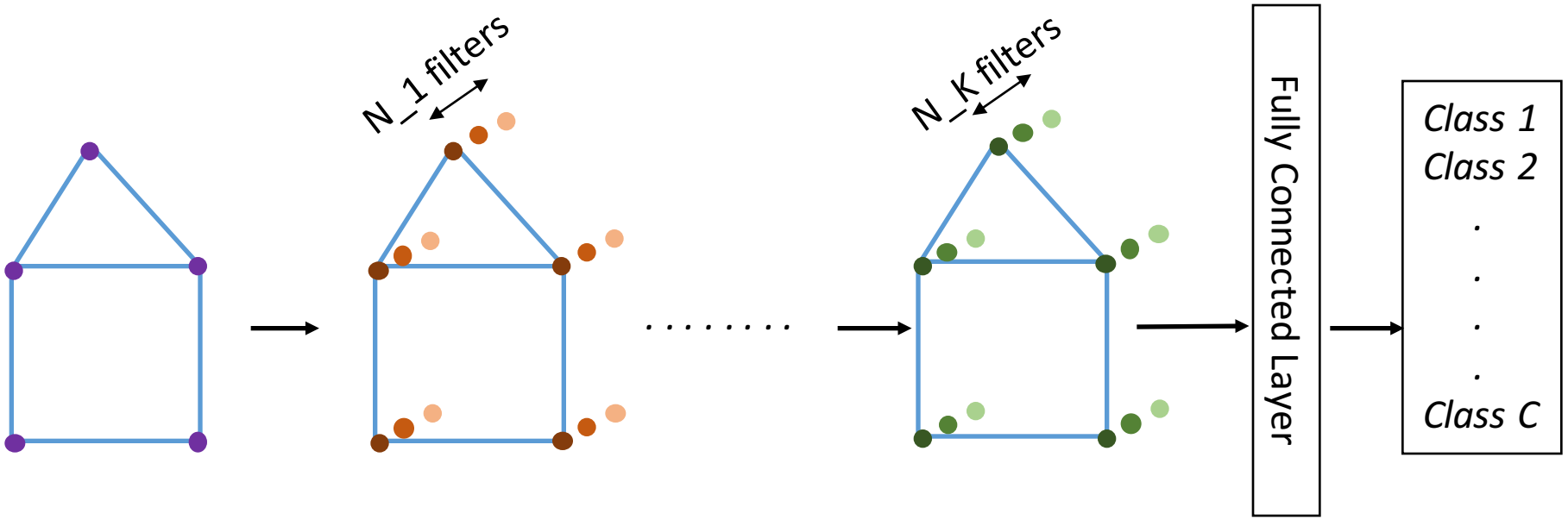
w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

vs

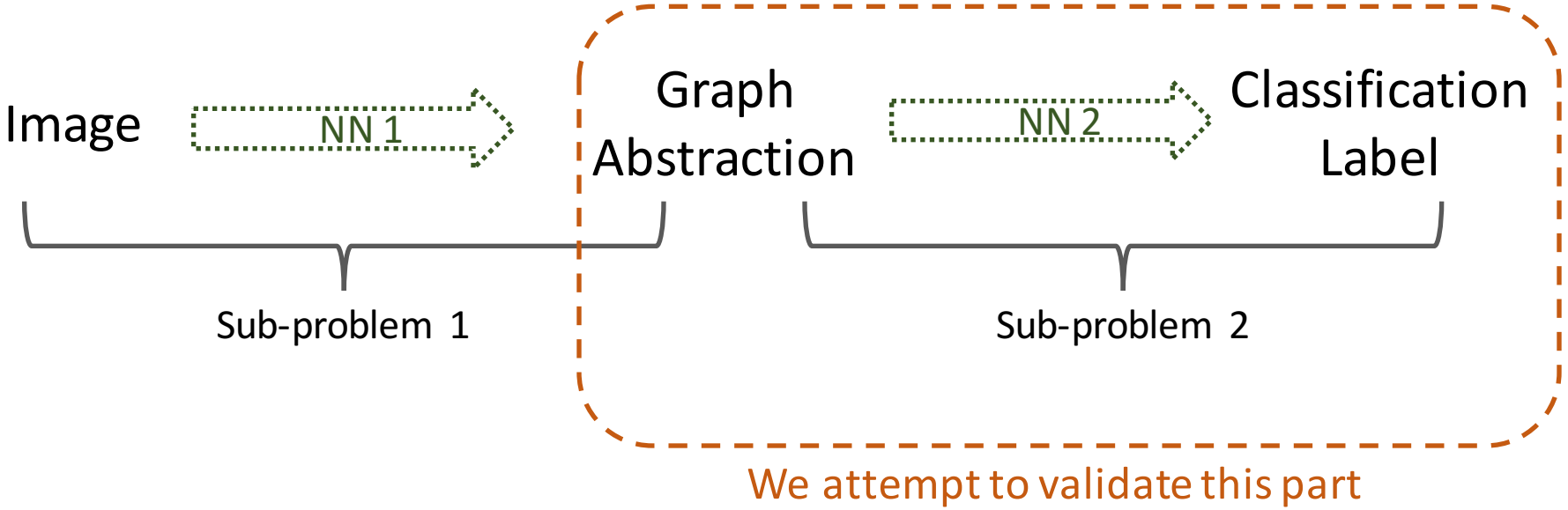


Graphs kernel

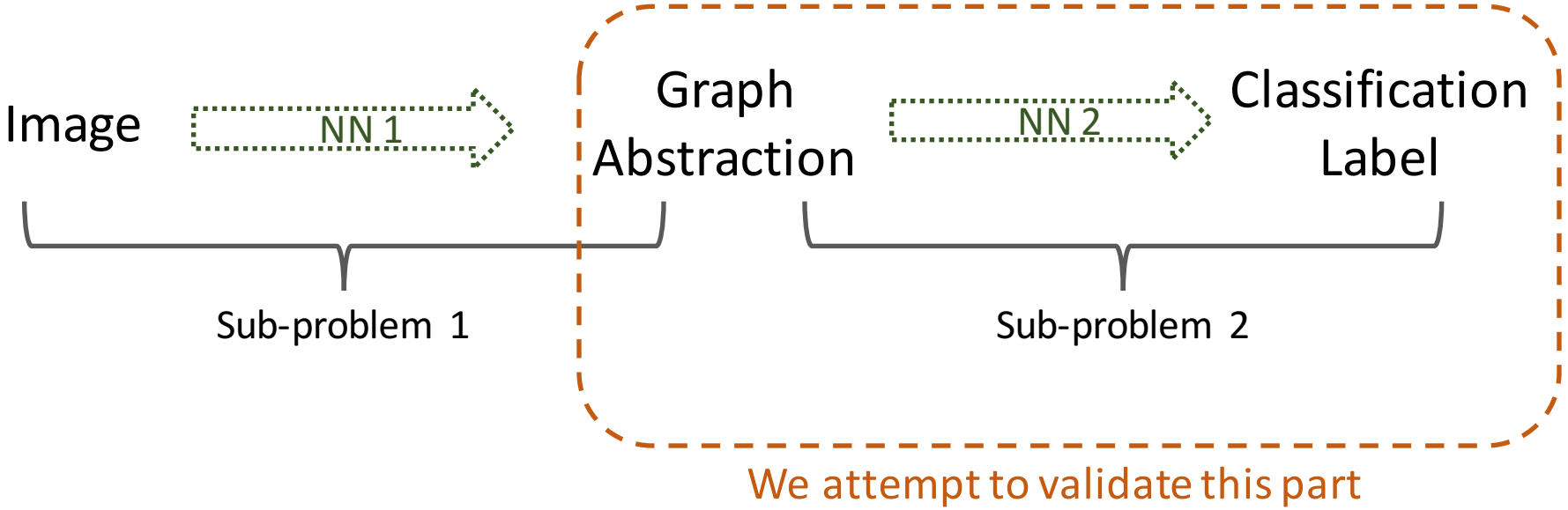
Architecture for sub-problem 2



Preliminary results



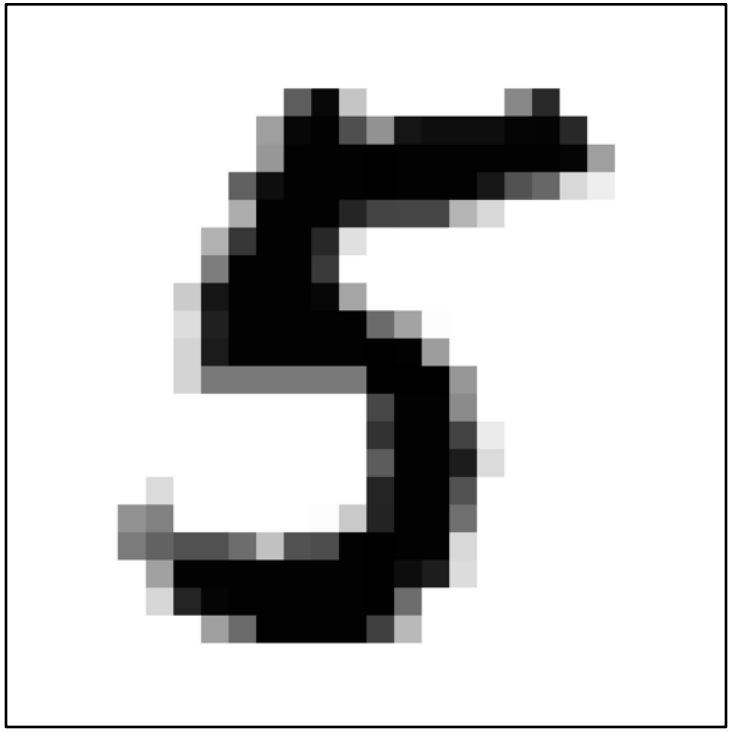
Preliminary results



- Generate graph inputs from images with known ground truths
  - Nodes
  - Edges

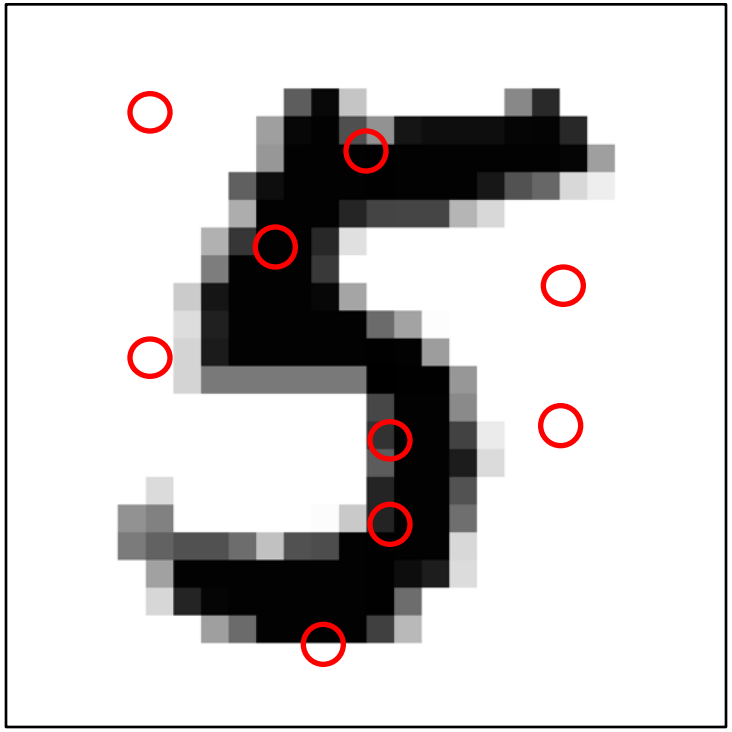
- We use MNIST dataset: images of handwritten digits (0,1,…,9)

Experiment setup to generate graphs



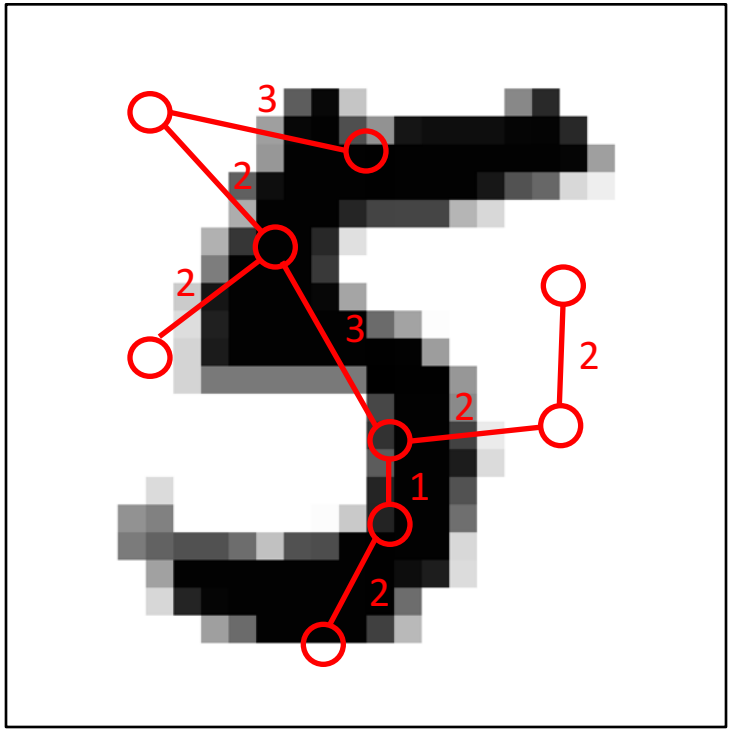
1. Take an image

Experiment setup to generate graphs



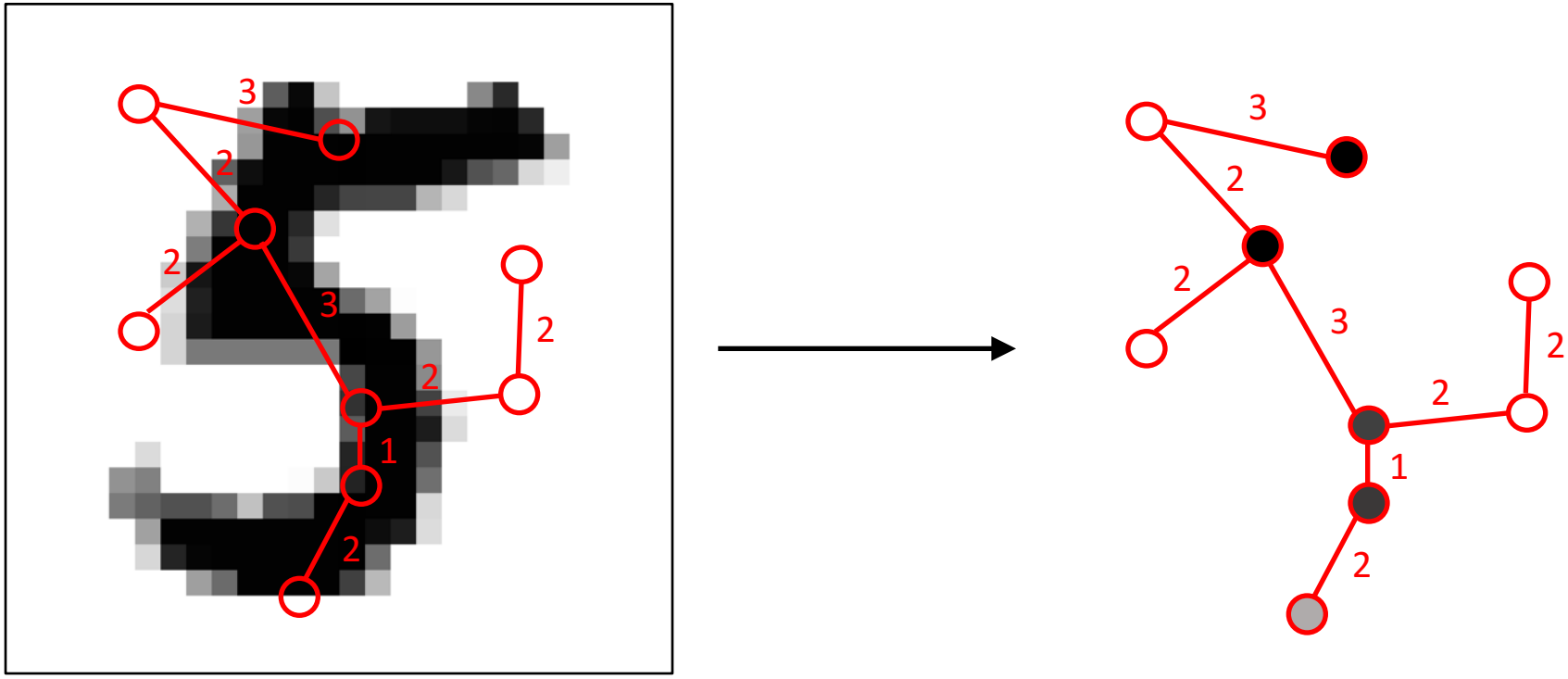
1. Take an image
2. Select N points at random

Experiment setup to generate graphs



1. Take an image
2. Select N points at random
3. Connect ever node to k-nearest neighbors (based on distance)
4. Extract the graph and color at the nodes

Experiment setup to generate graphs



Obtain f, A from the image

Preliminary results

1. As the number of nodes used to create the graph increases, accuracy increases

   With a fixed graph, 50 nodes : 84%

   200 nodes : 91%

Preliminary results

1. As the number of nodes used to create the graph increases, accuracy increases

   With a fixed graph, 50 nodes : 84%

   200 nodes : 91%

2. The technique also works for varying graphs (*A* changes)

   Classification of digits 0 and 1 : 92%

   digits 1 and 7 : 69%

   } 50 nodes

Summary

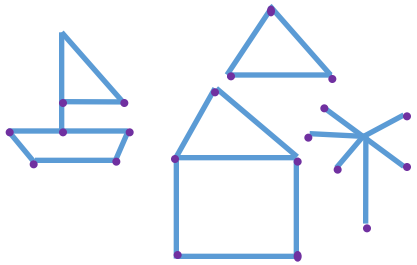
Image

Classical CNN

Classification label

Airplane : 3 %
Automobile : 10 %
…
Truck : 85 %

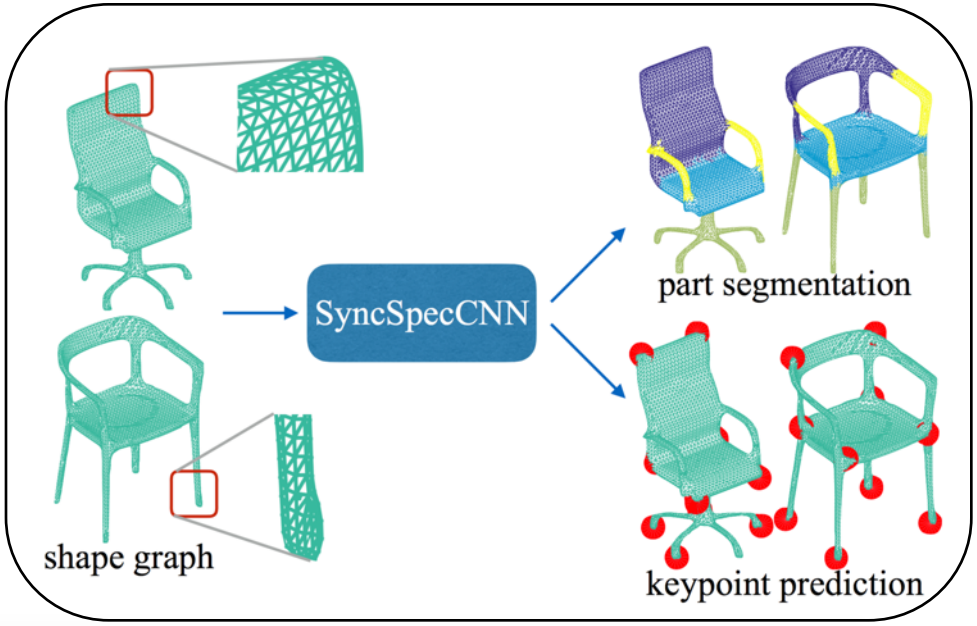
Graph abstraction



Other Applications of Geometric Deep Learning

- Analyzing meshes/ point cloud data (from Lidar)
  - Convert to graph and use Geometric Deep Learning techniques

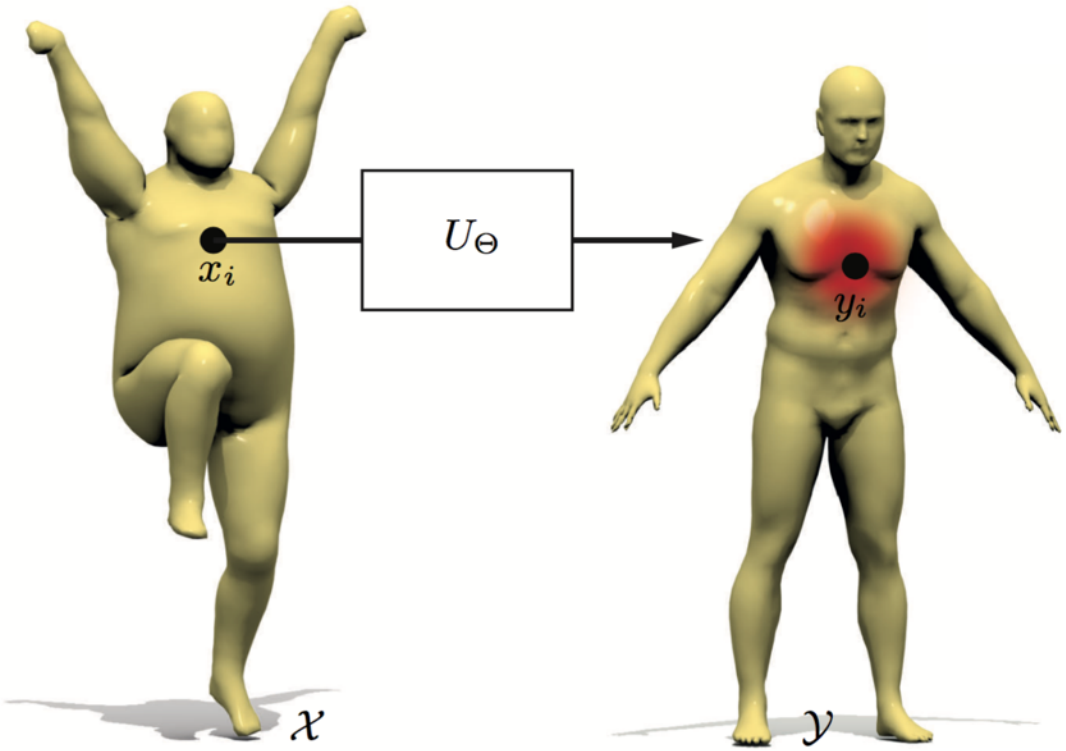
[Verma et. al. 2018]



[Yi et. al. 2017]

Other Applications of Geometric Deep Learning

- Shape correspondence for deformable objects



[Bronstein et. al. 2017]