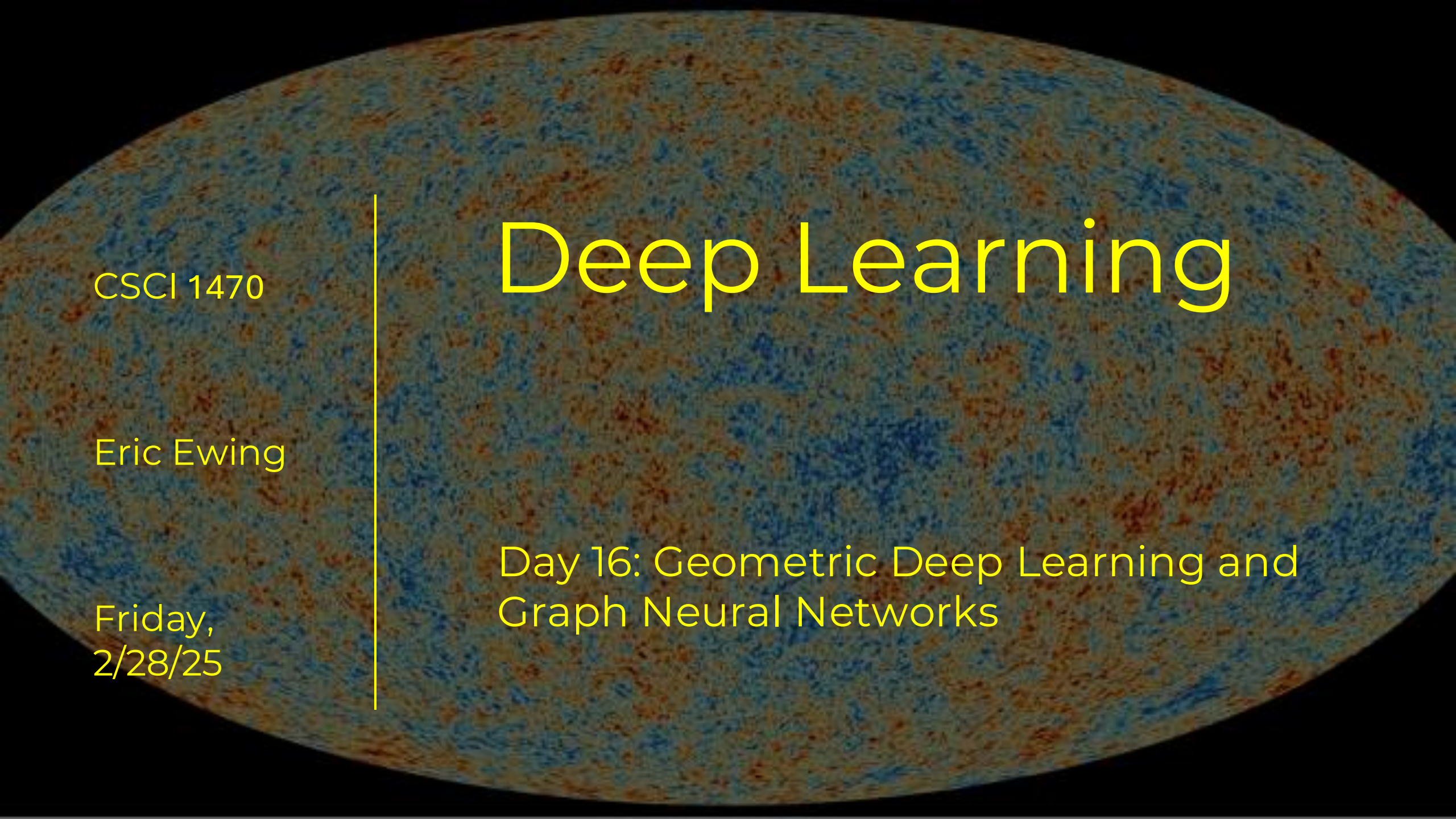


CSCI 1470

Eric Ewing

Friday, 2/28/25

Deep Learning

Day 16: Geometric Deep Learning and Graph Neural Networks

Deep Learning Architectures

- So far, we have covered two classes of neural networks:
  - Multi-Layer Perceptrons (Fully-connected/Dense)
  - Convolutional Neural Networks
- Each class was defined by their use of specific layer types
- Within each class there are individual network architectures
  - E.g., ResNet, VGG, AlexNet
- You should not focus on learning specific architectures, but the reason behind their choices

Why did we introduce convolutions in the first place?

What methods enable us to more have a more stable training process? Are there more general/principled ways to avoid overfitting than trial and error hyperparameter tuning?

What difficulties did researchers run into trying to scale AlexNet/VGG and how did they resolve these issues?

How can we apply these lessons outside of CNNs?

Goals For Today

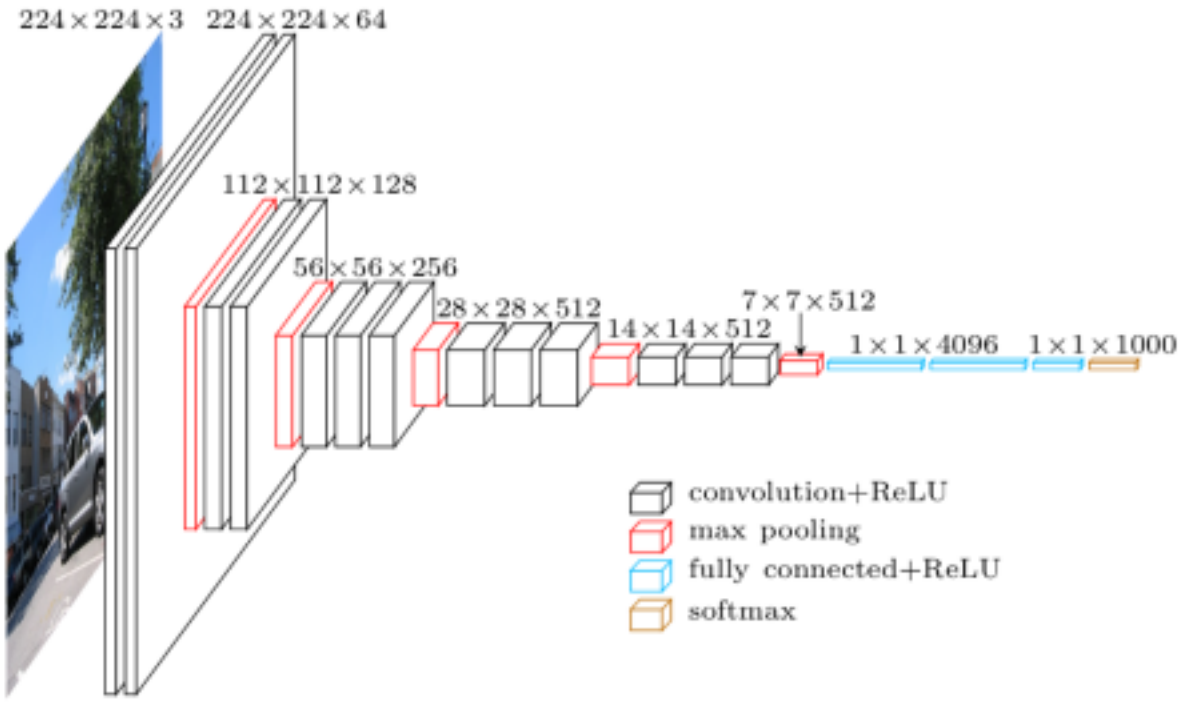
Learn about Geometric Deep Learning

1) Going beyond images and grids for input
2) Graph Convolutional Networks

Structured Data

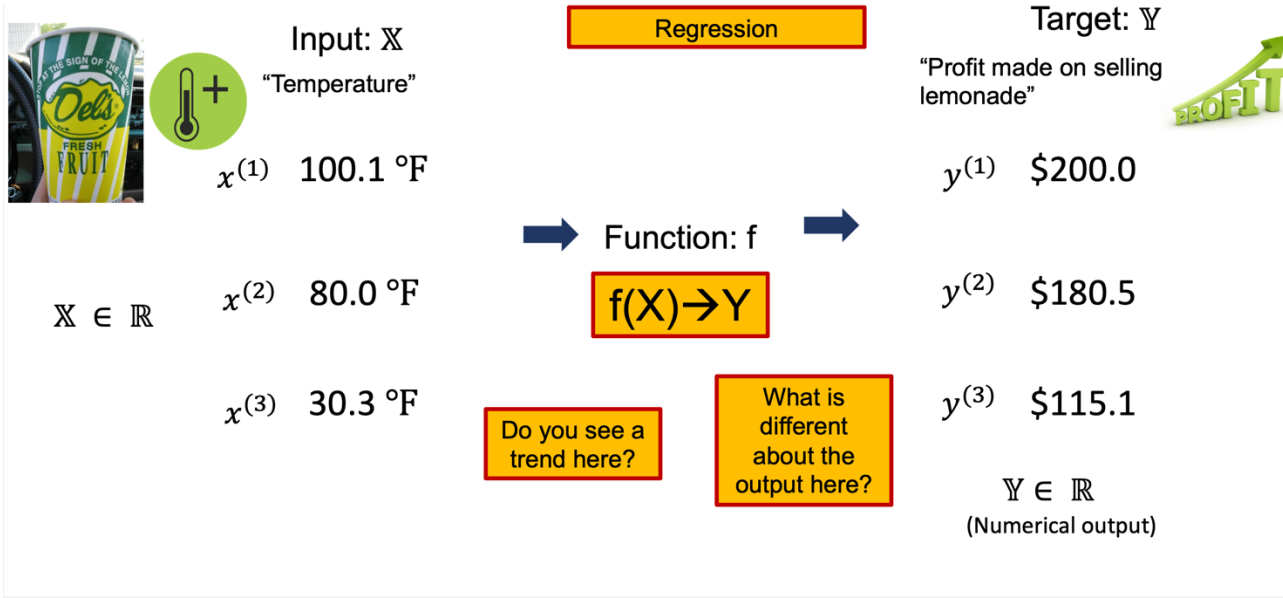
Images are a ***structured*** type of data

The structure (order and layout) of pixels matters

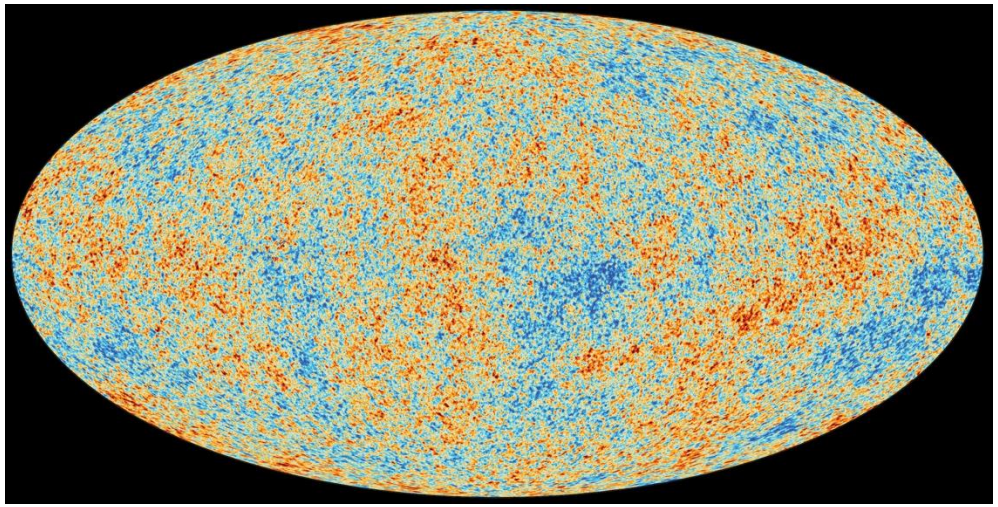


Lemonade stand data is ***not structured***

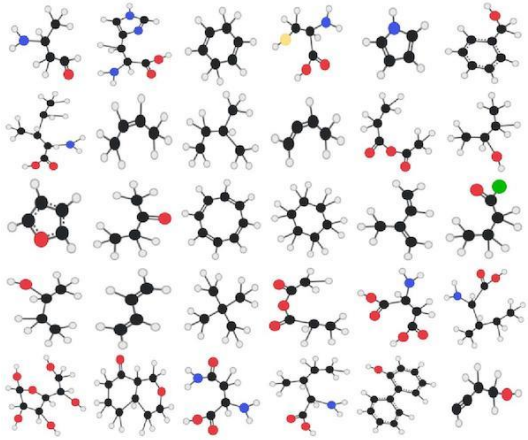
The order of features we use does not change training



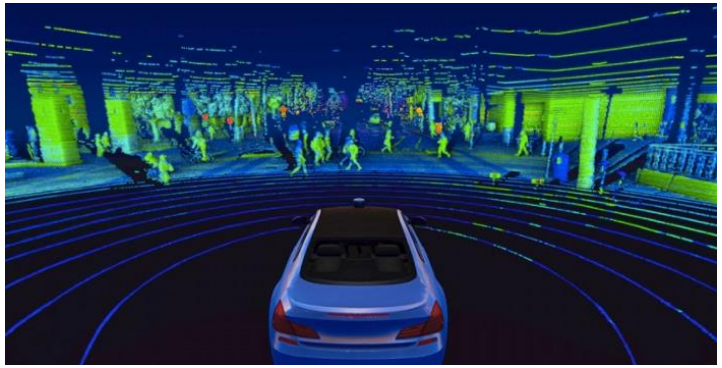
Other Types of Structured Data Types



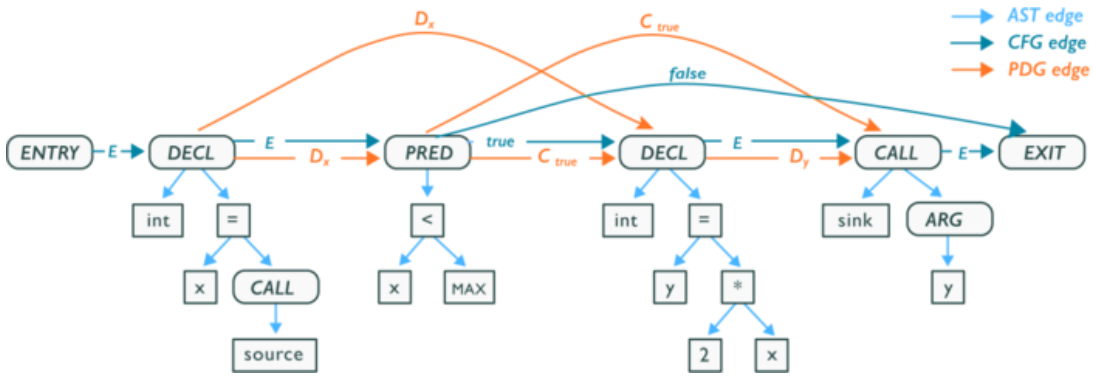
Microwave Background Radiation



Molecule Data



Point Clouds (from LIDAR)



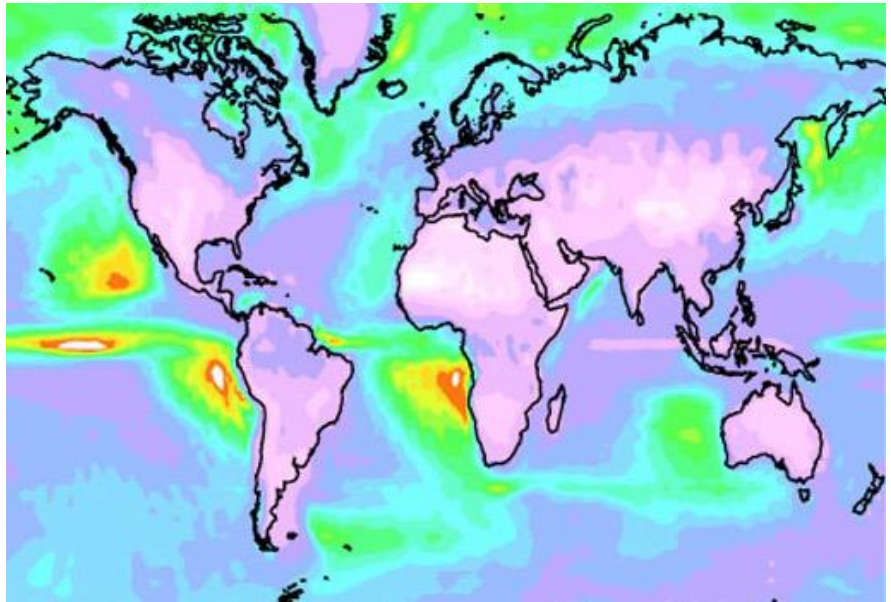
Code Graphs

Geometric Deep Learning

- We turned to convolutions for image data to give our networks “spatial reasoning”
- By Explicitly modelling the relationship of our data, we can achieve better results
- Geometric Deep Learning is the subfield of DL dedicated to learning representations of *structured* data.

Goal: Take as input data from satellites, predict carbon dioxide

Idea 1: Take all of the data as input, make it look like an image, use CNNs to learn to output carbon dioxide amounts

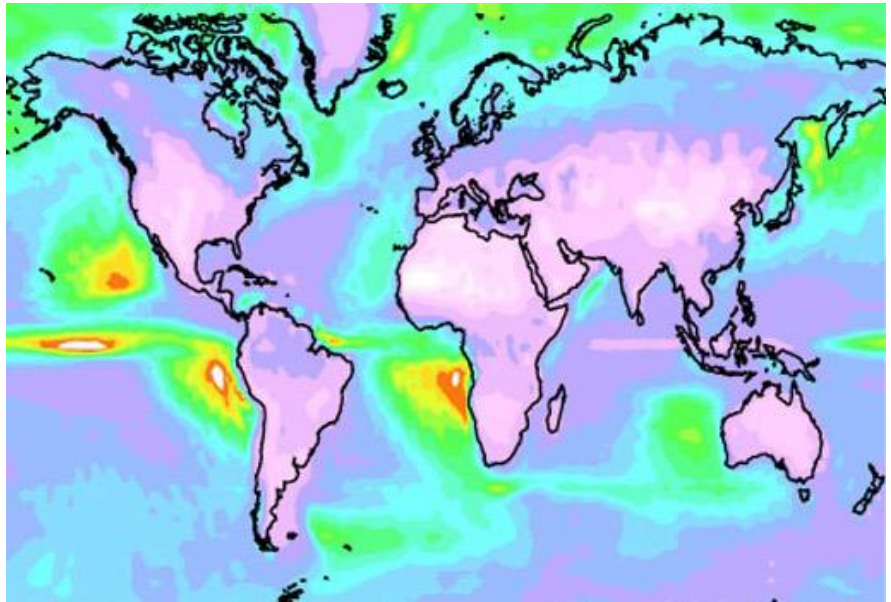


https://newton-climate.github.io/#research

Goal: Take as input data from satellites, predict carbon dioxide

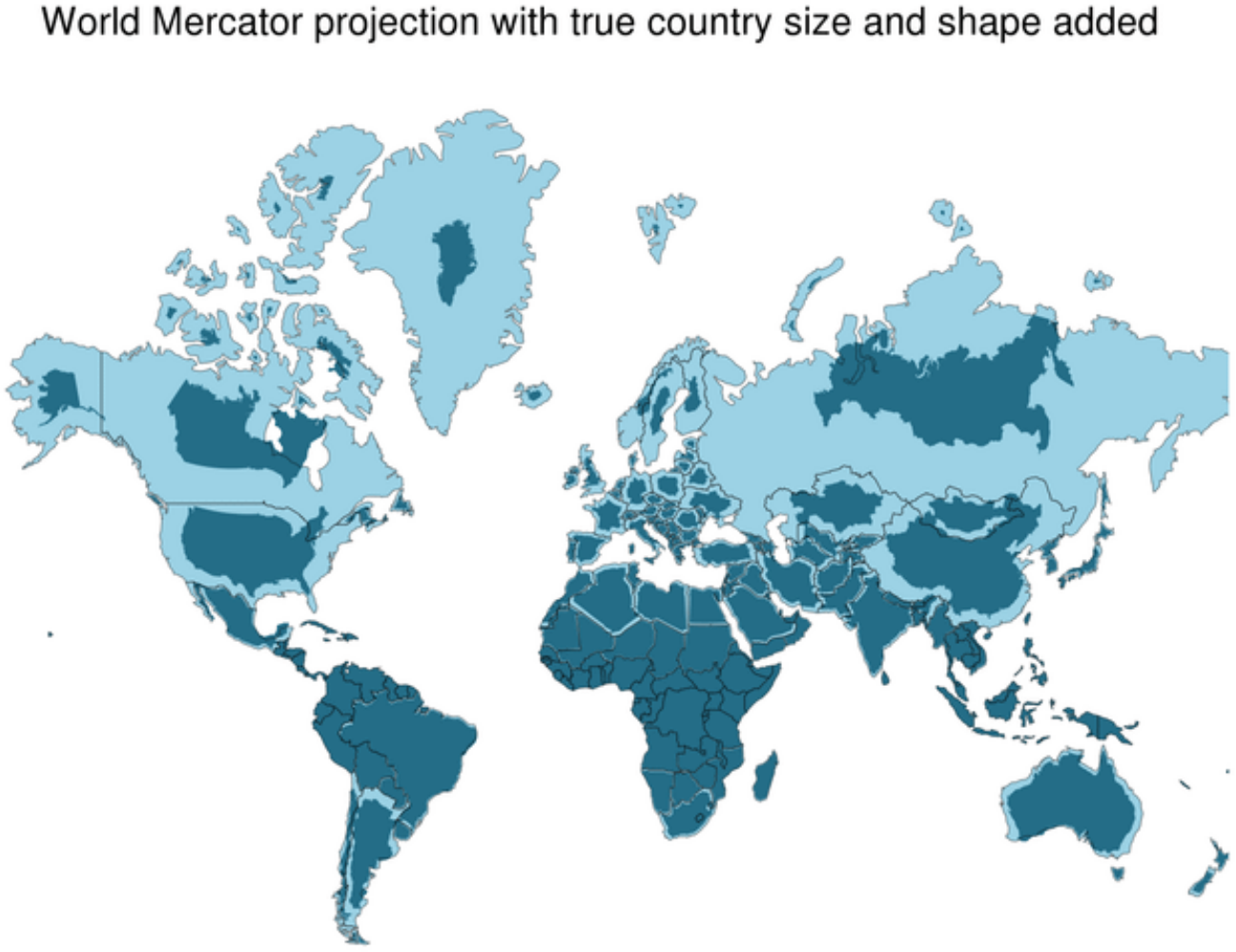
Idea 1: Take all of the data as input, make it look like an image, use CNNs to learn to output carbon dioxide amounts

Issue: Maps ☹



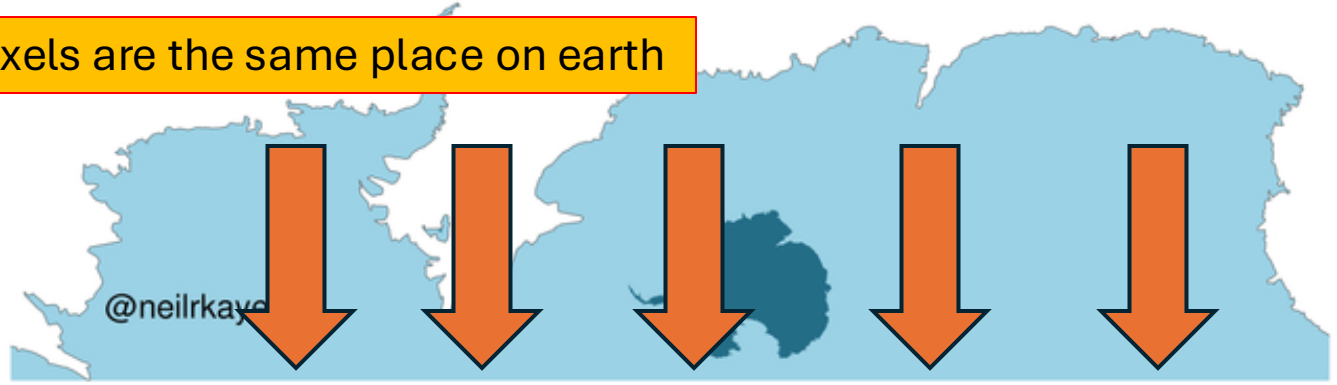
https://newton-climate.github.io/#research

Issue: Trying to represent a 3D sphere on a 2D plane



A 2D convolution doesn't make sense here...
But there's still structure (don't just want an MLP)
What about a Spherical Convolution!

All these pixels are the same place on earth



What should we pad the side of the image with?

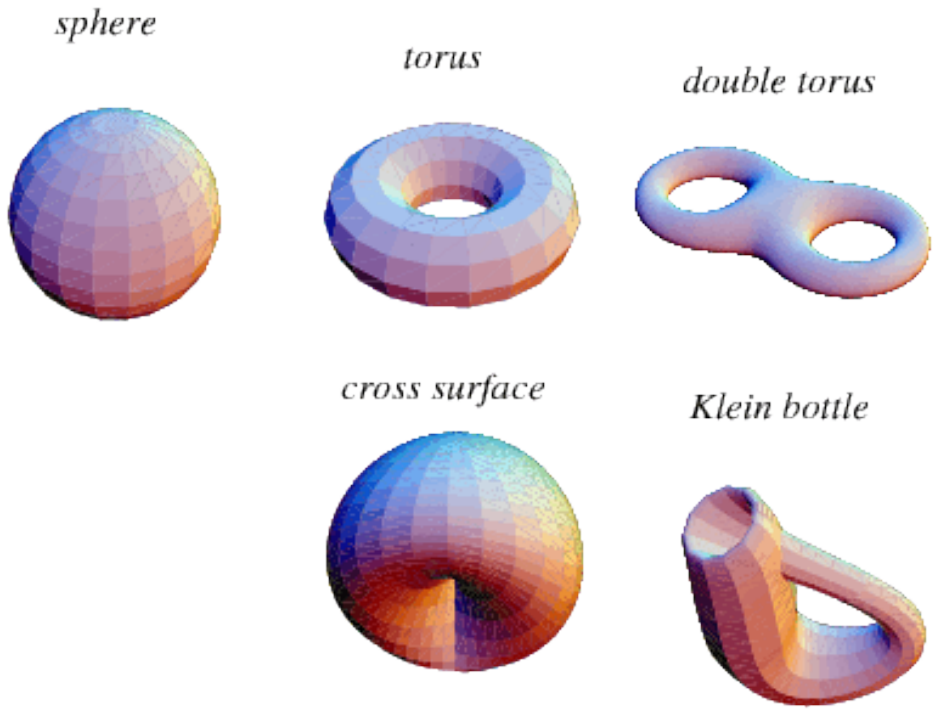
Instead of padding with 0's, "pad" with adjacent locations on earth

Convert 2D convolution on grids to work on Spheres

Manifolds

- Topological Space that looks locally like Euclidean space, but can be globally curved
  - For example, the Earth
- Can we generalize convolutions to more than just spheres?

Need to specify which points are “close” to other points, convolve neighborhoods of points together

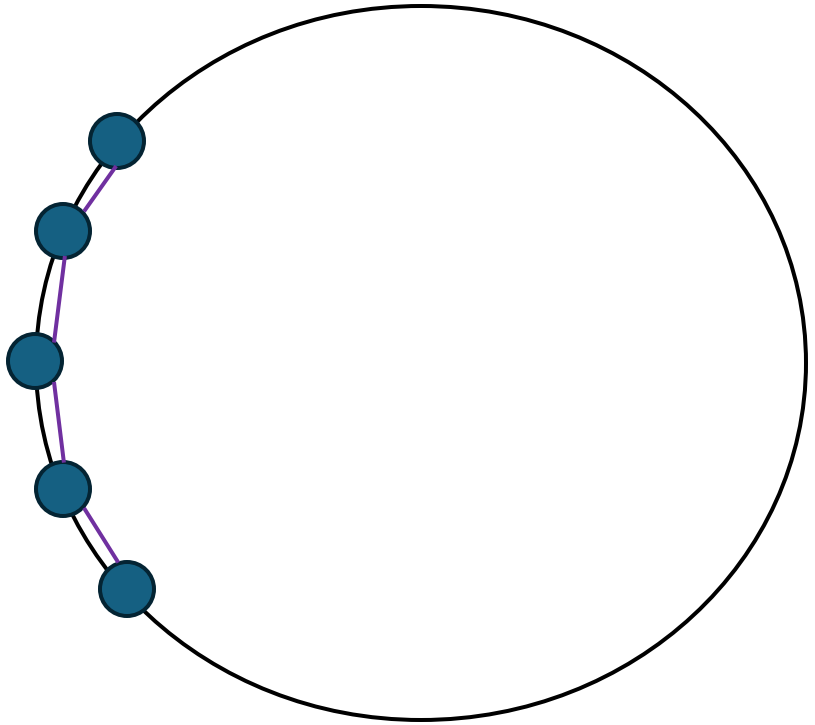


Extending Convolutions to Manifolds

A circle is a manifold in 2D
(like a sphere or taurus in 3D)

For every point on the manifold,
connect it to “close” points

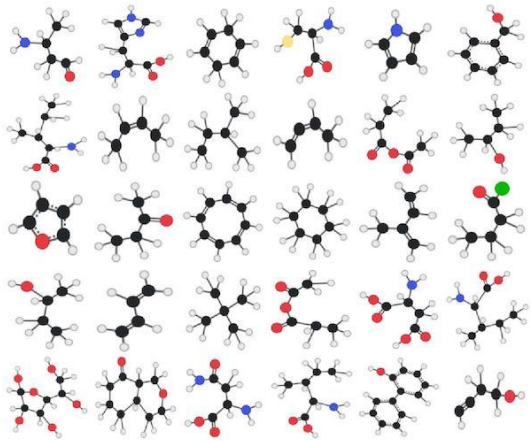
What data structure does this lead to?



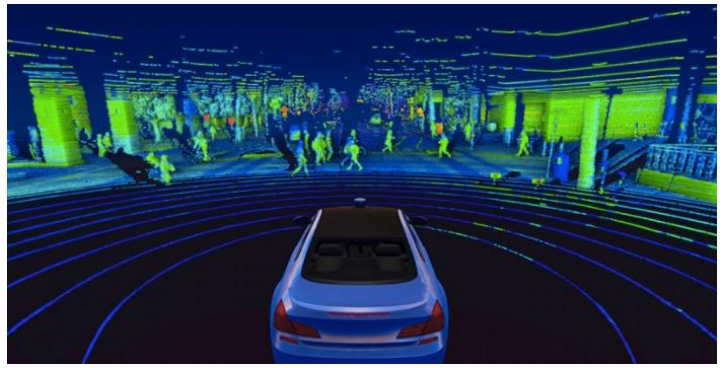
Geometric Deep Learning

- It is often confusing why much of Geometric Deep Learning deals with graphs…
  - Geometry happens in continuous space
  - Graphs are discrete

Molecules

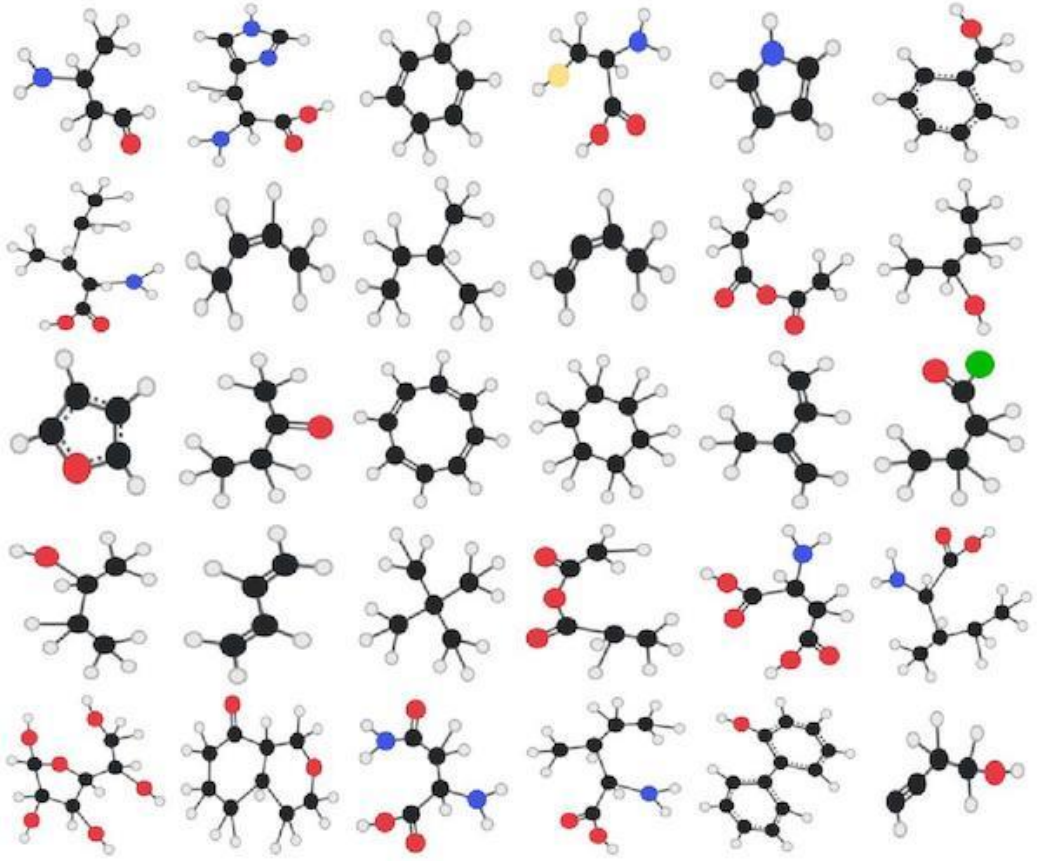


LIDAR Data



What's Hard About Neural Networks for Graphs?

- Can have different numbers of nodes
- Can be arbitrarily complex
  - Different numbers of edges
  - Different edge patterns
  - Weighted edges



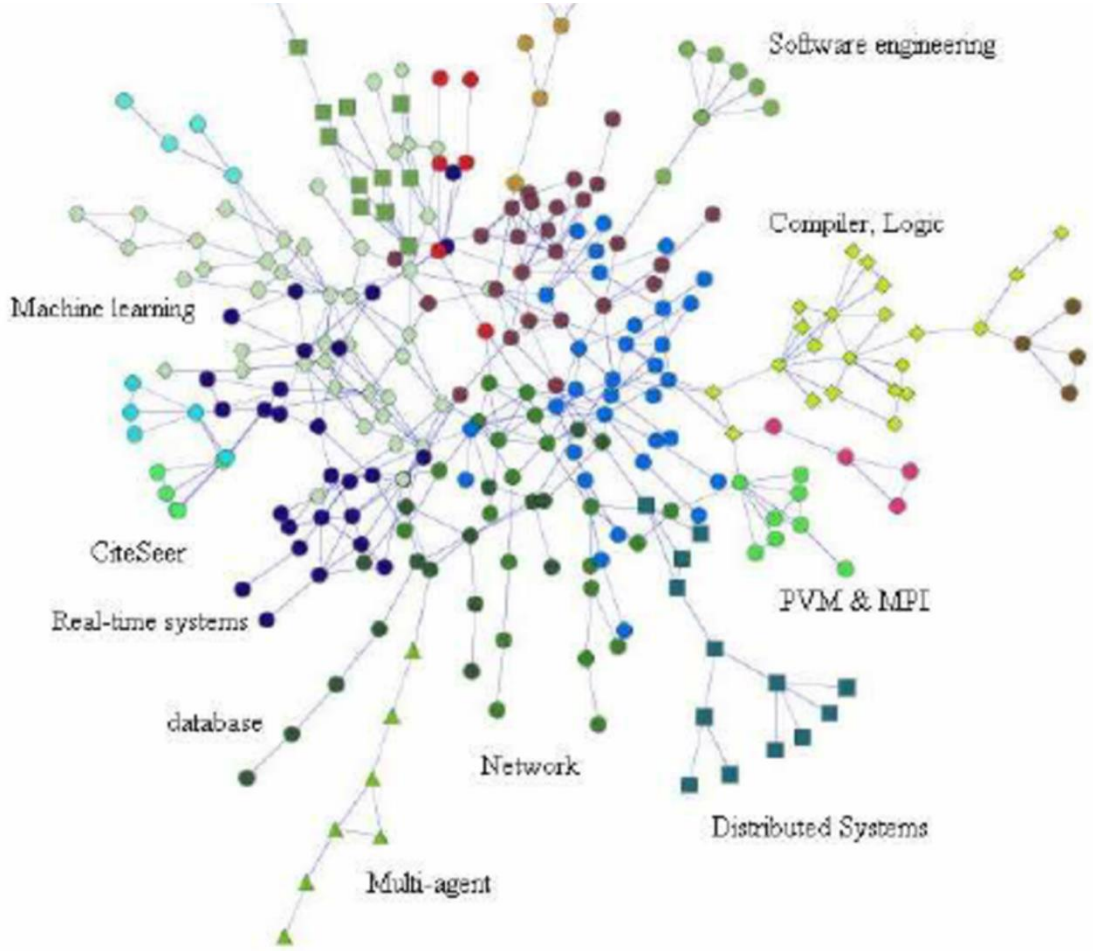
How will a Graph Convolution have to differ from a 2D convolution?

Example Dataset

CiteSeer Dataset:

Collection of academic papers:

- directed edges connect papers to the papers they cite
- Words used by each paper
  - Vector of length 3703 for each word in “vocabulary”, 1 if present 0 if not



Clustering CiteSeer collaboration network, k = 21. Best viewed in color.

Ruan, Jianhua & Zhang, Weixiong. (2006). Identification and Evaluation of Weak Community Structures in Networks.. Proceedings of the Twenty-First National Conference on Artificial Intelligence, July 16-20, 2006. Boston, Massachusetts.

Graph Convolutions: Message Passing

- Graph Neural Networks (GNNs) need to share information between connected nodes
- This sharing of information is called *message passing*

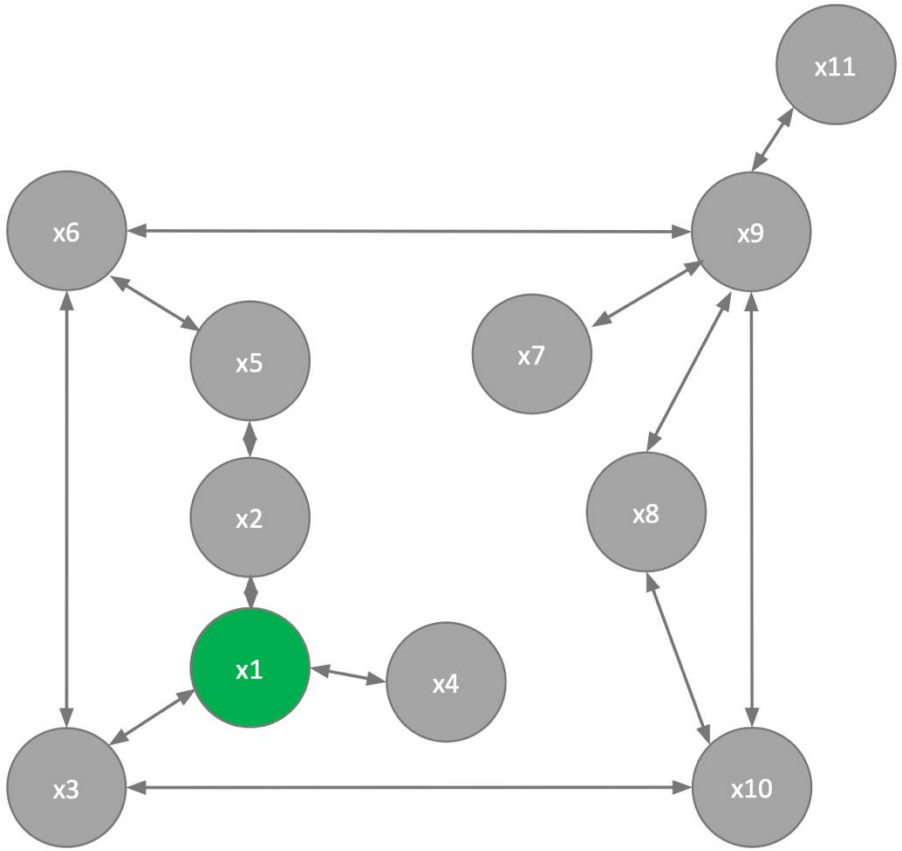
Graph Convolutions

For a graph G=(V, E)

Each node v_i has features x_i, each edge (v_i, v_j) can also have features (e.g., weight)

1. Start by mapping x_i to hidden features h_i

$$f_{init}(x_i) = h_i$$



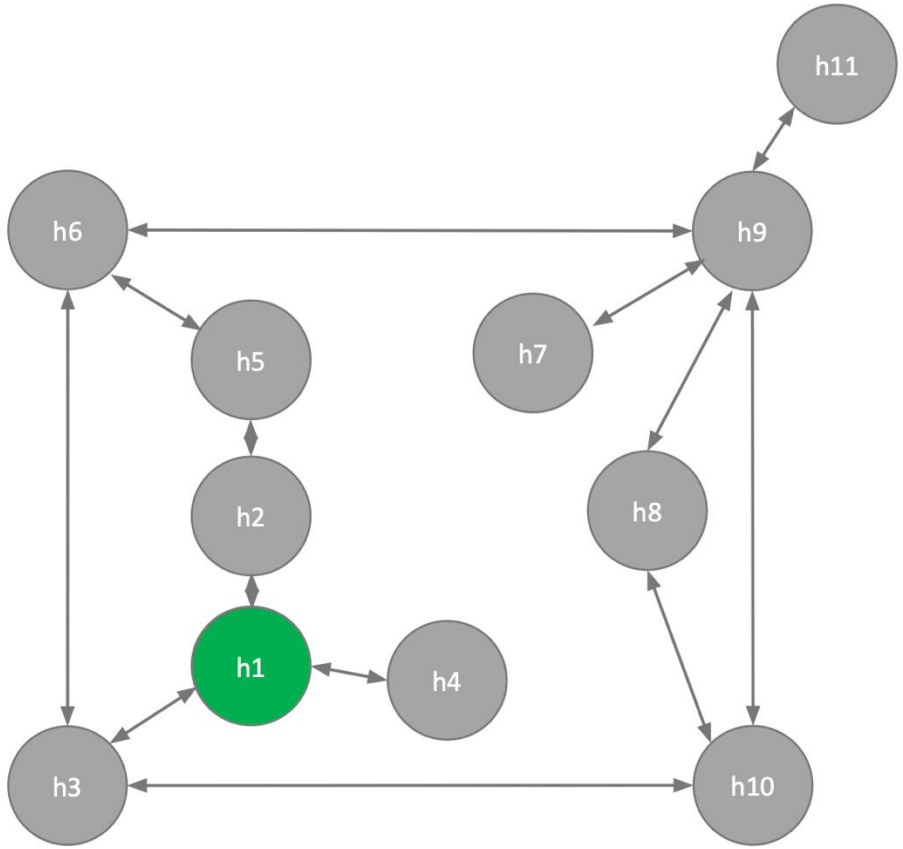
Graph Convolutions

For a graph G=(V, E)

Each node v_i has features x_i, each edge (v_i, v_j) can also have features (e.g., weight)

1. Start by mapping x_i to hidden features h_i

$$f_{init}(x_i) = h_i$$



Message Passing

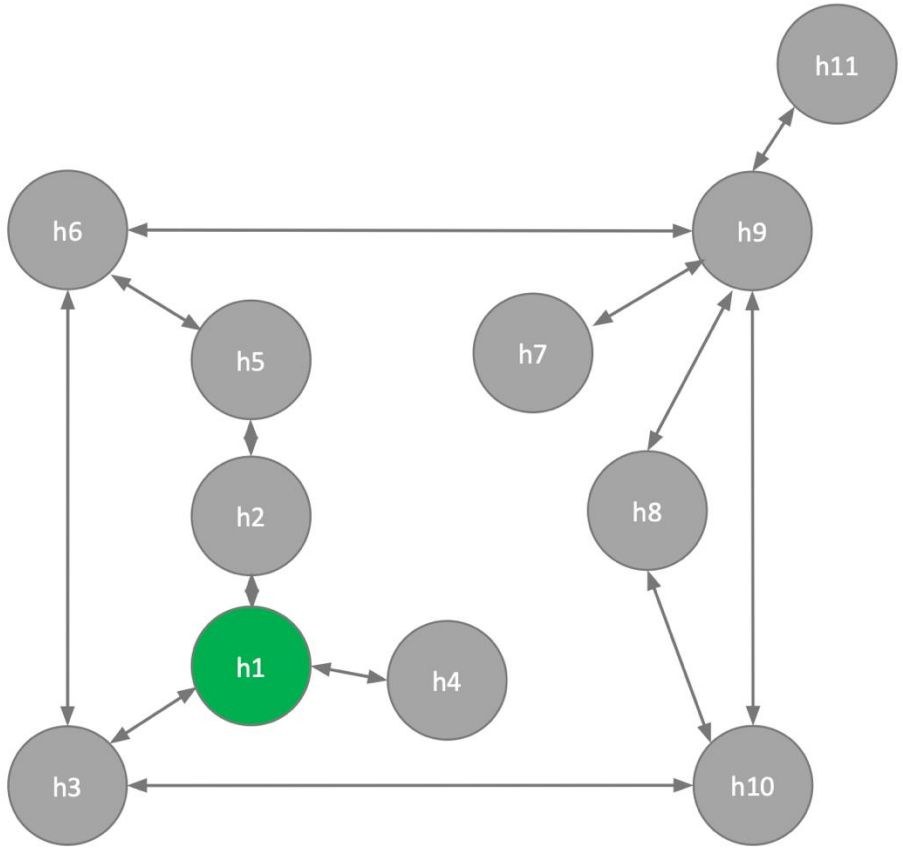
For a graph G=(V, E)

Each node v_i has features x_i, each edge (v_i, v_j) can also have features (e.g., weight)

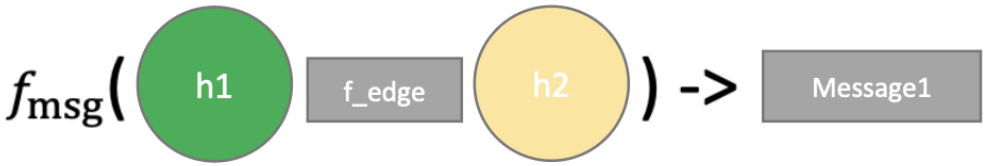
1. Start by mapping x_i to hidden features h_i

$$f_{init}(x_i) = h_i$$

2. Perform message passing among adjacent nodes

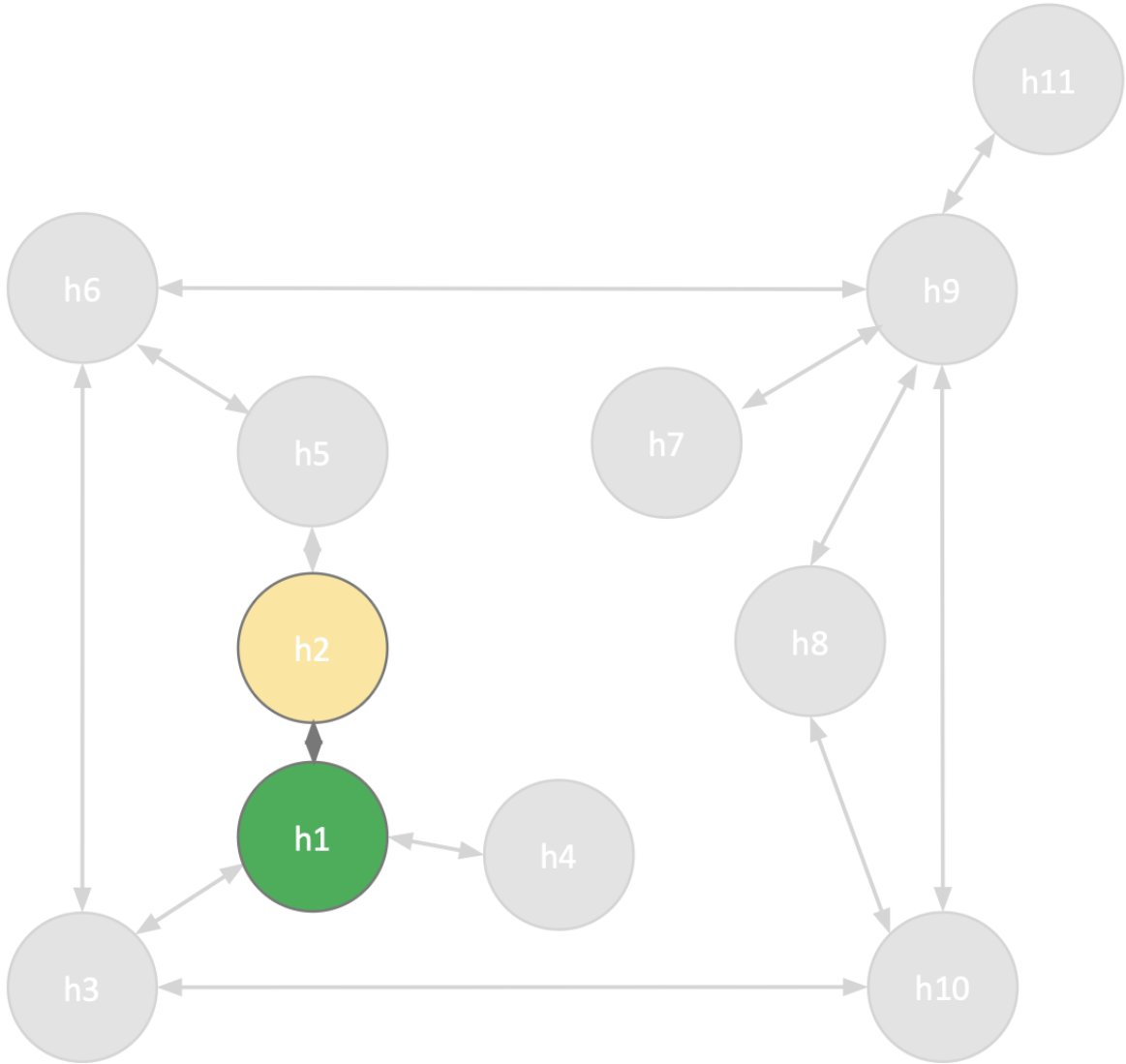


Message Passing

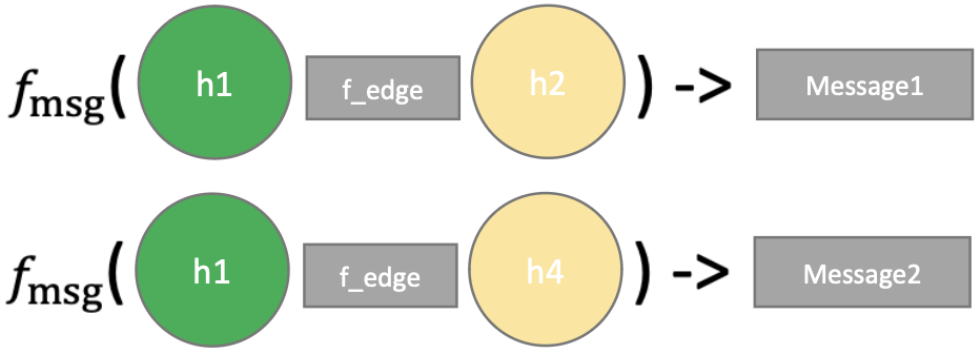


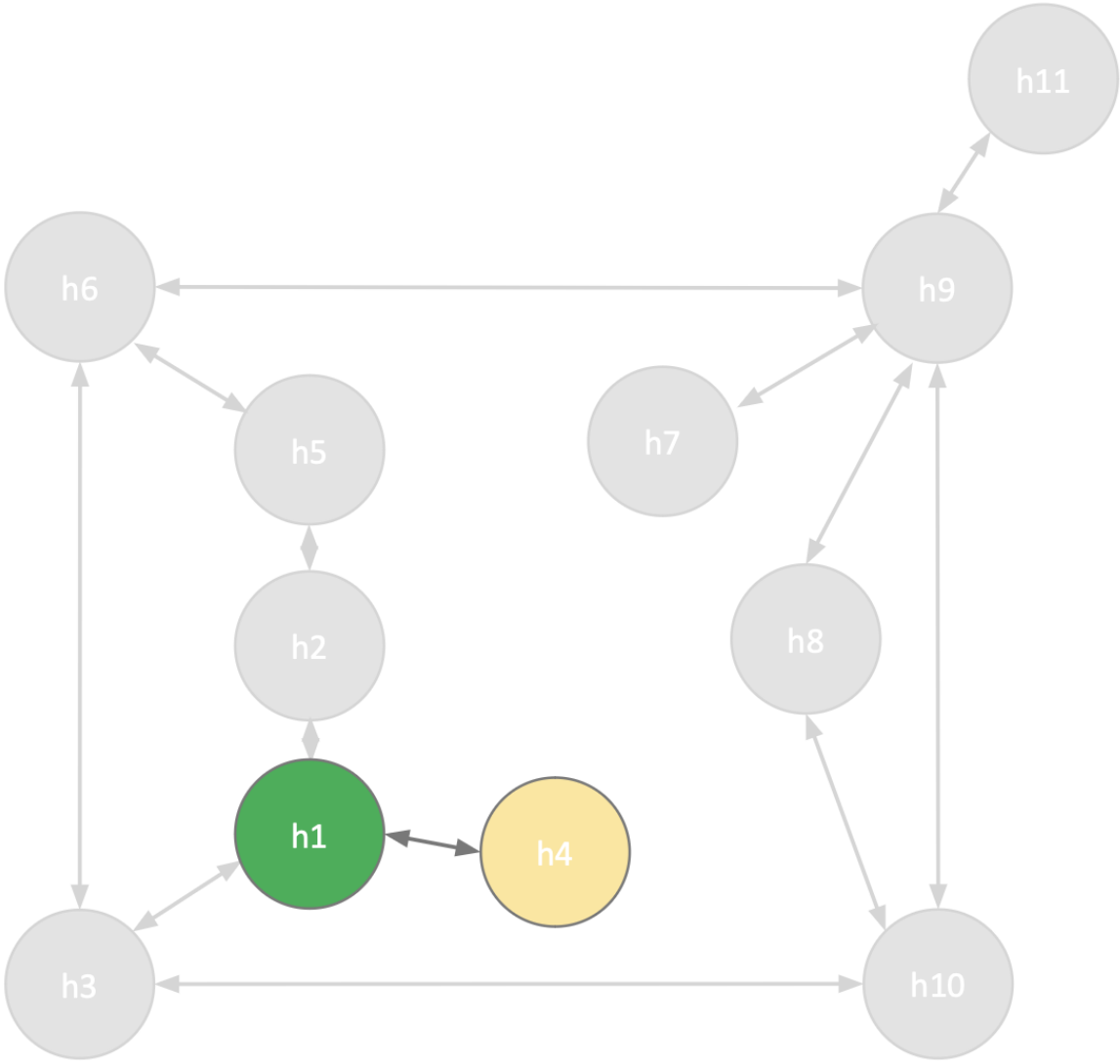
We use a neural network f_{msg} to compute a ***message*** between two nodes in the graph

- Inputs
  - Hidden states for each node
  - Features along the edge
- Output
  - Message vector

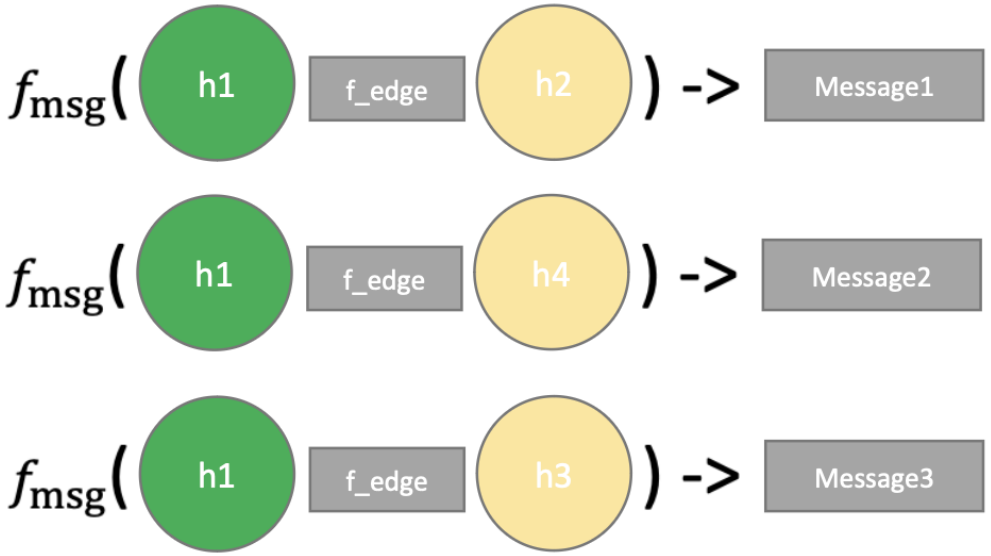


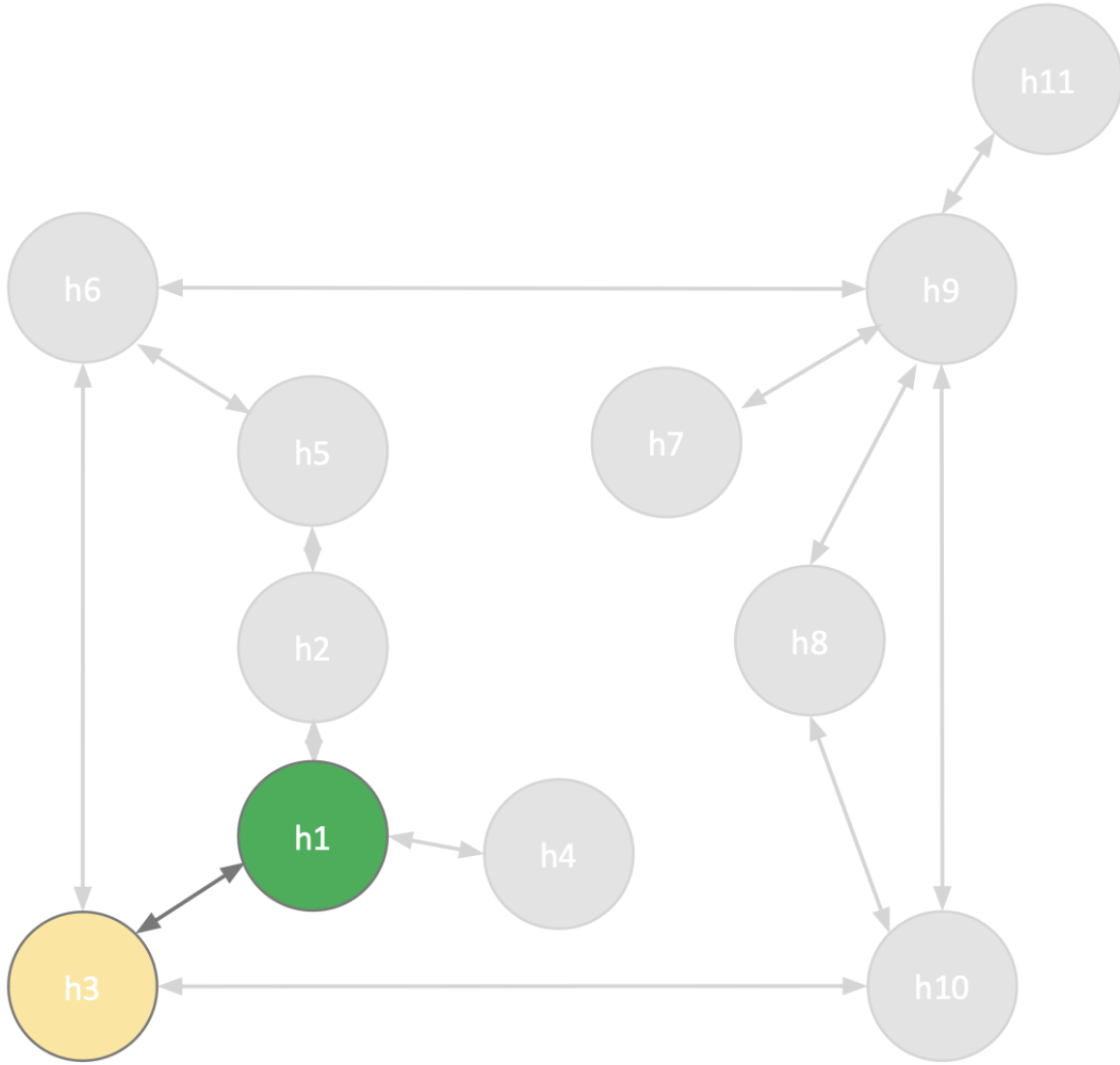
Message Passing



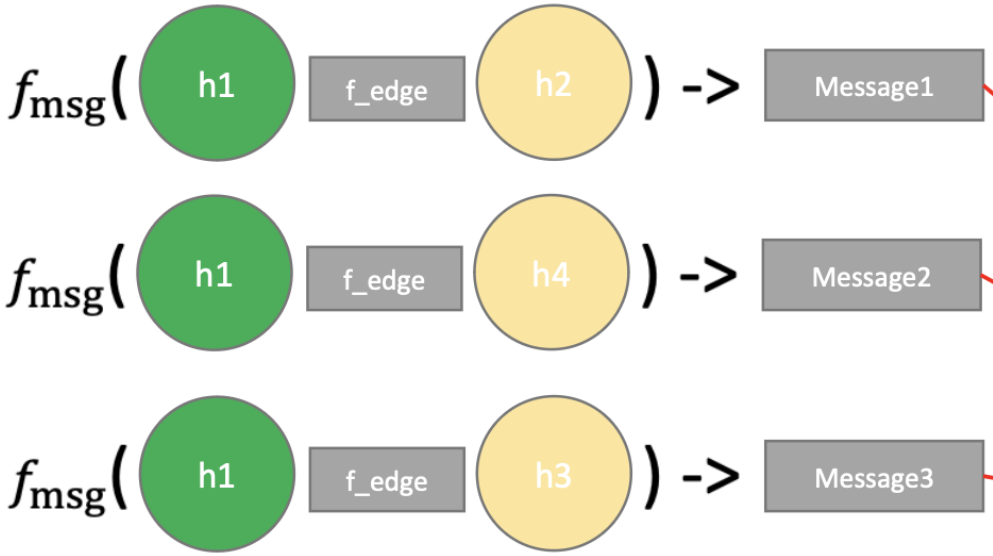


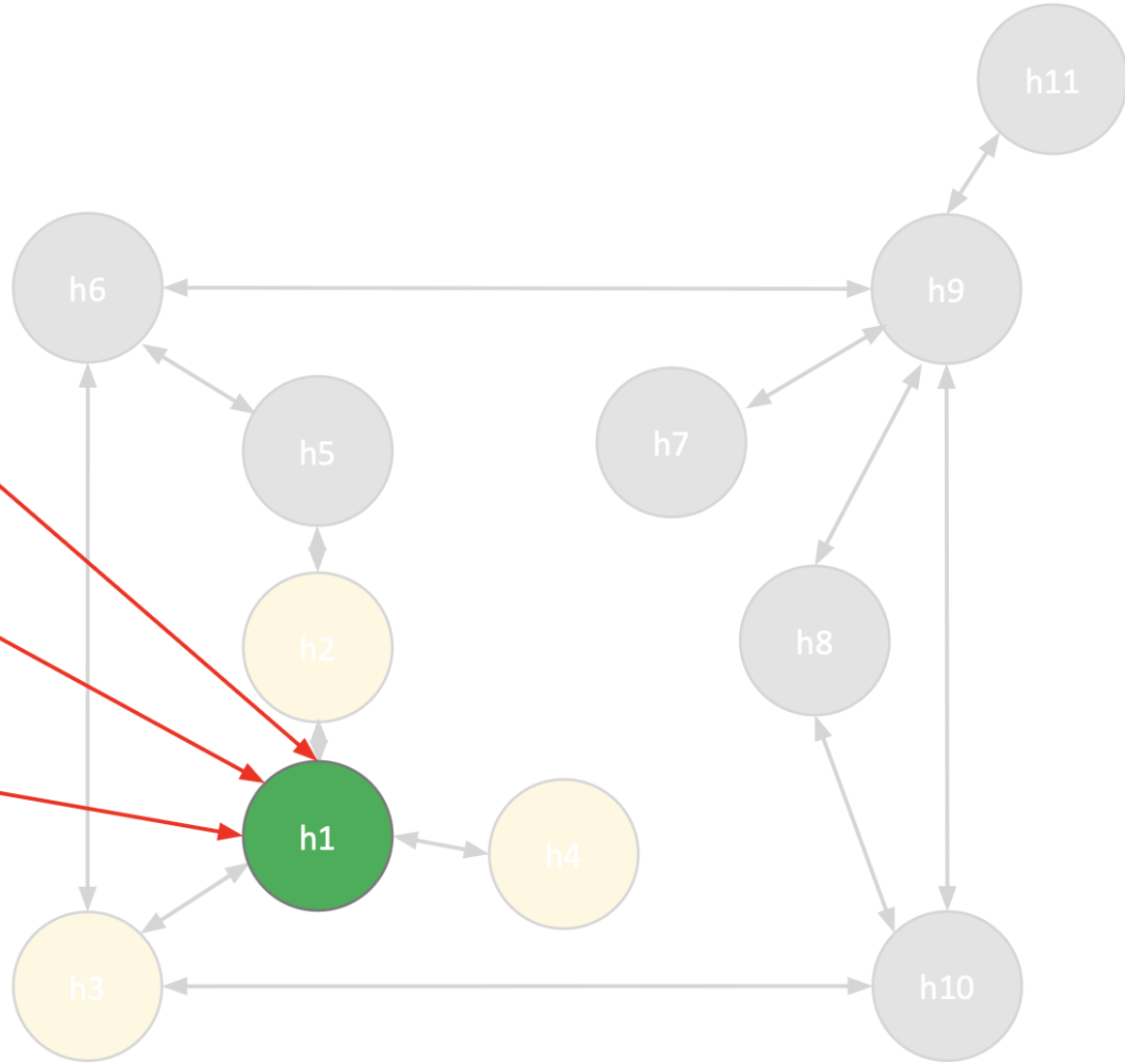
Message Passing





Message Passing





Aggregate all the messages to compute the output hidden state vector for $\mathbf{h}_1$

(e.g. sum up the messages)

NOTE: The aggregation function can differ leading to different formulations of GNNs

GraphSage

Average of neighbor's embeddings for previous round

Learnable weights B times previous embedding

$$h_v^k = \sigma\left(W_k \sum_{\{u \in N(v)\}} \frac{h_u^{k-1}}{|N(v)|} - B_k h_v^{k-1} \right) \forall k \in \{1, \ldots, K\}$$

Hidden features after k rounds of message passing

Weights layer for k'th round

Nonlinear activation function (e.g., ReLU)

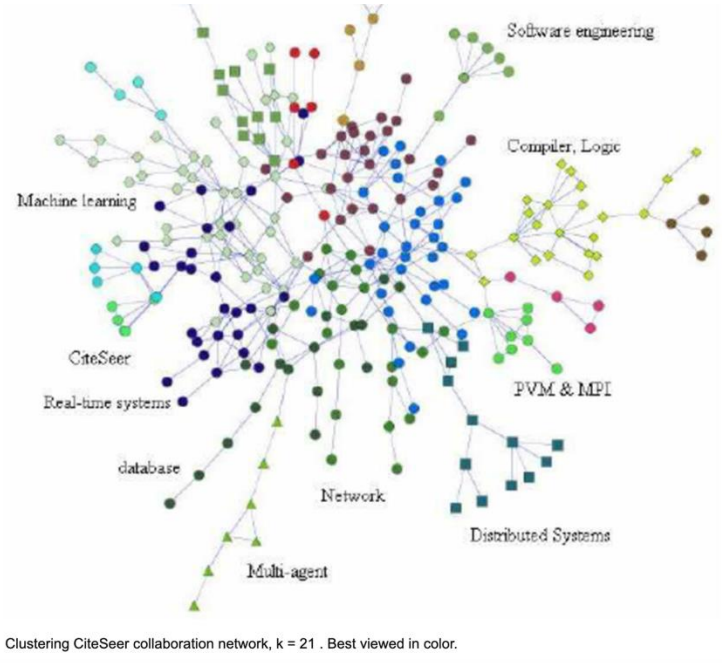
$$z_v = h_v^K$$

Embedding after all rounds of message passing

W and B are learnable parameters!

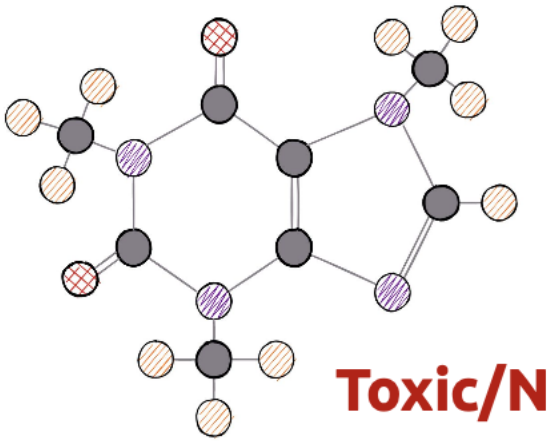
After K iterations, we end with a vector of features z_v for each node in our graph.

What might we actually want to predict?



Clustering CiteSeer collaboration network, k = 21 . Best viewed in color.

- Node level predictions: What is the topic of a paper?
- Link Prediction: For a new paper, what other papers might it cite?
- Graph level predictions



Toxic/Not toxic?

Think/Pair/Share:
How can we turn features for each node into node-level predictions?

What about link prediction?

Graph level predictions are hard, we have |V| vectors of features, which may vary between different graphs we take as input), how can we get a single output from variable sized input?

Node Prediction

Each node has a learned representation z_v, we can learn a fully-connected layer to go from features z_v to output

$$f(z_v) = \sigma(W z_v)$$

Link Prediction

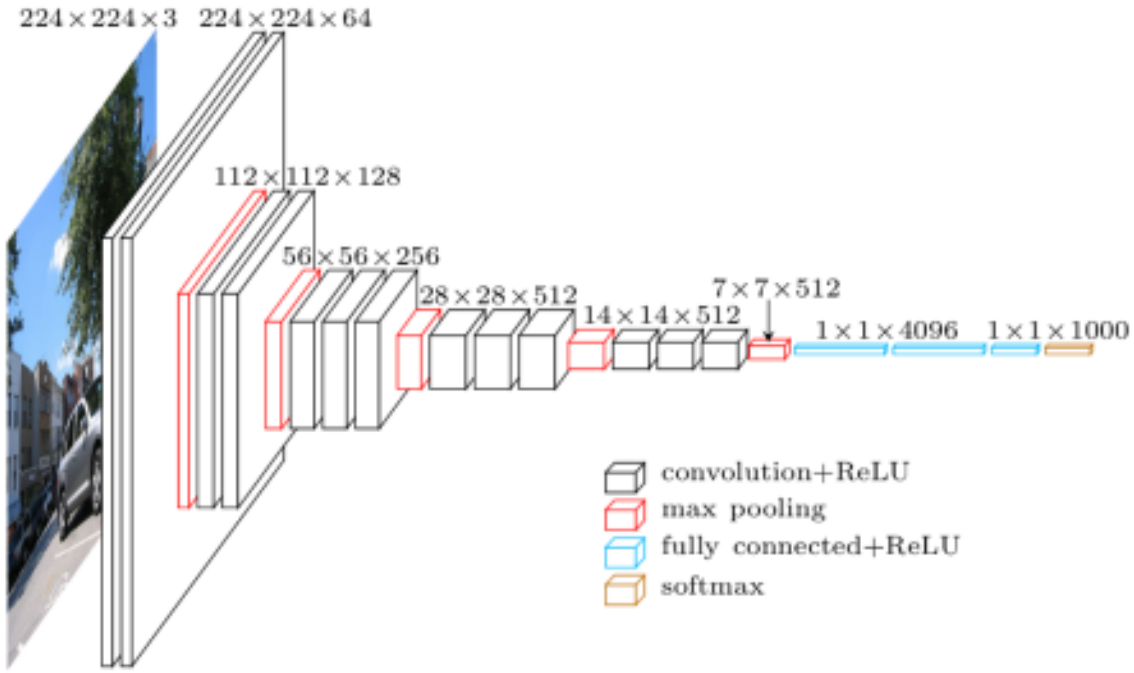
Learn a function that takes two nodes as input and predicts the presence (or absence) of an edge

$$f(z_v, z_u) = \sigma(W(z_v \odot z_u))$$

Element-wise product

Global Pooling

- We covered MaxPool in CNNs, which looks at the maximum value in a local neighborhood
- We can consider an alternative way to do MaxPool, which looks at the entire feature for all inputs and returns a single max value

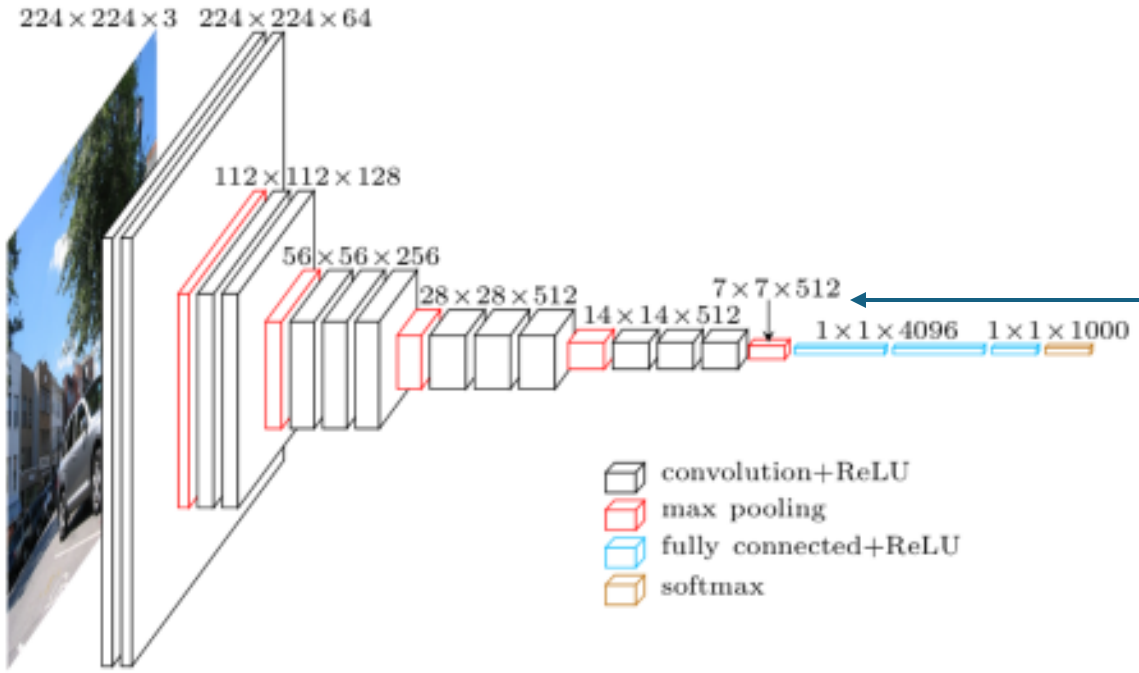


Normally, CNNs flatten the results of their final convolution and feed into a linear layer of $W \times H \times C$.

This has to be the same size for all examples!

Global Pooling

- We covered MaxPool in CNNs, which looks at the maximum value in a local neighborhood
- We can consider an alternative way to do MaxPool, which looks at the entire feature for all inputs and returns a single max value

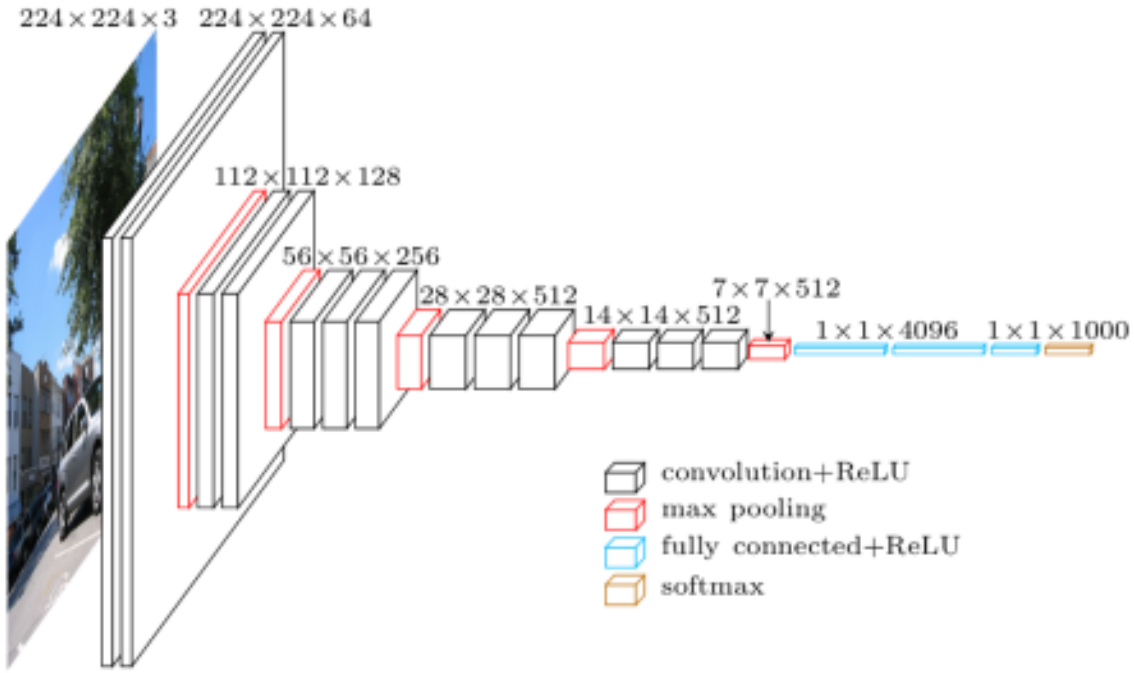


Alternatively, we can to the max over each channel in the final convolutional output

Output will have 512 features instead of 4096

Global Pooling

- We covered MaxPool in CNNs, which looks at the maximum value in a local neighborhood
- We can consider an alternative way to do MaxPool, which looks at the entire feature for all inputs and returns a single max value

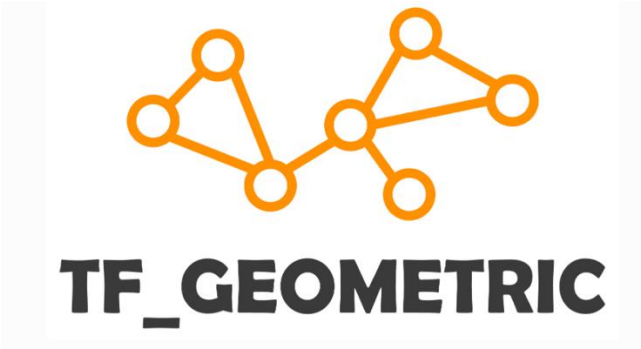


MaxPooling loses lots of data from each channel… AveragePool becomes more common when performing global pooling operations

Implementing GNNs

Best Resources: PyG
- Pytorch Geometric library
- Lots of industry users and well supported models and datasets

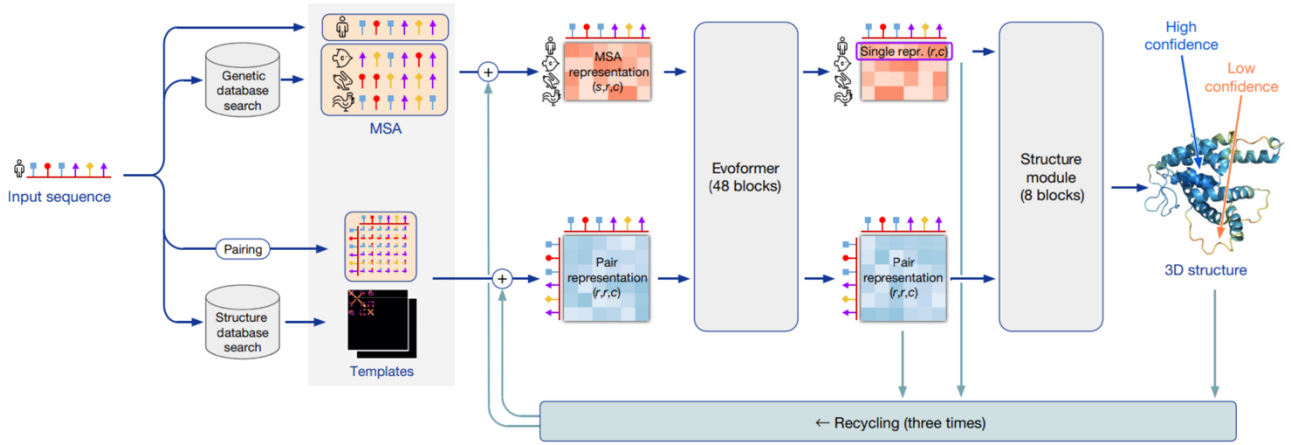
Tensorflow Version: tf_geometric
- Not as many users, but if you really like tensorflow, maybe it's for you



SAGEConv(in_channels, out_channels)

AlphaFold

- Goal: Given amino acid sequence, predict folding structure of protein
- Encodes each amino acid into a node and connects adjacent amino acids
- Alpha Fold 1 uses convolutions, Alpha Fold 2 uses attention (coming soon to a lecture near you)



Text Data

- We can apply convolutions to text data as well
  - Convert each word into a vector of features
  - Perform message passing with adjacent words
  - Figure out how to get an output…
- Perhaps there are better ways…

Next Topic: Recurrent Networks and Sequences/Language modelling

Recap

Geometric Deep Learning

- Motivation: Data on manifolds
- Connections between discrete graphs and continuous spaces

GNNs

- Graph Convolutions
- GraphSAGE