Graph Neural Networks

Shah Rukh Qasim
Department of Mathematical Modeling and Machine Learning
&&
Physik Institut
University of Zurich

shahrukh.qasim@physik.uzh.ch

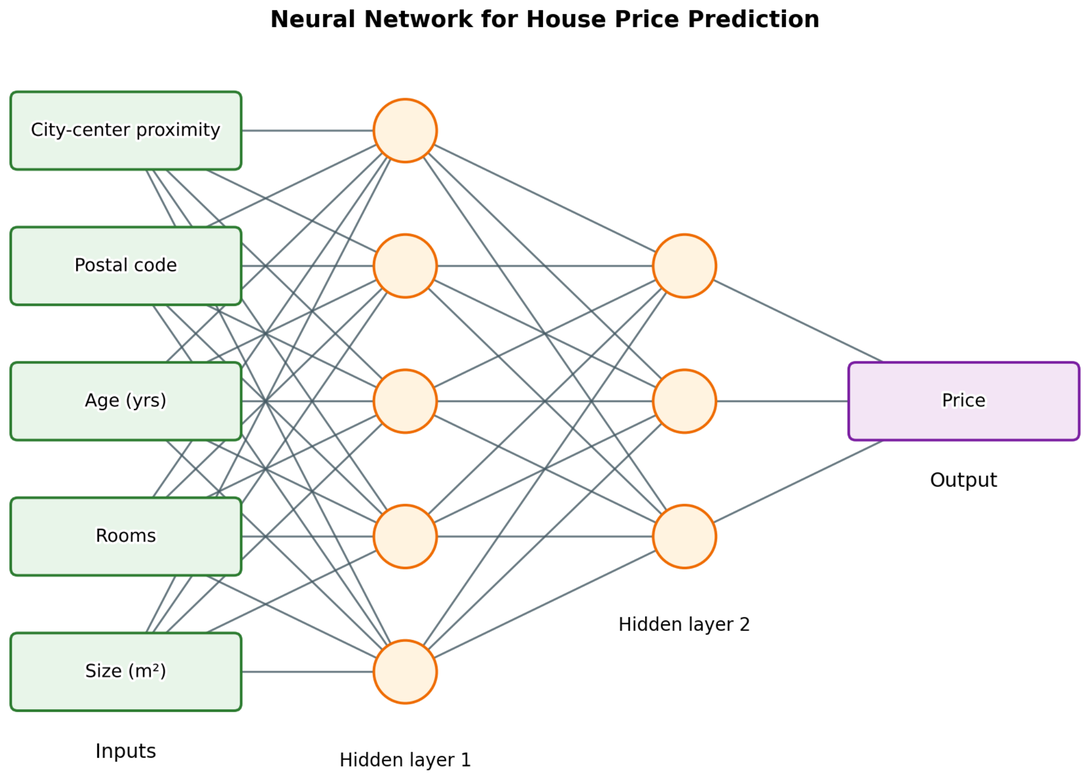
21.10.2025

Today's Agenda

- Graph Neural Networks (GNNs)
  - Part 1 (Lecture)
    - Why GNNs?
    - Modern GNN / with message passing
      - You will be able to get an understanding to be able to use GNNs
    - And then we'll have a little bit of a history lesson
    - And possibly discuss some advanced topics as well
- Part 2 (Tutorial)
  - We'll train a GNN for molecule classification on collab
  - The notebook is simple but contains all the barebones of modern GNNs so you can also adapt the code from there for your future graph problem

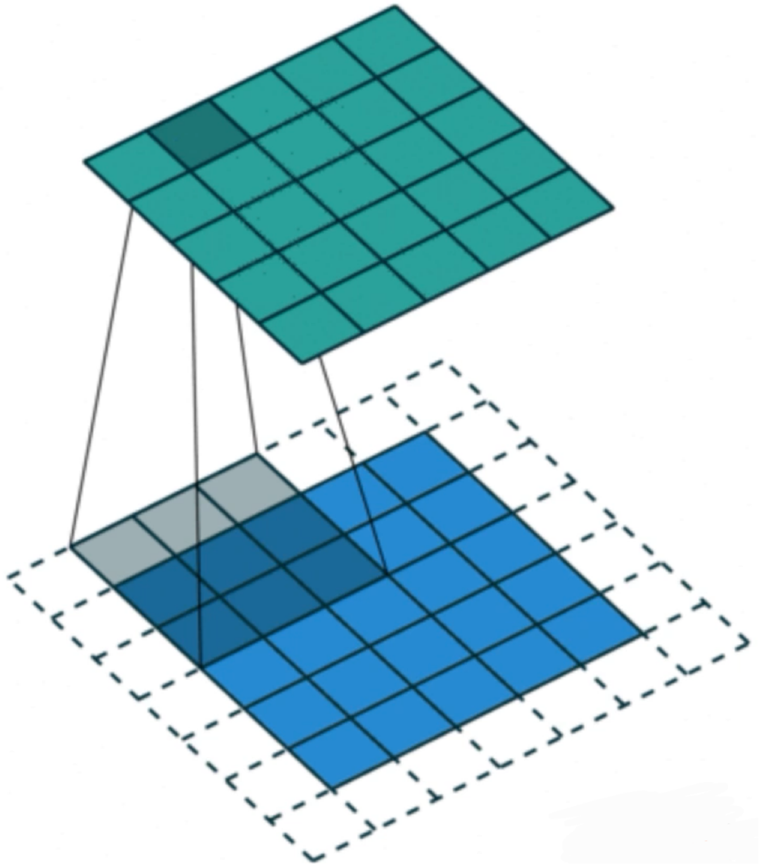
Why Graph Neural Networks (GNNs)?

- Let’s say we have a simple example on the right side
  - That’s the very basic fully connected neural network
- Still widely used and is very simple



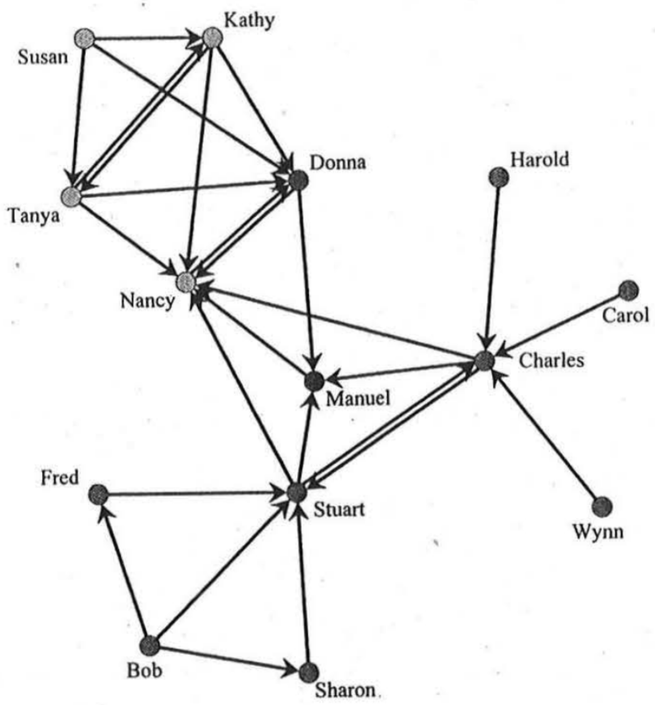
Why Graph Neural Networks (GNNs)?

- Let’s say now we are working with images
  - A classification task: Is that image of a cat or a dog?
  - A low res image would be 1000x1000 : 10^6 input dimensions
    - 10^6 * 10^6 => too big
- So we use Convolutional Neural Networks (CNNs)
  - Make use of the structure of the problem

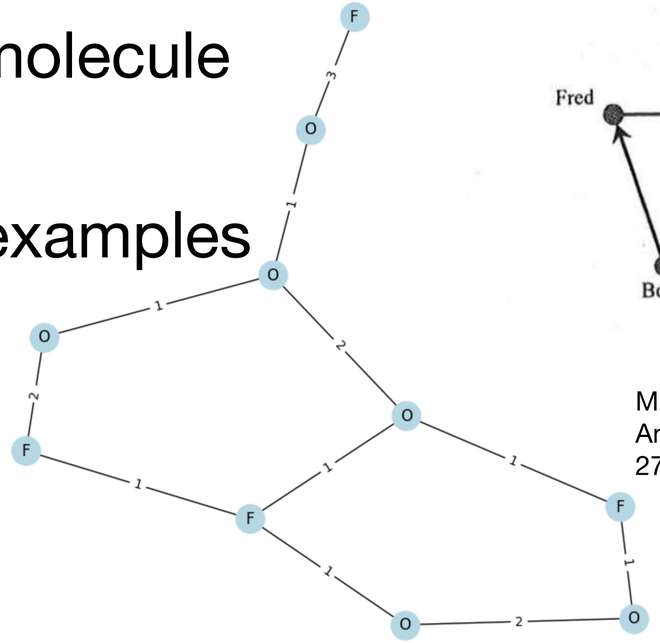


Why Graph Neural Networks (GNNs)?

- Let’s now complicate it a little bit more, we are working with for example:
  - Social Networks (predicting possible friends on Facebook)
  - Molecular Graphs (predicting if a molecule is cancerous or not)
- You can take a screenshots of your examples and apply a CNN
  - But perhaps we can do better



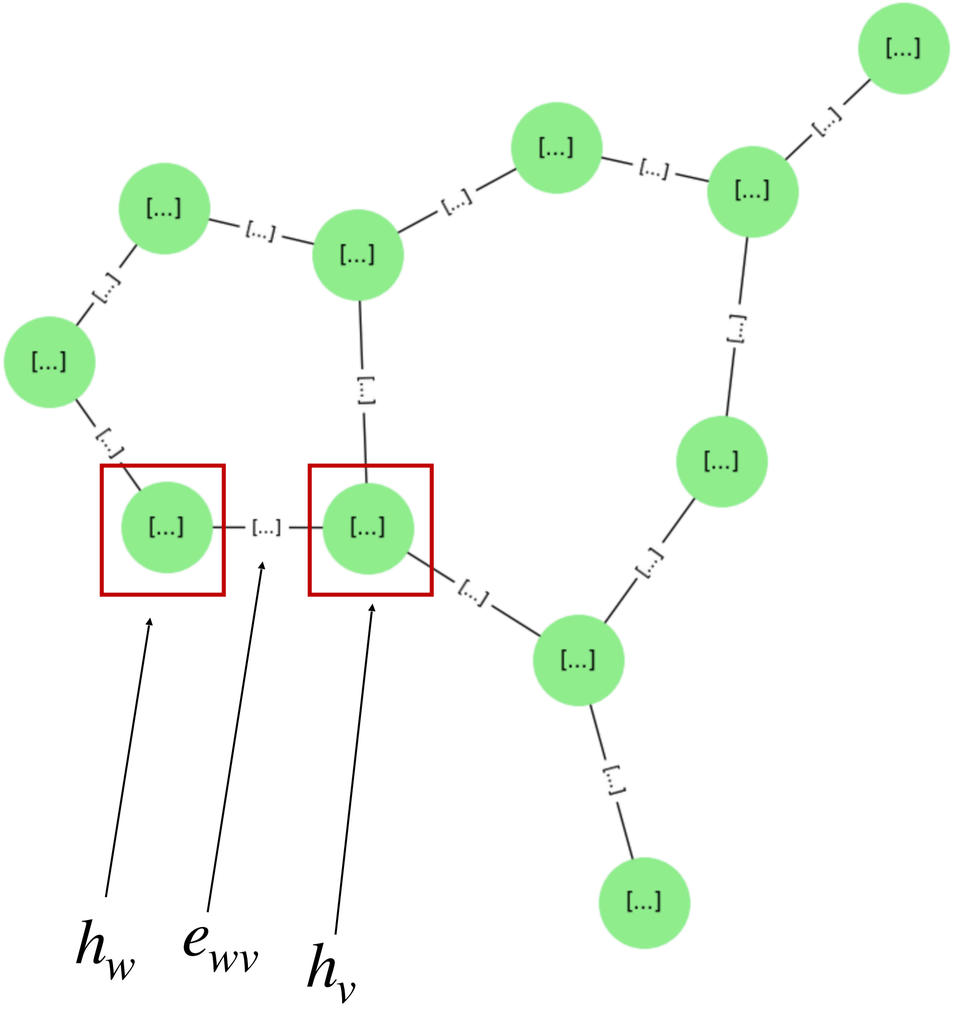
Mitchell, J. Clyde. "Social networks." Annual review of anthropology 3 (1974): 279-299.



Example of a Molecular Graph

Graphs

- Represent the data as graphs
- And use try to use neural networks on graphs
  - That's a Graph Neural Network
- Graph: $G = (V, E)$
  - Nodes / vertices (V): relate to an "object" in your data
  - Edges (E): captures the relationship between the nodes
- Using NNs, we want to have features:
  - Each vertex's features is a vector (h_w, h_v)
    - For example, the age, gender of a person in social networks
  - Same as edges (e_{wv}) => not necessary
- Graphs are generic: Everything is a graph
  - A set is a graph
  - An image is a graph
  - A sequential time datum is a graph

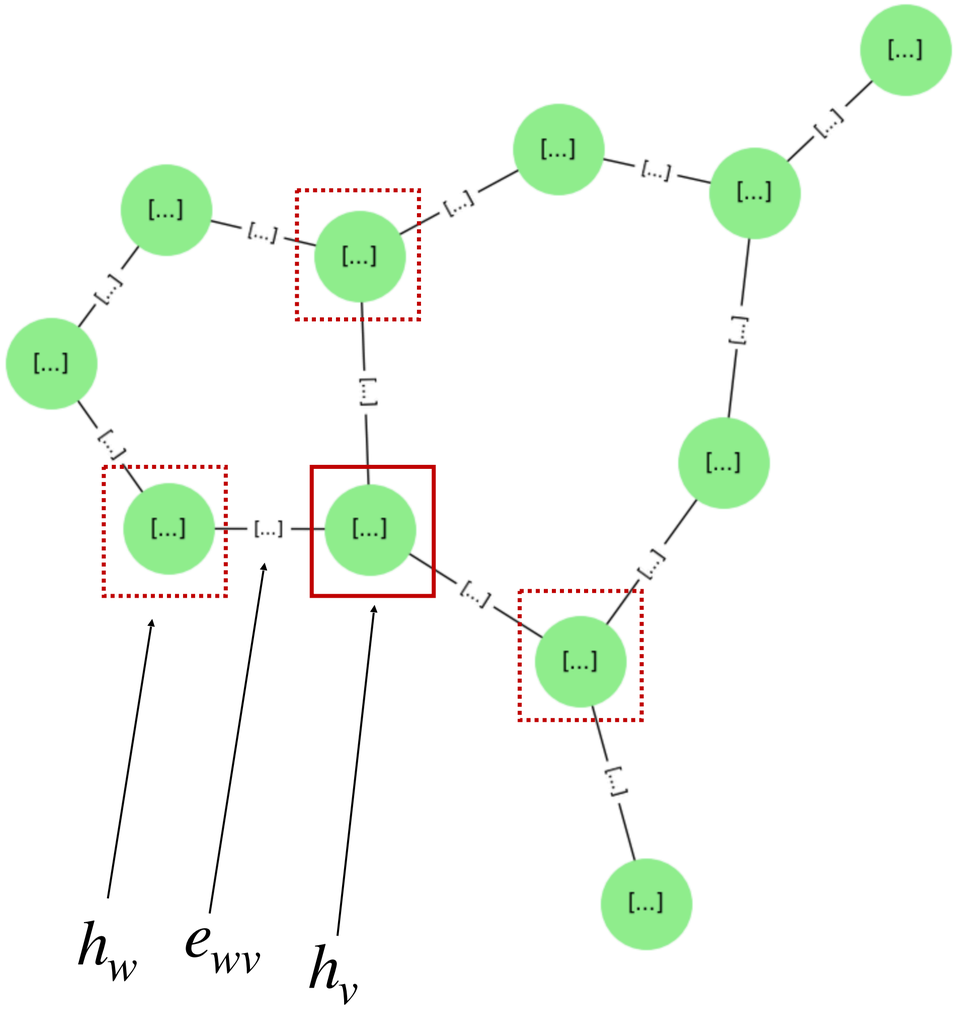


Graph Message Passing

- The core of modern GNNs is what's called Message Passing
  - The most important concept to understand
  - We'll adopt the notation from:
    Gilmer, Justin, et al. "Neural message passing for quantum chemistry." International conference on machine learning. PMLR, 2017.
- Sending messages
  - Messages are derived from features / hidden features of every node

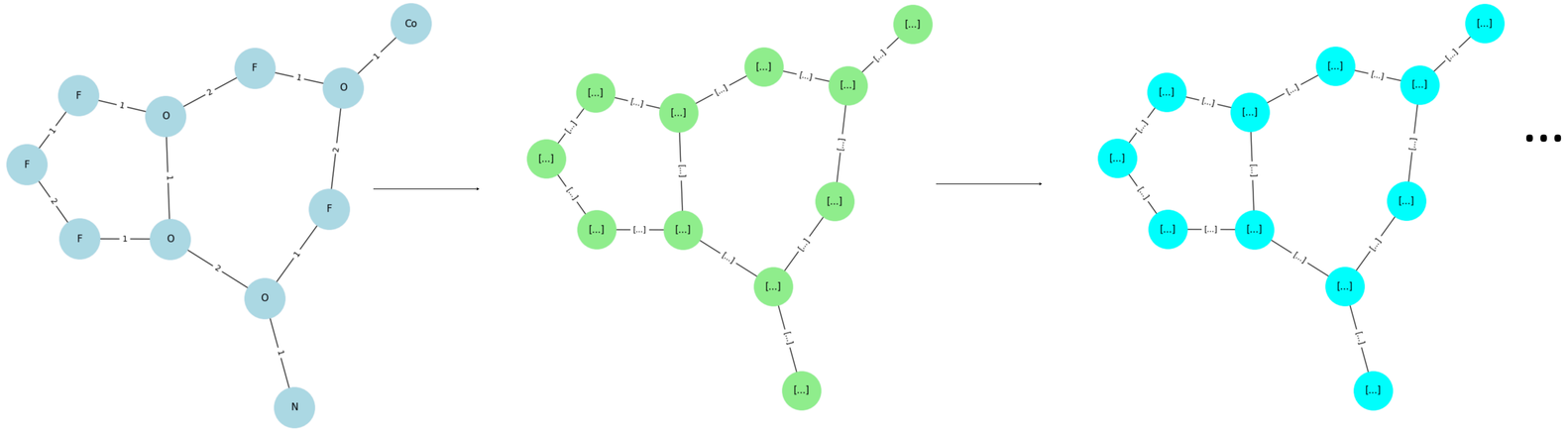
Graph Message Passing

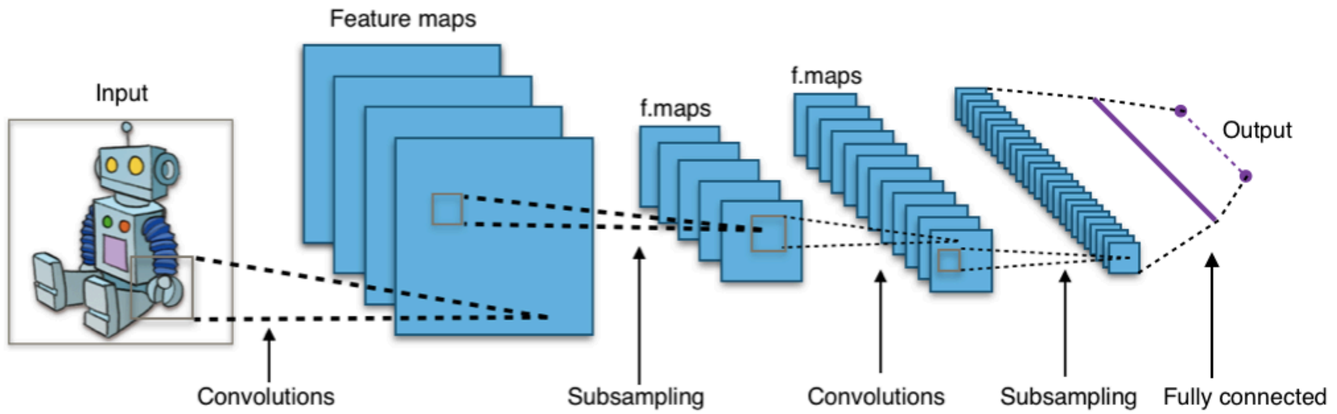
- A message from w to v:
  $m_{wv}^t = M_t(h_w^t, h_v^t, e_{vw})$
- And then v will aggregate from all it's neighbors
  - How?
  - Sum
    $m_v^{t+1} = \sum_{w \in N(v)} M_t(h_w^t, h_v^t, e_{wv})$
- Can also use min / max
  - Important: Ideally permutation invariant
- Update function:
  $h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$



Graph Neural Networks

- Similar to CNNs
- We do some message passing
  - and get another set of features
- And again
- And again
- The Message and Update functions are different across layers
  - Same As CNNs
- The graph structure remains the same
  - (In general)





How to train them?

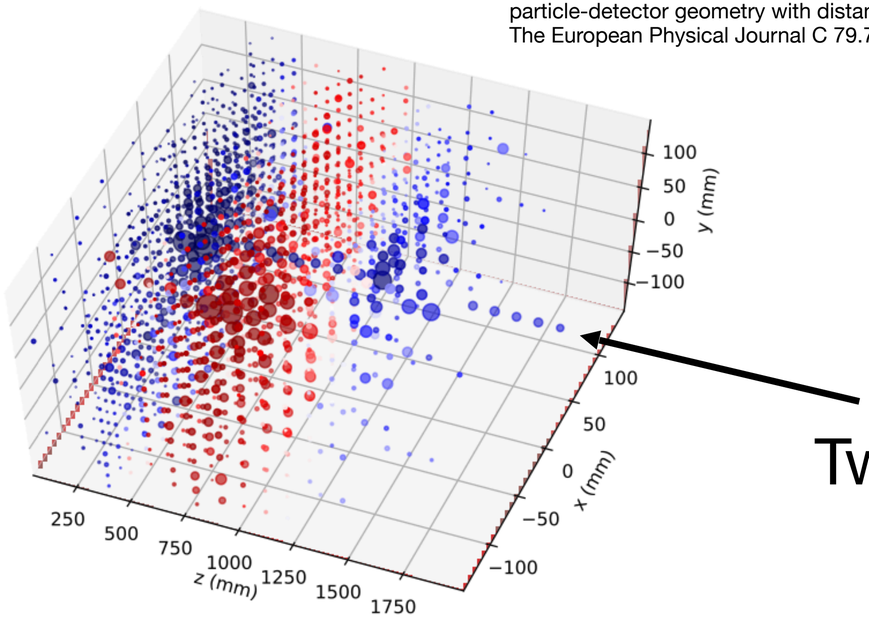
- Depends on the problem
- **(A)** Graph level task
  - Example: Molecular graph classification
  - Pool information from all the nodes
  - Mean / Min / Max

- Everything else is then the same
  - The same old cross entropy loss
  - Same old training loop with Adam optimizer

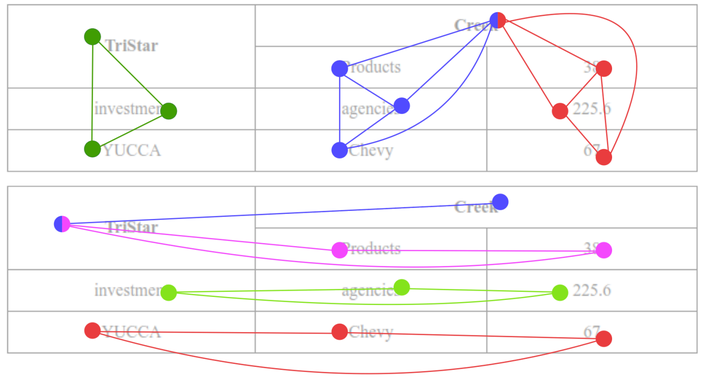
GNN Tasks

- Node level
  - Output: the feature vector at the last layer of your GNN
  - Example: segmentation
- Edge level
  - Don’t pool information from the graph
  - Use the output at every edge
  - If the edge features don’t exist, do subtraction or concatenation
  - Example: In a social network, if two people are friends or not

Qasim, Shah Rukh, et al. "Learning representations of irregular particle-detector geometry with distance-weighted graph networks." The European Physical Journal C 79.7 (2019): 1-11.



Two particles



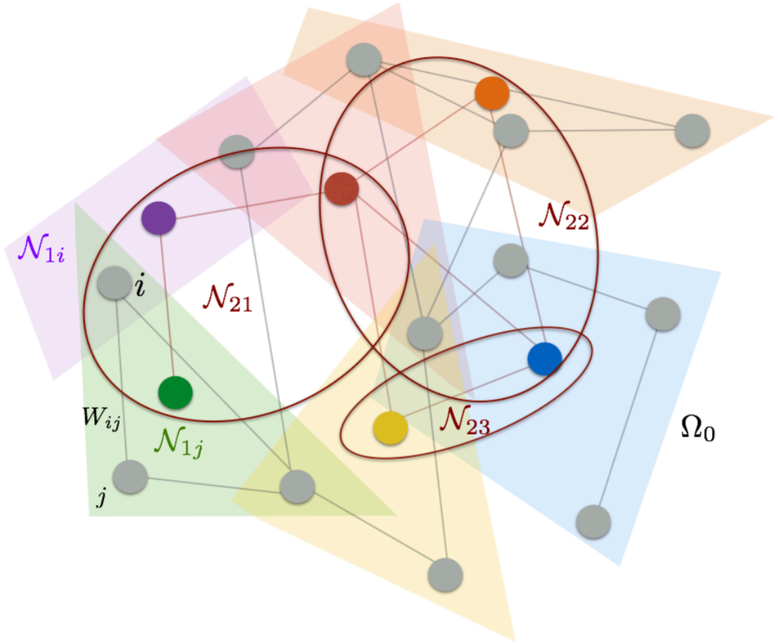
Qasim, Shah Rukh, Hassan Mahmood, and Faisal Shafait. "Rethinking table recognition using graph neural networks." 2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019.

GNNs in Practice: Storage

- Node features `x`: shape `[N, F]`
- Edges edge_index: shape `[2, E]`, `[src, dst]` integer indices (0-based)
- Edge features `edge_attr`: `[E, D]`
- Batch bundling
  - Nodes: `x_batch = torch.cat([x¹, x², ...], dim=0)` → shape `[ΣN_g, F]`
  - Edges: `edge_index_batch = torch.cat([edge¹ + offset¹, edge² + offset², ...], dim=1)`
  - Each `offset^g` = cumulative sum of previous `N_g` to shift node ids
  - Batch vector `B`: shape `[N]`
- PyG provides functions to bundle and unbundle

History: Starts with Spectral Networks

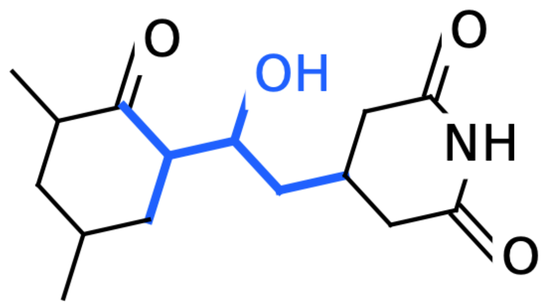
- Yan LeCun et al. paper
- They tried to generalise CNNs beyond the regular array dataset paradigm
- They replaced the translation-invariant kernel structure of CNNs with hierarchical clustering
- And, based on spectrum of the graph
- Taken from the adjacency matrix



Bruna, Joan, et al. "Spectral networks and locally connected networks on graphs." arXiv preprint arXiv:1312.6203 (2013).

History: Message Passing

- Traced back to
  Duvenaud, David K., et al. "Convolutional networks on graphs for learning molecular fingerprints." Advances in neural information processing systems 28 (2015).
- Introduced “convolutional neural network that operates directly on graphs”
- Different language, similar idea

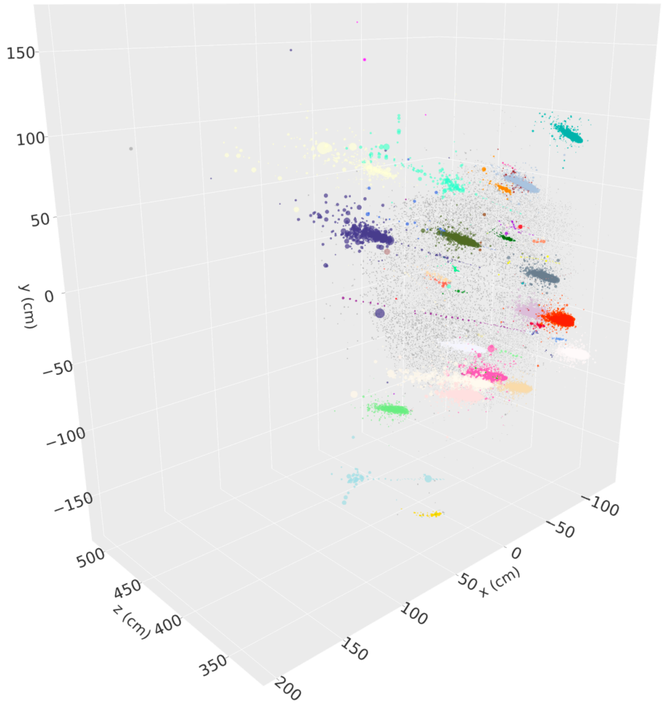


Algorithm 2 Neural graph fingerprints

Input: molecule, radius R, hidden weights $H_1^1 \dots H_R^5$, output weights $W_1 \dots W_R$
Initialize: fingerprint vector $\mathbf{f} \leftarrow \mathbf{0}_S$
for each atom a in molecule
  $\mathbf{r}_a \leftarrow g(a)$ ▷ lookup atom features
for $L = 1$ to R ▷ for each layer
  for each atom a in molecule
    $\mathbf{r}_1 \dots \mathbf{r}_N = \text{neighbors}(a)$
    $\mathbf{v} \leftarrow \mathbf{r}_a + \sum_{i=1}^{N} \mathbf{r}_i$ ▷ sum
    $\mathbf{r}_a \leftarrow \sigma(\mathbf{v} H_L^N)$ ▷ smooth function
    $\mathbf{i} \leftarrow \text{softmax}(\mathbf{r}_a W_L)$ ▷ sparsify
    $\mathbf{f} \leftarrow \mathbf{f} + \mathbf{i}$ ▷ add to fingerprint
Return: real-valued vector $\mathbf{f}$

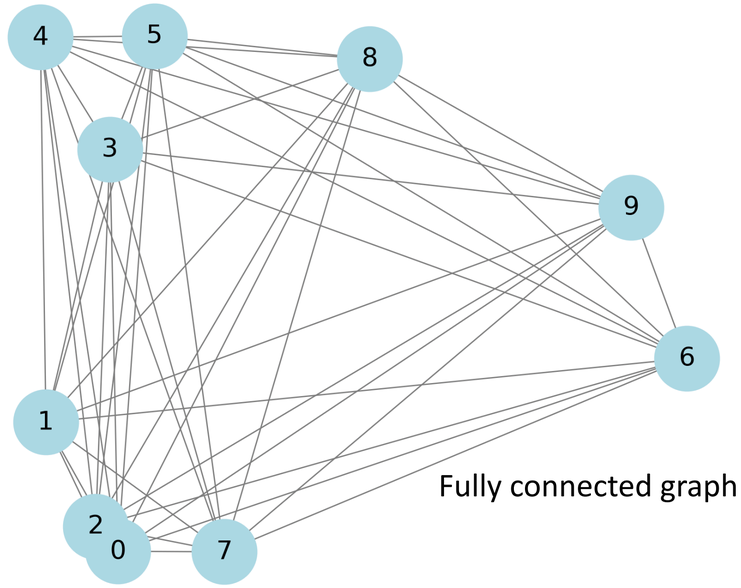
Point Clouds

- If the data is non structured as a bunch of points
  - aka a point cloud (or a set)
  - No graph structure defined
  - Very important for science, especially for physics
  - A data from a calorimeter or a tracker doesn’t have a graph defined
  - A set of particles in a jet don’t have a graph defined
- How do I apply machine learning here?

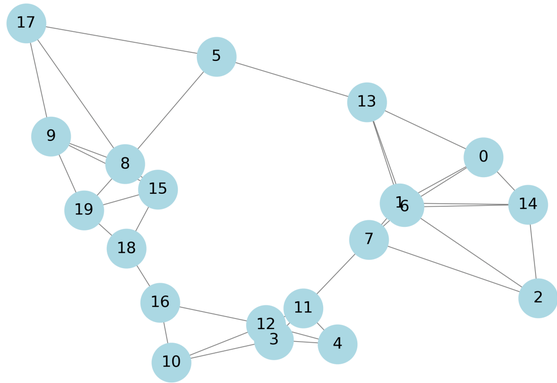


Point Clouds

- Answer 1:
  - Connect everything to everything
  - Will not scale!
- Answer 2:
  - Build a graph locally in
  - Radially in the input domain…
  - Or k Nearest Neighbour (kNN)



Fully connected graph



kNN graph

Dynamic GNNs

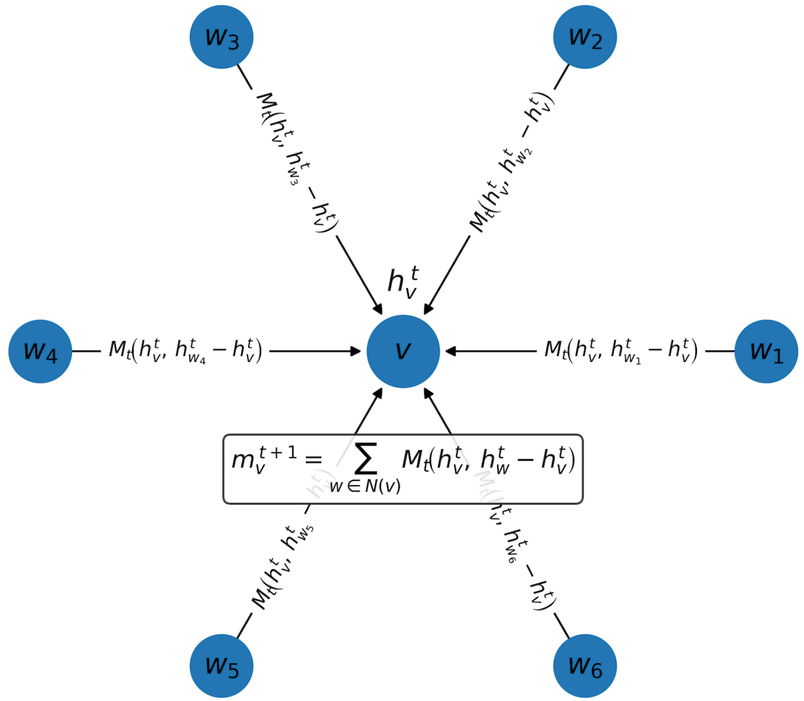
- Build the graph in a dynamically learned space
- Feature vector for every point
- A neural network to feature vectors of every point to get some latent space h_v
- Build a graph as KNN of every node using euclidean distance in h_v
  - So you have N*K edges
  - Perform message passing with permutation invariant aggregation functions
  - Dynamic GNN:
    - The KNN graph isn't built in the original space but in a learned feature space
    - (Breaking the static graph topology rule here)

Dynamic GNNs: Edge Conv

- kNN graph is built in h_t
  - Different than previous method: the graph changes across layers
- And then the next set of hidden features are constructed as:

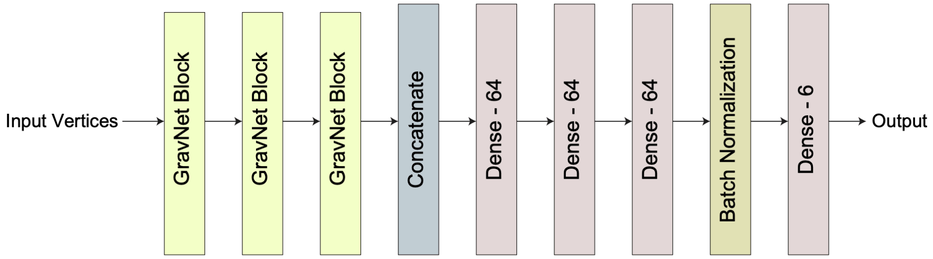
$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t - h_v^t)$$

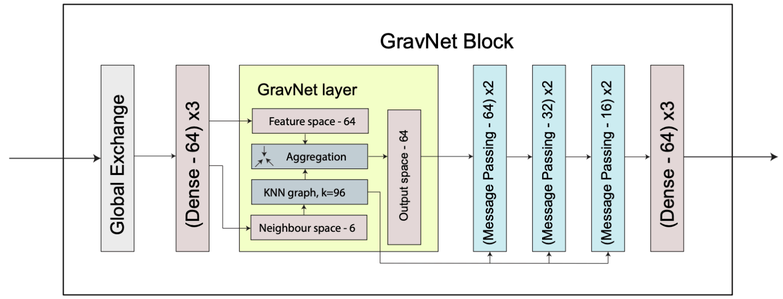
- Same story: Do it a bunch of times to build your GNN
- Useful observation: Use subtraction if you don't have edge features! (Even in non dynamic GNNs)



Wang, Yue, et al. "Dynamic graph cnn for learning on point clouds." ACM Transactions on Graphics (tog) 38.5 (2019): 1-12.

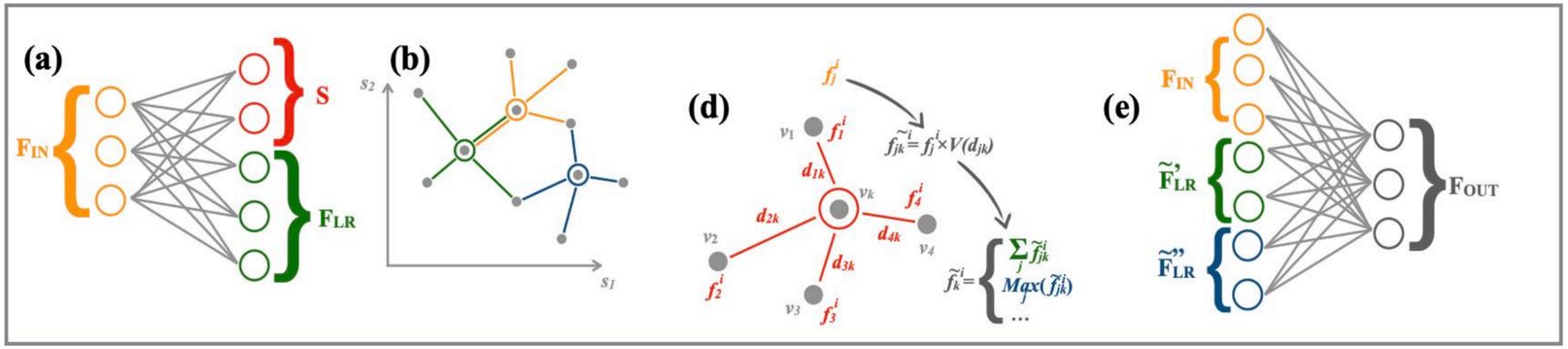
Dynamic GNNs: GravNet





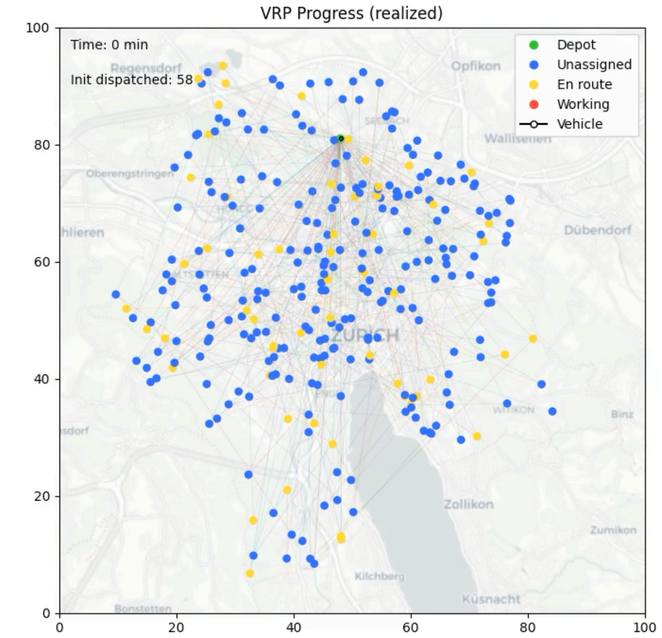
- Edge Conv:
  - You build the graph in the feature space at step t
  - Hope that the graph is good
    - Works very well in practice
- Can we do better?
  - At every layer, transform the features into two different spaces
    - Coordinate space ($_S$) – Low dimensional
    - Feature space ($_{F_{LR}}$) – Higher dimensional
  - $_{F_{LR}}$ is weighed by (neg exp. of) distance between nodes in $_S$
  - In this way, the coordinate space is learned such that the nodes are brought closer if it helps the gradient
- Also much more performant:
  - kNN building is very expensive: dominates the time taken
  - Much faster in smaller dimensions

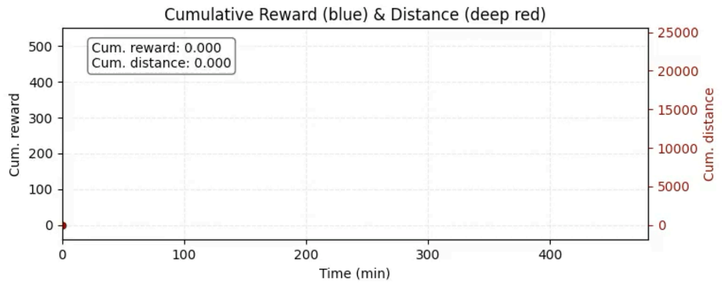
Qasim, Shah Rukh, et al. "Learning representations of irregular particle-detector geometry with distance-weighted graph networks." The European Physical Journal C 79.7 (2019): 1-11.



Point Clouds are Important

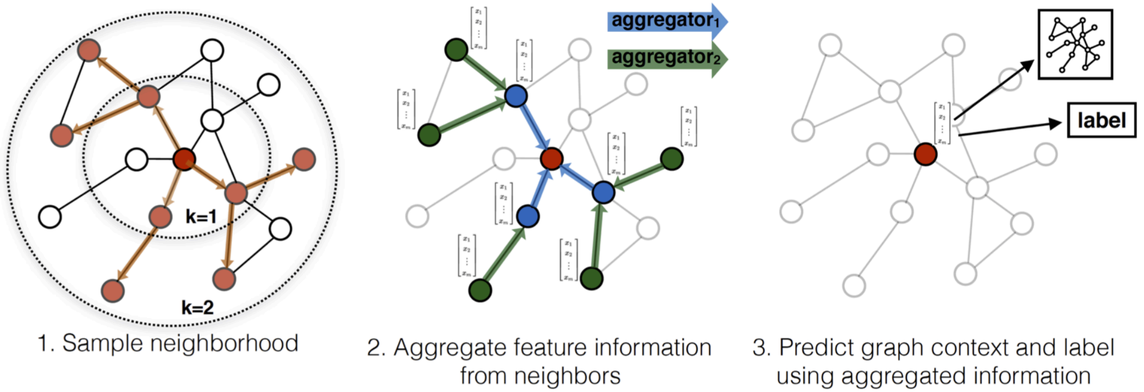
- Fluid dynamics: nodes = spatial points, edges = learned interactions.
- High-energy physics: reconstruct event interactions from particle clouds
- Social networks: maybe the person has more possible friends
- Routing problems





GNNs at Scale

- GraphSAGE (SAmple and AGgregate)
  Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." Advances in neural information processing systems 30 (2017).
- Sample in the neighborhood
- And then do the aggregation
- Here, the authors also explored LSTMs
  - Not permutation invariant though
- Applied on large graphs: citation and reddit graphs



1. Sample neighborhood

2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

Heterogeneous GNNs

- Common for graph problems to have different types of nodes and edges
  - E.g. publications
    - Nodes: author, paper, venue
    - Edge types: 1. paper cites paper, 2. author writes paper, 3. paper published in

- Can be represented like this roughly:
  - $G = (V, E, T_V, T_E)$

Summary

- GNNs are powerful
- And now also very popular
- Transformers overshadow everything
  - But in a lot of cases, GNNs make more sense
- Have been applied to a large sets of problems across multiple domains

- Email for consultations regarding graph problems:
  - shahrukh.qasim@physik.uzh.ch

Part 2: Hands on Tutorial

Find the Collab Notebook

- https://colab.research.google.com/drive/1bdf4cqDgQrBKOJn2v8GEabHW4gpPrOk7?usp=sharing

- https://tinyurl.com/mr3vsmvt

- It won’t save
  - Make a copy!

Intro to PyTorch

- PyTorch: PyTorchGeometric / PyG: https://pyg.org/
  - Academia standard
- TensorFlow: TF-GNN
  - Tried this last year but it’s disastrously bad for introducing new people
  - Can use tools like keras here
- Deep Graph Library (DGL): Framework agnostic

- I implemented a barebones dummy CNN for you
- The training loop is already implemented so you don’t need to worry about it
- The difference is that we define the layers with weights in the constructor and then call them in forward (other functions)
- You can find a PyTorch version of all other layers in Keras (Google it)

Intro to NetworkX

- NetworkX is a Python library for creating and analyzing graphs and networks
- Also provides tools for visualization with matplotlib
- Has a bunch of classical graph algorithms (matching, clustering etc)
- We are going to use it to plot our graphs

Implement the Model

- Implement the message passing layer
- And then you can make a few copies of this layer in your GNN
- Similar to CNNs, you add a couple more dense layers at the end (the classification head)

- We are going to spend the first 15 minutes in implementing the MPNN Layer
- And then 15 minutes in implementing the GNN
- And then we iterate