GRAPH-NETS IN HEP: A PRACTICAL TUTORIAL

Rute Pedro & Giles Strong

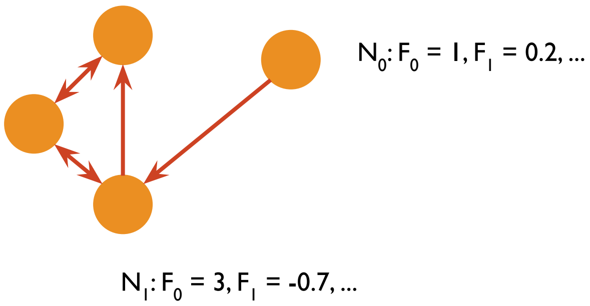
LIP Big Data 1st thematic workshop, Online - 12/07/21

OVERVIEW

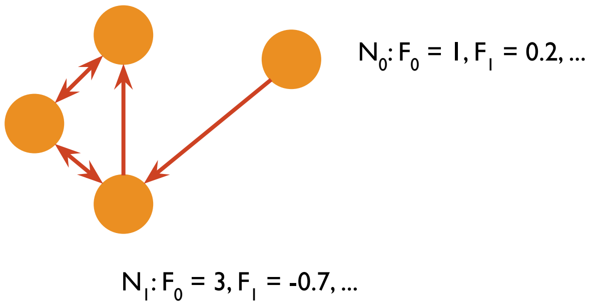
1. Graph data and tasks
2. Graph examples in HEP
3. A few architectures
4. Practical tutorial

GRAPHS

- A *graph* consists of:
  - *Nodes/Vertices*, each with a set of features
  - *Edges* which connect vertices together and imply an interaction between the pair of nodes
    - Edges can be unidirectional and not all possible edges have to exist
- Twitter example:
  - People (nodes) with tweets (features) connected by both uni- and bi-directional edges (following & mutual follows)
  - Not everyone is connected to everyone else (some edges don't exist)



GRAPHS



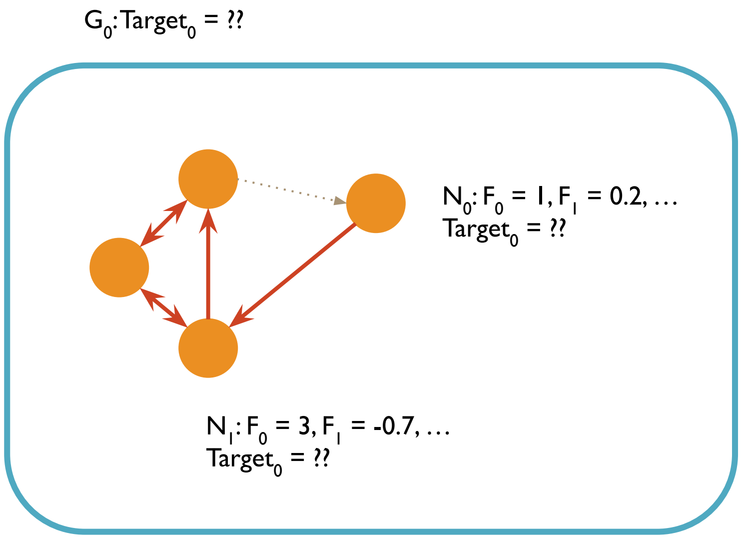
- Can represent graph as:
  - A matrix of nodes with features (N, F) or (F, N)
  - An (N,N) adjacency matrix of connections between nodes
  - Nodes are also connected to themselves

	F		
N	1	0.2	…
	3	-0.7	…
	…	…	…

	Receive			
Send	1	1	0	0
	0	1	1	1
	0	1	1	1
	0	0	1	1

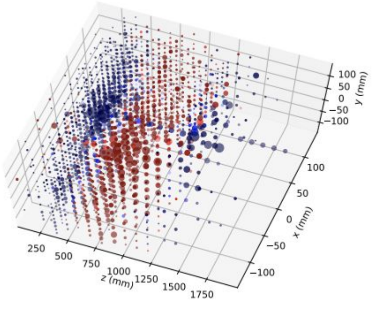
GRAPH-TASKS

- Three general task categories
- Node-level predictions:
  - Given a graph, predict the values of unknown features for every node
- Edge-level predictions:
  - Given a set of nodes, predict whether each possible edge-connection exists
- Graph-level predictions:
  - Given a graph, predict the values of unknown features for the entire graph

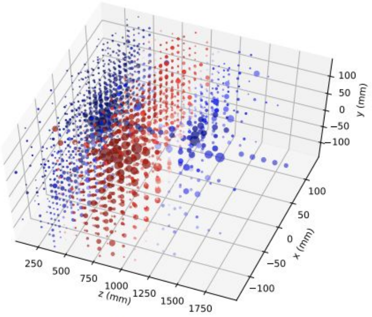


HEP EXAMPLES

- Generally in HEP, assume every node is connected to every other node
  - All edges exist and are bi-directional
- Node-level predictions: Assign detector hits to different showers (e.g. Qasim et al. 2019, right)
  - Hits are nodes with energy & position features
  - Graph is an event
- Graph-level predictions: Jet tagging (e.g. Qu & Gouskos 2019)
  - Particles are nodes with 4-momentum
  - Graph is a jet
- Hybrid: Particle flow (e.g. Kieseler 2020) - assign hits to objects and predict properties of objects



(a) Truth



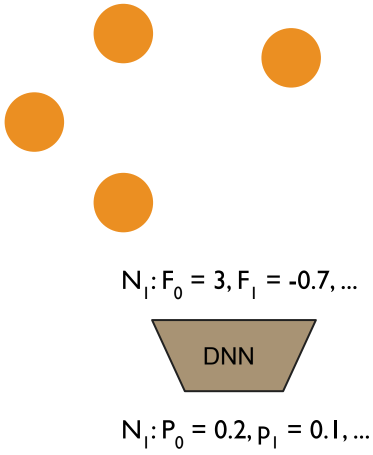
(b) Reconstructed

ADVANTAGES OF GRAPHS

- No assumed regularity of node positions:
  - CNNs rely on grid-layout of pixels, but what if you detector has an irregular layout?
  - Node features allow us to specify node position (or learn a latent-space embedding)
- No assumed ordering of nodes:
  - DNNs, CNNs, & RNNs require inputs to be ordered somehow
  - For some domains ordering is intuitive (start at top-left of image, read text in word-order, etc.)
  - But in HEP recorded particles *exist simultaneously* but we must order them by a criterion and hope it is optimal
  - Graphs also present all nodes simultaneously with no sense of priority
  - N.B. For graph-level predictions, care must be take to retain order-invariance (see next slide)
- Graphs are flexible: nodes and edges can be created or destroyed

SIMPLE APPROACH TO GRAPHS

- Take a single DNN and apply it to every node in the graph:
  - Inputs are node features
  - DNN weights are shared like a CNN
  - Provides set of predictions per node
- Node-level tasks:
  - The DNN predictions are the target features for the node

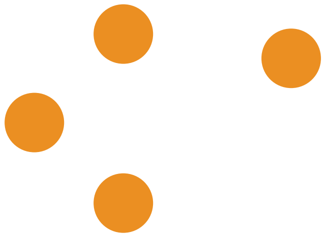


Apply same DNN to every node

SIMPLE APPROACH TO GRAPHS

- Graph-level predictions:
  - Aggregate the node predictions, either:
    - Take the average/maximum of every node prediction - retains order invariance but loses information
    - Reshape node predictions - requires nodes to be ordered but retains information
  - Feed aggregate features through second DNN to get graph-level prediction

N_0: $P_0 = -0.5$, $p_1 = 0.8$, ...



N_1: $P_0 = 0.2$, $p_1 = 0.1$, ...

Average nodes

$\langle p_0 \rangle = -0.3$, $\langle p_1 \rangle = 0.2$, ...

G_0: $P_0 = 0.7$, $p_1 = -0.4$, ...

Flatten nodes

$p_{0,0}=-0.5, p_{0,1}=0.8, \ldots, p_{1,0}=0.2, p_{1,1}=0.1, \ldots$

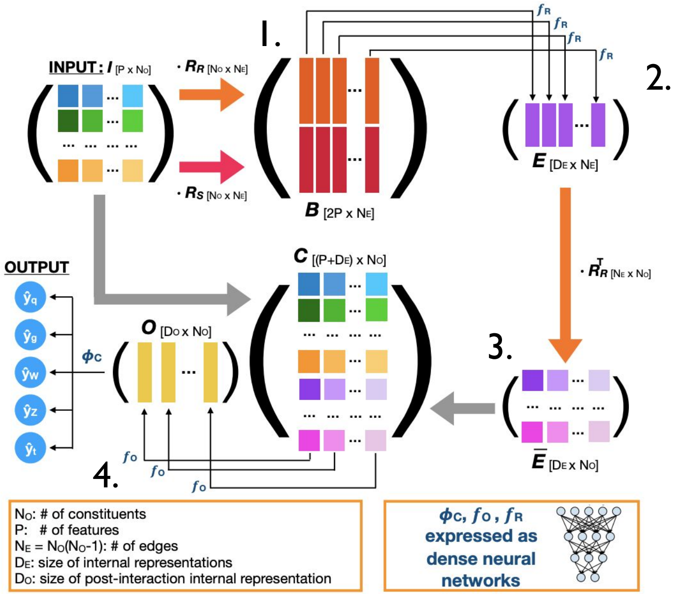
G_0: $P_0 = 0.7$, $p_1 = -0.4$, ...

GRAPH-NETS

- The simple approach works but ignores the connections to other nodes in the graph
- A Graph Neural Network still provides predictions per node, but also has a mechanism to consider the features of the other nodes when predicting each node
  - The mechanism (*message passing*) varies according to GNN architecture

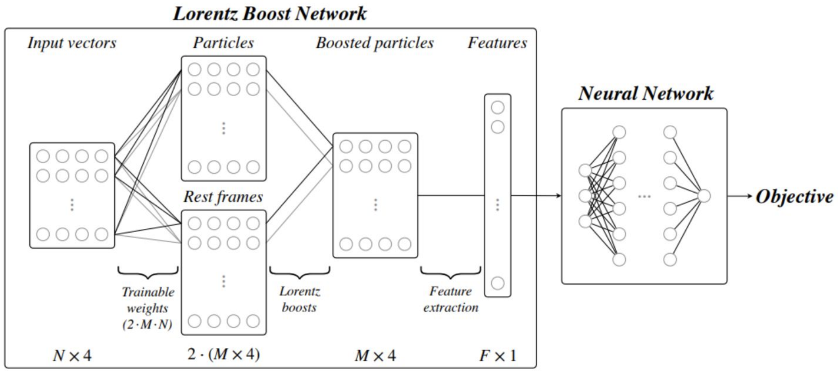
INTERACTION NET

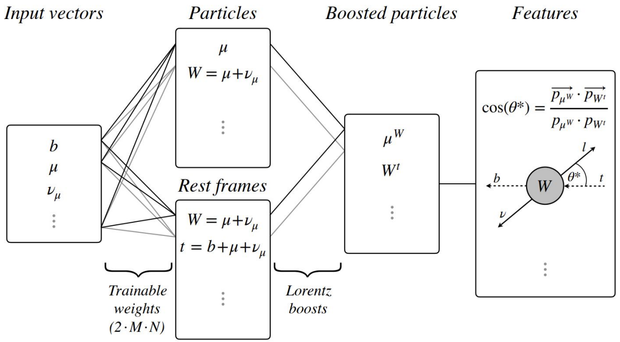
- Originally for physical simulations (Battaglia et al. 2016)
  - Applied to HEP by Moreno et al. 2019
- 1. Combine features along edges
  - Implemented by fixed sending & receiving matrices
- 2. Apply DNN to learn internal transformation for each node
- 3. Apply transformation to each node and concat with original features
- 4. Apply 2nd DNN to learn output features per node



LORENTZ-BOOST NETWORK

- Erdmann et al., 2018
- HEP-specific arch for learned feature extraction from 4-momenta
  - Creates new boosted particles by learning both new particles and rest-frames by combining input particles
    - Particles & restframes are linear combinations of inputs, with learnable coefficients
  - Computes pre-specified high-level features using (combinations of) the boosted particles
    - Lorentz boost requires inputs are *physical*
- Reduces impact of HL-feature selection/specification by providing means to learn optimal particles for the chosen features
  - LUMIN implementation offers a further relaxation by replacing the fixed feature extraction with a pair of DNNs:
    - One extracts N features each particle
    - The other extracts M features from every combination of particle pairs



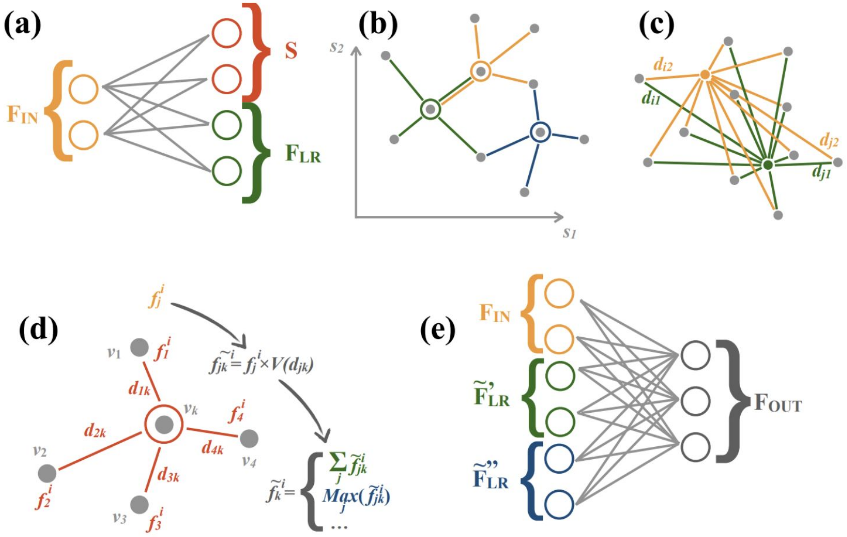


DEALING WITH LARGE GRAPHS

- Typical problem with graphs: slow to evaluate with many nodes
- Greedy approach: prediction of a single node depends on all connected nodes + self
- Heuristic approach: learn which other nodes are actually required

GRAVNET

- Quasim et al., 2019
- A) Initial DNN learns new features (F_{lr})+ latent-space coords (S) per node based on node features (F_{IN})
- B) Graph constructed by only connecting each node to its *k*-nearest neighbours in latent-space (Euclidean separation in S)
- D) Node features (f_j^i) "seen" by a given node are:
  - Weighted by a potential according to Euclidean distance, e.g. exp(-10*d_{jk}^2) ($\tilde{f}_{jk}^i$)
  - Aggregated by order-invariant functions, e.g. average & maximum ($\tilde{f}_k^i$)
  - The neighbour-features are then concatenated with the original features of the node
- E) A second DNN computes the output features per node based on the F_{IN} & $\tilde{f}_k^i$) features



PRACTICAL TUTORIAL - GILES

TOP TAGGING

- Binary classification of jets (0=QCD, 1=Top)
  - Inputs are 4-momenta of 1st 200 sub-jets (p_t ordered)
  - Full details
- For GNN task:
  - Sub-jets are nodes
  - 4-momenta are features
  - Jets are graphs
  - Graph-level classification problem

	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$) single	mean	median	#Param
CNN [25]	0.981	0.930	914±14	995±15	975±18	610k
ResNeXt [40]	0.984	0.936	1122±47	1270±28	1286±31	1.46M
TopoDNN [27]	0.972	0.916	295±5	382± 5	378 ± 8	59k
Multi-body N-subjettiness 6 [33]	0.979	0.922	792±18	798±12	808±13	57k
Multi-body N-subjettiness 8 [33]	0.981	0.929	867±15	918±20	926±18	58k
TreeNiN [52]	0.982	0.933	1025±11	1202±23	1188±24	34k
P-CNN	0.980	0.930	732±24	845±13	834±14	348k
ParticleNet [56]	0.985	0.938	1298±46	1412±45	1393±41	498k
LBN [28]	0.981	0.931	836±17	859±67	966±20	705k
LoLa [31]	0.980	0.929	722±17	768±11	765±11	127k
LDA [63]	0.955	0.892	151±0.4	151.5±0.5	151.7±0.4	184k
Energy Flow Polynomials [30]	0.980	0.932	384			1k
Energy Flow Network [32]	0.979	0.927	633±31	729±13	726±11	82k
Particle Flow Network [32]	0.982	0.932	891±18	1063±21	1052±29	82k
GoaT	0.985	0.939	1368±140		1549±208	35k

RUNNING THE TUTORIAL

- Dedicated software repo: https://github.com/GilesStrong/workshop_LIP_GNN
- Either run-locally, or use Google Colab: https://colab.research.google.com/github/GilesStrong/workshop_LIP_GNN/blob/main/GravNet_for_top_tagging.ipynb
- Subsampled, preprocessed data available from https://cernbox.cern.ch/index.php/s/YsKrkmIM6rBcnfG/download
    - Link will be deactivated on 18/07/21 - afterwards use official source (notebook contains the preprocessing code, but the full dataset is large)

PRACTICAL TUTORIAL - RUTE

HIGGS ML KAGGLE

- ATLAS 2012 MC full simulation with Geant 4
- Signal: Higgs to di-tau
- Backgrounds: Z→ $\tau\tau$, tt, and W decay
- Events selected for the semi-leptonic channel: $\tau\tau$ → (e | μ) + τh
- 250,000 labelled events for training, 550,000 unlabelled events for testing
- 31 features:
  - 3-momenta of main final-states and upto two jets (pT ordered)
  - High-level features: angles, invariant masses, fitted di-tau mass (MMC), et cetera

- Solutions must predict signal or background for each test event
  - Solutions ranked via their Approximate Median Significance
  - Quick, accurate, analytical approximation of full discovery significance
  - s = sum of weights of true positive events (signal events determined by the solution to be signal)
  - b = weights of false positive events (backgrounds events determined by the solution to be signal)
  - br = constant term (set to 10 for the challenge)

$$\text{AMS} = \sqrt{2\,(s + b + b_r)\log\left(\left(1 + \frac{s}{b + b_r} - s\right)\right)}$$

RUNNING THE TUTORIAL

- Colab link: https://colab.research.google.com/drive/1QQyGakWRfFIpALV2dYpq0M3X7bgjxh4v?usp=sharing

FURTHER READING

ASSORTED FURTHER READING

- GNN tutorial + explanation
- Should Graph Neural Networks Use Features, Edges, Or Both?
- JEDI-net
- GravNet
- Object condensation
- Attention is all you need
- Point Cloud Transformers