GRAPH NEURAL NETWORKS FOR HIGH LUMINOSITY TRACK RECONSTRUCTION
EP-IT DATA SCIENCE SEMINAR, CERN, 18 MAY 2022

DANIEL MURNANE
ON BEHALF OF THE EXATRKX AND L2IT PROJECTS
AND THE ATLAS COLLABORATION
HIGH LUMINOSITY TRACK RECONSTRUCTION
WHY HIGH LUMINOSITY PHYSICS?

1. Better reach for Supersymmetry discovery:
   a) Electroweakino particles produced by much greater range of chargino masses
   b) Gluino exclusion from channels across 0.7-2.0TeV to channels across 2.5-3.2TeV

2. Sensitive to resonances (W', Z') up to 6-8TeV

3. W mass precision improvement from $\pm 9.4\text{MeV}$ to $\pm 6\text{MeV}$
In order to perform the analysis that leads to discovery (e.g. of dark matter, extra dimensions, SUSY, ...), need to make sense of the detector read-out.

There are many tasks required to reconstruct the physics event behind the read-out.

- Vertex Reconstruction
- Jet Tagging
- Pile-up Removal
- Missing Energy Reconstruction
In order to perform the analysis that leads to discovery (e.g. of dark matter, extra dimensions, SUSY, ...), need to make sense of the detector read-out.

There are many tasks required to reconstruct the physics event behind the read-out.

These all require accurate track reconstruction.
TRACK RECONSTRUCTION

- Protons collide in center of detector, “shattering” into thousands of particles
- The charged particles travel in curved tracks through detector’s magnetic field (Lorentz force)
- A track is defined by the hits left as energy deposits in the detector material, when the particle interacts with material
- The goal of track reconstruction:

Given set of hits from particles in a detector, assign label(s) to each hit.

Perfect classification: All hits from a particle (and only those hits) share the same label
REPRESENTATION OF COLLISIONS
COMPUTE SCALING FOR HIGH LUMINOSITY

ATLAS Computing Requirements Over Time

<table>
<thead>
<tr>
<th>Year</th>
<th>Annual CPU Consumption [MHS09/years]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020</td>
<td>5</td>
</tr>
<tr>
<td>2022</td>
<td>10</td>
</tr>
<tr>
<td>2024</td>
<td>15</td>
</tr>
<tr>
<td>2026</td>
<td>20</td>
</tr>
<tr>
<td>2028</td>
<td>25</td>
</tr>
<tr>
<td>2030</td>
<td>30</td>
</tr>
<tr>
<td>2032</td>
<td>35</td>
</tr>
<tr>
<td>2034</td>
<td>40</td>
</tr>
<tr>
<td>2036</td>
<td>45</td>
</tr>
</tbody>
</table>

- **ATLAS Preliminary**
- 2022 Computing Model - CPU
- Run 3 (μ=55)
- Run 4 (μ=88-140)
- Run 5 (μ=165-200)

- Conservative R&D
- Aggressive R&D
- Sustained budget model (+10% +20% capacity/year)

ML Image Classification Efficiency Over Time

- 5 million parameters
- 62 million parameters

4x less compute required to get to AlexNet performance 7 years later (linear scale)

CERN-LHCC-2022-005
TEASER: GRAPH-BASED PIPELINE FOR TRACK RECONSTRUCTION

- Using **graph-based ML**, can perform track reconstruction on High Luminosity detector events
- Comparable efficiency and fake rates to traditional algorithms
- Scaling that is approximately linear in event size (on open-source TrackML dataset)
HOW SHOULD WE REPRESENT PARTICLE COLLISIONS?

Assuming we want to use deep learning, how can we represent a particle collision?

Image? Sequence? Set/Point Cloud?

For event collision as point cloud, with relationships between points, this is a graph.

Convolutional Neural Networks with Event Images,... ATLAS Collab.

Particle Track Reconstruction with Deep Learning, Farrell et al

Deep Sets based Neural Networks for Impact Parameter,... ATLAS Collab
WHAT IS A GRAPH?

A COLLECTION OF NODES

NODE
WHAT IS A GRAPH?

AND EDGES

EDGE
WHAT IS A GRAPH?

NODES + EDGES = DOUBLETS
WHAT IS A GRAPH?

NODES CAN HAVE FEATURES

NODE FEATURE
- e.g. “West Oakland”
WHAT IS A GRAPH?

**EDGES CAN HAVE FEATURES**

**EDGE FEATURE**
- e.g. “Under Maintenance – Single Track”
THE WHOLE GRAPH CAN HAVE FEATURES

GRAPH FEATURE e.g. “Sunday Timetable”
GRAPHS ARE A NATURAL WAY TO REPRESENT TRACKS

Given hits on layers of a detector
GRAPHICS ARE A NATURAL WAY TO REPRESENT TRACKS

Connect the hits in some way
Tracks should be found amongst the connected nodes.

Note the trade-off: Rather than needing to classify or cluster nodes with many labels, we only need binary classification of edges.

However, introduce the extra step of building tracks from classified edges.
INTRO TO GRAPH NEURAL NETWORKS
GRAPH NEURAL NETWORK APPLICATIONS

- Travelling Salesman Problem
- Knowledge Graph Comprehension
- Image Comprehension
- Molecular Chemistry
- Protein Comprehension
GRAPH NEURAL NETWORK PROCEDURE

Node features → Encoder → Node channels → Message passing → Node Aggregation → Task output layer → Node channels
**STEP 1: MESSAGE PASSING MECHANISM**

For each node neighborhood:

a) Pass node channels through a multi-layer perceptron (MLP) encoder

b) Pass encoded channels along each edge to the central node of the neighborhood

**Note:** This is quite inexpensive since we store $N_{\text{nodes}}$ for backpropagation

---

Figure inspired by [Koshi et. al.](#)
STEP 2: AGGREGATION

At each node:
Sum all messages

Note: Called isotropic message passing. Introduced as “Graph Convolution Network”

Figure inspired by Koshi et. al.
**EDGE CHANNELS**

- Isotropic message passing can’t differentiate importance of neighbors
- Anisotropic message passing: encode a combination of node and neighbor along each edge
- Much more expensive – now need to store $N_{edges}$ for backpropagation
- But much more powerful

Found in “Graph Attention Network” and “Interaction Network”
**EDGE CHANNELS**

- Can access *contextual* relationships
THE LANDSCAPE OF GNNS

- RNNs
- Transformers
- CNNs
- Cellular Automata
- Graphical Automata
- MLP + 1-pixel Equivalency
- Grid adjacency
- Fully-Connected
- Linearly Connected
- = reduces to

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.554.4395&rep=rep1&type=pdf
GNNS ELSEWHERE IN PARTICLE PHYSICS

- Very large and active field of study!

High Lumi Generic Tracking
High Lumi CMS Calorimetry
LArTPC Particle Reconstruction
FPGA-based Track Reconstruction
Quantum Track Reconstruction
GRAPH-BASED TRACK RECONSTRUCTION
WHO IS INVOLVED?

- Two groups worked on the results in this presentation, and both first tested methods on TrackML, based on the GNN-based reconstruction introduced in arxiv:2003.11603
- **L2IT**: Laboratoire des deux Infinis, institute based at the University of Toulouse, within the Institute of Nuclear Physics and Particle Physics
- **Exa.Trkx**: A DoE Office of Science-funded collaboration of LBNL, Caltech, FNAL, SLAC and a collaboration of US institutions including Cincinnati, Princeton, Urbana-Champaign, Youngstown State, and others
The goal of track reconstruction:
Given set of hits in a detector from particles, assign label(s) to each hit.
Perfect classification: All hits from a particle (and only those hits) share the same label

What does it mean to represent an event with a graph?
- Treat each hit as a node
- A node can have features (e.g. position, energy deposit, etc.)
- Nodes can be connected by edges, that represent the possibility of belonging to the same track

Goal: Use ML and/or graph techniques to segment or cluster the nodes to match particle tracks
Proof-of-concept: TrackML community challenge dataset with simplified simulation
PIPELINE OVERVIEW

- Current pipeline of the L2IT-Exatrkx collaborative effort
- Each stage offers multiple independent choices, depending on hardware and time constraints
DATASETS

- Two datasets used to study this pipeline. For absolute clarity, when citing a result specific to one dataset, will place the badge of TrackML or ATLAS ITk on slide:

  ![TrackML Badge](image)
  ![ATLAS ITk Badge](image)

- Mean number of spacepoints: 110k
- Simplified simulation: No secondaries and optimistic charge information

- Mean number of spacepoints: 310k
- Full simulation
**ATLAS ITk GEOMETRY**

- **Generation script*** using Athena, $t\bar{t}$ at $\mu = \langle 200 \rangle$: with statistics dominated by soft interactions

- ITk consists of barrel and endcap, each with pixels and strips:

  - **Spacepoints** (3D representations of track hits) are defined depending on strip or pixel:

    - Strip: Each spacepoint maps to two clusters – one on either side of strip, which can map to many particles

    - Pixel is trivial: Each spacepoint maps to one cluster, which can map to many particles

- **0**: Pixel barrel
- **1**: Pixel endcap
- **2**: Strip barrel
- **3**: Strip endcap

*Thanks Noemi Calace*
Fiducial particles are \textit{charged}, with \( \eta \in \left[ -4, 4 \right] \), and \textit{production radius} < 260mm.

Each event has \( O(15k) \) fiducial particles, \( O(300k) \) spacepoints.

We define \textbf{background} spacepoints as including:

- Those left by non-fiducial or intermediate particles (i.e. any particle barcodes not retained during simulation), or
- Those mis-constructed in the strip regions as \textit{ghost} spacepoints

An event has \( O(170k) \) background spacepoints.

\textbf{Ghost spacepoint}: Incorrectly constructed from clusters left by different particles.
Graph Construction

Hits → Metric Learning [or Module Map] → Graph
**EDGE TRUTH DEFINITIONS**

Target particle:
- $p_T > 1$ GeV, and
- At least 3 SP on different modules, and
- Primary

Therefore, define efficiency and purity (note that we mask out sequential non-target) for a graph with edges $e$

\[
\text{Efficiency} = \frac{|e \cap t_{Seq}|}{|t_{Seq}|}, \quad \text{Purity} = \frac{|e \cap t_{Seq} - \tilde{t}_{Seq}|}{|e - \tilde{t}_{Seq}|}
\]
**MODULE MAP - DOUBLETS**

- **The idea**: Build a map of detector modules, where a connection *from* module A to module B means that at least one true track has passed sequentially through A to B.

- **Step 1**: Build all combinations of sequential doublets for an event, register an A-to-B entry if a doublet passes through. O(90k) events used to build these combinations.

- **Step 2**: For each A-to-B entry, also register/update the max and min values of a set of geometric observables. Apply these cuts when building the graph in inference.

\[
\text{Map} = \{m_1 : m_2, m_2 : m_3, \ldots, m_5 : m_6\}
\]
**MODULE MAP – TRIPLETS**

- **The idea**: Build a map of detector modules, where a connection from module A to module B to module C means that at least one true track has passed sequentially through A to B to C.

- **Step 1**: Build all combinations of sequential triplets for an event, register an A-to-B-to-C entry if a triplet passes through.

- **Step 2**: For each A-to-B-to-C entry, also register/update the max and min values of a set of geometric observables. Apply these cuts when building the graph in inference.

---

**Step 1**

Connections added:
1 -> 2 -> 3
2 -> 3 -> 4
3 -> 4 -> 5
4 -> 5 -> 6

**Step 2**

- $z_0 = z_{h1} - r_{h1} \times \frac{\Delta r}{\Delta \eta}$
- $\Delta \phi = \phi_{h2} - \phi_{h1}$
- $\Delta \eta = \eta_{h2} - \eta_{h1}$

- $\Delta \Delta \eta = \frac{\Delta \eta_{12}}{\Delta \eta_{12}} - \frac{\Delta \eta_{23}}{\Delta \eta_{23}}$
### METRIC LEARNING INTUITION

- Encode / embed input into N-dimensional space
- Reward (low loss) matching pairs within unit distance
- Punish (high loss) mismatching pairs within unit distance
- Repeat for many pairs

#### “Contrastive” hinge loss

\[
l_n = \begin{cases} 
  x_n, & \text{if } y_n = 1, \\
  \max\{0, \Delta - x_n\}, & \text{if } y_n = -1,
\end{cases}
\]
**METRIC LEARNING**

- **The idea:** Teach an MLP to embed spacepoint features (spatial and cell information).
- In this embedded space, all doublets in a given particle track are trained to be near each other (Euclidean distance $x$), using a contrastive loss function $L$:
- A hit in a track is trained to be closest to its preceding and succeeding track hits.

The loss function $L$ can be expressed as:

$$ L = \begin{cases} 
  x, & \text{if true pair} \\
  \max(0, r - x), & \text{if false pair} 
\end{cases} $$
FAST GRAPH CONSTRUCTION

• Nearest neighbor search is a bottleneck of the graph construction stage

• **FAISS** finding K=500 for N=100,000 ~ 700ms

• KNN is overkill – we don’t need explicit list of K sorted neighbours

• Built custom library on Fixed Radius Nearest Neighbour (FRNN) search algorithm

• Cell-by-cell grid search is much faster: [*The complexity of finding fixed-radius near neighbors.* Bentley, et al 1977]

Fast fixed-radius nearest neighbors: Interactive Million-particle Fluids, Hoetzlein (NVIDIA), 2014

Fixed Radius NN Search vs Pytorch3D’s KNN

- 172974 points -16x speed up
- 543652 points -12x speed up
- 437645 points -15x speed up
- 100000 points -30x speed up

Accelerating NN Search on CUDA for Learning Point Clouds, Xue 2020
METRIC LEARNING - FILTERING

- Output graph of metric learning is impure: 0.2%
- Can pass edges through a simple MLP filter to filter out the easy fakes
- Improves purity to 2%, so graph can be trained entirely on a single GPU
GRAPH CONSTRUCTION RESULTS

• Drop in efficiency at low $\eta$ due to poor barrel strip resolution (will discuss further!)

• Drop in efficiency at high $p_T$ due to low training statistics
Graph Neural Network

Edge Labeling

Edge Scores
EDGE CLASSIFICATION WITH
GRAPH NEURAL NETWORK

1. Node features (spatial position) are encoded
2. Encoded features are concatenated and encoded to create edge features
3. Edge features are aggregated around nodes to create next round of encoded node features (i.e. message passing)
4. Each iteration of message passing improves discrimination power

\[
v^k_0 = \phi(v^k_0, \Sigma e^k_{0j}),
\]

\[
e^k_{ij} = \phi(v^k_i, v^k_j, e^k_{ij})
\]

MEMORY MANAGEMENT

- Graph construction leads to very large graphs O(1m) edges, cannot fit training on A100 GPU with 32Gb memory
- Should not split the graphs up (leads to lower GNN accuracy)
- Solution A: Were previously using a compromising form of “gradient checkpointing” – reduced memory by 4x
- Now using maximal checkpointing, reduce memory further by 2x – just fits on A100
TRAINING SOLUTIONS

- **Solution B**: Model offloading
- Each layer of GNN placed on GPU for forward and backward pass, but held on CPU otherwise
- Works well with TensorFlow, enabling training of $O(1m)$ edge graphs
- Unable to integrate with Pytorch pipeline

ZeRO-Offload: Democratizing Billion-Scale Model Training
arXiv: 2101.06840
LOSS FUNCTION DESIGN

- The **target** of the GNN and track reconstruction is edges from primary particles with pT>1 GeV that have left at least 3 hits on different modules in the detector (see slide 12)

- Have very small set of target edges (1-2% of edges are true target $t_{Seq}$)

- **Solution:** $t_{Seq} y = 1$ weighted up by $\times 10$, sequential background $\tilde{t}_{Seq}$ masked, all others $y = 0$

- Weighting gives much better performance at high-efficiency

- Masking gives much better performance around the 1 GeV cutoff
GNN EDGE CLASSIFICATION RESULTS

ROC CURVE & EDGEWISE PERFORMANCE VS. $p_T$

- **ATLAS Simulation Preliminary**
  - $\sqrt{s} = 14$ TeV, $tt$ or $tt$, $p_T > 1$ GeV
  - Background rejection
  - Signal efficiency
  - GNN classifier
  - Naive classifier
  - Edge classification score $s = 0.5$

- **GNN per-edge efficiency**
  - $\sqrt{s} = 14$ TeV, $tt$ or $tt$, $p_T > 1$ GeV
  - using Module Map
  - $|\eta| > 2$
  - $|\eta| < 2$

- Edge cut of 0.5 on output of GNN edge classifier
GNN EDGE CLASSIFICATION RESULTS

EDGewise PERFORMANCE VS. $\eta$

Again, see a drop in performance at low $\eta$
BARREL STRIP MISCLASSIFICATION

Nature of false positive edges

- 43%: “True” ghosts
- 37%: Fakes

Location of false positive edges
BARREL STRIP MISCLASSIFICATION

“True” ghost edges: 43%
Edges between SP from particle A, and a ghost SP of clusters from particle A and particle B. i.e. The GNN is “right”, the construction is “wrong”

Fake edges: 37%
Edges between SP from particle A and particle B. i.e. The GNN is “wrong”
STRIP MODULES: GHOSTS AND Z-RESOLUTION

- Since spacepoints are constructed from pairs of clusters in the strip, could mis-construct and form a ghost.
- These ghosts can be cleaned up in later stages of the reconstruction chain.
- However, even for correctly matched clusters, there remains low z-resolution.
- Consider this example.
- Easily confuses GNN!
- Could fix by including underlying cluster information somehow... (e.g. heterogeneous node features)

Image courtesy of Jan Stark – thanks!
Graph Segmentation

- Edge Scores
- Connected Components
- Track Candidates

3

or

Connected Components + Walkthrough
We now have labelled edges. Want to now label each node depending on connectivity.

Two distinct approaches: **component-based** segmentation, or **path-based** segmentation.

**Component-based**

E.g. connected components algorithm:

- Classified edges
- Ignore cut edges
- Label connected components

- Pros: Fast - $O(N_{nodes})$
- Cons: Can merge tracks into one candidate

**Path-based**

E.g. walkthrough algorithm:

- Classified edges, starting node
- Choose high score junctions
- Remove a high-scoring path

- Pros: Handles hits as a sequence, as a track should be
- Cons: Potentially slow - $O(N_{edges})$, needs a directed graph
We now have labelled edges. Want to now label each node depending on connectivity.

Two distinct approaches: **component-based** segmentation, or **path-based** segmentation.

**Component-based**
- Classified edges
- Ignore cut edges
- Label connected components

E.g. connected components algorithm:
- Pros: Fast - $O(\text{nodes})$
- Cons: Can merge tracks into one candidate

**Path-based**
- Classified edges
- Score junctions
- Remove a high-scoring path

E.g. walkthrough algorithm:
- Cons: Potentially slow - $O(\text{edges})$, needs a directed graph

Both methods by construction associate each hit with only one track.
1. Connected Components

- Classified edges
- Cut score < 0.2
- Label simple candidates

2. Walkthrough, a.k.a “Wrangler”

- Walk through paths from starting node, count length $L$
- Assign longest path as candidate
TRACK MATCHING DEFINITIONS

- $N(P_i, C_j)$ is the number of spacepoints shared by particle $i$ and candidate $j$

- Particle $i$ is called “matched” if, for some $j$, $\frac{N(P_i, C_j)}{N(P_i)} > f_{\text{truth}}$

- Candidate $j$ is called “matched” if, for some $i$, $\frac{N(P_i, C_j)}{N(C_j)} > f_{\text{reco}}$

- Particle $i$ and candidate $j$ are called “double matched” if, for some $i$ and $j$, $\frac{N(P_i, C_j)}{N(P_i)} > f_{\text{truth}}$ and $\frac{N(P_i, C_j)}{N(C_j)} > f_{\text{reco}}$

- $eff = \frac{\sum_i P_i(\text{matching condition})}{\sum_i P_i}$, $pur = \frac{\sum_j C_j(\text{matching condition})}{\sum_j C_j}$

**Standard matching**: single-matched particles with $f_{\text{truth}} = 0.5$

**Strict matching**: double-matched particles with $f_{\text{reco}} = 1.0$
TRACK RECONSTRUCTION RESULTS

**ATLAS Simulation Preliminary**

- $\sqrt{s} = 14$ TeV, $t\bar{t}$, ($\mu$) = 200, primaries (t$\bar{t}$ and soft interactions) $p_T > 1$ GeV
- Using Module Map

Matching to truth particles without track fit:
- Red: Standard matching
- Black: Strict matching

- Standard matching: single-matched particles with $f_{\text{truth}} = 0.5$
- Strict matching: double-matched particles with $f_{\text{reco}} = 1.0$

- Fake rate is $O(10^{-3})$ using standard truth matching
Physics is important, but GNNs shine in scaling behavior

When development began, graph-based pipeline started required 15 sec for TrackML

Implemented custom Fixed Radius Nearest Neighbor (FRNN) algo., cuGraph Connected Components algo., and Mixed Precision inference

Now have sub-second TrackML inference on 16Gb V100 GPU

Inference time scales approximately linearly across size of event, in TrackML
ONGOING WORK
ONGOING WORK: HETEROGENEOUS NODE FEATURES

- Motivated by inconsistent performance across detector:
- Currently each node in graph uses same input feature set – spacepoint \( s = (r, \phi, z) \)
- We could imagine using cluster-level information, e.g. position and shape of energy deposit
- *But:* this is not consistent across detector. Need different node and edge networks depending on detector region
ONGOING WORK: HETEROGENEOUS NODE FEATURES

- To get intuition, consider simple filter MLP applied to two pixel nodes:

- To apply a filter MLP to a pixel (single cluster) and strip (double cluster) node combination, need a different MLP:

- Already gives better than homogeneous filter MLP (~2x construction purity)
ONGOING WORK: HETEROGENEOUS GRAPH NEURAL NETWORK

- Exact same logic applies to GNN networks
- For a four-region heterogeneous GNN, we have four node encoders/networks \((N_0, N_1, N_2, N_3)\) and ten edge encoders/networks \((E_{00}, E_{01}, E_{02}, E_{03}, E_{11}, \ldots, E_{34}, E_{44})\)
- Thus, is a larger model and takes longer to train
- But reduces GNN inefficiency and fake rate by approximately half
### ONGOING WORK: ACTS & ATHENA INTEGRATION

#### ACTS (A Common Tracking Software)
- A library for tracking that is independent of particular experiment or geometry
- Written in highly performant c++ and parallelized

<table>
<thead>
<tr>
<th>Tracking</th>
<th>Geometry (TrkDetDescr)</th>
<th>Event Data Model (TrkEvent)</th>
<th>Extrapolation (TrkExtrapolation)</th>
<th>Fitting (TrkFitter)</th>
<th>Calibration, general (TrkTools)</th>
<th>Alignment (TrkAlignment)</th>
</tr>
</thead>
</table>

*A. Salzburger, et al.*
# Ongoing Work: ACTS & Athena Integration

Integration of GNN pipeline with ACTS

- **Integration complete**, with generic **TrackFindingMLBased** interface
- Uses TorchScript to call ML models (OnnxRuntime not yet fully compatible with GNN methods)
- Replaces seeding and track finding stages, produces protoTracks

---

A. Salzburger, et al.

---

## Tracking

<table>
<thead>
<tr>
<th>Geometry (TrkDetDescr)</th>
<th>Event Data Model (TrkEvent)</th>
<th>Extrapolation (TrkExtrapolation)</th>
<th>Fitting (TrkFitter)</th>
<th>Calibration, general (TrkTools)</th>
<th>Alignment (TrkAlignment)</th>
</tr>
</thead>
</table>

- Spacepoints in each event: std::vector<Spacepoints>
- A list of track candidates: std::vector<std::vector<int>>
ONGOING WORK: ACTS & ATHENA INTEGRATION

Athena

- Framework for ATLAS event generation, simulation, digitization, reconstruction and analysis

Integration of GNN pipeline with Athena:
- This is ongoing!
OTHER ONGOING WORK

- Extending TrackML inference timing and scaling studies to ATLAS ITk.
- Investigating training and inference performance on lower $p_T$ tracks (i.e. < 1 GeV) and high $p_T$ tracks (i.e. > 10 GeV).
- Investigating performance on large radius tracks and dense track environments.
- Direct comparison with combinatorial Kalman filter (current algorithm) efficiency and track parameter resolution.
CONCLUSION

- A graph-based representation of particle collisions is intuitive and rich
- GNNs and other graph techniques are well-suited even to high luminosity events
- Produced first public results on official ATLAS ITk geometry using GNN-based track reconstruction pipeline
- Promising reconstruction performance, well-positioned for comparison with traditional algorithms
- This is very early in development – many more improvements are in progress within ExaTrkx+L2IT
- Also new techniques being invented in GNN/ML community every day

THANKS FOR TUNING IN!

Links
- ExaTrkx website
- L2IT website
- ExaTrkx paper
- L2IT paper
- Codebase