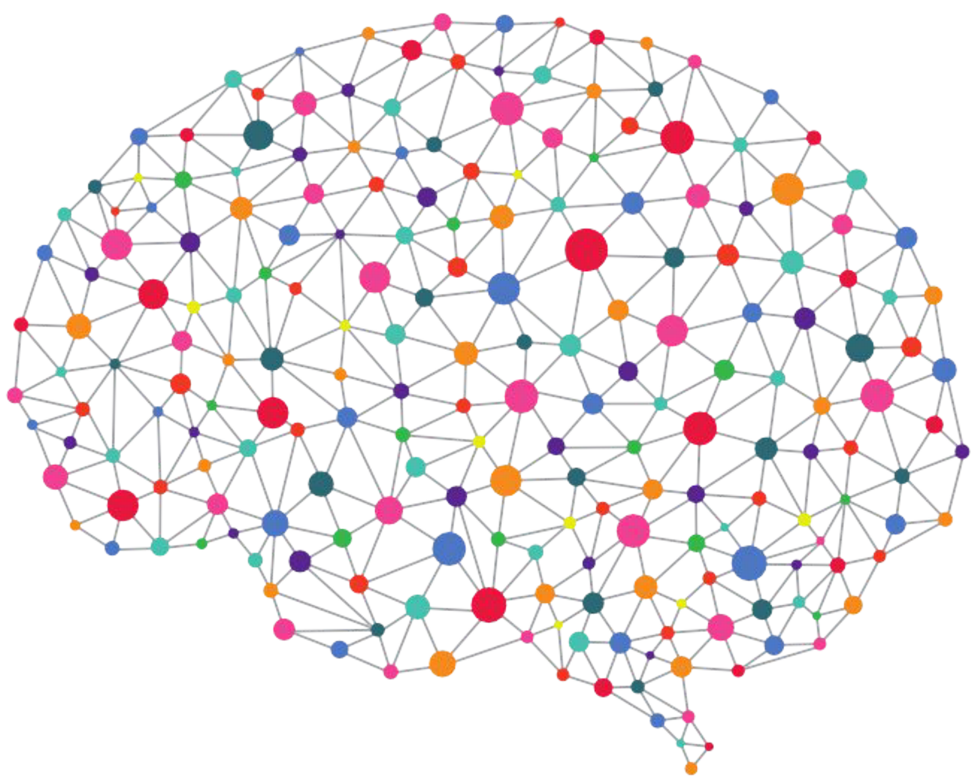
UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Graph Networks for Track Reconstruction

Learning to Discover: Advanced Pattern Recognition, IPa Workshop 2019

Marcel Kunze, Heidelberg University



Data Representation

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

- **Plain vector**
- Sequences
- Trees
- Graph

Numeric: made of numbers

0.1	0.2	0.8	0.5	0.3	1.0

Categorical: made of words

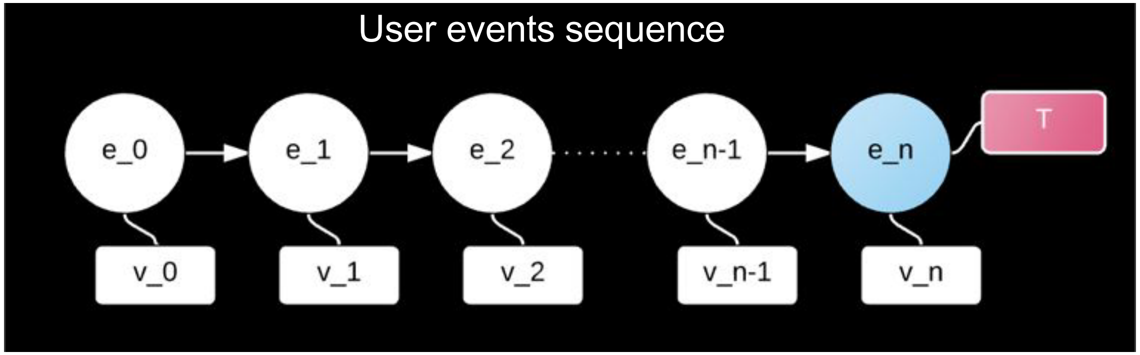
Female	1	0
Male	0	1

The numbers do not represent a complex relationship or hierarchy

Data Representation

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

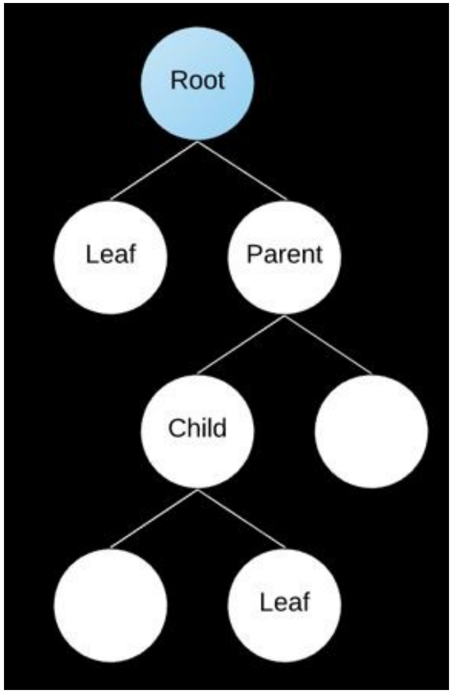
- Plain vectors
- **Sequences**
- Trees
- Graph



The data objects have a sequential (time) dependence

Data Representation

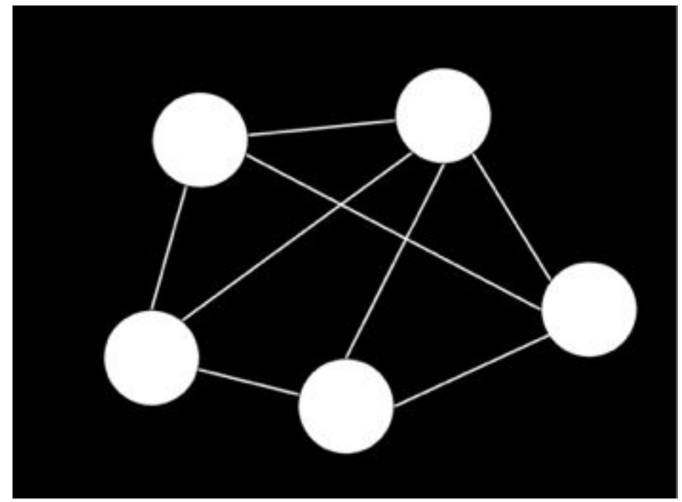
- Plain vectors
- Sequences
- **Trees**
- Graph

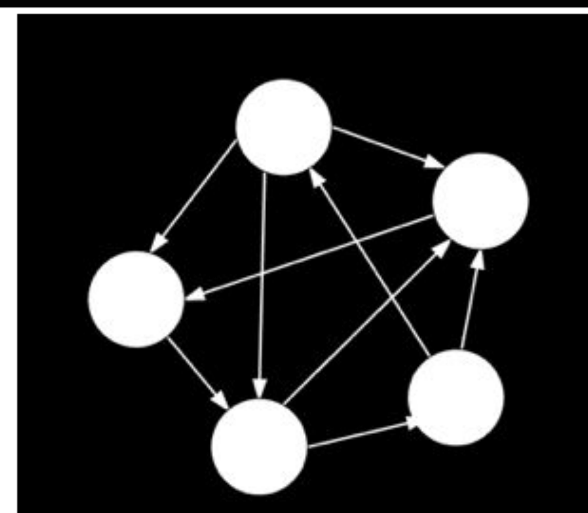


The data objects have a hierarchical dependence

Data Representation

- Plain vectors
- Sequences
- Trees
- **Graph**



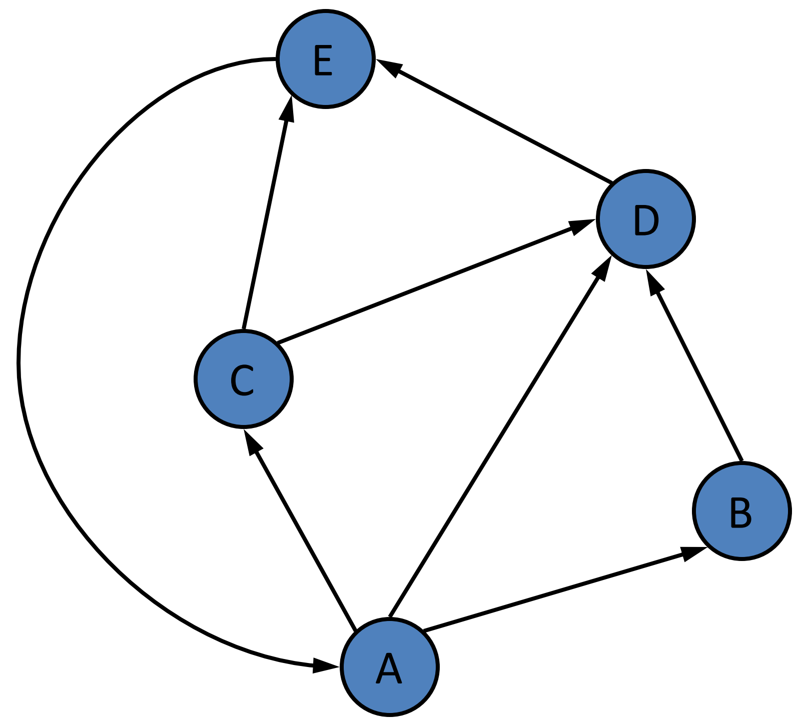


The data objects have an arbitrary dependence
- Model complex relationship / hierarchy

Directed Graphs

A directed Graph is a graph of nodes whose edges are all directed

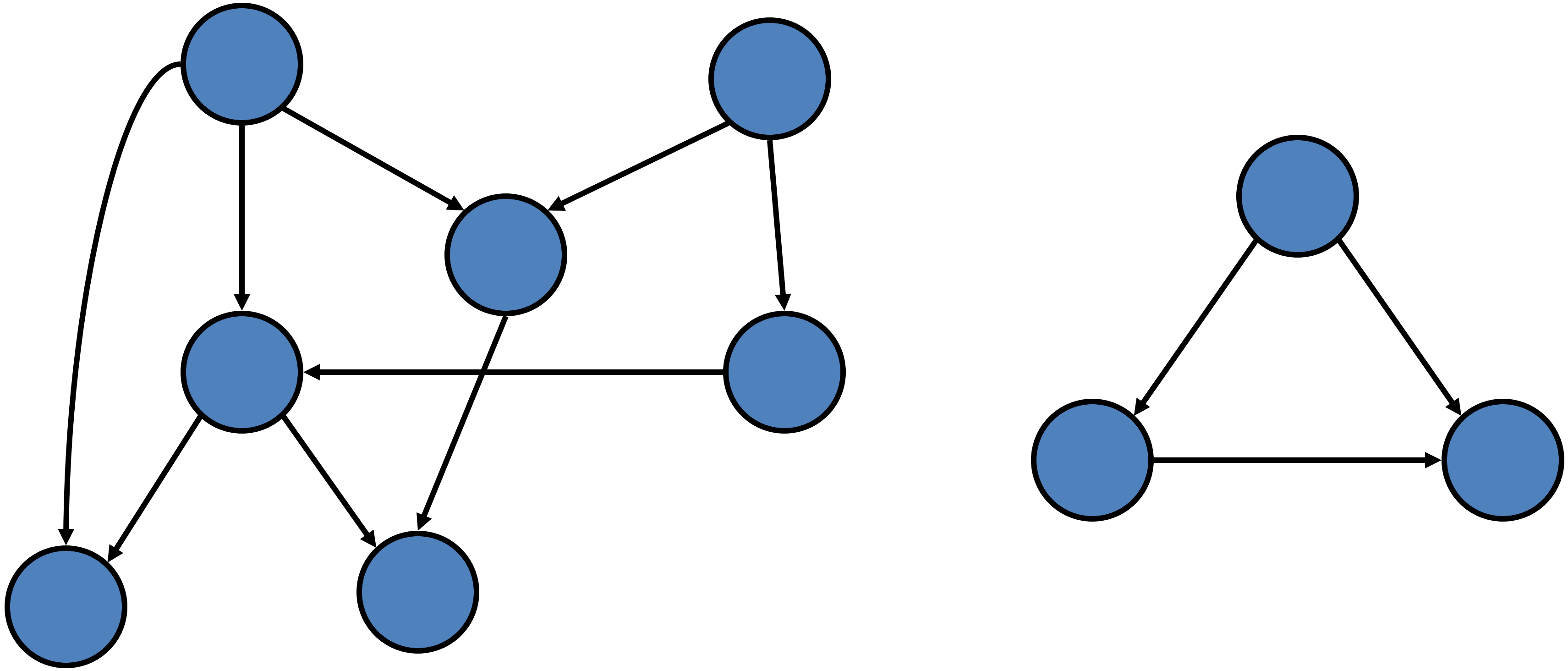
Applications
- one-way streets
- flights
- task scheduling
- …



Directed Acyclic Graphs (DAG)

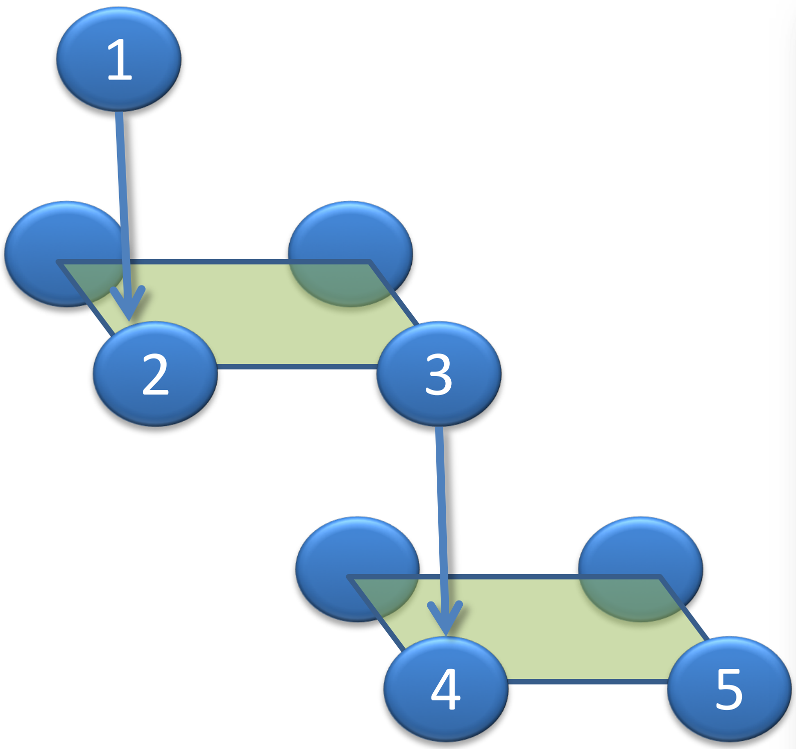
UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

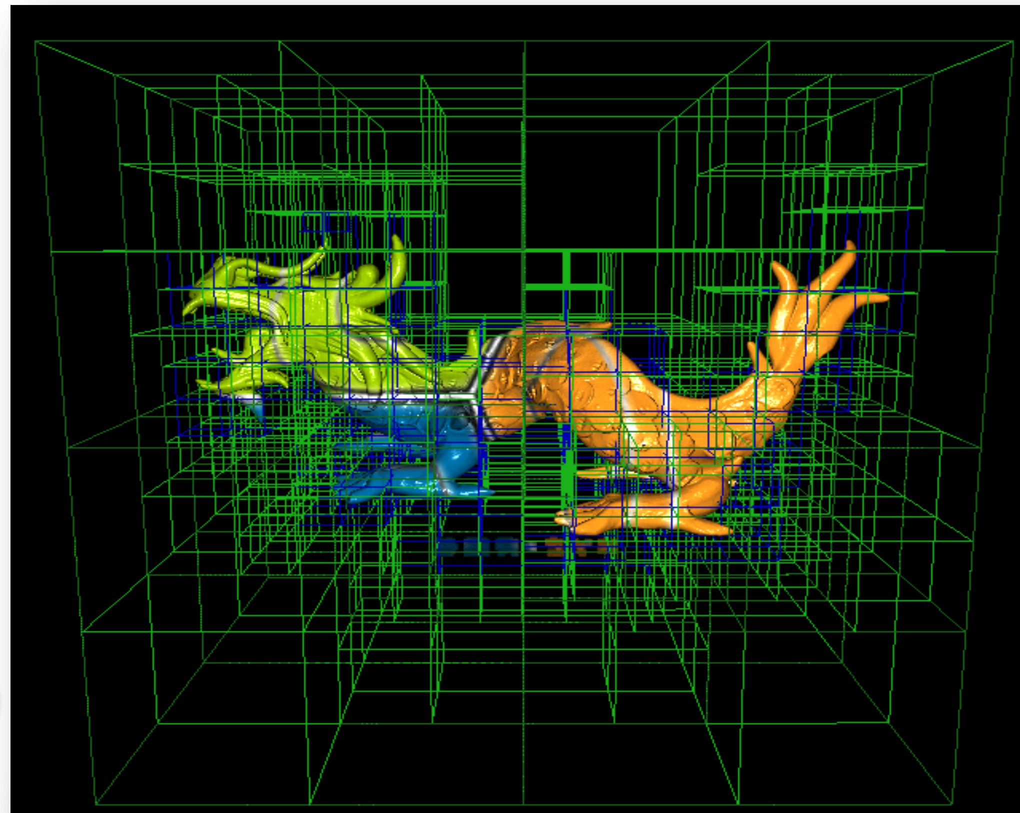
A *directed acyclic graph* or *DAG* is a directed graph with no directed cycles:



Gaming: Sparse Voxel Octrees (SVO)

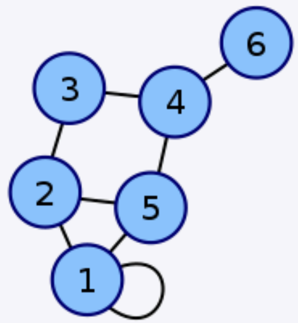
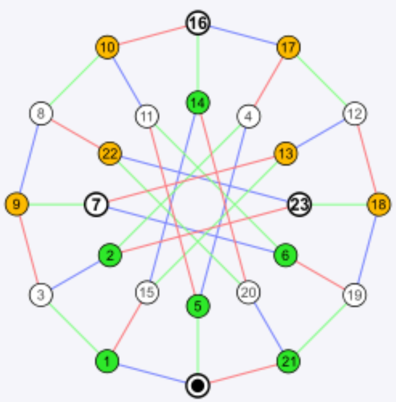
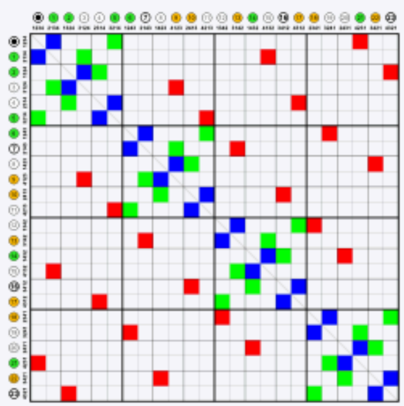
UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386





- Raytracing
- Compression of data
- Multi-scale resolution

Adjacency Matrix

Labeled graph	Adjacency matrix
	$\begin{pmatrix} 2 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ Coordinates are 1–6.
 Nauru graph	 Coordinates are 0–23. White fields are zeros, colored fields are ones.

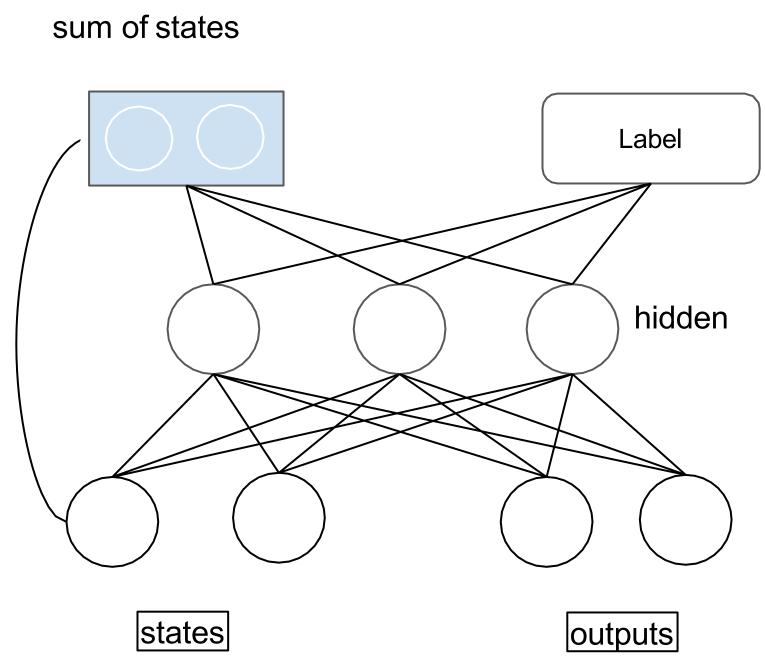
- The adjacency matrix is a square matrix to represent a graph.
- The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph.

In the mathematical field of graph theory, the **Nauru graph** is a symmetric bipartite cubic graph with 24 vertices and 36 edges.

Graph Neural Network

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

- A graph is processed node by node in a random order
- For a node in graph, the sum of the state vectors of neighboring nodes are computed and concatenated to its own label vector
- The algorithm guarantees a convergence of the state nodes to a stable and unique solution



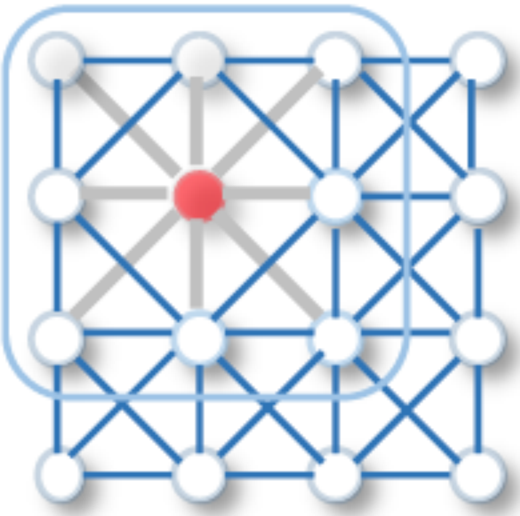
Taxonomy

UNIVERSITÄT HEIDELBERG
ZUKUNFT
SEIT 1386

Category	Models
Recurrent Graph Neural Networks (RecGNNs)	Spectral Spatial
Spatial-temporal Graph Neural Networks (STGNNs)	
Graph Autoencoders (GAEs)	Network embedding Graph Generation
Convolutional Graph Neural Networks (ConvGNNs)	

A Comprehensive Survey on Graph Neural Networks
arXiv:1901.00596v3 8 Aug 2019

Convolution



2D Convolution

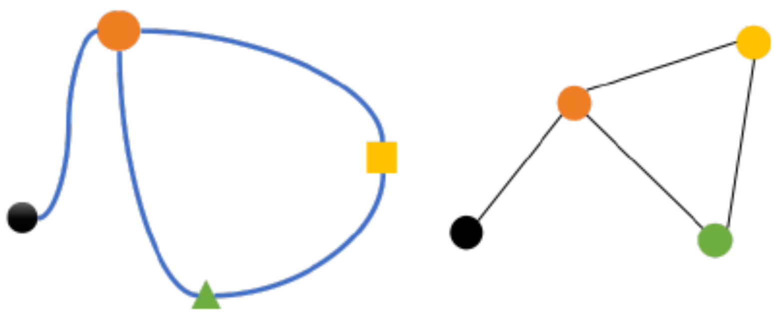
Analogous to a graph, each pixel in an image is taken as a node where neighbors are determined by the filter size. The 2D convolution takes the weighted average of pixel values of the red node along with its neighbors. The neighbors of a node are ordered and have a fixed size.

Graph Convolution

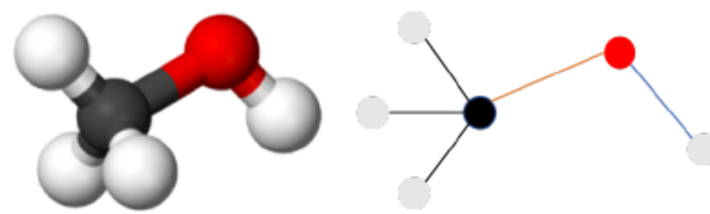
To get a hidden representation of the red node, one simple solution of the graph convolutional operation is to take the average value of the node features of the red node along with its neighbors. Different from image data, the neighbors of a node are unordered and variable in size

Application Domains

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386



Computer Science:
Routing + tracking algorithms



Chemistry:
Molecular engineering

Feature Mapping

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

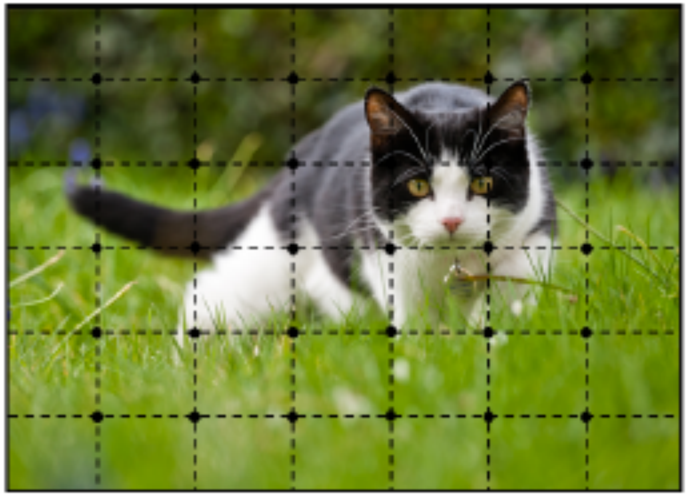
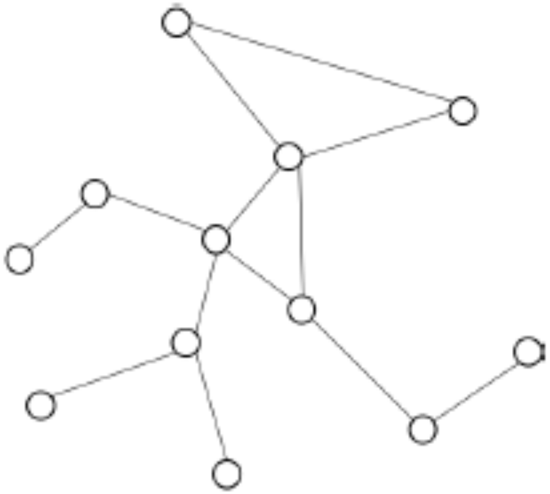


image in Euclidean space



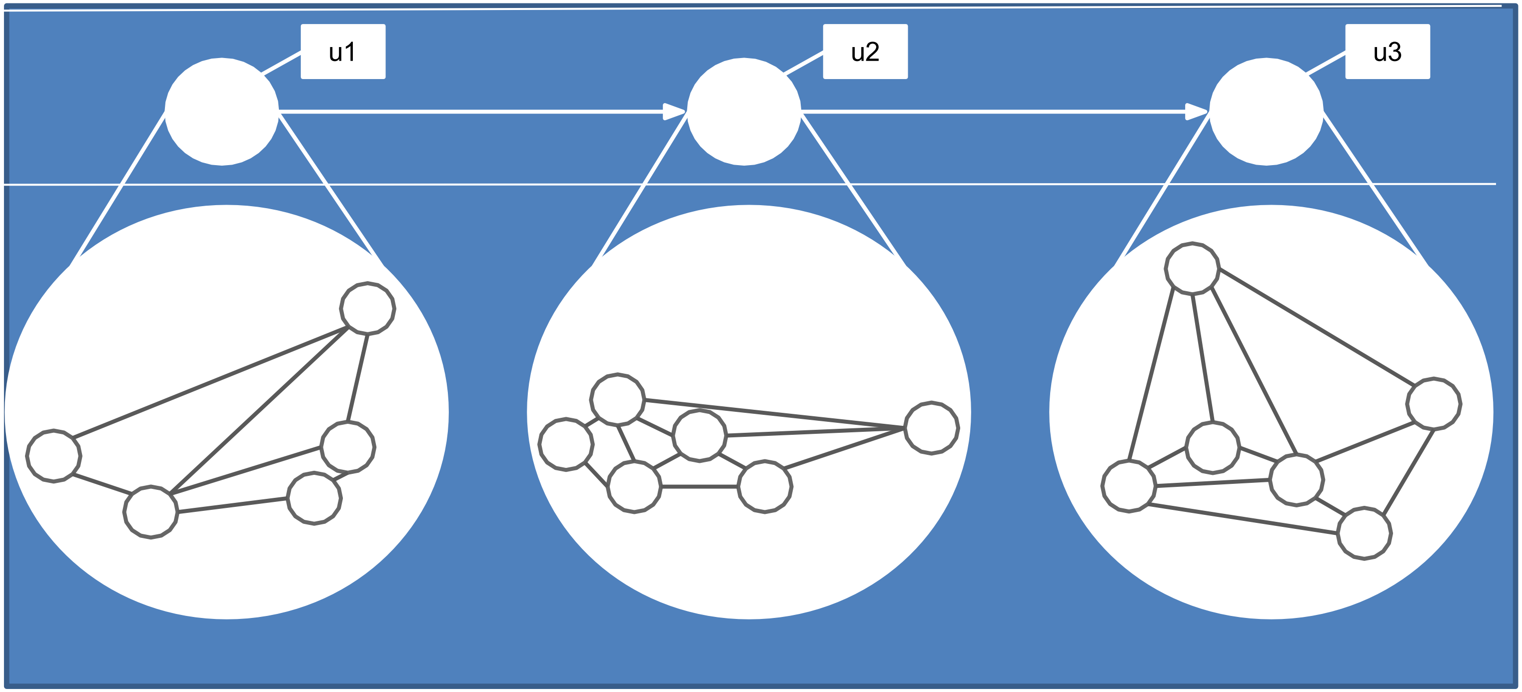
graph in non-Euclidean space

Human Action Recognition

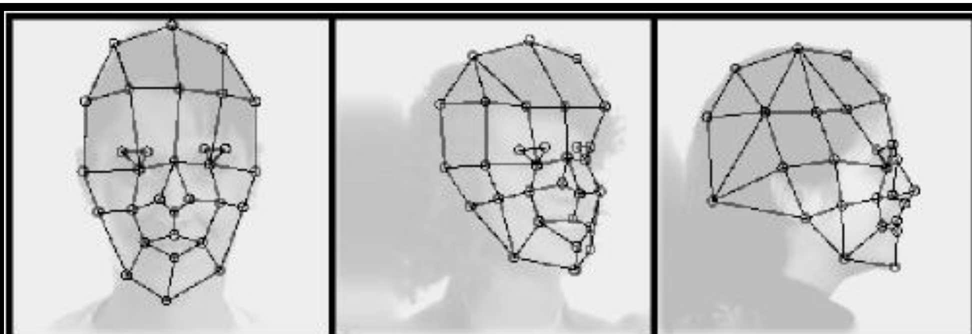
Video sequence

Level 1
Sequence of frames

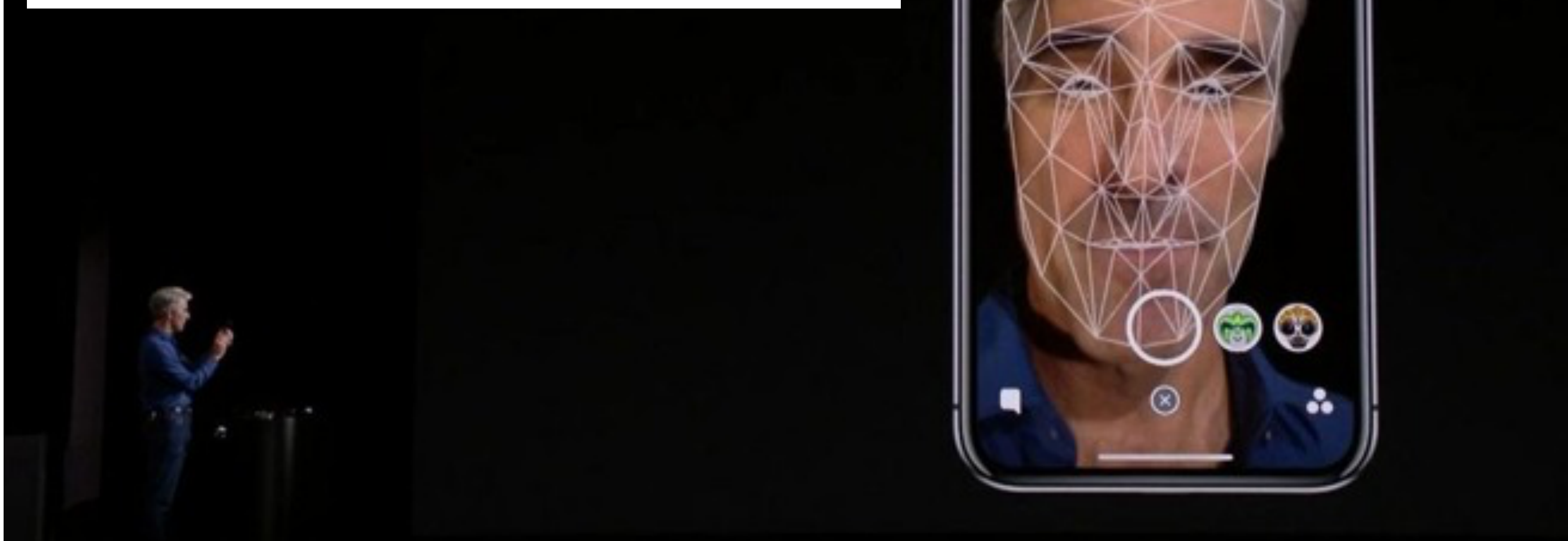
Level 2
Delaunay Triangulation of each frame



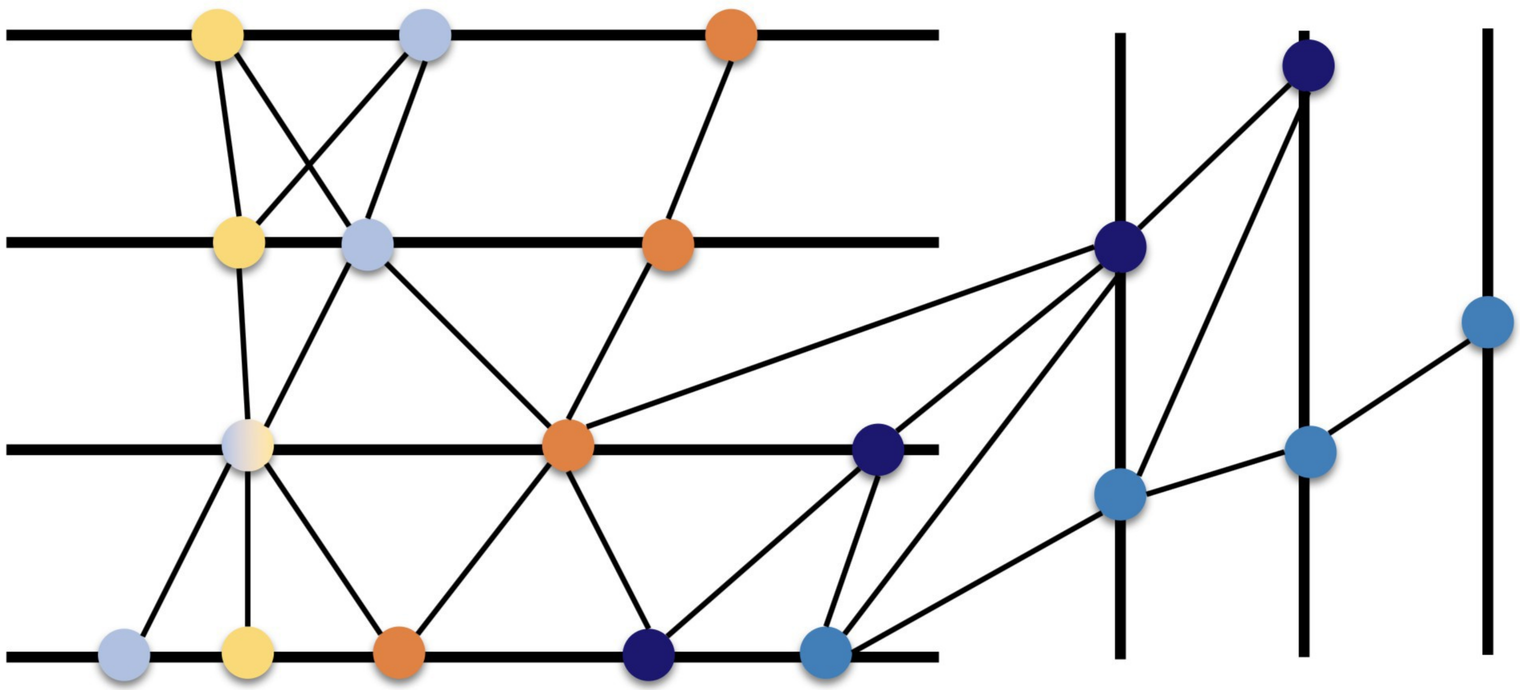
Face Recognition: Elastic Graph Matching



Face recognition by elastic bunch graph matching
Institute of Neuro Informatics research project, 1993



Tracker Hit Graph

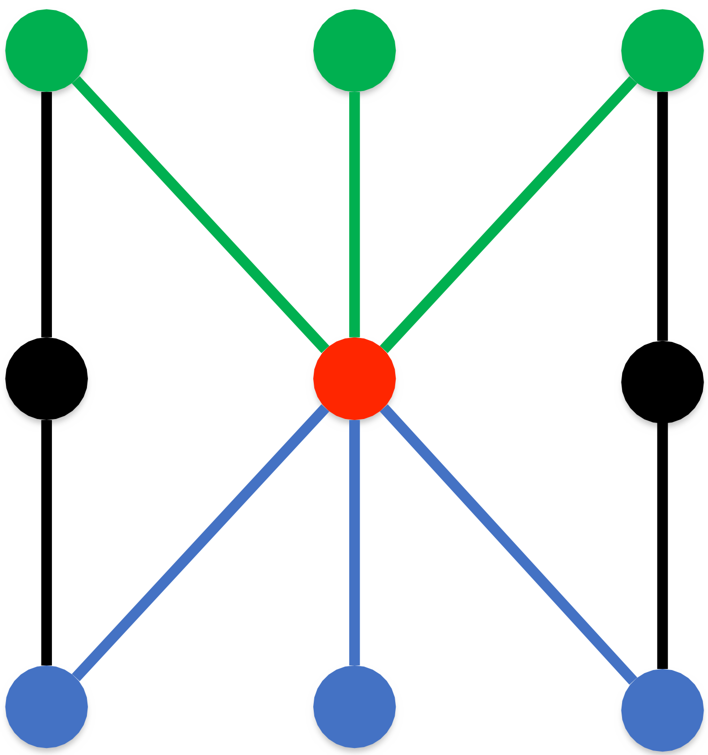


Graph construction

- One tracker hit ≡ one node
- Sparse edges constructed from geometrical consideration

➜ Edge classification ≡ reconstructing the trajectory of particles

GNN Architecture

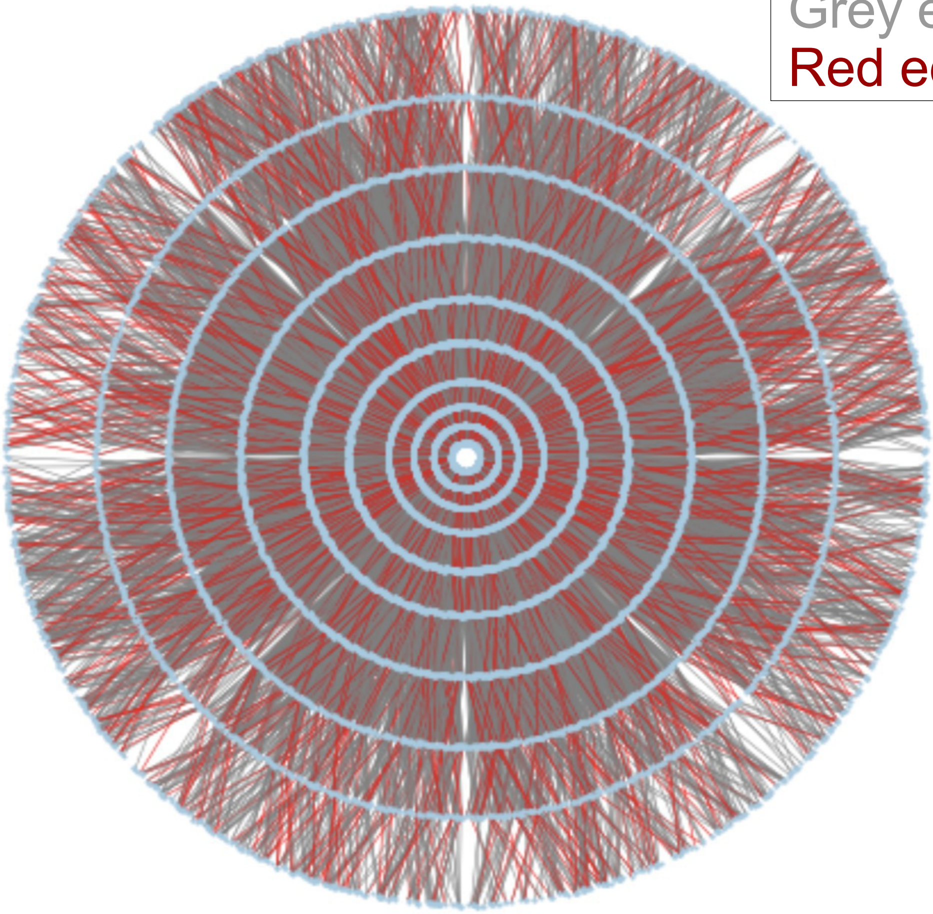


Three components operate on graph:

- **Input network** computes hidden node features
- **Edge network** computes **edge scores** from node features
- **Node network** computes hidden node features from aggregated **weighted** **incoming** and **outgoing** node features

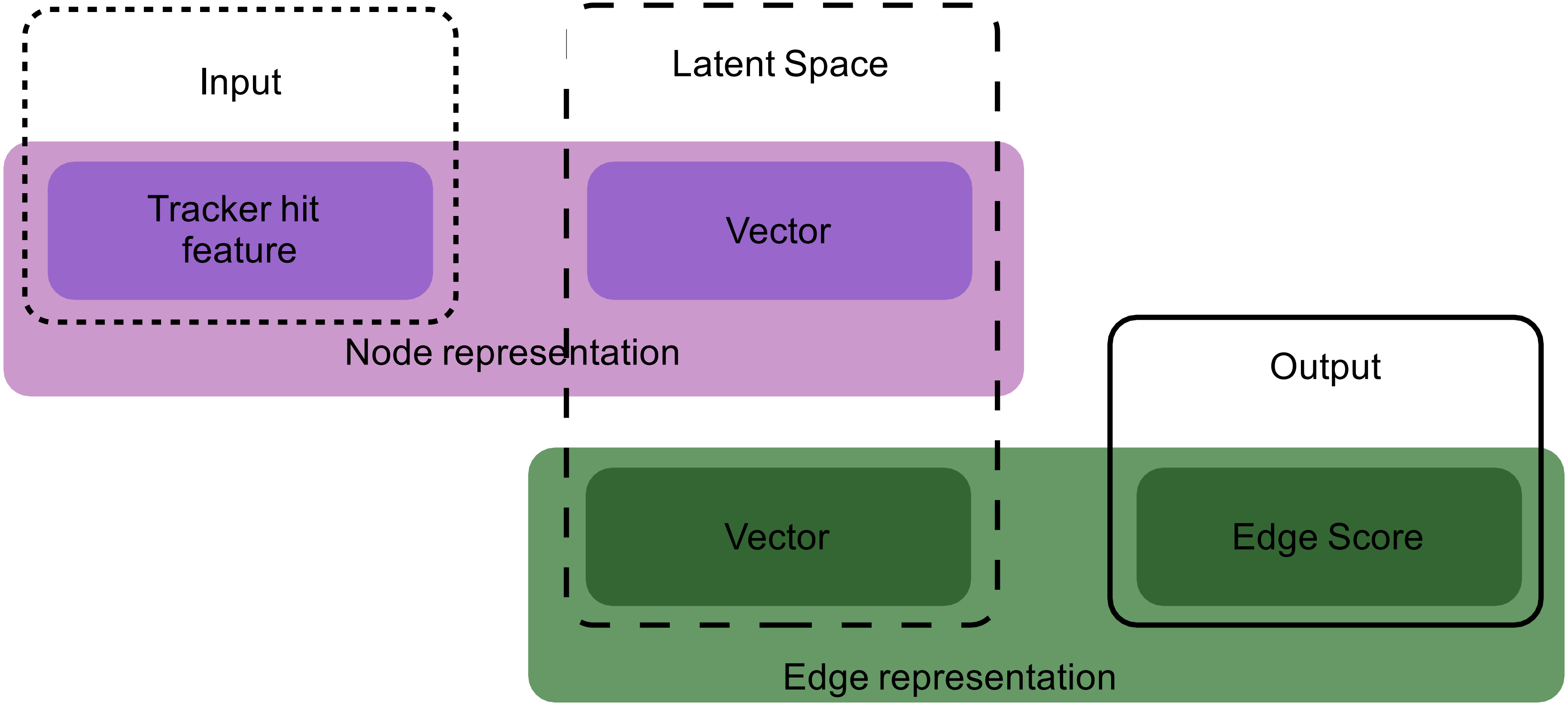
Incoming and outgoing nodes with higher weights get more "attention"

Edge Classification

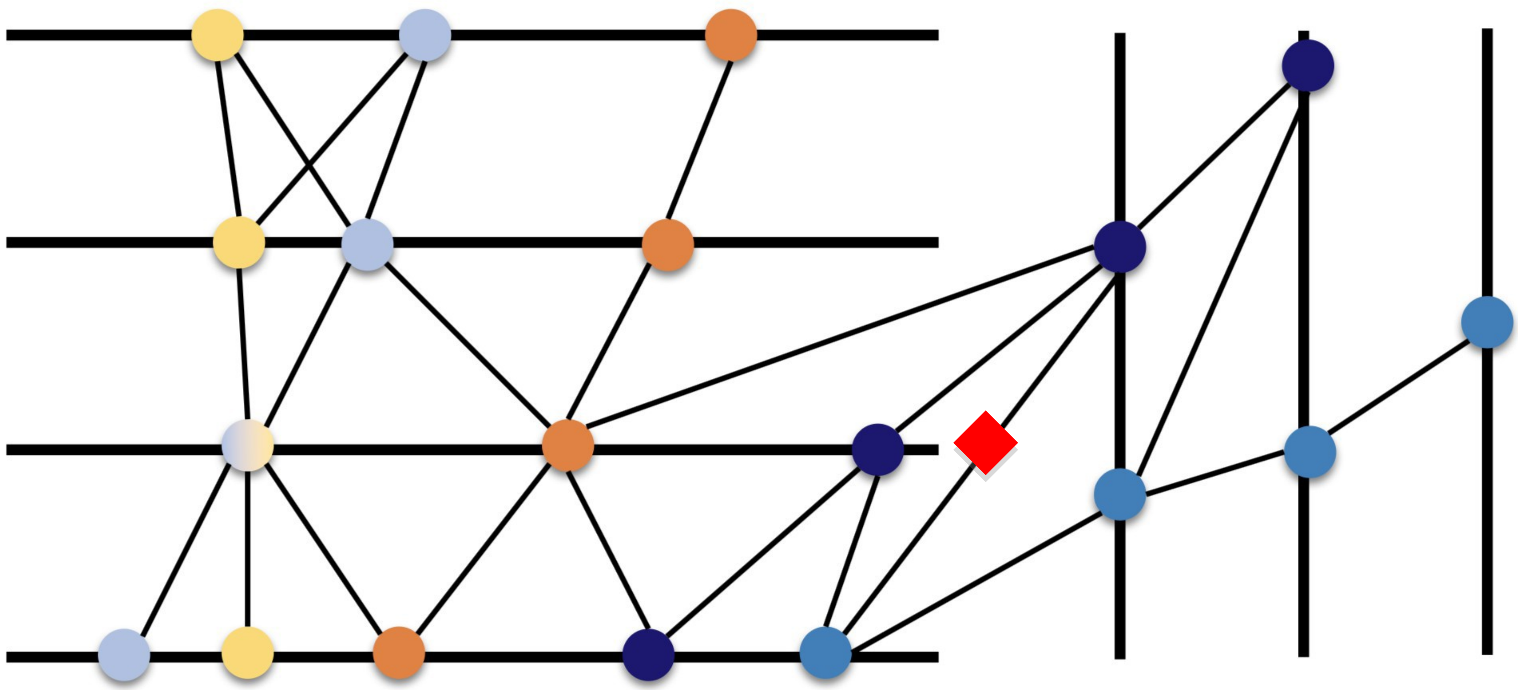


Grey edge: fake
Red edge : true

Node & Edge Representations

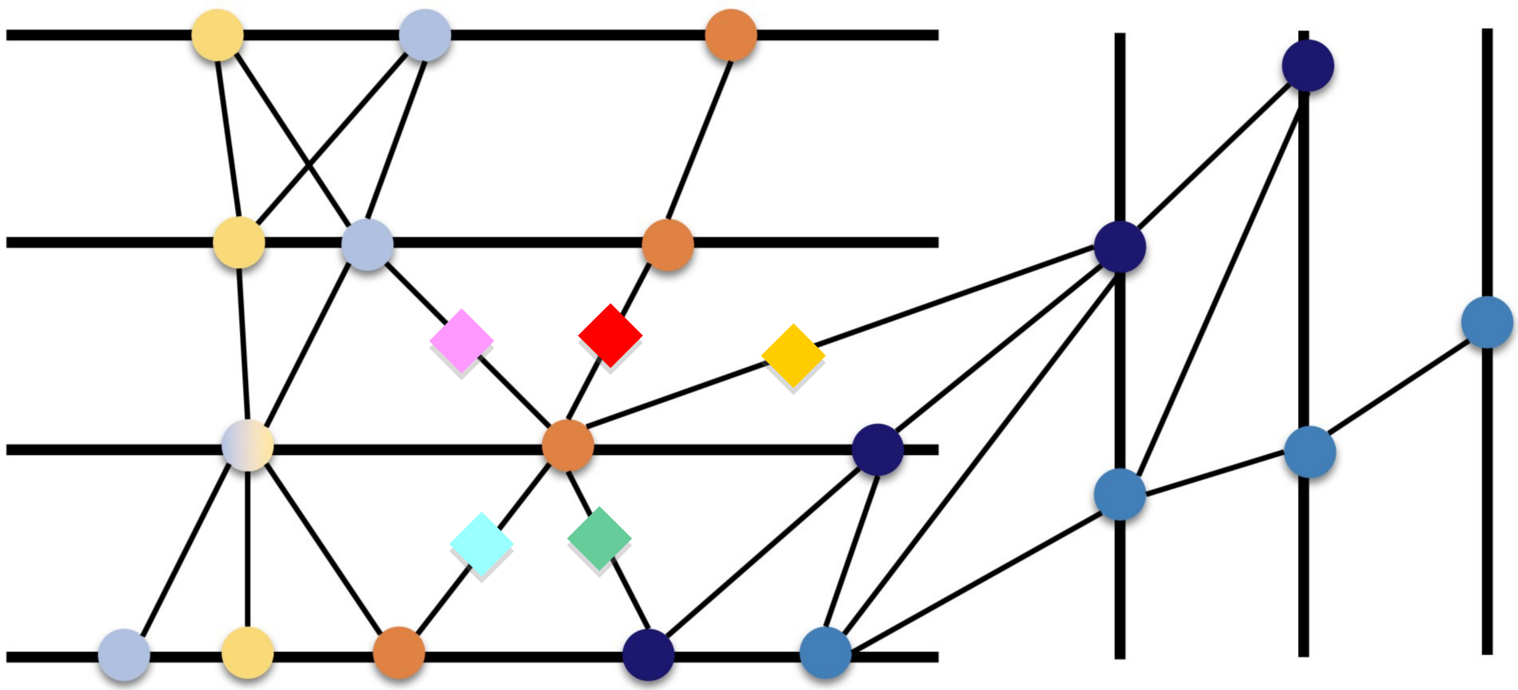


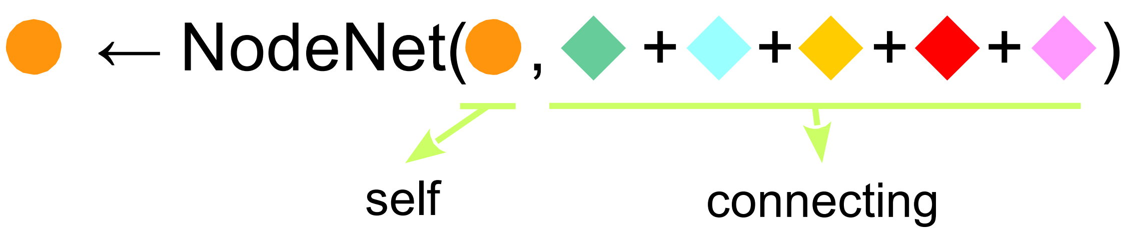
Edge Network



◆ ← EdgeNet(●,●)

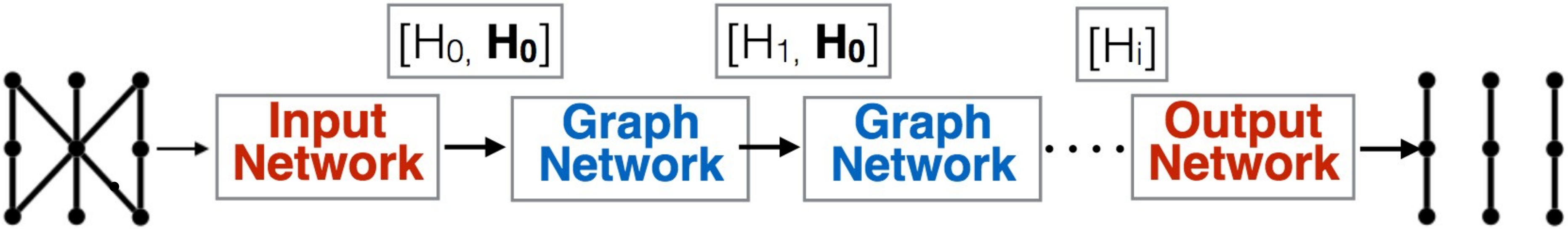
Node Network





Message Passing Model

HEP.TrkX

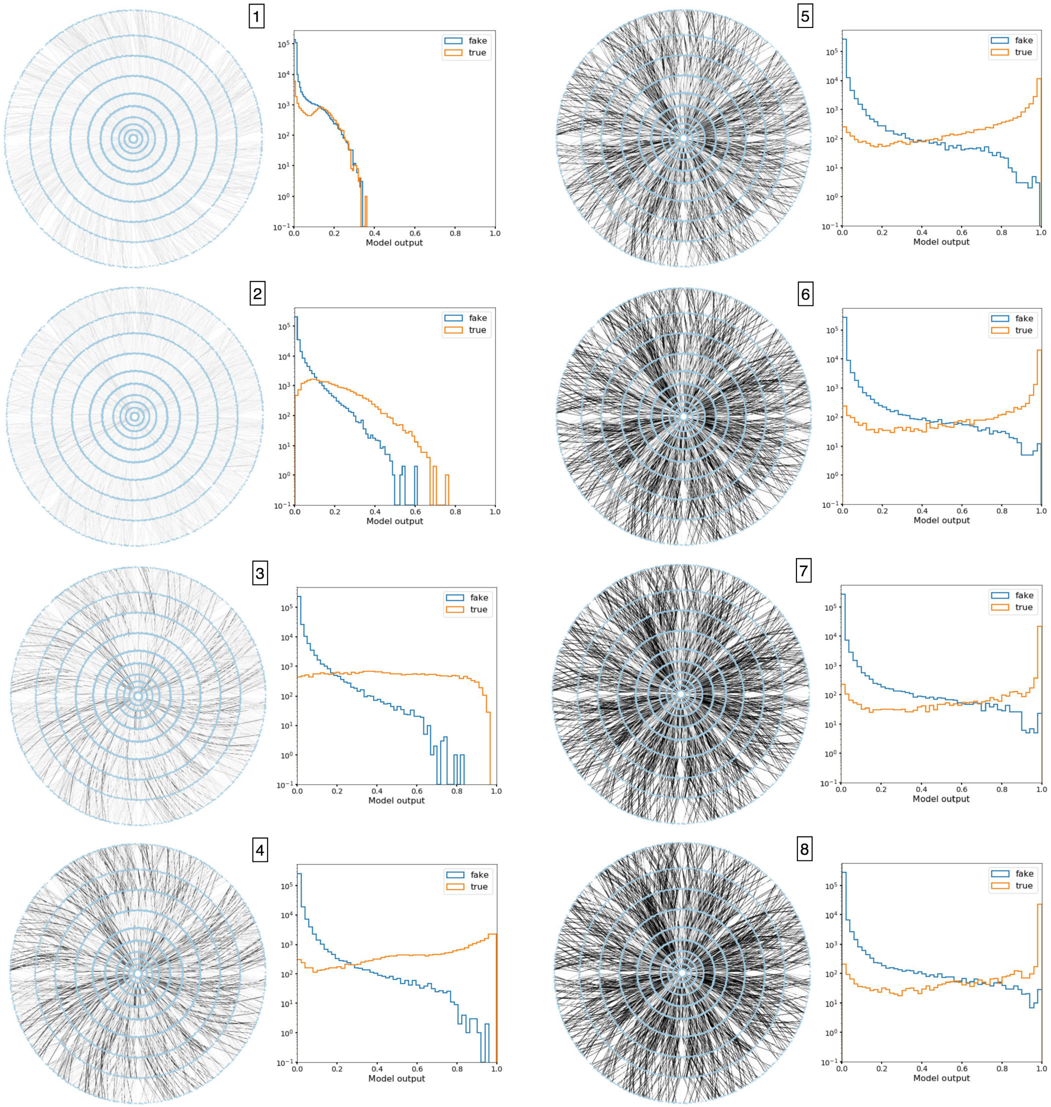


Graph is sparsely connected from consecutive layers
Edge representation computed from features at the ends
Node representation computed from the sum over all connected edges

- Correlates hits information through multiple (8) iterations of (Graph Network)
- Uses https://github.com/deepmind/graph_nets TF library

Information Flow

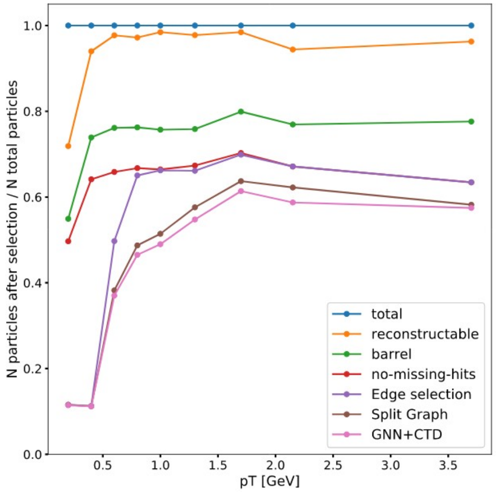
HEP.TrkX



- Checking edge score after each step of graph network.
- Effective output of the model is in step 8.
- Full track hit assignment learned in last stages of the model.
- Tracklets learned in intermediate stages.

Performance

one-event	N-particles	ratio w.r.t Total	ratio w.r.t Reconstructable	relative ratio
Total	11170	100%		100%
Reconstructable	9635	86%	100%	86%
Barrel	7492	67%	78%	78%
No-missing hits	6600	59%	69%	88%
Edge selection	3114	28%	32%	**47%**
Split graph	2668	24%	28%	**86%**
GNN	2590	23%	27%	**97%**



Tracks formed with a simple algorithms that traverse the hit graph over high-score edges.

→ Promising performance, once passed acceptance cut for training purpose

TrackML Throughput phase 3rd place

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Phase 2 cloudkitchen

Author: Marcel Kunze

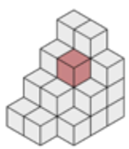
Accuracy: 0.93
Time/event: ~7 sec
Memory: 0.7 Gb

partly based on top quarks Phase 1 solution

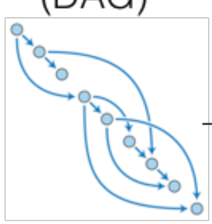
1	—	Top Quarks	0.92182	10	2mo

Algorithm outline

hits → sorted in voxels

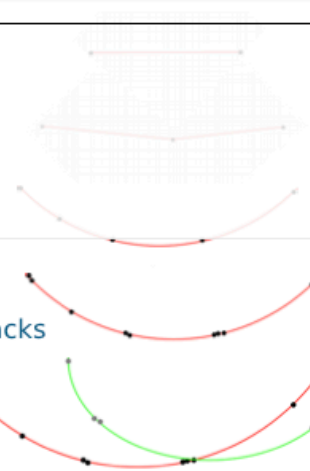


→ organised in direct acyclic graphs (DAG)



Main steps

- Select promising pairs
  - 7 million / 0.99
- Extend pairs to triples
  - 12 million / 0.97
- Extend triples to tracks
  - 12 million / 0.95
- Add duplicate hits to tracks
  - 12 million / 0.96
- Assign hits to tracks
  - 90% of hits / 0.92



Triplet finder: NN3

doublet finder: NN1, NN2

DAGs are pre-trained on ~25 events ground truth

DAGs are used to fast navigate through voxel space → Disc section / Radial section (Threaded)

Voxel (Volume Pixel)

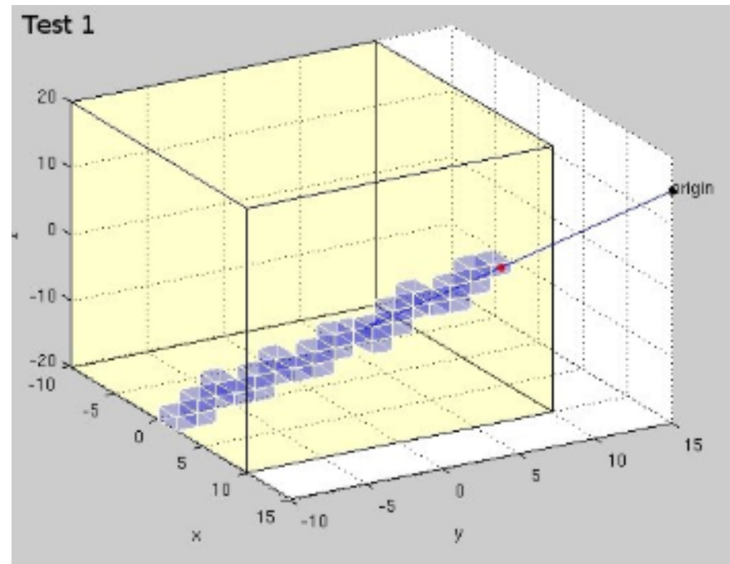
UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Define spatial elements in $\phi * \theta$ (voxel)

- Organize the voxels in DAGs according to track evolution in radial direction

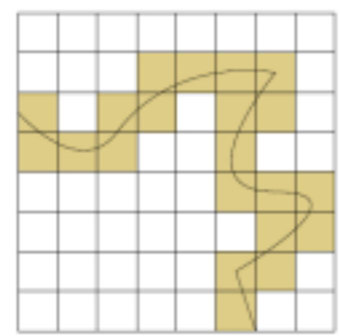
index = (phi<<32) | (theta<<24) | (layer<<16) | module;

- Flexible to model even arbitrary paths (kinks, missing hits, outliers, random walk, ..)
- Training is done with MC tracks of typically 15-25 events



Multiscale resolution (Better use SVOs?)

- 2*1 DAGs for pair finding (slices)
- 12*14 DAGs for triple finding (tiles)



Path finding

- Sort event hits into the trained DAGs
- Seed and follow the path strategy

Pattern Recognition with Machine Learning

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Intuition

- Model free estimator
- Start with basic quantities
- Coordinates, simple derived values
- Only very basic detector specific information

Input parameter space

- Polar coordinates (R_t, ϕ, z)
- Directional cosines
- Simple helix calculation (score)

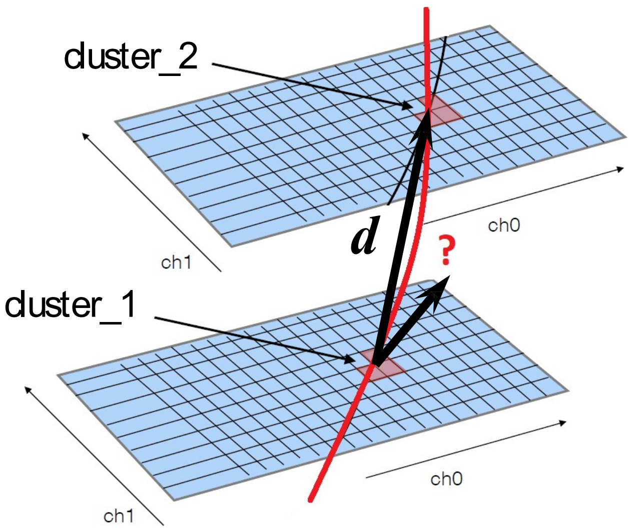
} In principle not needed, but speeds the things up !

Training

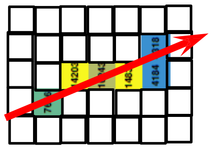
- Supervised: presenting MC ground truth
- Unsupervised: presenting probability density function

Input Parameter Space

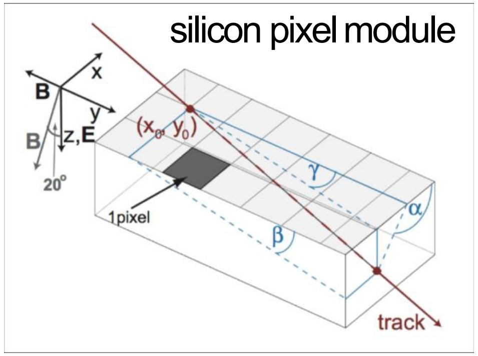
Given two hits (clusters of silicon cells): predict if they belong to the same track



- Estimate track direction from the cluster shape:



eigenvector of covariance matrix of the silicon cells



Features for the training

- Polar coordinates of the hit doublet: (r_1,ϕ_1,z_1), (r_2,ϕ_2,z_2)
- Triplet finder works the same with a hit triplet
- Simple helix score
- Angle/length deviations of the vector ***d*** projection from the values predicted by the shape of cluster 1
- Angle/length deviations of the vector ***d*** projection from the values predicted by the shape of cluster 2

Input Parameter Folding

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

The tracking problem is symmetric wrt. polar coordinates

- Fold the input parameter space into an octagon slice using “abs” function
- Considerable improvement of the separation strength of the parameters
- Need less statistics / yield better results

```
: ----------------------------------
: Rank : Variable    : Separation
: ----------------------------------
:    1 : log(score) : 5.039e-01
:    2 : rz3        : 5.491e-04
:    3 : phi3       : 7.552e-05
:    4 : z3         : 4.986e-05
:    5 : rz2        : 1.519e-05
:    6 : rz1        : 9.568e-06
:    7 : phi2       : 4.101e-06
:    8 : z1         : 1.967e-06
:    9 : z2         : 1.965e-06
:   10 : phi1       : 1.503e-06
: ----------------------------------
```

→

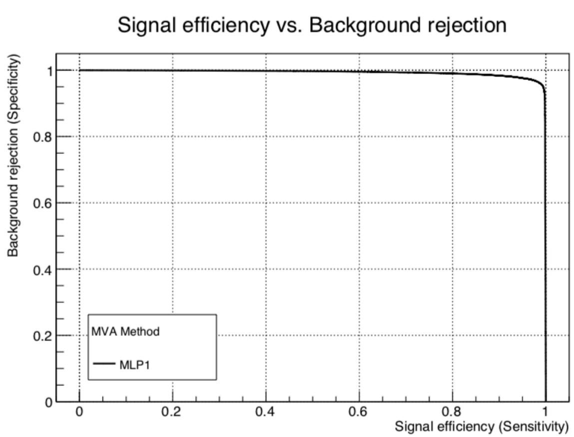
```
: -----------------------------------------------------
: Rank : Variable                          : Separation
: -----------------------------------------------------
:    1 : log(score)                        : 5.978e-01
:    2 : rz3                               : 6.329e-04
:    3 : abs(abs(phi3)-1.57079632679)      : 1.317e-04
:    4 : abs(z3)                           : 5.522e-05
:    5 : rz2                               : 2.067e-05
:    6 : rz1                               : 1.675e-05
:    7 : abs(abs(phi2)-1.57079632679)      : 4.335e-06
:    8 : abs(z1)                           : 3.592e-06
:    9 : abs(abs(phi1)-1.57079632679)      : 3.038e-06
:   10 : abs(z2)                           : 2.963e-06
: -----------------------------------------------------
```

Hit Doublet / Triplet Classification: MLP

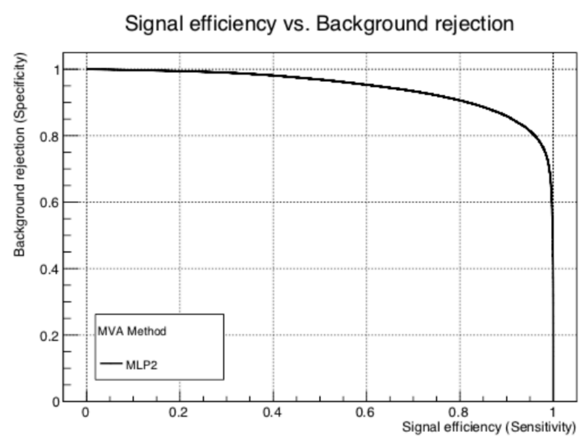
“Shallow learning” ;)

- **Classify the doublets and triplets with neural networks**
  - Multi Layer Perceptron: MLP1 8-15-5-1 / MLP2 9-15-5-1 / MLP3 10-15-5-1
  - Input: hit coordinates, directional cosines towards the clusters, helicity score wrt. origin
  - Output: doublet/triplet quality, supervised training with Monte-Carlo ground truth
  - Training: Typically 10 events, O(Mio) patterns, 500 epochs, one hour on standard PC
  - “*Receiver Operation Characteristics*” (ROC) curves indicate good quality

Doublet finder (disc)

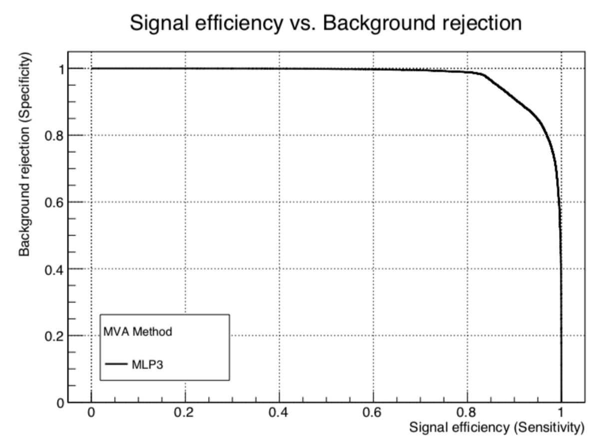


Doublet finder (tube)



Worse due to vertex shift !

Triplet finder



Multi Threading

- **Well defined algorithmic steps for pattern recognition**
- **Efficient parallelism on the basis of DAGs**
  - Form doublets from seeding hits in a DAG (MLP1, MLP2)
  - Extend the doublets to triplets (MLP3)
  - Extend the triplets to path segments
  - The path segments are merged into tracklets
  - Remove duplicate solutions

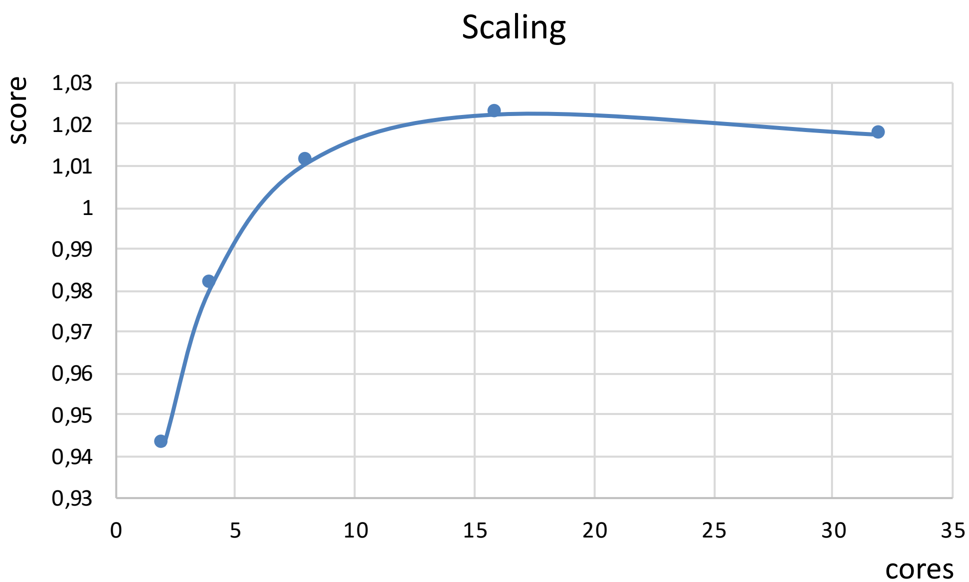
The tracklets are merged into a common tracking solution by a **serial task**

Scaling Behavior

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Scaling tests have been performed with Amazon EC2

- Instance type c5n.9xlarge (36 cores)
- Core power comparable to CodaLab cores
- Code scales up to 16 cores (Score: 1.022, accuracy 92.3%, 1.7s)
- Limited by serial code: Sorting tracklets into tracks (improve by use of OpenMP ?)



Amdahls Law: Speedup is the fraction of code P that can be parallelized:

$$speedup = \frac{1}{1-P}$$

Machine Learning Advantage

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Model free estimator

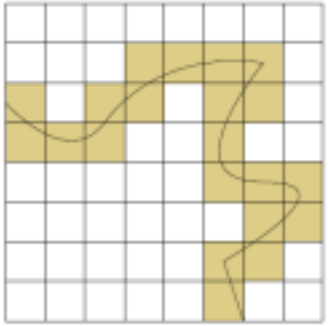
- Solution may be easily transferred to a different context

Graceful degradation in presence of changes

- Geometry
- Dead channels
- Calibration
- …

The DAGs may represent arbitrary tracking paths

- Inhomogeneous magnetic field
- Kinks
- …



Can we do better ?

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

- **Supervised** training makes us dependent on truth (wishful thinking)
- Is it possible to organize an **unsupervised** learning process that is driven by the data itself?

Process of Self-Organization

- System parameters are defined externally
- Final system state is not known a priori
- System state propagates in a self organized process
- System is mostly influenced by neighborhood relations
- No central planning

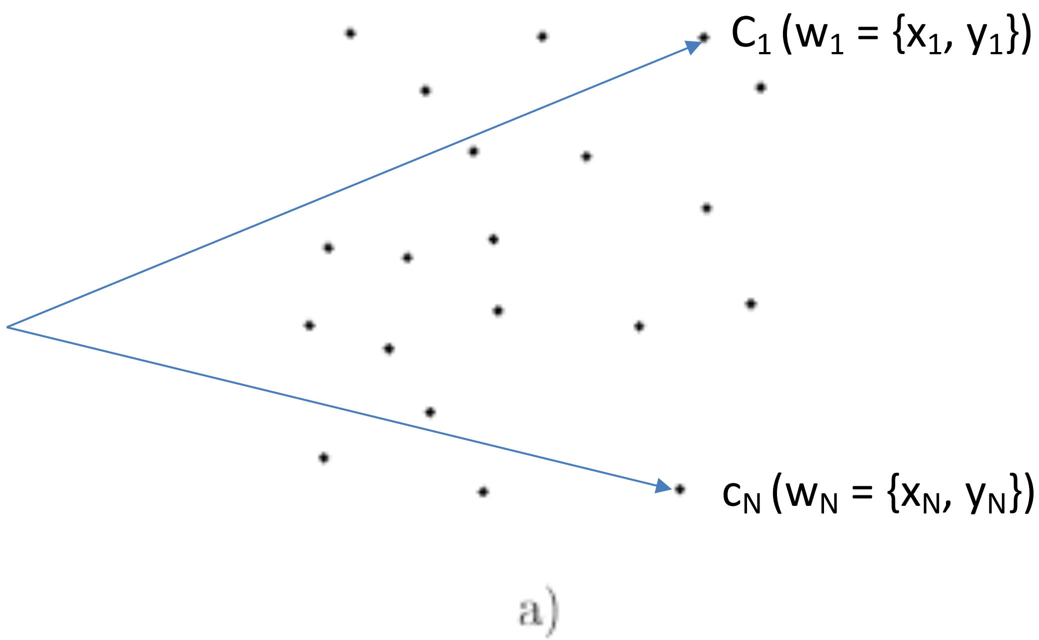
Example: Economy of a state, swarm of birds, …

Vector based Graph Neural Networks

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Structure

- The network state is defined by N n-dimensional reference vectors.
- The input feature input space is n-dimensional as well.
- A network consists of $A = \{c_1 \ldots c_N\}$ neurons / units.
- Each unit c is connected to a reference vector $\mathbf{w}_c \in \mathrm{R}^n$



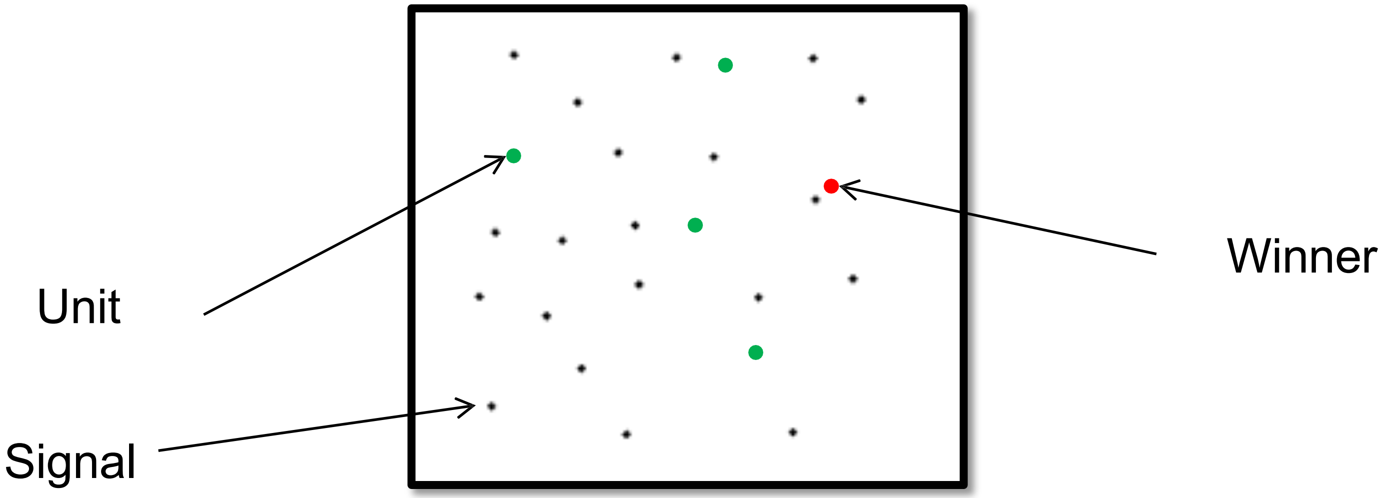
Vector based Graph Neural Networks

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Winner

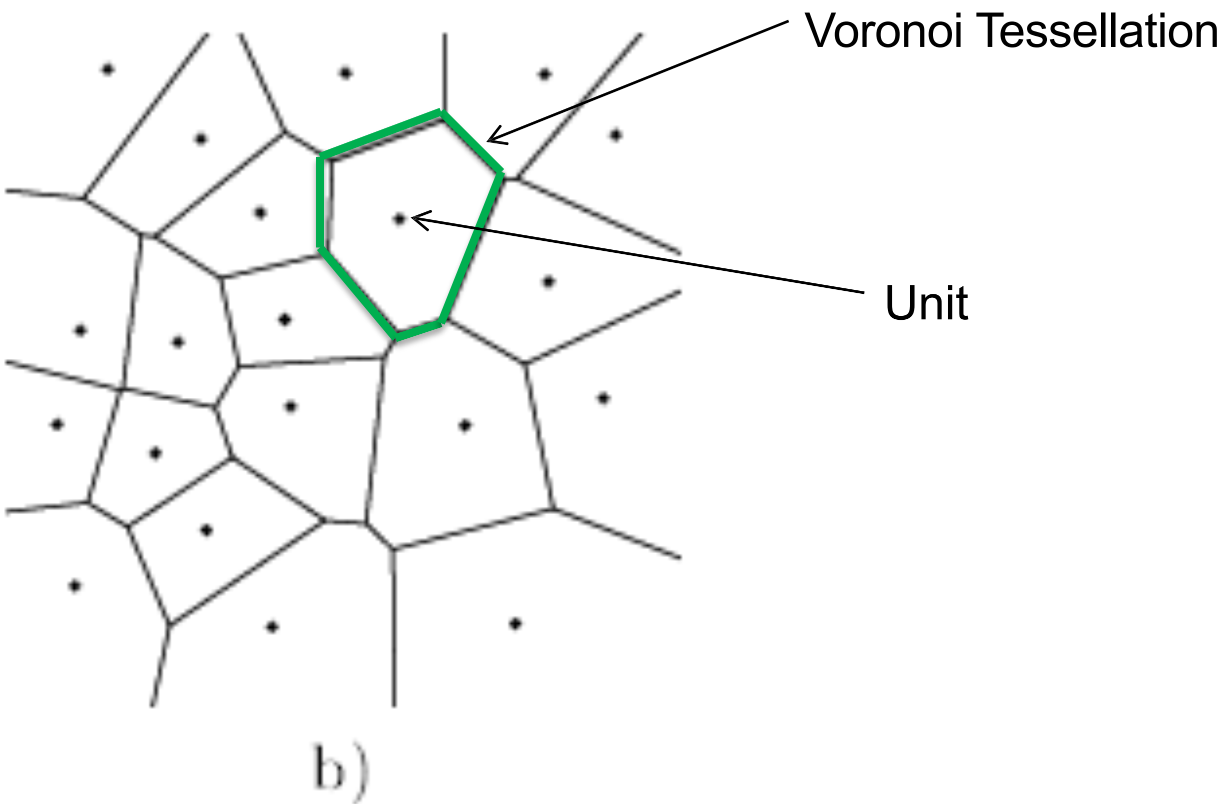
The winner is the unit with the reference vector next to the input signal:

$$s(\boldsymbol{\xi}) = \arg\, \min_{c \in \mathcal{A}} \|\boldsymbol{\xi} - \mathbf{w}_c\|$$



Vector based Graph Neural Networks

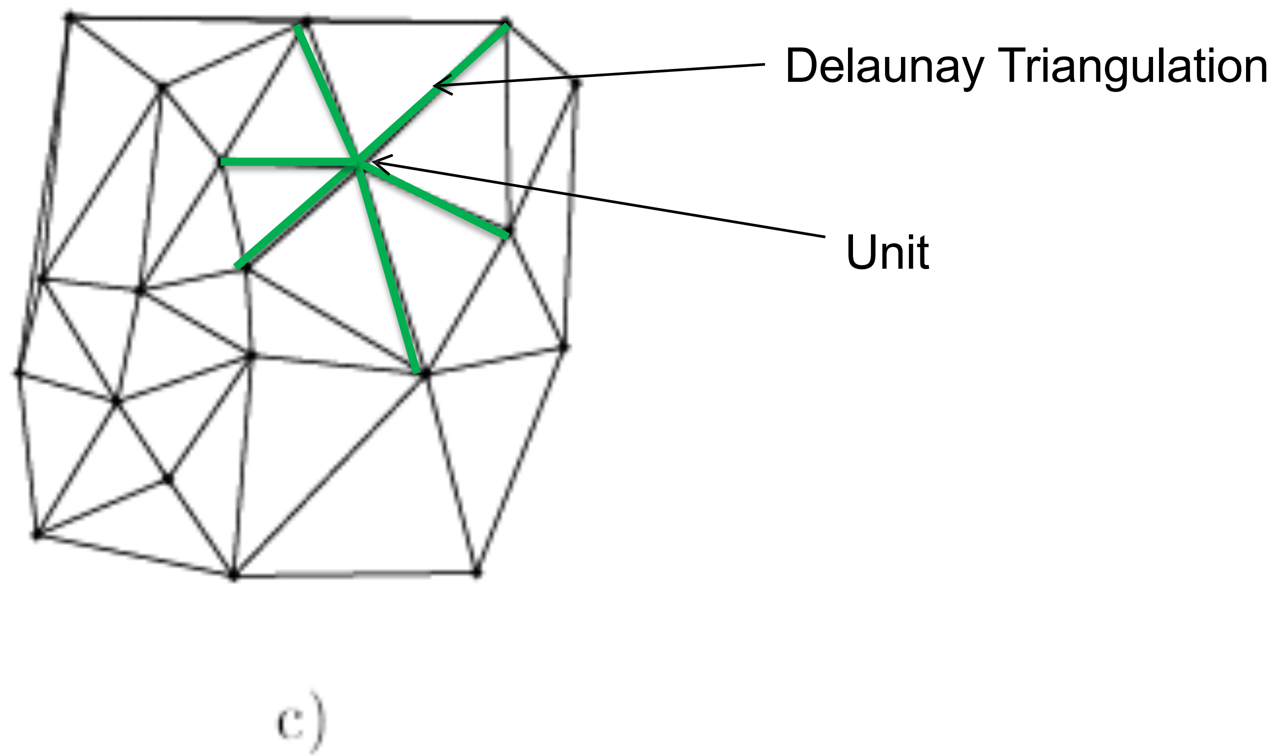
Voronoi Tessellation



The Voronoi tesselation defines the region in which a unit is a winner.

Vector based Graph Neural Networks

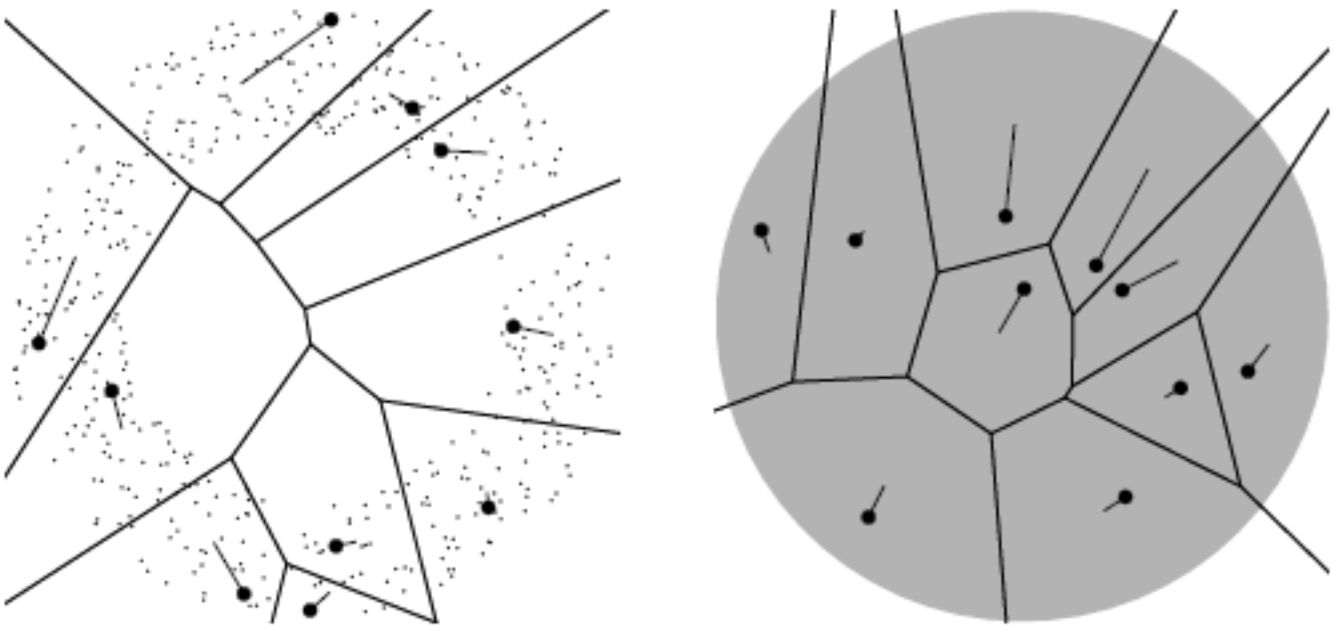
Delaunay Triangulation



The Delaunay triangulation defines the neighborhood in which a unit is a winner.

Vector based Graph Neural Networks

Learning by Voronoi Tessellation

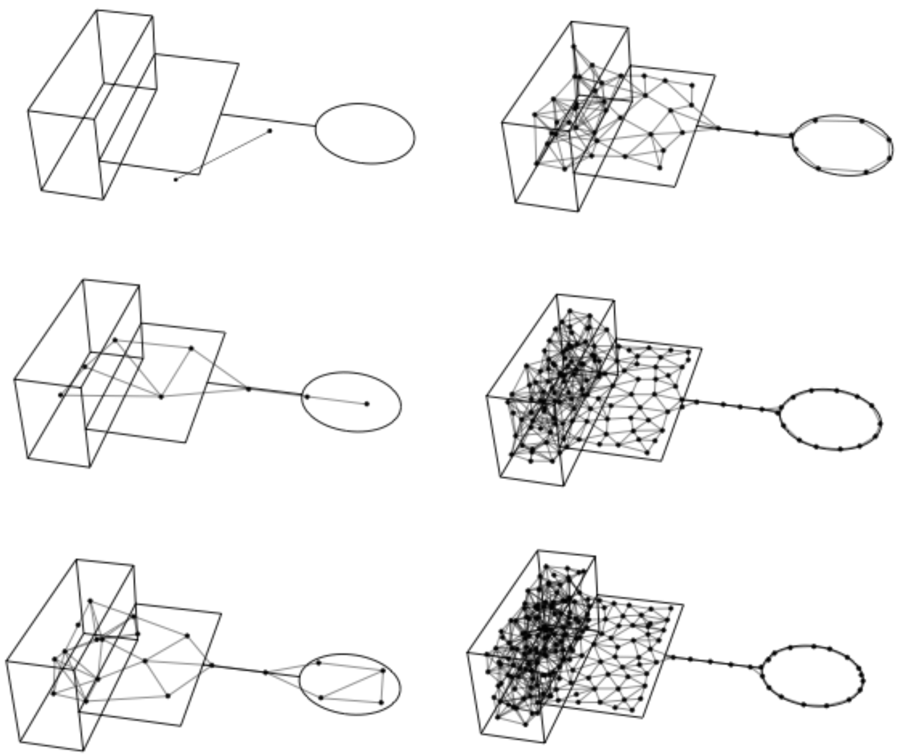

a) endliche Datenmenge   b) kontinuierliche W-Verteilung

Move the reference vector of the winner unit into the direction of the signal.

Growing Neural Gas (GNG)

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

The neural gas is a simple algorithm for finding optimal data representations based on feature vectors. The algorithm was coined "neural gas" because of the dynamics of the feature vectors during the adaptation process, which distribute themselves like a gas within the data space.

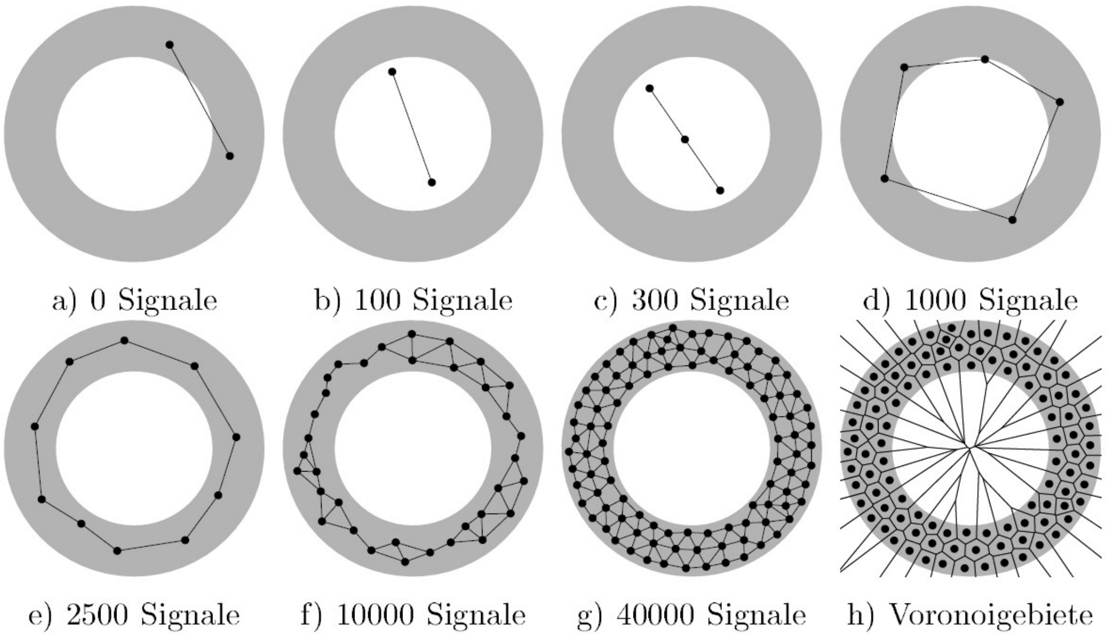


Unsupervised Learning

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

By use of **unsupervised training** we may
- fit probability density functions without a-priori knowledge of truth
- identify patterns by examination of density variations

Self organizing neural networks
e.g. Growing Neural Gas (GNG)



Hebbian competitive learning
+
Adaptation of reference vectors
+
Dynamic growth

Example:
GNG learns ring-shaped probability density function

Description of the Approach

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

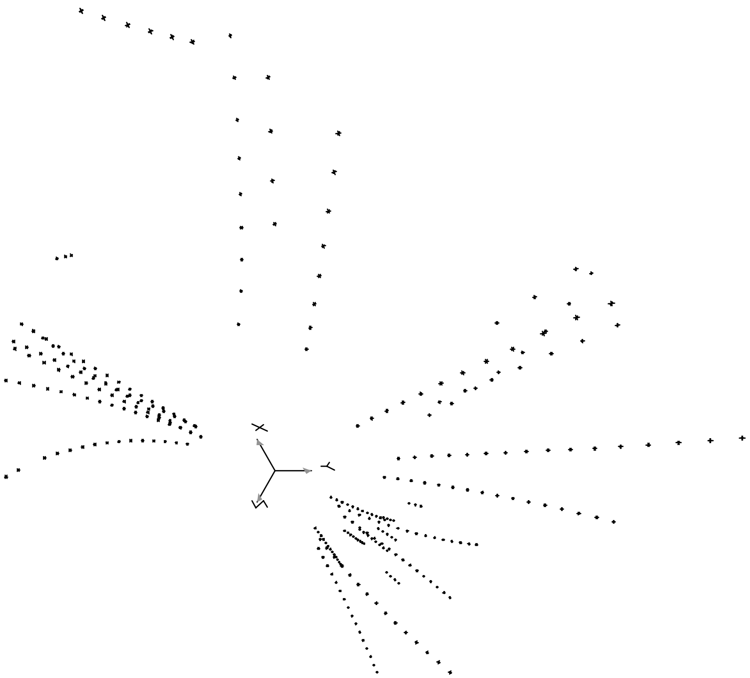
GNG is developed as a self-organizing network that can dynamically increase and remove the number of neurons in the network. A succession of new neurons is inserted into the network every λ iterations near the neuron with the maximum accumulated error. At the same time, a neuron removal rule could also be used to eliminate the neurons featuring the lowest utility for error reduction:

1. GNG starts with a set A of two units a and b at random positions w_a and w_b in R_n
2. In the set A find two nearest neighbors s_1 and s_2 to the input signal x.
3. Connect s_1 and s_2, with an edge and set the edge age to zero.
4. Adjust the positions of s_1 and its neighborhood by a constant times $(x-s_1)$. (ε_b for s_1 and ε_n for the neighborhood)
5. Remove edges in the neighborhood that are older than a_{max}.
6. Place a new node every λ cycles between the node with greatest error and its nearest neighbor.
7. Reduce error of the node with the maximum error and its nearest neighbor by α%, and add the removed error to the new node.
8. Reduce error of all nodes by a constant (β) times their current error.

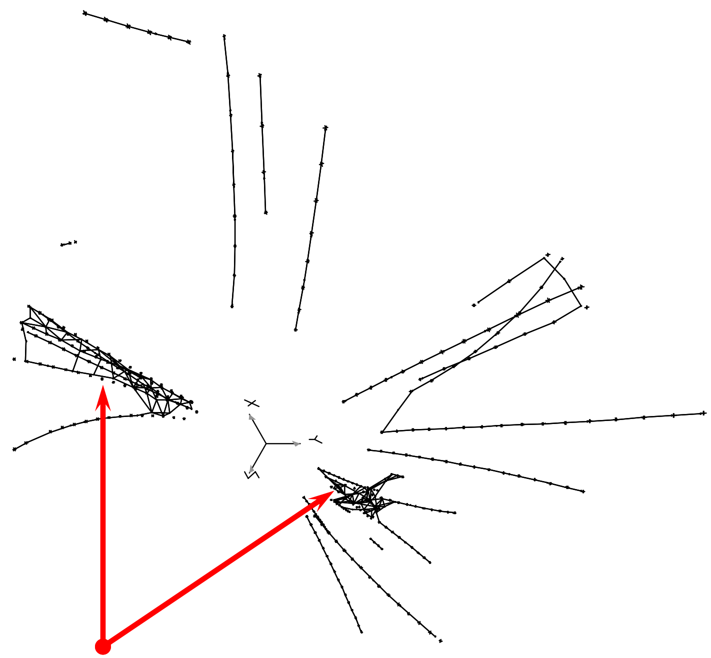
Tracking Example

Intuition: Use a GNG self organizing neural network to grow along particle trajectories

- Move and connect nodes according to the density of measured points



Example: 22 Tracks, 309 Points



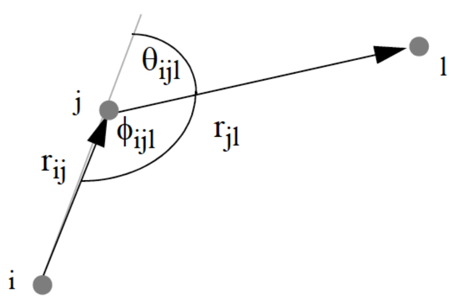
Convergence of training after 60 epochs
Problem: crossover connections between tracks in regions of high track density

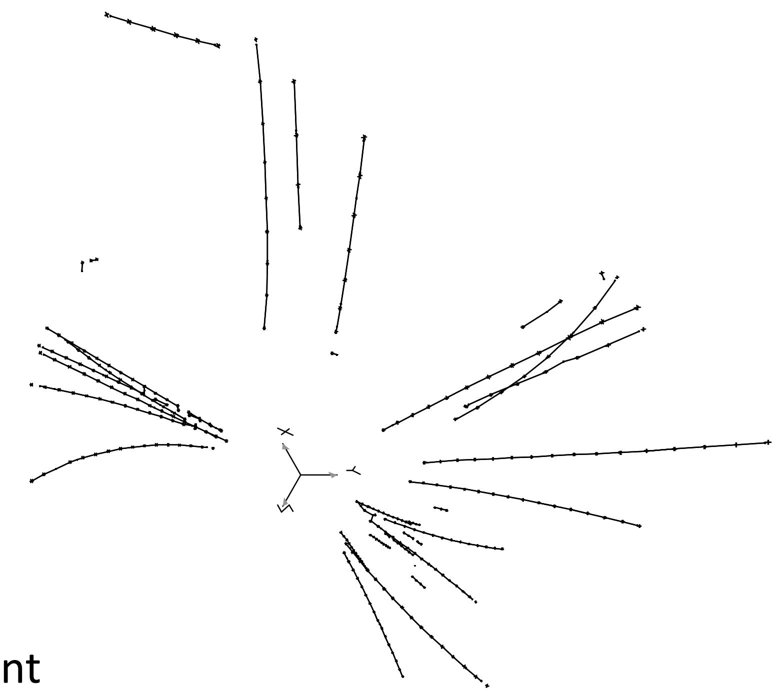
Tracking Example (cont‘d)

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

How to get straight tracks?

- insert new nodes at the neighboring measured point
- introduce a „rotor model“ to support the track representation
- remove connections with large angle to suppress crossover connections: require $|\cos\Phi| < \varepsilon$



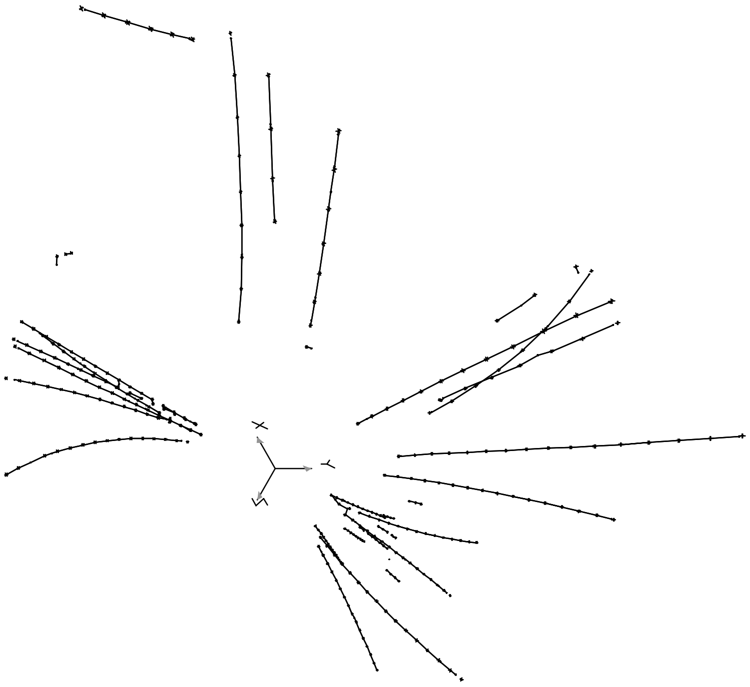


Challenges

- Difficult to handle high track density
- Compute intense, training happens for each event

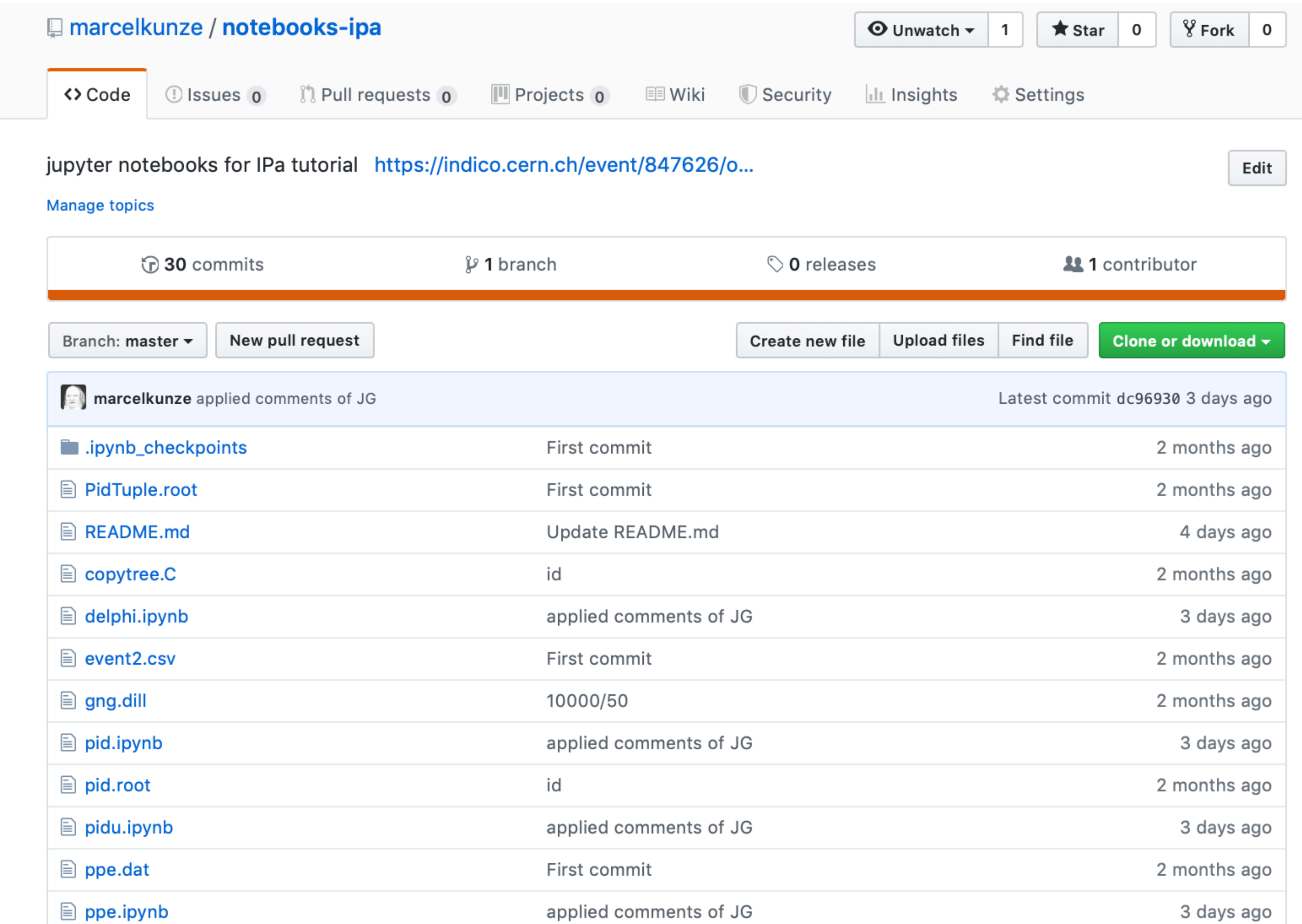
Live Demonstration

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386



Jupyter Notebooks

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386



https://github.com/marcelkunze/notebooks-ipa

Machine Learning Notebooks

This set of jupyter notebooks supports the machine learning tutorials prepared for the workshop Learning to Discover : Advanced Pattern Recognition at Institute Pascal / Paris-Saclay University.

delphi.ipynb - Tracking by growing a neural gas (Unsupervised training)

This tutorial demonstrates the use of a Growing Neural Gas (GNG) network to learn the topology of an event as measured by the Delphi TPC at CERN. Learning of the tracks happens in unsupervised mode ie. implicitely driven by the density of the measured space points without knowledge of ground truth.

Data files: event2.csv

ppe.ipynb - Tesselation of a Dalitz plot (Unsupervised training)

This tutorial analyzes the density of a Dalitz plot of the antiproton proton annihilation into the 3 particle final state π π η (data from Crystal Barrel experiment at CERN). A GNG network adapts the probability density function of the plot yielding Voronoi regions of the same statistics.

Data files: ppe.dat

pid.ipynb - PID Classification with neural networks (Supervised training)

This example illustrates the classification of particle types using tensorflow/keras neural networks. The supervised training is based on a Multilayer Perceptron (MLP) with labeled MC generated data of BaBar.

Data files: PidTuple.root

pidu.ipynb - PID Classification with neural networks (Unsupervised training)

This example illustrates the classification of particle types using tensorflow/keras neural networks. The unsupervised training is based on a Growing Neural Gas (GNG) network with unlabeled MC generated data of BaBar.

Data files: PidTuple.root

Machine Learning Software: Neural Network Objects

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Neural Network Objects (NNO) is a C++ class library for Machine Learning based on the ROOT framework

Supervised models
- Multi-Layer Perceptron (TMLP, TXMLP)
- Fisher Discriminant (TFD)
- **Supervised Growing Cell Structure (TSGCS)**
- **Supervised Growing Neural Gas (TSGNG)**
- Neural Network Kernel (TNNK)

Unsupervised models
- Learning Vector Quantization (TLVQ)
- **Growing Cell Structure (TGCS)**
- **Growing Neural Gas (TGNG)**

Published on https://github.com/marcelkunze/rhonno

Takeaway Messages

Graph Networks:

- Very natural data representation for a lot of scientific problems.
- Promising performance in pattern recognition tasks.
- Allow to map high dimensional input to lower dimensions preserving neighborhood relationship.

Unsupervised Training:

- Purely data driven without knowing the a-priori ground truth.
- The analysis of the results may be challenging (k-means,…).
- Training often happens during productive operations: Computationally intensive task that requires further work on scaling.
- Allows for permanent alignment of graph structure to changes in probability density function of feature vectors (e.g. compensate the shifting calibration of a measurement device).

Literature

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

M.Kunze: **Jupyter notebooks to support the tutorial**
https://github.com/marcelkunze/notebooks-ipa.git

M.Kunze (aka cloudkitchen): **TrackML accuracy phase, 3rd place winning solution**
https://github.com/marcelkunze/trackml.git

Z.Wu et al: **A Comprehensive Survey on Graph Neural Networks**
arXiv:1901.00596v3 8 Aug 2019

X.Ju: **HEP.TrkX Charged Particle Tracking using Graph Neural Networks,** Connecting the Dots 2019
https://indico.cern.ch/event/742793/contributions/3274328/

M.Kunze: **Growing Cell Structure and Neural Gas - Incremental Neural Networks**, New computing techniques in physics research IV (Ed. B. Denby), World Sci., 1995

B Fritzke: **A growing neural gas network learns topologies**, Advances in neural information processing systems, 625-632

Dr. Marcel Kunze
marcel.kunze@uni-heidelberg.de
Im Neuenheimer Feld 293 / 106
D-69120 Heidelberg

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386