Neural Networks Architectures
Practical Hands-On

2nd Terascale School of Machine Learning
1-11 March, 2021


Jean-Roch Vlimant (California Institute of Technology)
jvlimant@caltech.edu @vlimant

Outline

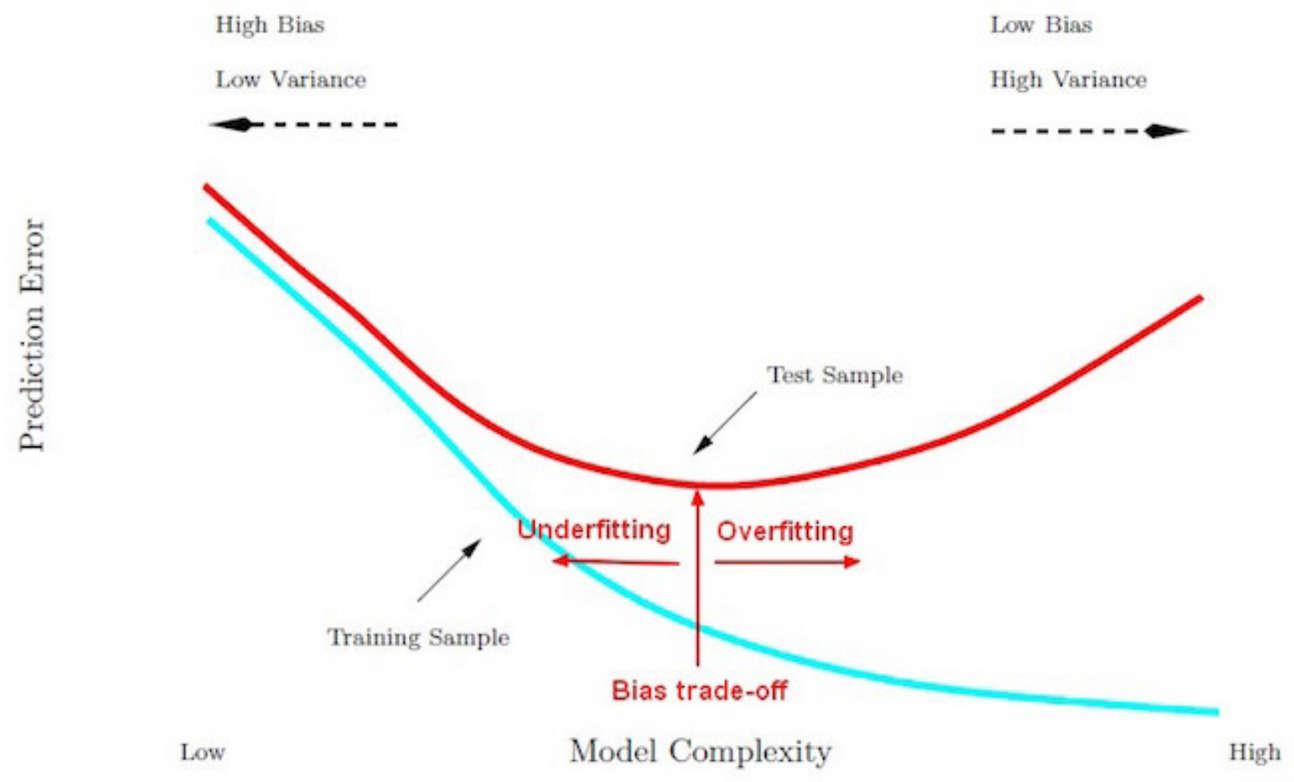
- Model convergence.
- Model comparison with K-folding.
- Data efficiency of model types.
- Arch search with Bayesian optimization.
- Graph network for particle-flow reconstruction.
- Jet tagging with interaction network.
- Vertexing with set2graph models.

Github repository for the hands-on
https://github.com/vlimant/NNArchTeraScale2021

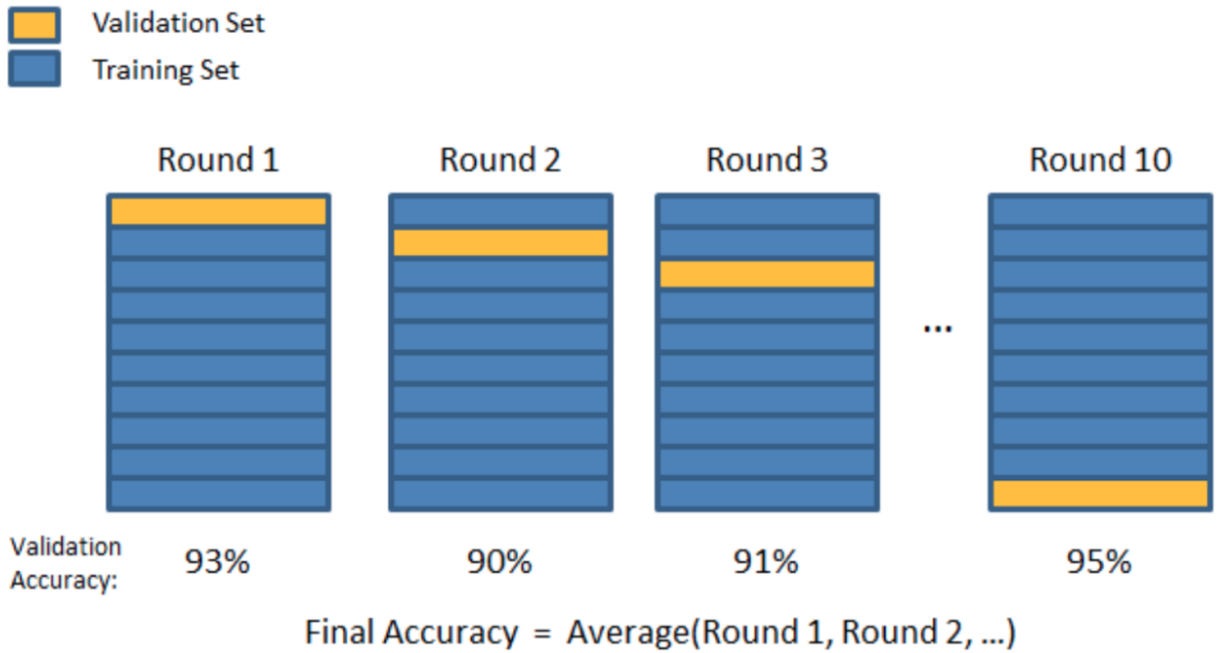
Model Comparison

Generalization

- Systematic error ≡ bias
- Sensitivity of prediction ≡ variance
- A good model is a tradeoff both



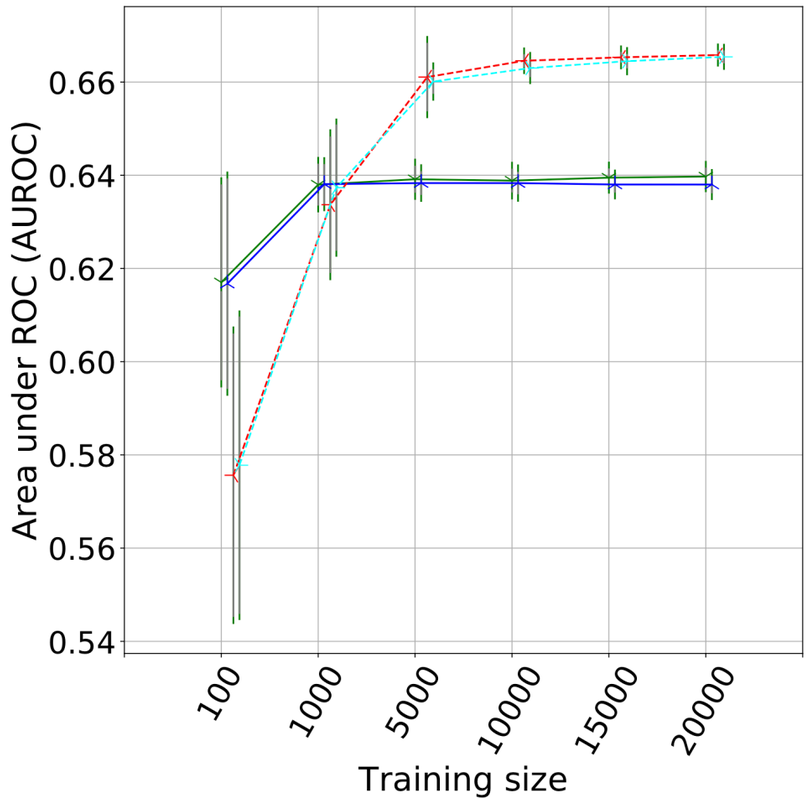
Cross Validation



- Model selection requires to have an estimate of the uncertainty on the metric used for comparison
- K-folding provides an un-biased way of comparing models
- Stratified splitting (conserving category fractions) protects from large variance coming from biased training
- Leave-one-out cross validation : number folds ≡ sample size

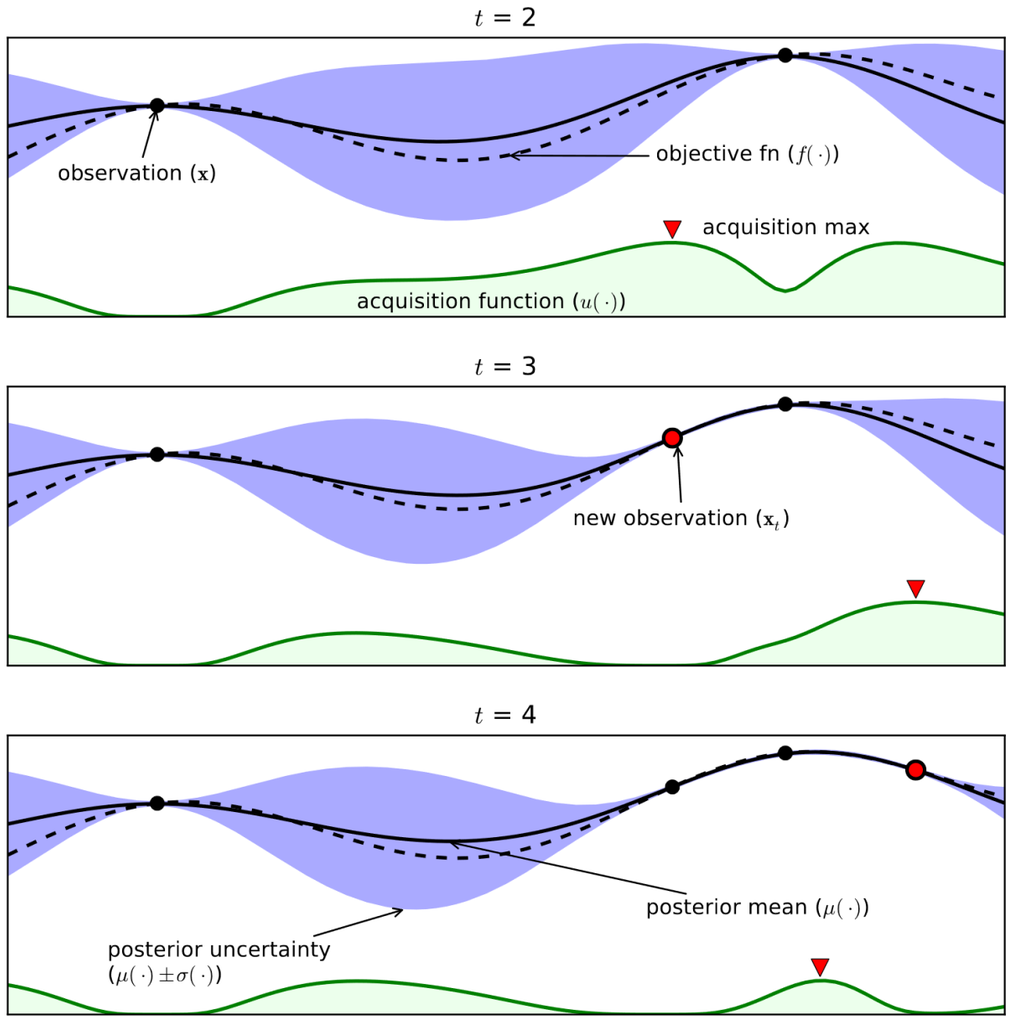
Need for Data

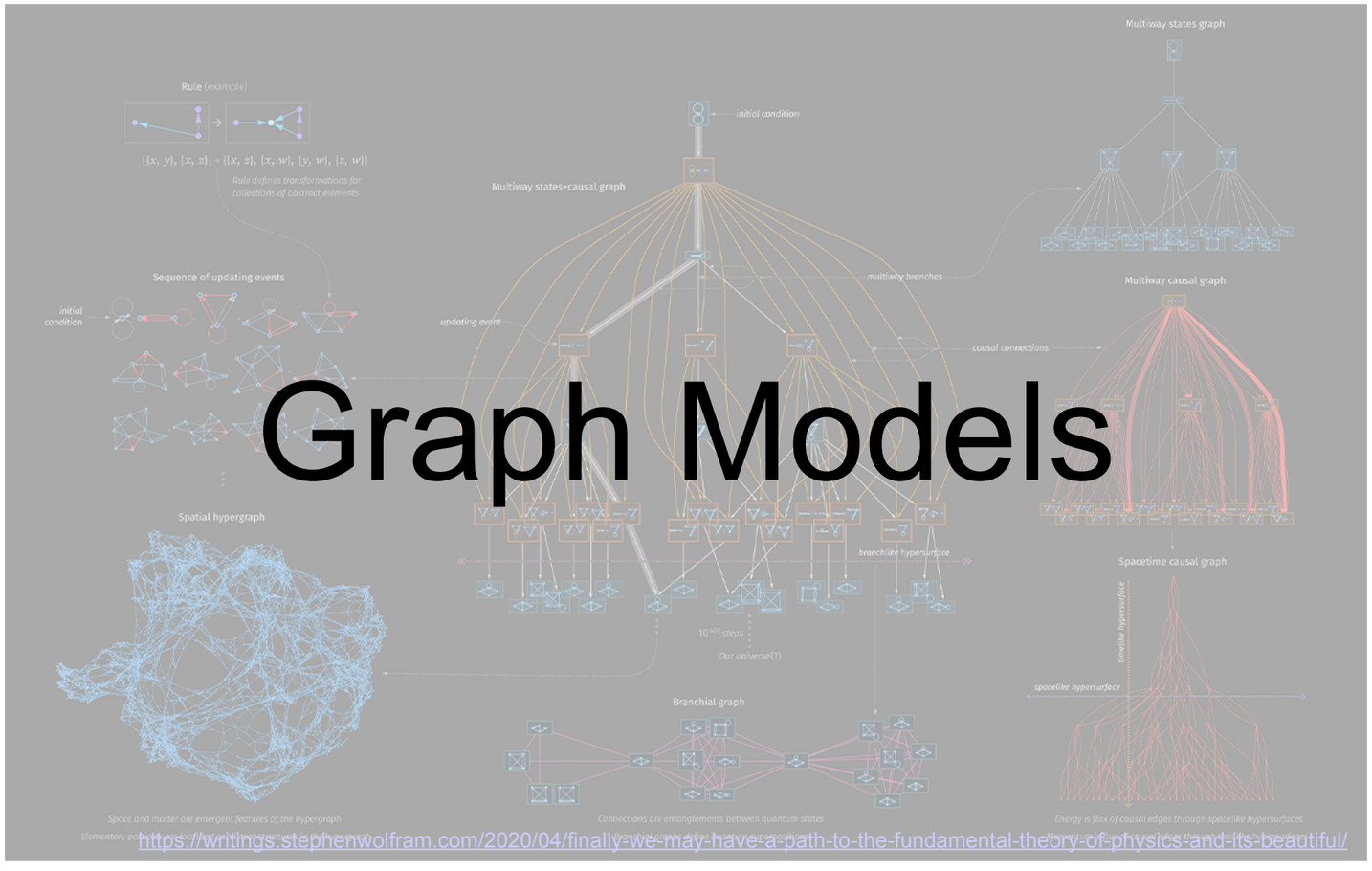
- "What is the **best performance one can get** ?" rarely has an answer
- When comparing multiple models, one can answer "what is the **best of these models, for this given dataset** ?"
- It does not answer "what is the **best model at this task** ?"



Bayesian Optimization

- Applicable to optimize function **without close form** and that are **expensive to call** (numerical gradient impractical)
- Approximate the objective function with **Gaussian processes** (GP)
- Start at random points, then sample according to optimized acquisition function
  - Expected improvement
    - $EI(x) = -E(f_{GP}(x) - f(x_{best}))$
  - Lower confidence bound
    $LCB(x) = \mu_{GP}(x) + \kappa\sigma_{GP}(x)$
  - Probability of improvement
    - $PI(x) = -P(f_{GP}(x) \geq f(x_{best}) + \kappa)$



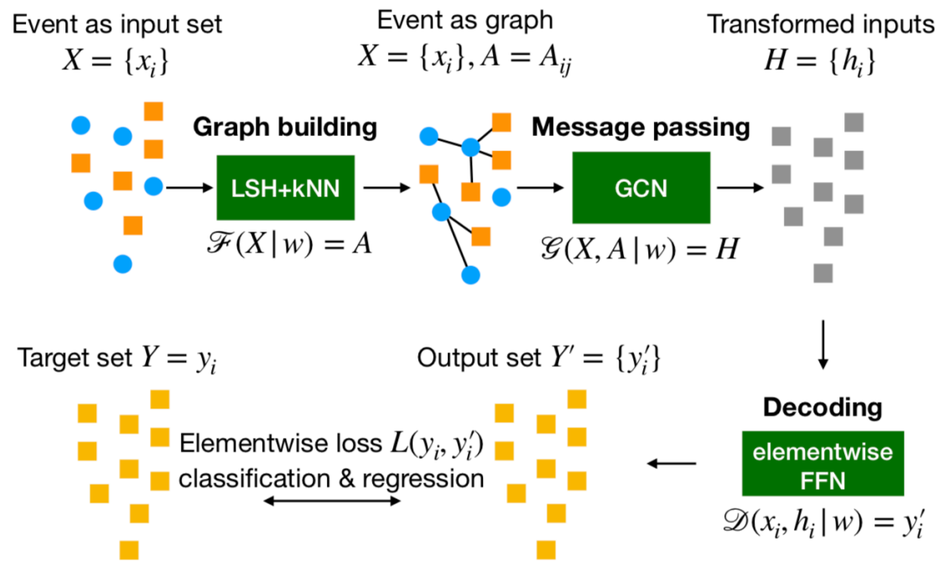


Graph Models

https://writings.stephenwolfram.com/2020/04/finally-we-may-have-a-path-to-the-fundamental-theory-of-physics-and-its-beautiful/

Particle Flow Reconstruction

MLPF: Efficient Machine-Learned Particle-Flow Reconstruction Using Graph Neural Networks [2101.08578]

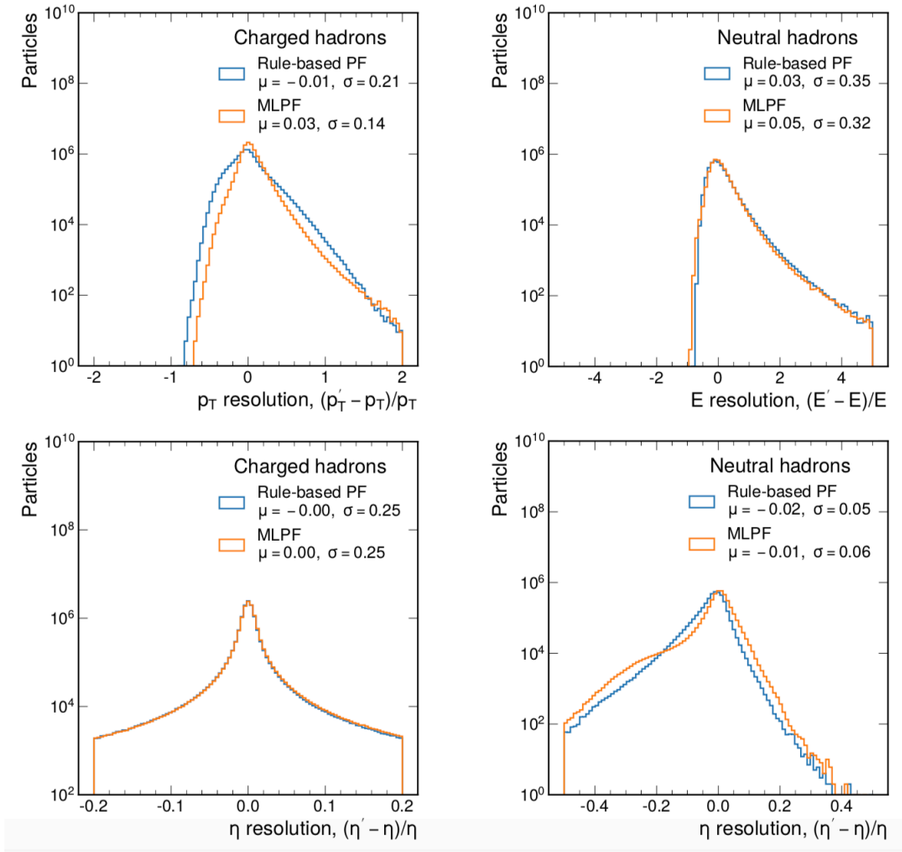


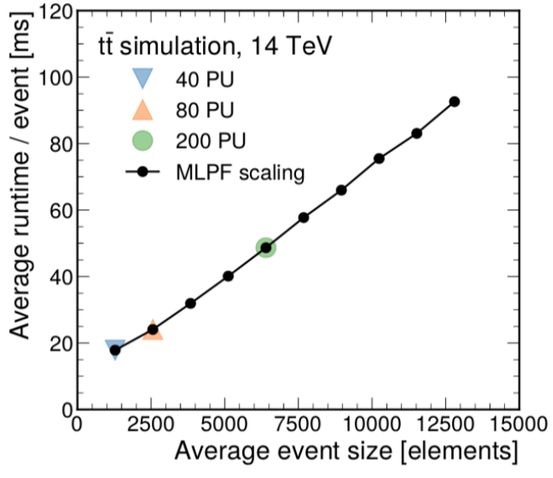
$x_i = [\text{type}, p_T, E_{ECAL}, E_{HCAL}, \eta, \phi, \eta_{outer}, \phi_{outer}, q, \ldots], \text{type} \in \{\text{track, cluster}\}$

$y_i = [\text{PID}, p_T, E, \eta, \phi, q, \ldots], \text{PID} \in \{\text{none, charged hadron, neutral hadron}, \gamma, e^{\pm}, \mu^{\pm}\}$

$h_i \in \mathbb{R}^N, N = 256$

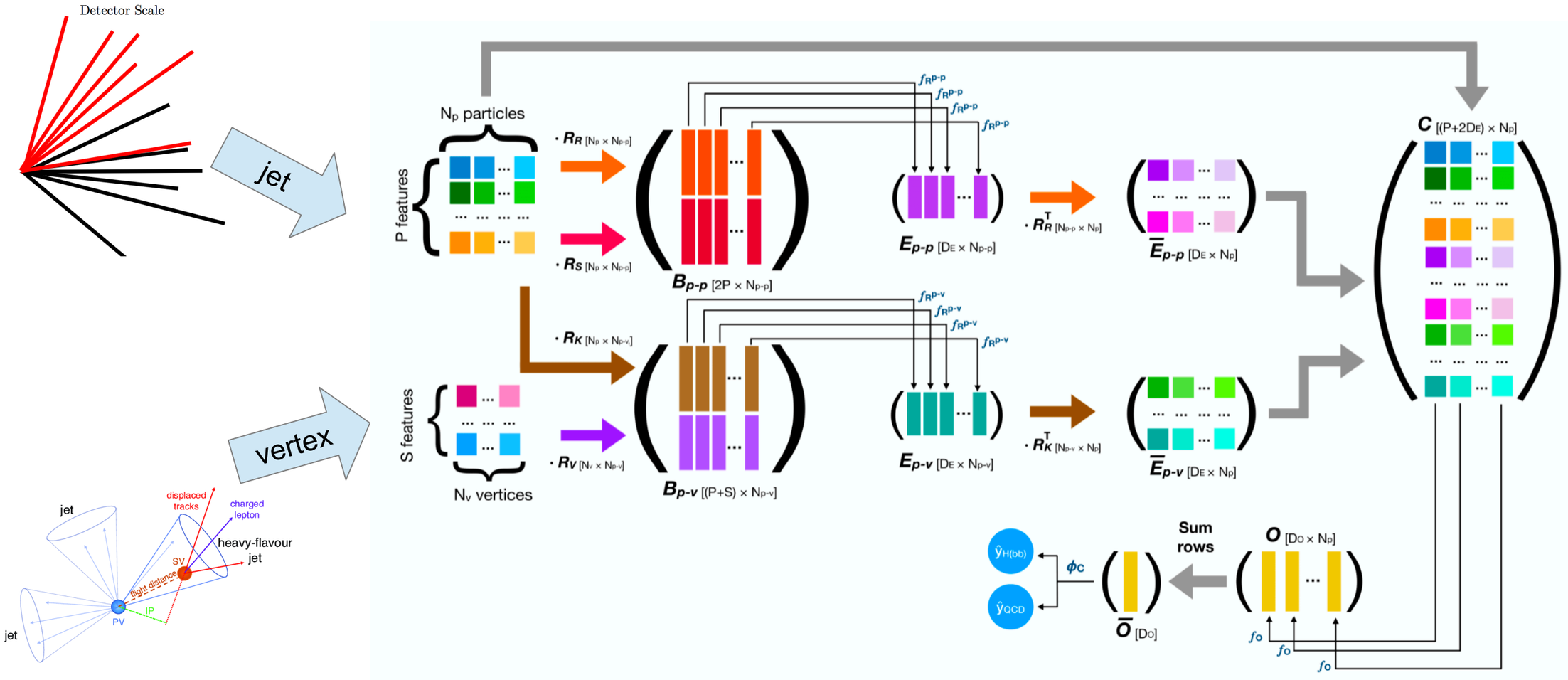
Trainable neural networks: $\mathcal{F}, \mathcal{G}, \mathcal{D}$





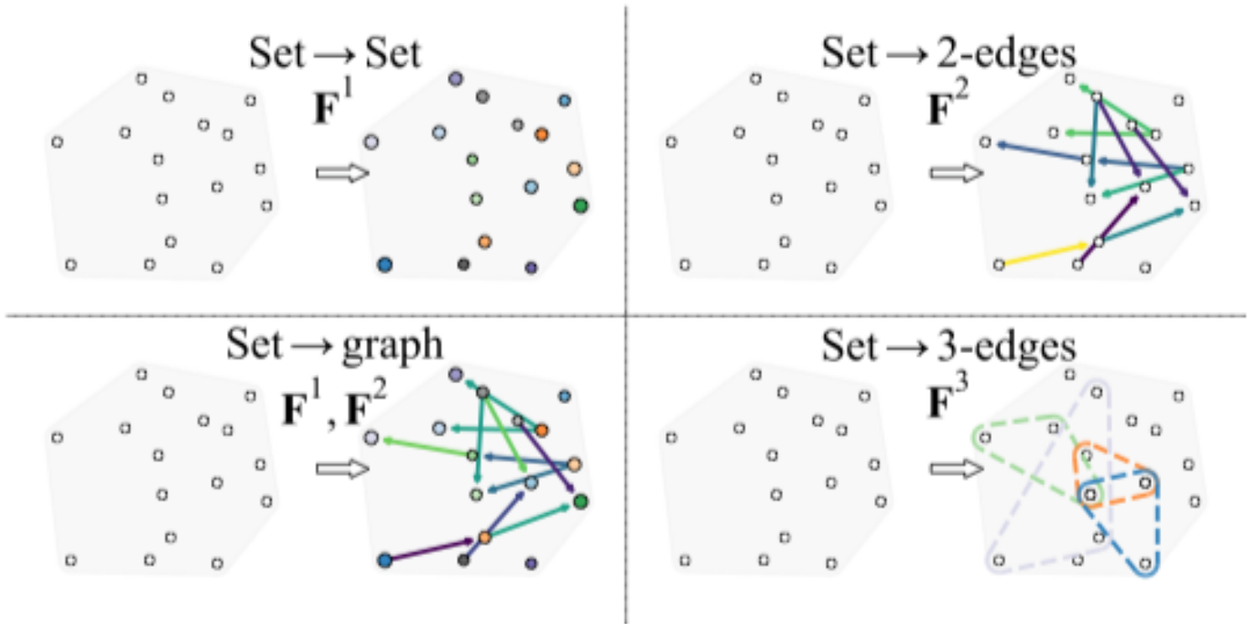
- Set of tracks & clusters in input
- Classify sub-set of graph nodes
- Regress parton kinematics
- Execution time linear with PU

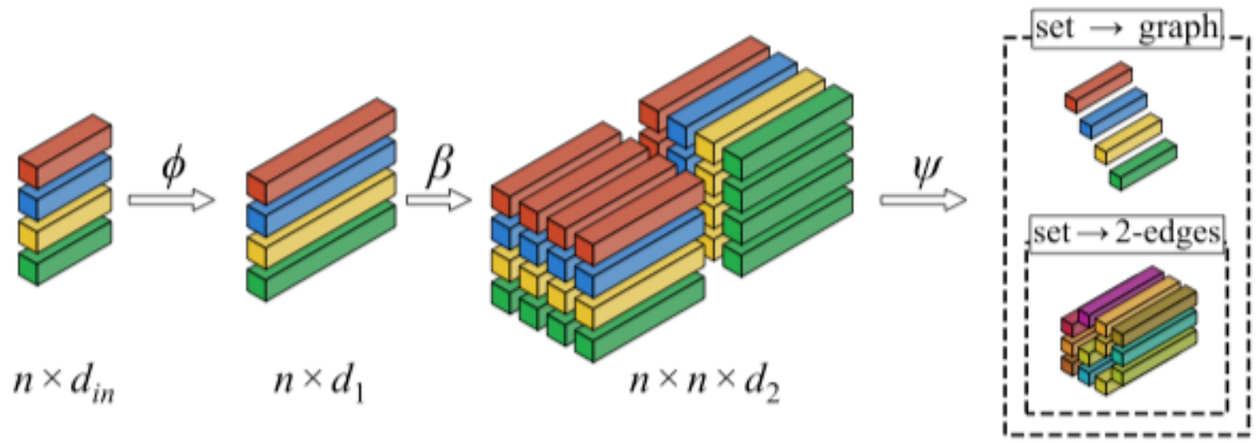
Jet-id with Graph Network



All particles of a jet, and vertex added on an all-to-all message passing graph network.
Graph-level classification (binary or multi-class)

Vertexing with set2graph





Learning graphs from sets, applied to vertexing
[2002.08772]