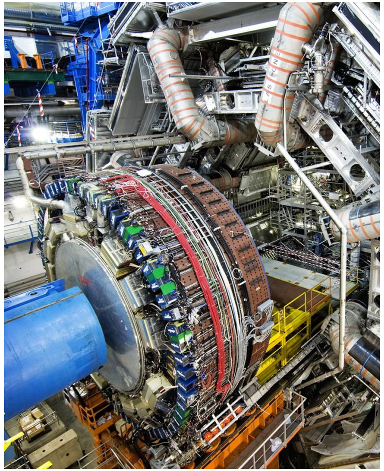
Interaction networks for the identification of boosted $H \to b\bar{b}$ decays

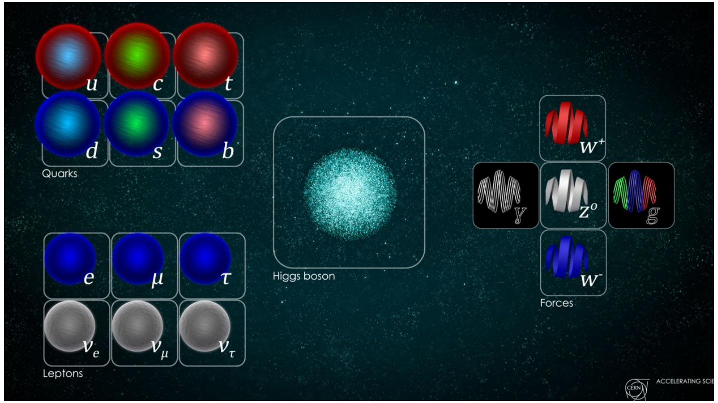
Arturo Sorensen, Danylo Drohobytsky, Fredy Ramirez, and Zihan Zhao

UC San Diego

Background

- Colliding ‘primary particles’, produces jets of quarks and gluons
- A hadron is composed of multiple quarks held together by strong force (exchange of gluons)
- Hadrons compose jets, understanding them provides insight about primary particles in the collision
    - Higgs produces b quarks of 1.5ps lifetime, creating a second identifiable vertex where the decay occurs

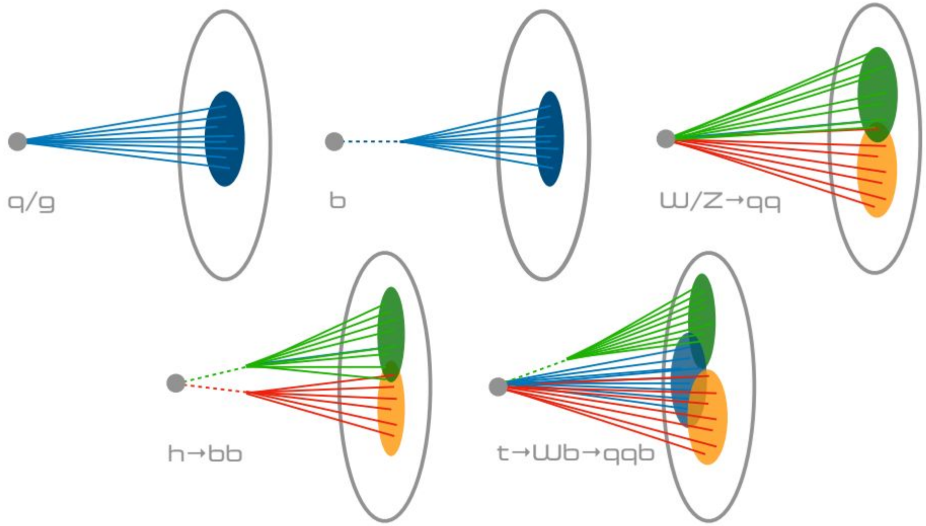




https://atlas.cern/Discover/Detector/Calorimeter
https://press.cern/resources/image/physics/infographics-gallery Rolf Landua

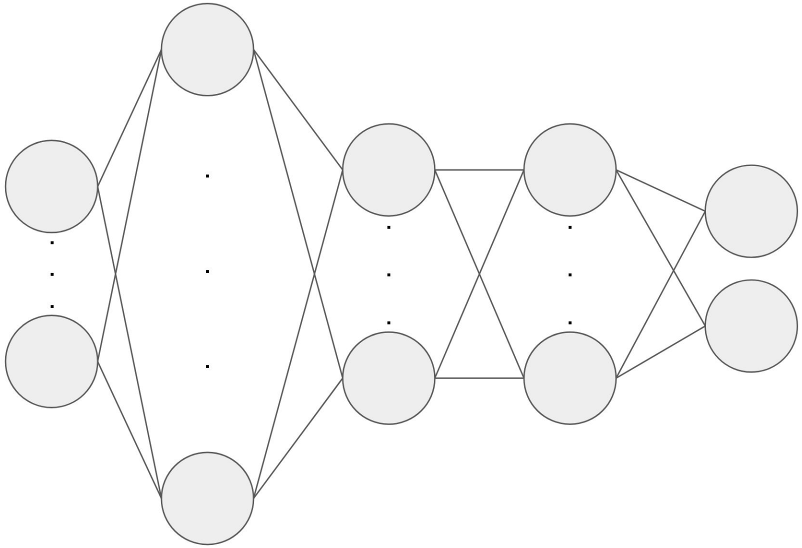
Pictorial Representation of Jets

- Bottom quarks have a unique jet signature because hadrons containing these bottom quarks have a long enough lifetime in order for there to be a detectable displacement from the point of particle collision and their decay. The dotted lines of the b jets represent this.
- This results in a secondary vertex (SV) displaced from the primary vertex (PV)
- The focus is on H→bb (Higgs boson decaying to bottom quark antiquark pair), as the goal is to achieve higher accuracy in jet tagging by correctly classifying the input jets as either H→bb jets (signal) or QCD jets (background)

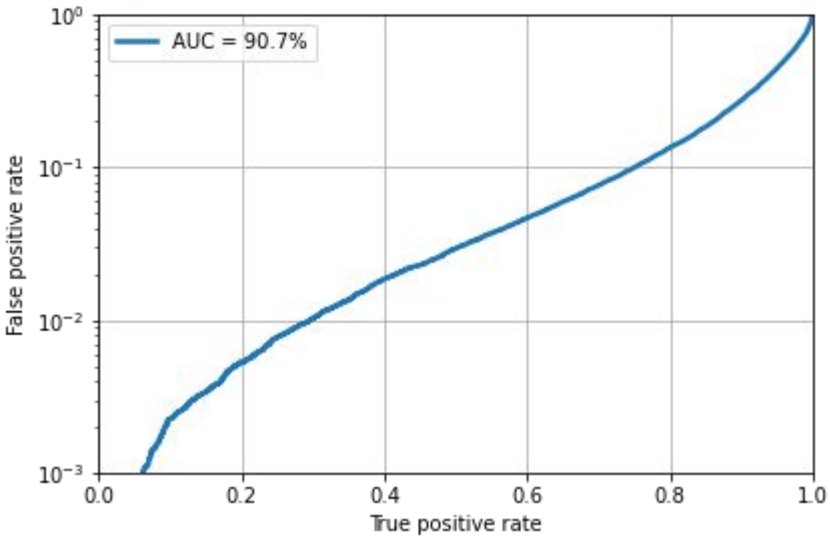


Benchmark Architecture

- Dense Keras model
- Input layer the same size as the number of features (in this case 27)
- Batch normalization
- Three hidden layers of sizes (64,32,32) with ReLU activation
- Output layer the same size as the number of labels (in this case 2) with softmax activation
- Trained using Adam optimizer
- Batch size of 1024
- For up to 100 epochs
- Enforcing early stopping on the validation loss with a patience of 10 epochs
- Categorical cross-entropy loss



Benchmark Performance

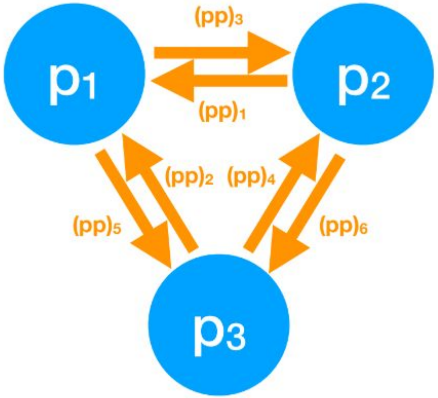


Graph Neural Network (GNN)

- While existing DL approaches have been successfully applied to jet tagging, particle jets involve multiple entities that are not easily encoded as images or lists.
  - Graphs provide a natural representation for such relational information.

- By placing charge particles and secondary vertices on a graph, the network can learn a representation of each particle-particle and particle-to-vertex interaction.
  - We can exploit this to categorize a given jet as a signal ($H \to b\bar{b}$) or background (QCD).

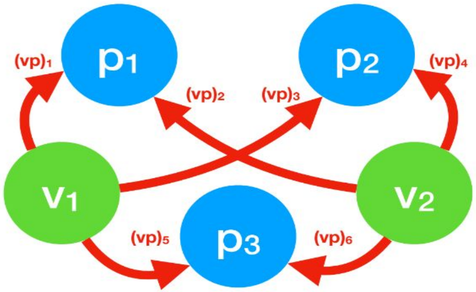
- The particle graph $\mathscr{G}_p$ is constructed by connecting each particle to every other particle through $N_{pp} = N_p\,(N_p\text{-}1)$ directed edges.
  - For the graph $\mathscr{G}_p$, a receiving matrix (R_R) and sending matrix (R_S) are defined.



$$R_R = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cccccc} (pp)_1 & (pp)_2 & (pp)_3 & (pp)_4 & (pp)_5 & (pp)_6 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

$$R_S = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cccccc} (pp)_1 & (pp)_2 & (pp)_3 & (pp)_4 & (pp)_5 & (pp)_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

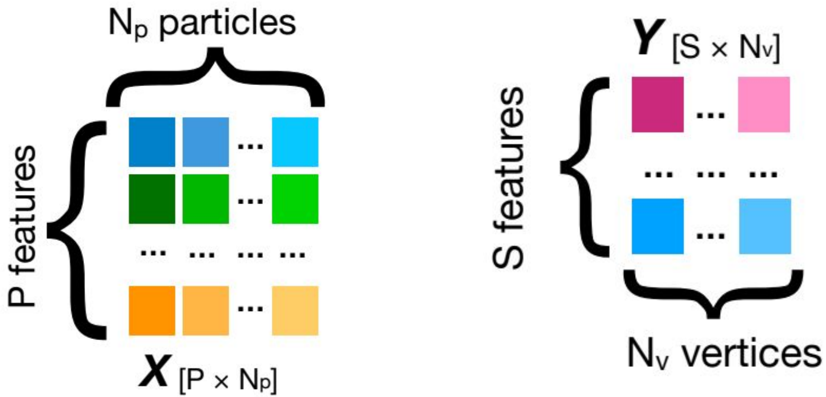
Particles and Vertices: Two Graphs



$$R_K = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cccccc} (vp)_1 & (vp)_2 & (vp)_3 & (vp)_4 & (vp)_5 & (vp)_6 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

$$R_V = \begin{array}{c} \\ v_1 \\ v_2 \end{array} \begin{array}{c} \begin{array}{cccccc} (vp)_1 & (vp)_2 & (vp)_3 & (vp)_4 & (vp)_5 & (vp)_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

- Similarly, a particle-vertex graph $\mathscr{G}_{pv}$ is constructed by connecting each vertex to each particle through $N_{pv} = N_p N_v$ directed edges.
  - We can also define matrices R_K and R_V, which connect particles and vertices.



- To setup the network between $\mathscr{G}_p$ and $\mathscr{G}_{pv}$, we use two input collections:
  - N_p particles, each represented by a feature vector of length P.
  - N_v vertices, each represented by a feature vector of length S.
- For a single jet, the input consists of an X matrix containing input features of charged particles and Y matrix containing the input features of the SVs.

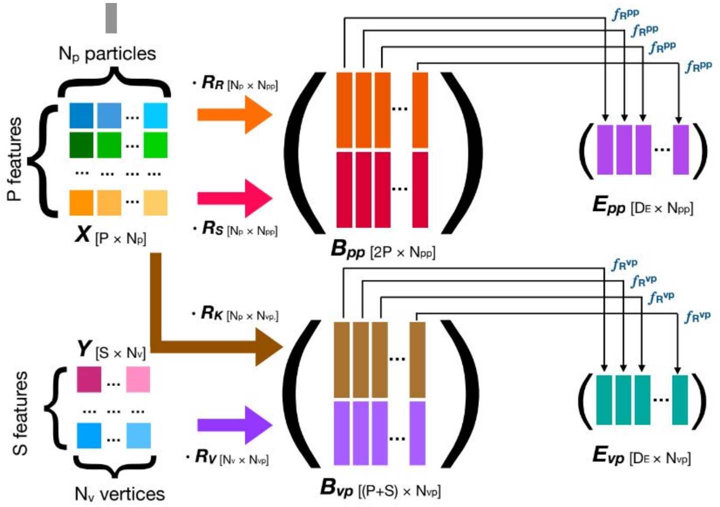
Interaction Network (IN) Model

- The particle feature matrix X is multiplied by the receiving and sending matrices R_R and R_S to build the particle-particle interaction feature matrix B_{pp}.
    - Similarly the particle-vertex interaction feature matrix B_{vp} is built (this uses X & Y matrix).

- These pairs are processed by the interaction functions f^{pp}_R and f^{vp}_R to build an internal representation of the particle-particle and particle-vertex interaction.
    - This results in an effect matrix E_{pp} and E_{vp}.

- The interaction functions f^{pp}_R and f^{vp}_R are expressed as a sequence of 3 dense layers with ReLu activation function after each layer.

- Although not shown here, we propagate the particle-particle (E_{pp}) and particle-vertex (E_{vp}) interaction back to the particles receiving them.



$$B_{pp} = \begin{pmatrix} X \cdot R_R \\ X \cdot R_S \end{pmatrix}$$

$$B_{vp} = \begin{pmatrix} X \cdot R_K \\ Y \cdot R_V \end{pmatrix}$$

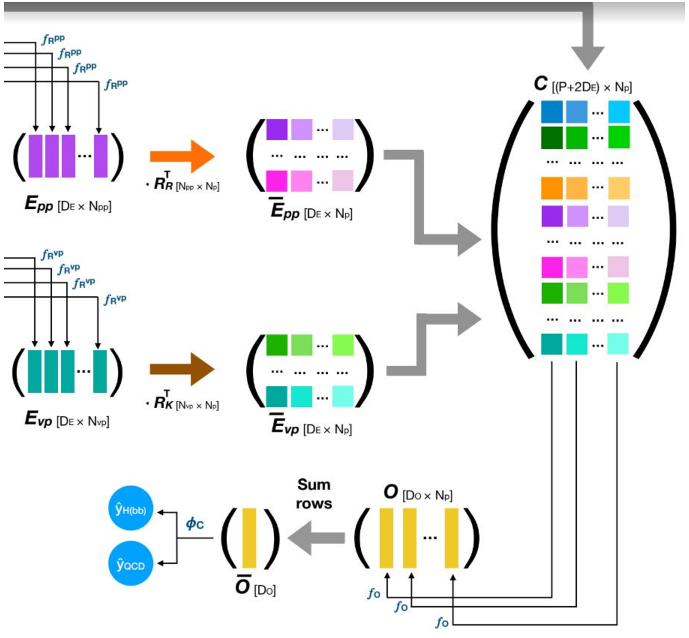
Interaction Network (IN) Model (cont.d)

- The next step consists of building the C matrix, by combining the input information for each particle (X) with the learned representation of the particle-particle ($\overline{E}_{pp}$) and particle-vertex ($\overline{E}_{vp}$) interactions.

- The final aggregator (f_O) combines the input and interaction information to build post-interaction representation of the graph, summarized by the matrix O.

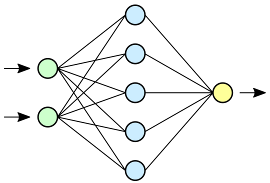
- The final function that computes the classifier output preserves the permutation invariance of the input particles and vertices.
  - Here the sum along each row (corresponding to a sum over particles) of O is done to produce a feature vector $\overline{O}$.

- From here $\overline{O}$ is passed on to function φ_C, which produces the output of the classifier.



$$C = \begin{pmatrix} X \\ \overline{E}_{pp} \\ \overline{E}_{vp} \end{pmatrix}$$

Data processing



Download root files	Process root files	Read .h5 files during training and testing

- Source: http://opendata.cern.ch/record/12102
- Store in `/home/ziz078/teams/group-2/Reproduction_of_IN/data`
-

- Run 'make_dataset.py'
- Benefits:
  - a. Choose desired features
  - b. Facilitate training and testing
  - Preprocess data
- Store processed data in `/home/ziz078/teams/group-2/Reproduction_of_IN/data/processed`

- Load feature arrays, truth labels, etc.

Training the IN

- Hyper parameters

Hyper parameter	Value
Batch size	512
Number of epochs	70
Training dataset size	300k
Validation dataset size	100k

- Optimization algorithm: Adam
  - Learning rate: 1e-4

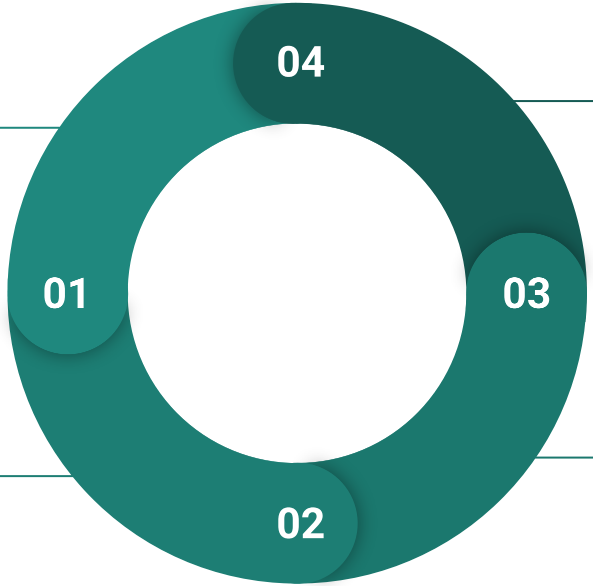
One epoch of training the IN

Training

- Load the training data in batches
- Forward pass
- Back propagation

Validation

- Load the data in batches
- Forward pass



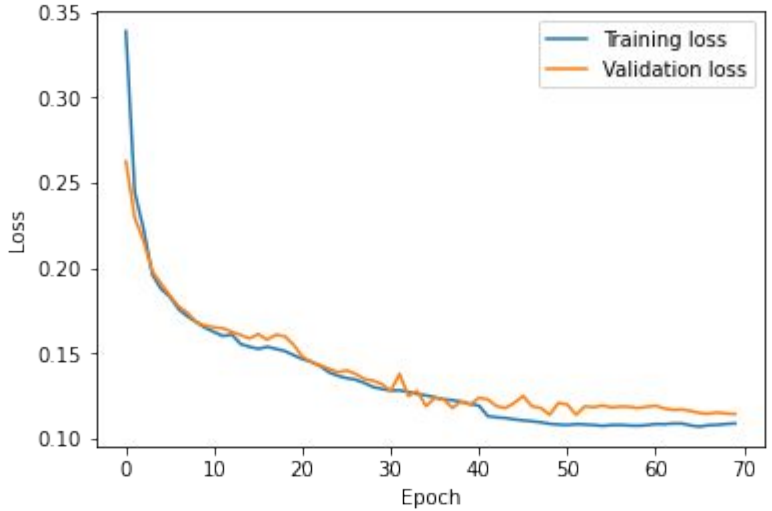
Display results

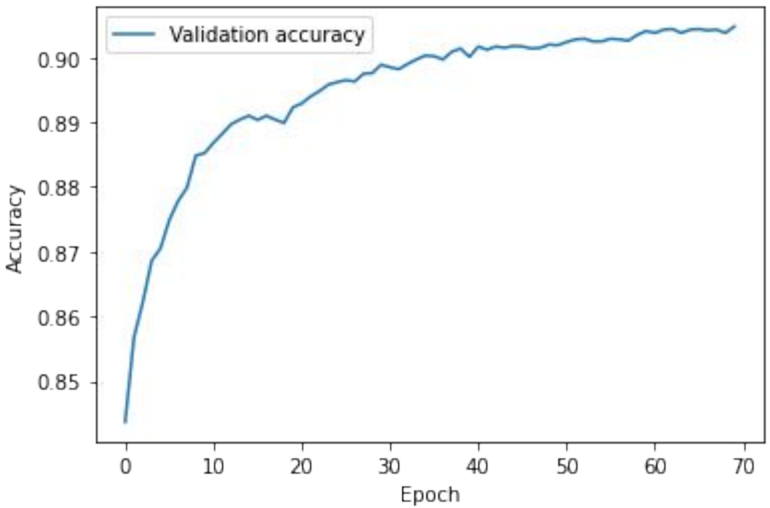
- Training loss
- Validation loss
- Validation accuracy
- Time
- New best model

Save model

- Only save the model if the validation accuracy is higher than all previous epochs.

Training result





Evaluation

To be continued