Selective Background Monte Carlo Generation

feat. Deep Full Event Interpretation (T. Boeckh)

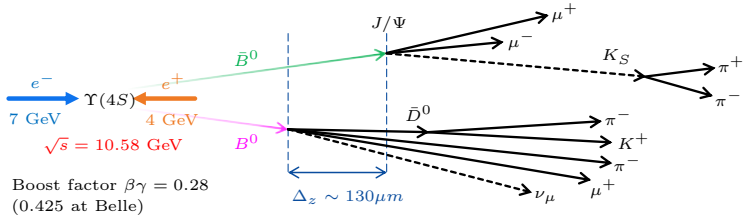
James Kahn (KIT, Prof. Bernlochner)
Andreas Lindner (LMU, Prof. Kuhr)
Kilian Lieret (LMU, Prof. Kuhr)
Emilio Dorigatti (LMU, Hackathon mentor)

October 1, 2019

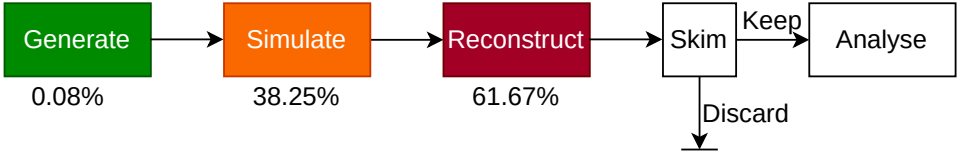
Dresden deep learning hackathon

Dresden deep learning hackathon

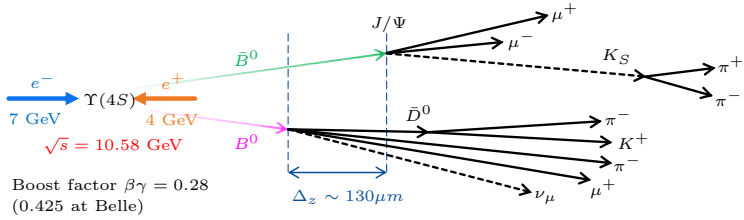
Project Overview



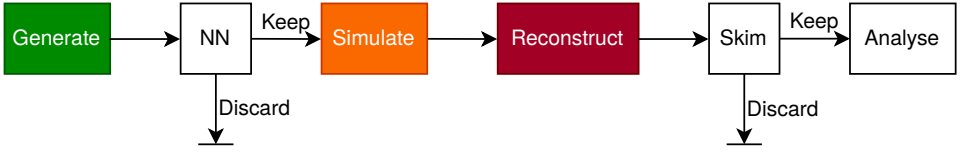
- Simulation of particle collisions computationally expensive, but most of the results are uninteresting and thrown away
- Idea: Can we figure out whether a collision is uninteresting at an early stage?



Project Overview



- Simulation of particle collisions computationally expensive, but most of the results are uninteresting and thrown away
- Idea: Can we figure out whether a collision is uninteresting at an early stage?
  - **Skip expensive steps**



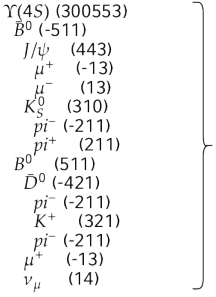
Dataset

Three categories of decay data:

- Full charged B meson reconstruction
- Full neutral B meson reconstruction
- Time-dependent CP violation

For each category:

$\sim$ 300,000 particle collision processes with binary classification labels



Feature	Definition
PDG code	Identifier of particle type and charge.
Mother PDG code	Particle parent PDG code.
Mass	Particle mass in GeV/c^2.
Charge	Electric charge of the particle.
Energy	Particle energy in GeV.
Momentum	Three momentum of the particle in Gev/c.
Production time	Production time in ns relative to $\Upsilon(4S)$ production.
Production vertex	Coordinates of particle production vertex.
Status bit	Bitmask representing MC production conditions.

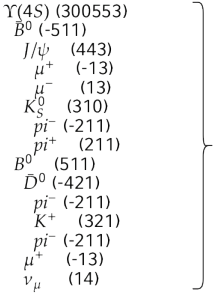
Dataset

Three categories of decay data:

- Full charged B meson reconstruction
- Full neutral B meson reconstruction
- **Time-dependent CP violation**

For each category:

$\sim$ 300,000 particle collision processes with binary classification labels

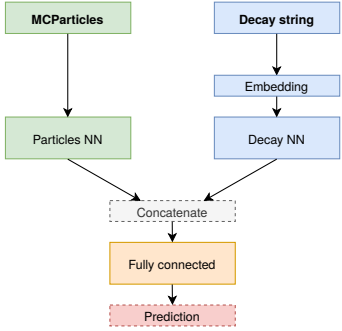


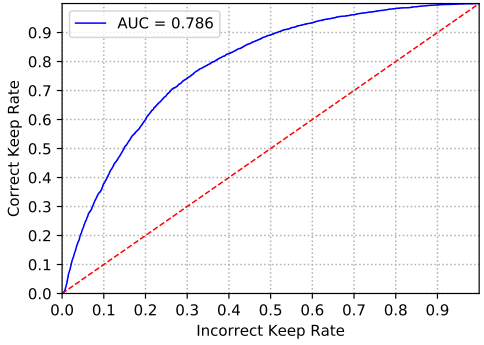
Feature	Definition
PDG code	Identifier of particle type and charge.
Mother PDG code	Particle parent PDG code.
Mass	Particle mass in GeV/c^2.
Charge	Electric charge of the particle.
Energy	Particle energy in GeV.
Momentum	Three momentum of the particle in Gev/c.
Production time	Production time in ns relative to $\Upsilon(4S)$ production.
Production vertex	Coordinates of particle production vertex.
Status bit	Bitmask representing MC production conditions.

Goals

Previous work by James Kahn:

- Convolutional neural networks: residual, recurrent, vanilla...





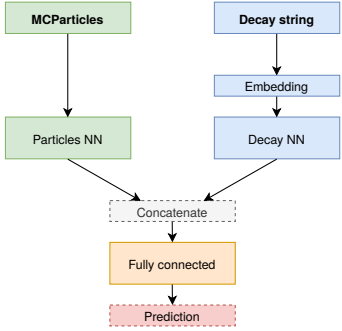
Goals for hackathon:

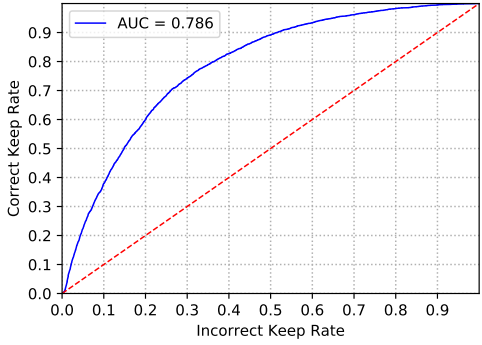
- Implement graph convolutional network
- Decorrelate network from selected kinematics

Goals

Previous work by James Kahn:

- Convolutional neural networks: residual, recurrent, vanilla...





Goals for hackathon:

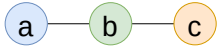
- Implement graph convolutional network
- ~~Decorrelate network from selected kinematics~~

Original Graph Convolutional Networks (GCN)

Propagation rule of layer activations $H^{(l)}$

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\,H^{(l)}W^{(l)}\right)$$

$$H^{(0)} = X$$
$$\tilde{A} = A + I_N$$
$$\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$$



$$\tilde{A}^{N\times N} = A + I = \begin{array}{c} \\ a \\ b \\ c \end{array}\begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \end{array}$$

Thomas N. Kipf, Max Welling, *Semi-Supervised Classification with Graph Convolutional Networks* (ICLR 2017)

Modified GCN

$$H^{(l+1)} = \sigma\left(H^l\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}W_1^{(l)}\right)^T W_2^{(l)}\right)$$

Intuition: **custom weights for each node, considering neighbors**

- Weight vector for every node: W_1 (size: $N \times F$)
- Multiply by $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} \longrightarrow$ weight vector for every node as the average of the vectors of neighboring nodes (output size: $N \times F$ again)
- Transpose and multiply by $W_2 \longrightarrow$ custom dense layer for each node (W_2 size: $N \times U$, result size: $F \times U$)
- Multiply by H: transform every node in the previous layer with its own custom layer (H size: $N \times F$, result size: $N \times U$)

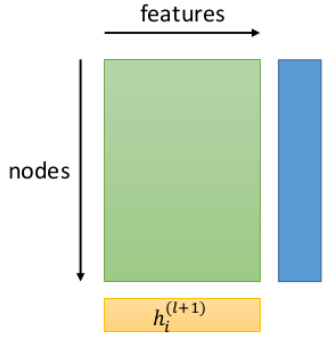
Special case of **graph isomorphism network**:
K. Xu, W. Hu, J. Leskovec, S. Jegelka, *How Powerful are Graph Neural Networks?* (CoRR 2018)

Node Aggregation

- **Soft attention** – weights given by softmax (D) and values given by $\tanh(E)$
- E, D: Independent dense layers applied to features of each node

$$h^{(l+1)} = \sum_i \left(\text{softmax}\left(D_{(i)}^{(l)} \right) \odot \tanh\left(E_i^{(l)} \right) \right)$$

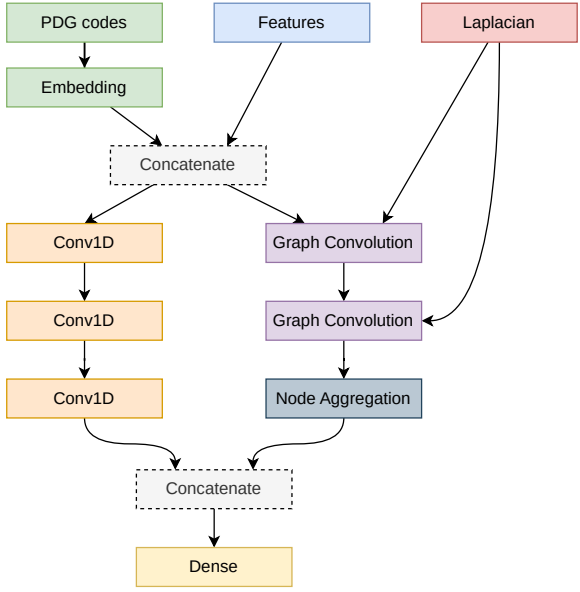
$$= \sum_i \frac{\exp D_{(i)}^{(l)}}{\exp \sum_k D_{(k)}^{(l)}} \tanh E_i^{(l)}$$



Adapted from: Yujia Li, Richard Zemel, Marc Brockschmidt, Daniel Tarlow, *Gated Graph Sequence Neural Networks*, (ICLR 2016)

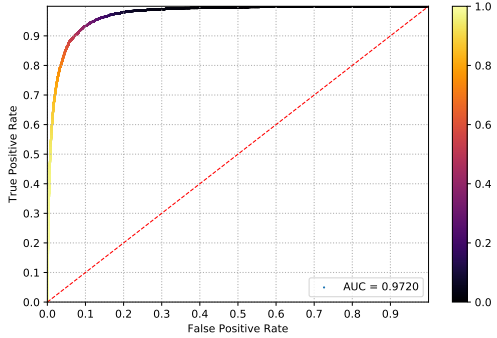
Results

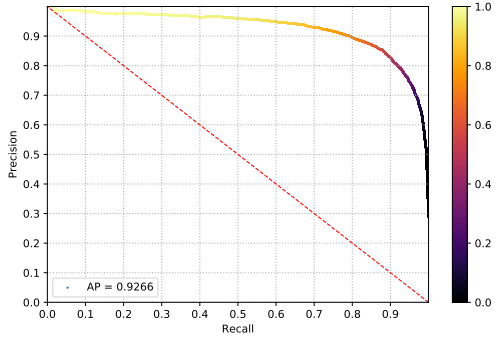
- Train with random subset of 225k processes (validate on 20%)
- Test on 75k independent processes
- Class weights
- Early stopping
- Reduce learning rate on plateau
- Model checkpoint – save only best



Results

- Train with random subset of 225k processes (validate on 20%)
- Test on 75k independent processes
- Class weights
- Early stopping
- Reduce learning rate on plateau
- Model checkpoint – save only best

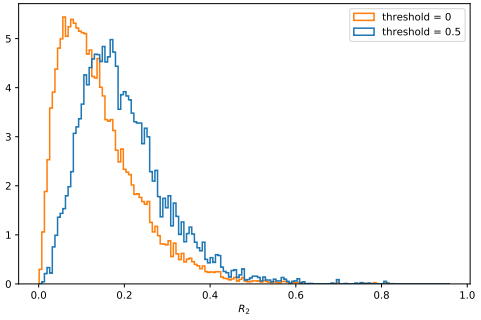


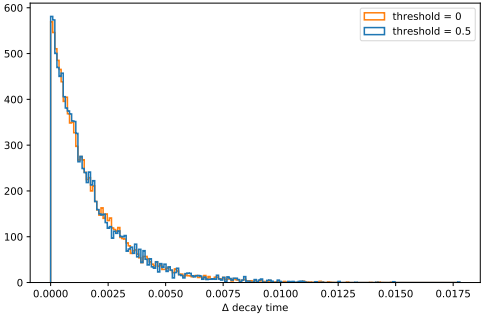


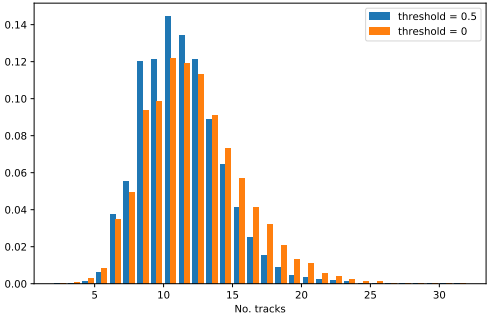
Current work

Bias control

- Studying impact of NN on resulting physics
- Selected event-level kinematic variables
- Implementing adversarial decorrelation



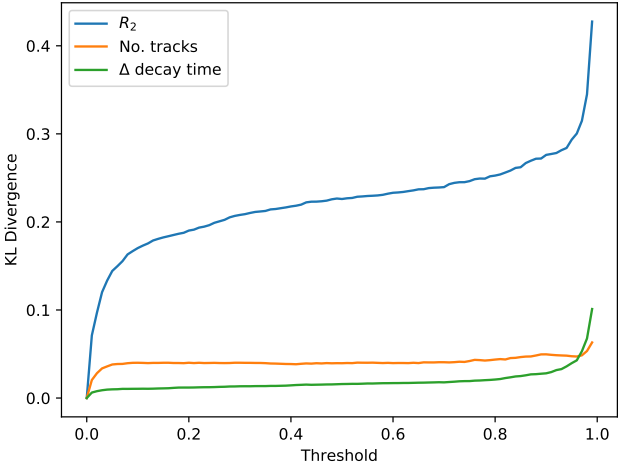




Current work

Bias control

- Studying impact of NN on resulting physics
- Selected event-level kinematic variables
- Implementing adversarial decorrelation



Next steps

- Implementation and timing in analysis framework
- Hyperparameter optimisation
- Training on large dataset and other skims
- Talk accepted at CHEP2019 (Computing in High Energy Physics) – proceedings

Bias Mitigation/Decorrelation:
Many approaches being explored:

- Divergence term in loss
- Adversarial setup
- Distance decorrelation (G. Kasieczka)
- Mutual information regularization (J. Tan, Belle II)
- ...

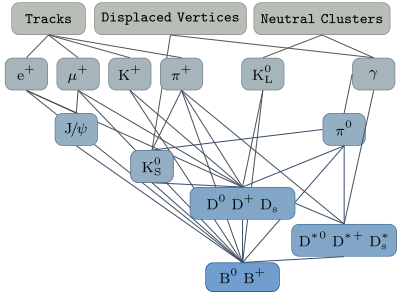
Common problem in physics ML usage

Deep Full Event Interpretation (deepFEI)

Overview

Existing generic reconstruction algorithm at Belle II:

- Hierarchical approach
- Reconstruct multiple *hard-coded* decay channels per particle
- Stacked boosted decision trees
- $\epsilon(B_{\mathrm{SL}}) = \mathcal{O}(1\%)$, $\epsilon(B_{\mathrm{H}}) = \mathcal{O}(0.1\%)$



Deep Full Event Interpretation (deepFEI)

Overview

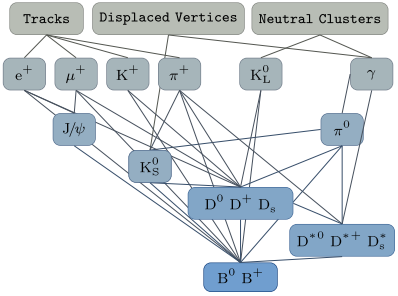
Existing generic reconstruction algorithm at Belle II:

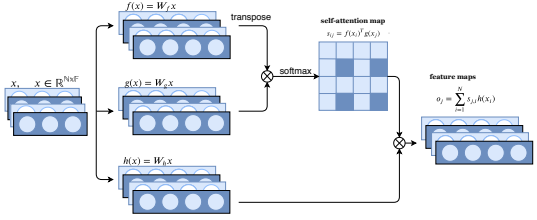
- Hierarchical approach
- Reconstruct multiple *hard-coded* decay channels per particle
- Stacked boosted decision trees
- $\epsilon(B_{\mathrm{SL}}) = \mathcal{O}(1\%)$, $\epsilon(B_{\mathrm{H}}) = \mathcal{O}(0.1\%)$

deepFEI:

- Generic solution to learn common decays by example
- Stacked self-attention maps to group particles – interpret as parent

Addresses generic problem:
How to build a tree from the leaves alone?



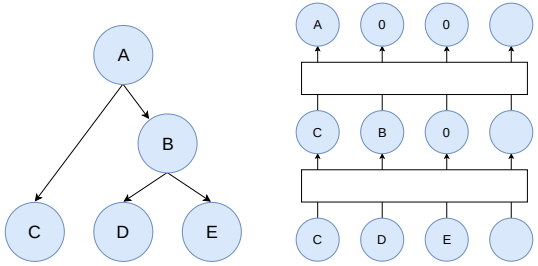


Deep Full Event Interpretation (deepFEI)

Progress

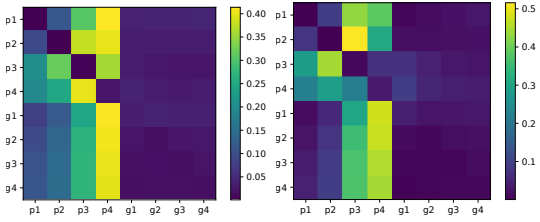
Implemented at hackathon:

- Transformer network setup
- Reconstruct $B \to D(\to K\pi\pi^0)\pi$ decays
- Permutation invariant loss function: *Kuhn-Munkres/Hungarian algorithm* (shipped with SciPy)



Results:

- Difficult to interpret attention maps
- No clear way to identify candidate decay chains
- Converging on **graph-based** solution



Closing remarks

Experiences from machine learning in Belle II:

- Sharing individual project progress interesting but ultimately not useful (without a lot of personal effort)
- Abstraction to generic, situational solutions significantly more useful
- Difficult to get overview of (latest) machine learning landscape

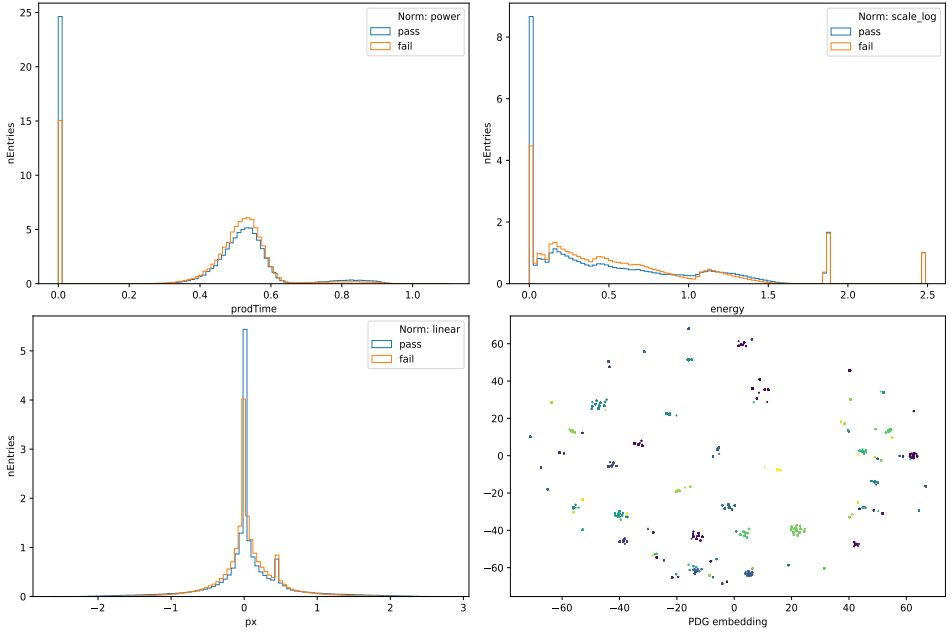
Example recurring topics:

- Quantifying systematic uncertainties
- Decorrelating network outputs
- Variable or irregular-shape inputs

Thank you

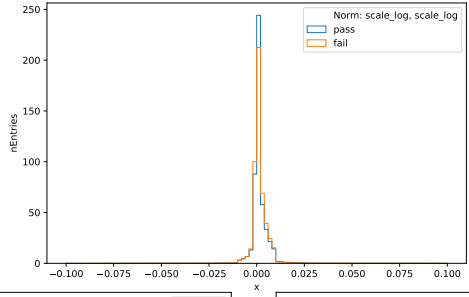
Backup

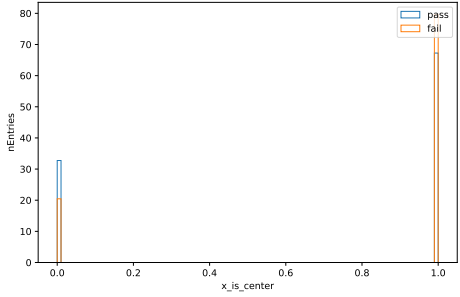
Features

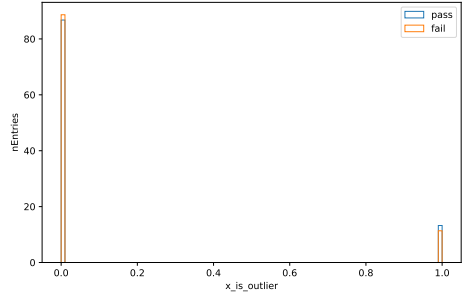

Norm: power
pass
fail
nEntries
prodTime
Norm: scale_log
pass
fail
nEntries
energy
Norm: linear
pass
fail
nEntries
px
PDG embedding

Features II

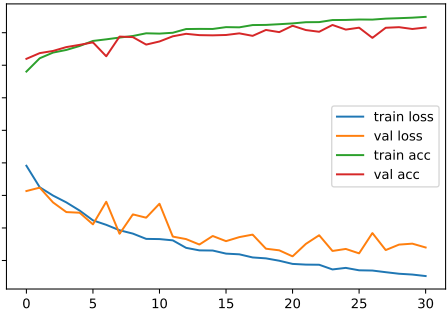
x, y, z coordinates: Hard to normalize

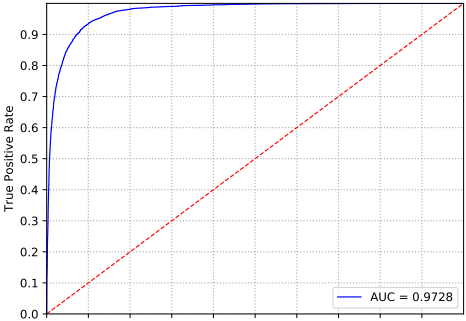


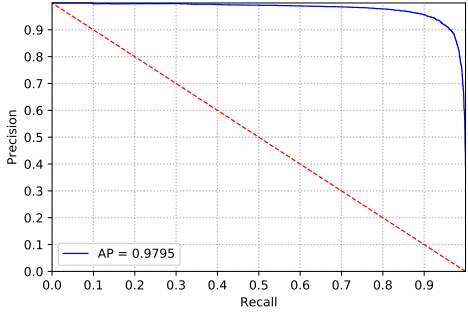




Results (Smaller network)







Adjacency matrix

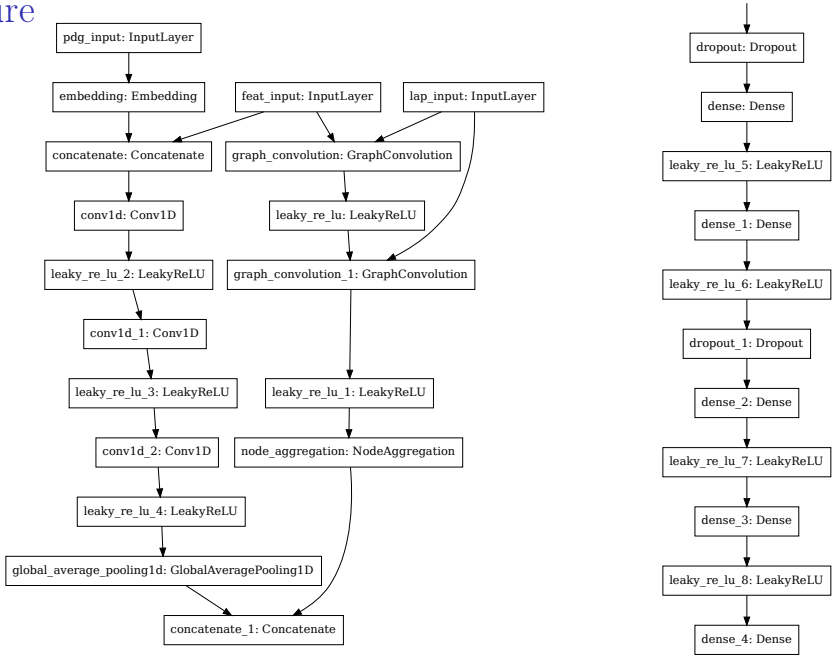
Add my plot from notebook

Architecture

```
p = GraphConvolution(128, kernel_regularizer=regularizers.l2(l2_strength))([feature_input, laplacian_input])
p = layers.LeakyReLU()(p)
p = GraphConvolution(128, kernel_regularizer=regularizers.l2(l2_strength))([p, laplacian_input])
p = layers.LeakyReLU()(p)
p = NodeAggregation(256, kernel_regularizer=regularizers.l2(l2_strength))(p)

m = layers.Concatenate()([pdg_l, feature_input])
m = layers.LeakyReLU()(layers.Conv1D(128, 5)(m))
m = layers.LeakyReLU()(layers.Conv1D(128, 5)(m))
m = layers.LeakyReLU()(layers.Conv1D(128, 5)(m))
m = layers.GlobalAveragePooling1D()(m)

l = layers.Concatenate()([m, p])

l = layers.Dropout(0.5)(l)
l = layers.LeakyReLU()(layers.Dense(512, kernel_regularizer=regularizers.l2(l2_strength))(l))
l = layers.LeakyReLU()(layers.Dense(256, kernel_regularizer=regularizers.l2(l2_strength))(l))
l = layers.Dropout(0.3)(l)
l = layers.LeakyReLU()(layers.Dense(128, kernel_regularizer=regularizers.l2(l2_strength))(l))
l = layers.LeakyReLU()(layers.Dense(32, kernel_regularizer=regularizers.l2(l2_strength))(l))

output_layer = layers.Dense(1, activation='sigmoid')(l)
```

- Class weights
- Early stopping
- Reduce learning rate on plateau
- Model checkpoint – save only best

Architecture



Architecture