b-quark Identification with Recurrent Neural Network

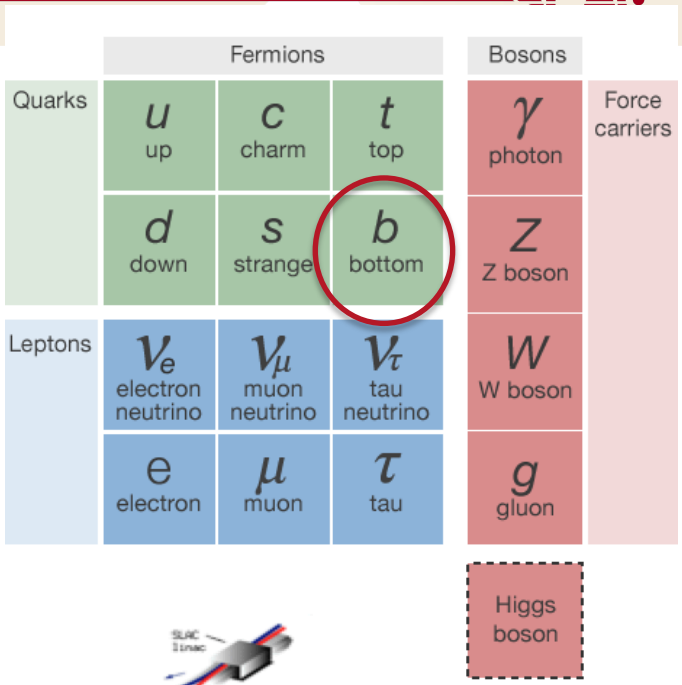
Michael Kagan, Zihao Jiang, Qi Zeng

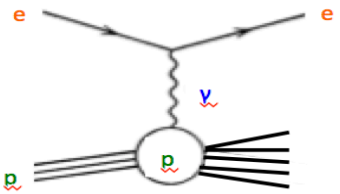
U.S. DEPARTMENT OF ENERGY
Office of Science

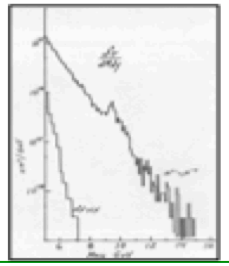
SLAC NATIONAL ACCELERATOR LABORATORY

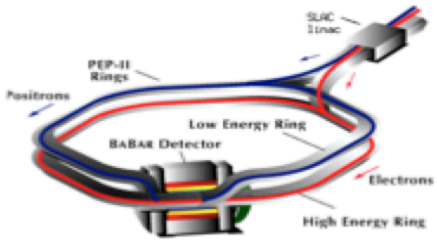
Introduction to b-quark Identification

- What is b-quark?
  - One of elementary particles
- Why b-quark?
  - Important signature of many physics process
  - It leaves “long traces” in our detector









Quarks Discovered at SLAC

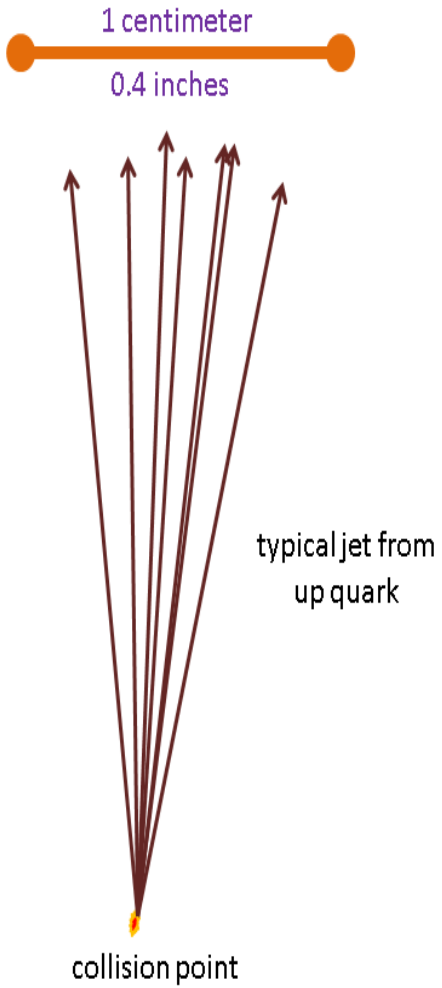
Bottom Quark Discovered at FermiLab

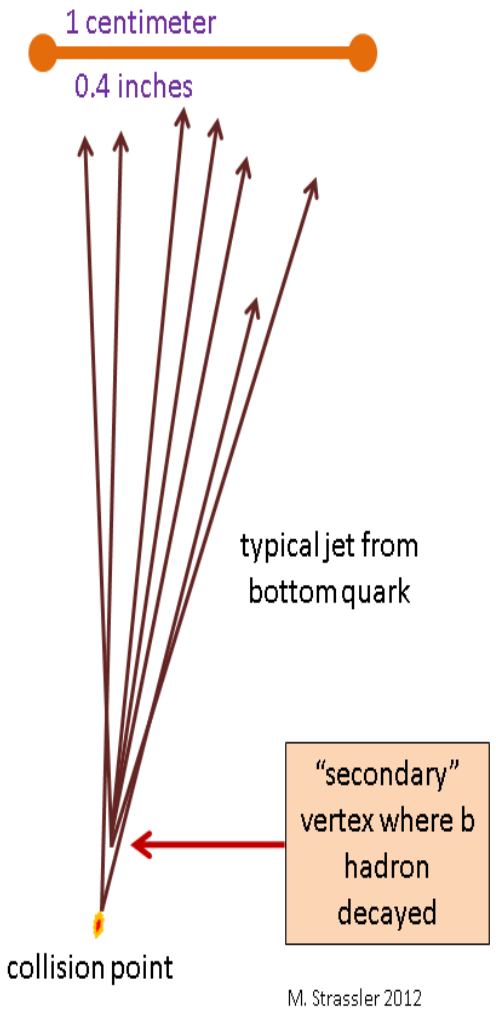
Bottom Quark “Factories”: E.g. BaBar Experiment at SLAC

1969 — 1973 — 1977 — 1983 — 1999

Introduction to b-quark Identification

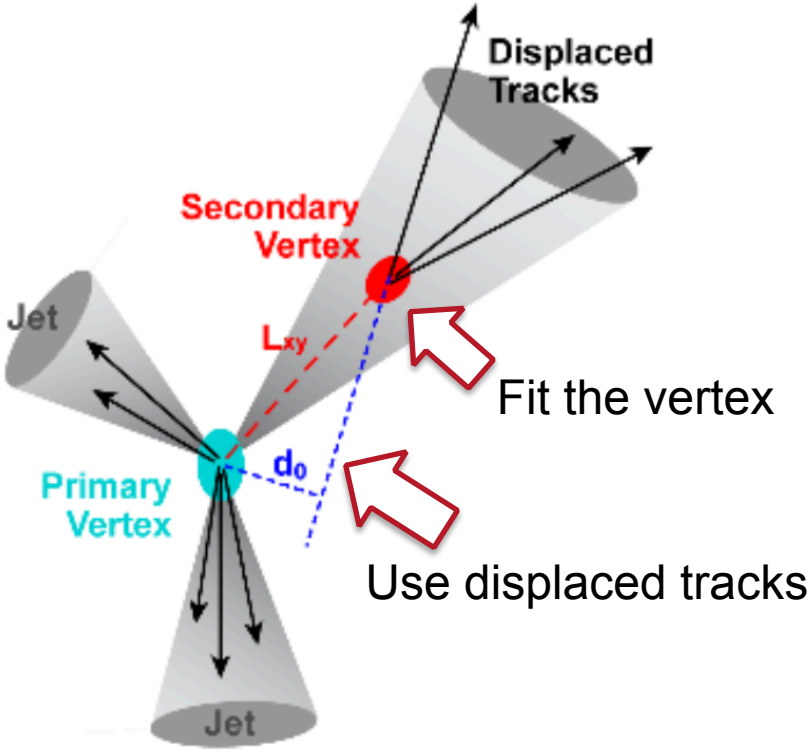
- How to identify b-quark?
  - "quark" is not directly measurable
  - Quark will be transformed into a bunch of collimated particles immediately after it is produced (hadronization)
  - These bunch of particles (hadrons) will either be stable, or go through further decay, collectively called "jets"
  - b-quark is special:
    - B-hadron will travel a long distance before it decays





Introduction to b-quark Identification

- Various sophisticated algorithms are developed inside ATLAS collaboration for this task
- Two complementary approaches:
  - Fit the vertex!
  - Use special track-by-track properties
- Problem setup:
  - Input: a jet and tracks associated to it
  - Output: Binary classification of b-jet or light-jet
  - Comment: In nature, there are many other types of jets. We simplify it to a binary case here



Baseline: Naïve Bayes Model --- IP3D

SLAC

- Input features:
  - d_0 and z_0 significance, track quality of each track
  - A list of tracks associated to a jet, which we want to identify
- IP3D
  - A Naïve Bayes Model to do the classification
  - Write down the likelihood as:

    $$L(b) = \prod Prob(trk_i|b) \qquad L(l) = \prod Prob(trk_i|l)$$

  - Use log likelihood ratio as final discriminant variable:

    $$score = log(L(b)/L(l))$$

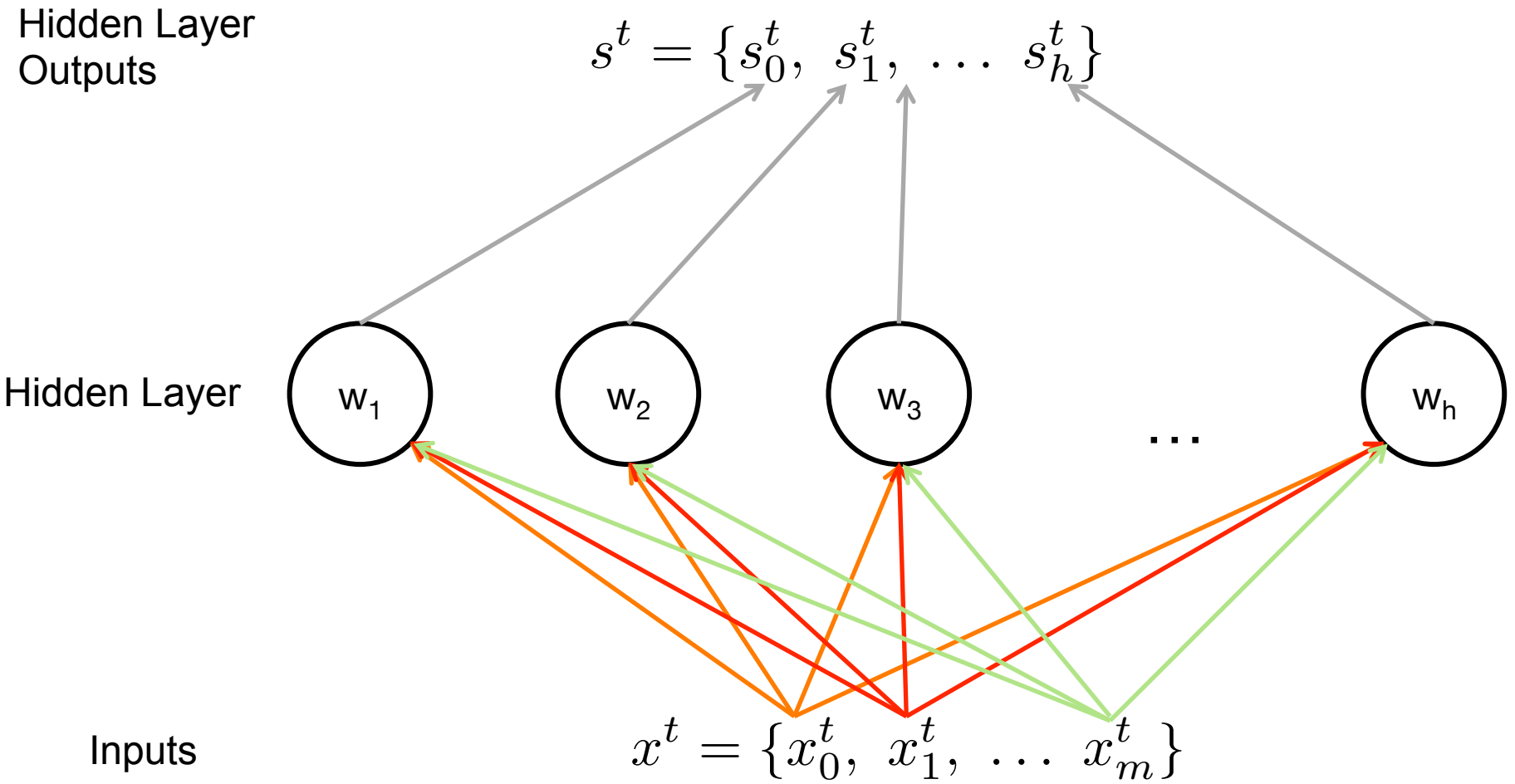
  - Template $Prob(trk_i|b)$ and $Prob(trk_i|l)$ are built from training sample

Improvement: Recurrent Neural Network

SLAC

- An obvious drawback of IP3D is that it ignores the correlation between tracks inside a jet
  - For a b-jet, given a track with high d0, there will be another track with high d0, due to momentum conservation
- How to take the correlation into account?
  - Build a huge template $Prob(trk_1, trk_2, ..., trk_n|b)$?
    - Needs huge amount of training sample
    - Number of tracks per jet is non-fixed
    - Will not be robust
  - Recurrent neural network!
    - An elegant extension to ordinary neural network
    - Widely applicable to time-series analysis

Let's start with ordinary neural network …

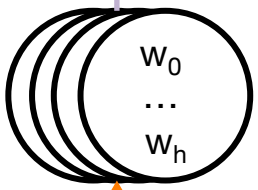


Densely connected, single hidden layer network:
every input feature $\{x_0^t,\ldots,x_m^t\}$ connects to every hidden node

Now, let's forget about all the details in hidden layer …

Hidden Layer Outputs

$$s^t = \{s_0^t,\ s_1^t,\ \ldots\ s_h^t\}$$



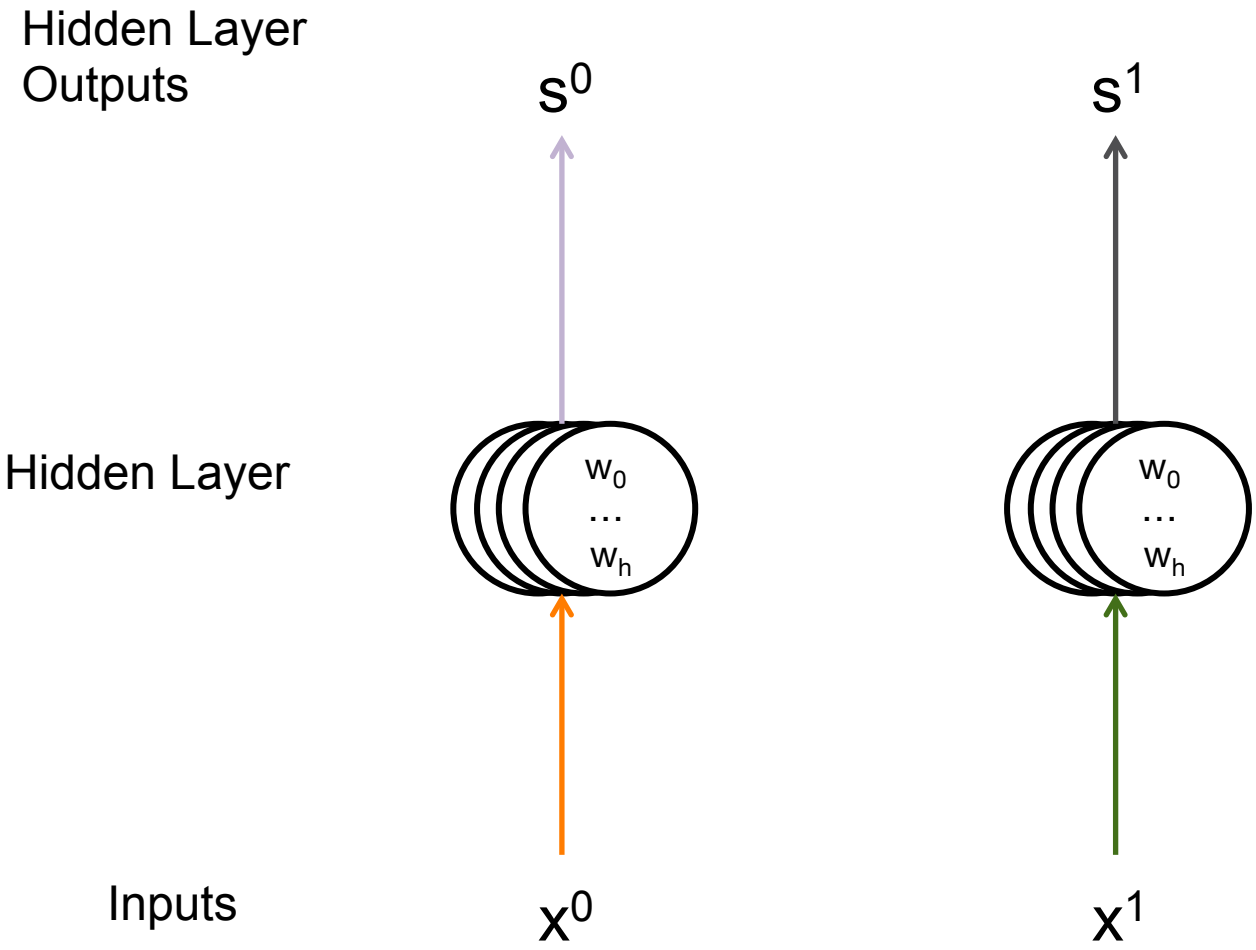
Hidden Layer

Inputs

$$x^t = \{x_0^t,\ x_1^t,\ \ldots\ x_m^t\}$$

"Side View" of densely connected network

Build a NN for each “time-step”

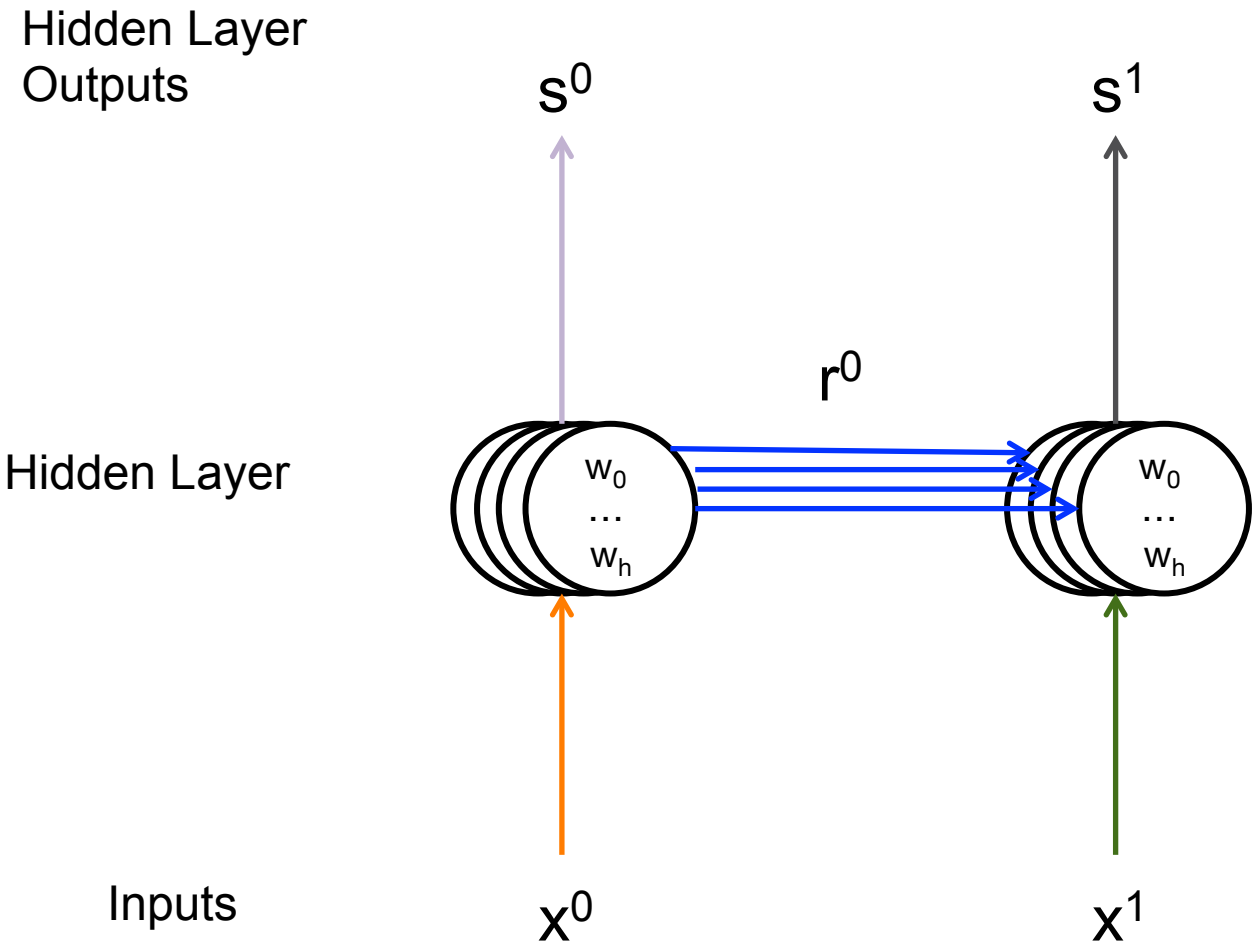


Another Time Step:

here evaluated with same dense network

on each time step independently

Connect between different time-steps!

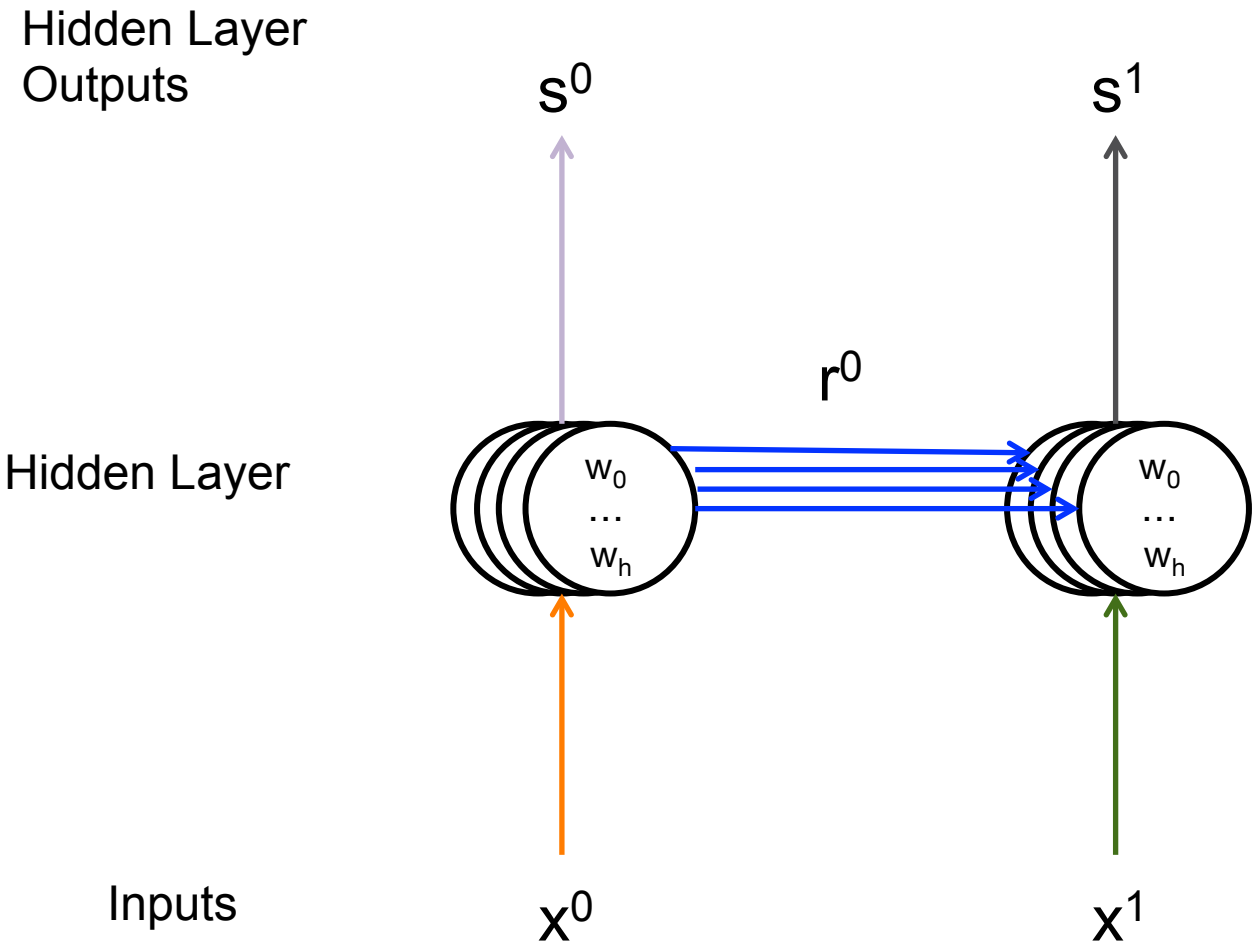


Recurrence: $r^t = \{r_0^t,\ r_1^t,\ \ldots\ r_h^t\}$

Use a second output on each time step

Feed new output to next time step

Connect between different time-steps!

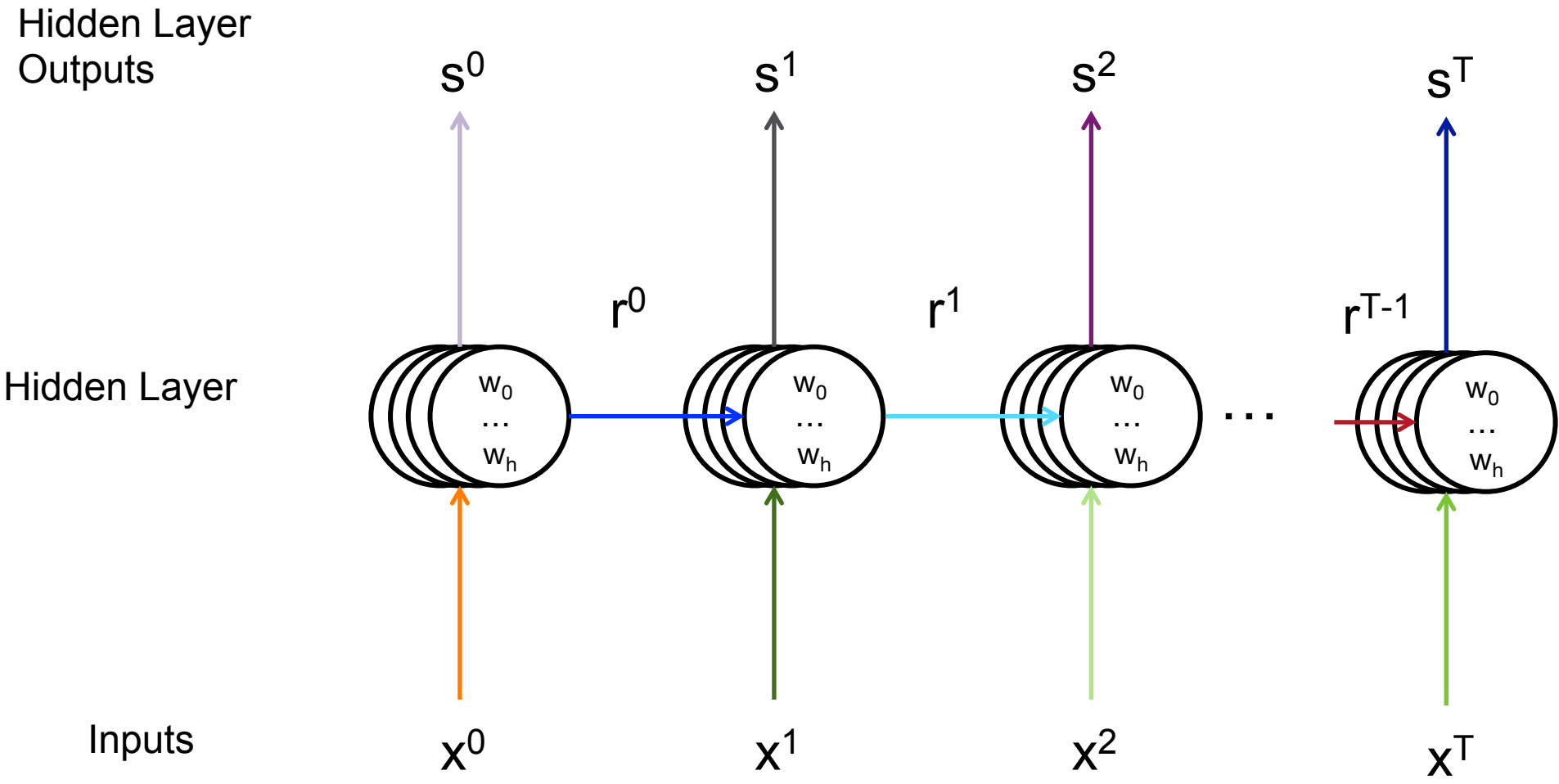


Recurrence:

Feeds information to next step → conditional probability

$$s^0 \simeq P(x^0), \qquad s^1 \simeq P(x^1|x^0),$$

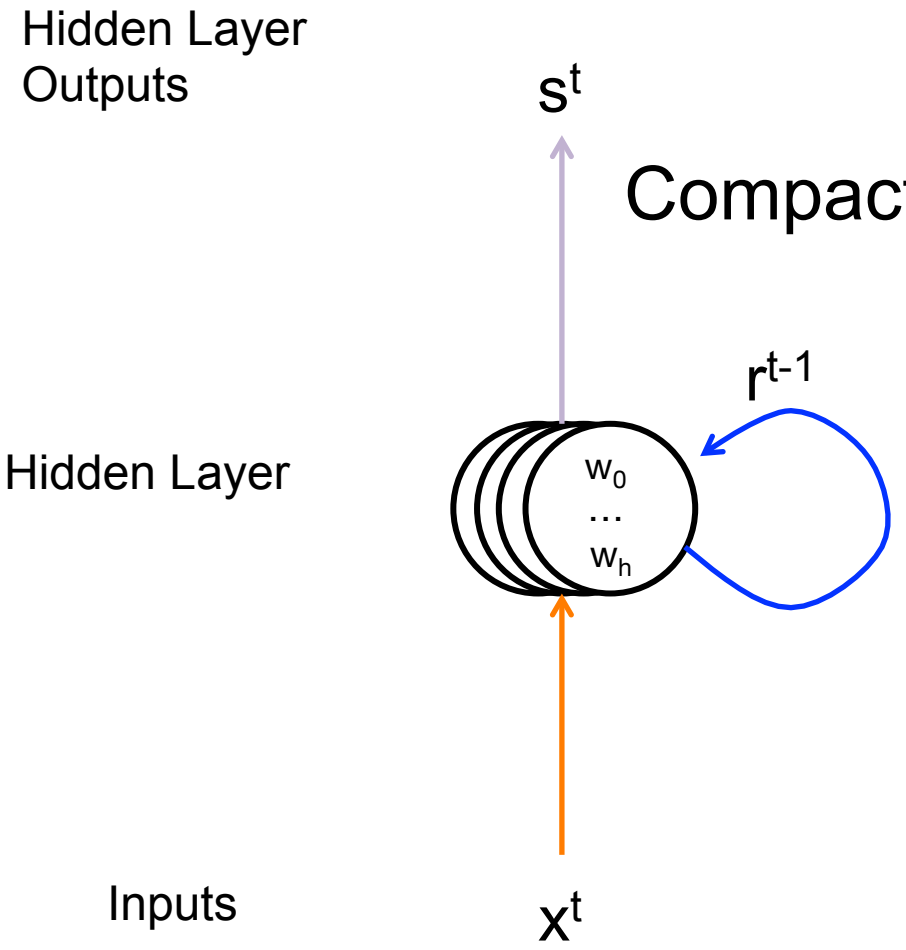
Expand to the whole time-series



Continue iterating over time / sequence steps

$$s^0 \simeq P(x^0), \qquad s^1 \simeq P(x^1|x^0), \quad \ldots \quad s^T \simeq P(x^T|x^0,\ x^1,\ \ldots x^{T-1})$$

Compact Graphical Recurrence

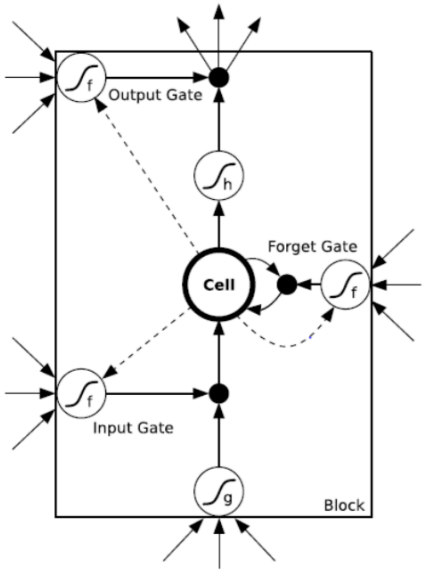


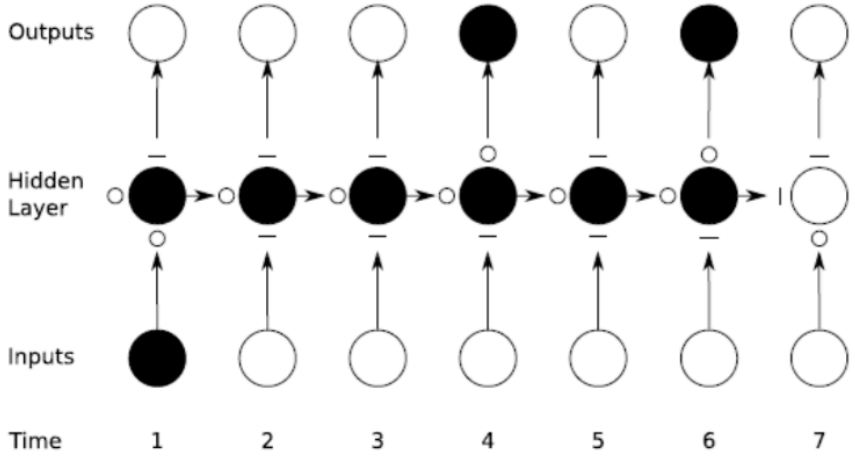
Incorporate correlations between time steps using conditional probabilities, since:

$$P(x^0, x^1, \ldots, x^T) = P(x^0) \cdot P(x^1|x^0) \cdot P(x^2|x^0, x^1) \ \ldots \ P(x^T|x^0, \ x^1, \ \ldots x^{T-1})$$

Long-short-term-memory (LSTM)

- But!
  - It is a well-known issue for RNN that the memory of early input can be easily “forgotten” in the end of epoch
  - “Vanishing Gradient”
- Long-short-term-memory cell is introduced to preserve the memory



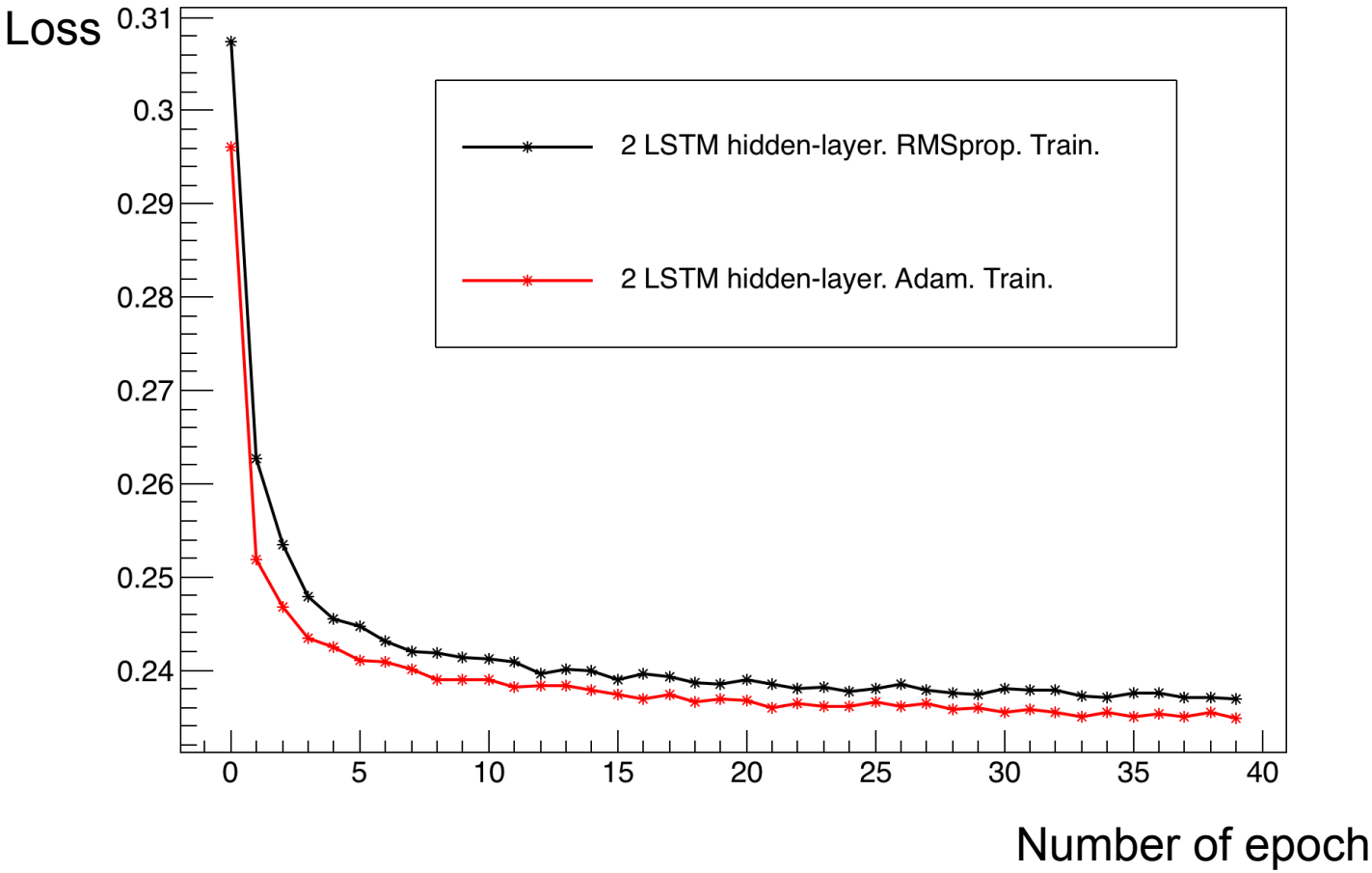


RNN Architecture for b-quark Identification

SLAC

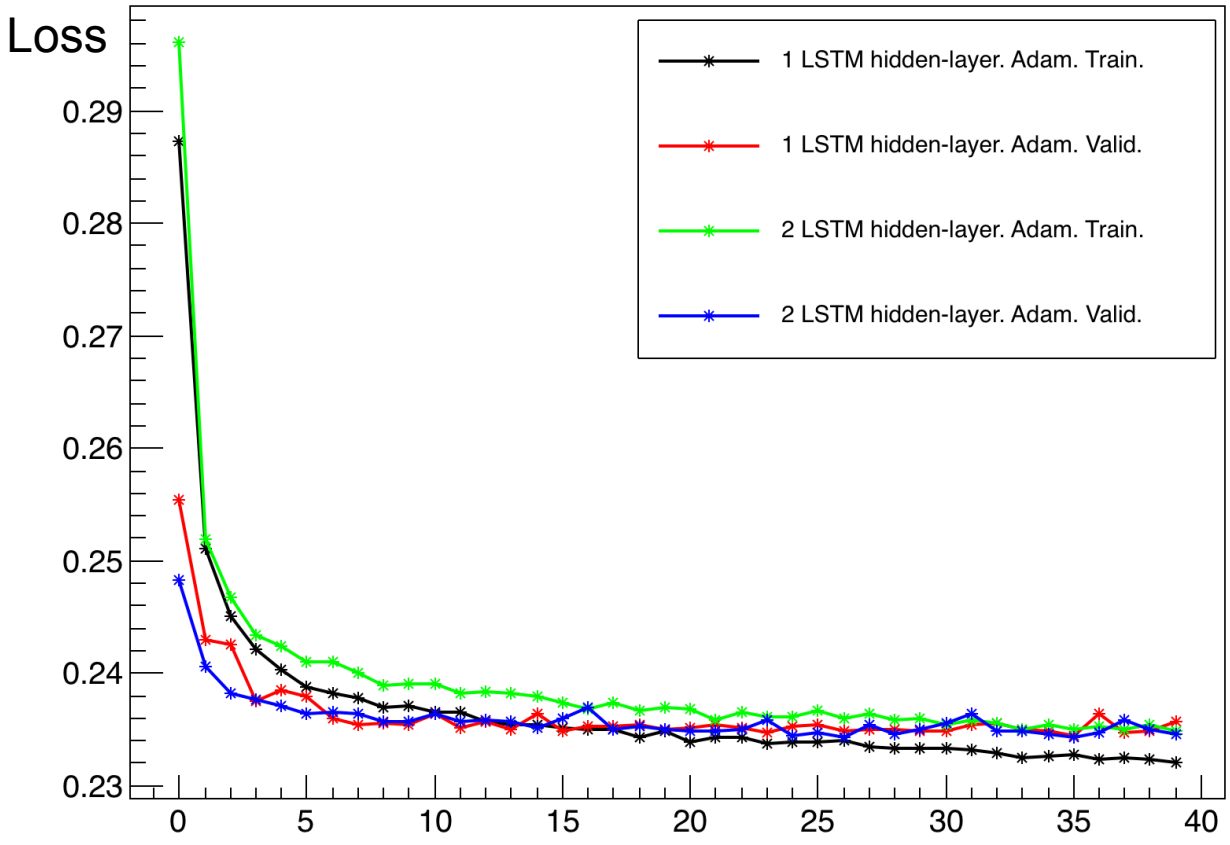
- Input features: exactly same as IP3D
  - Significance of d0
  - Significance of z0
  - Track quality: “embedding” – another layer combines all qualities in a linear way
- Ordering:
  - We are not really time-series data
  - Each track is a “time-step”
  - But, we order tracks by d0 significance
- RNN architecture:
  - 1-hidden LSTM layer, with 25 nodes.
  - 2-hidden LSTM layer, with 25 nodes
  - Connected to output layer only in last time-step (only one label per sequence!) (25-in-1-out)
- Optimizer:
  - Mini-batch gradient descendent
  - “RMSProp”
  - “Adam”

Comparison between different optimizers



- Loss v.s. number of epoches
- “Adam” looks to converge faster

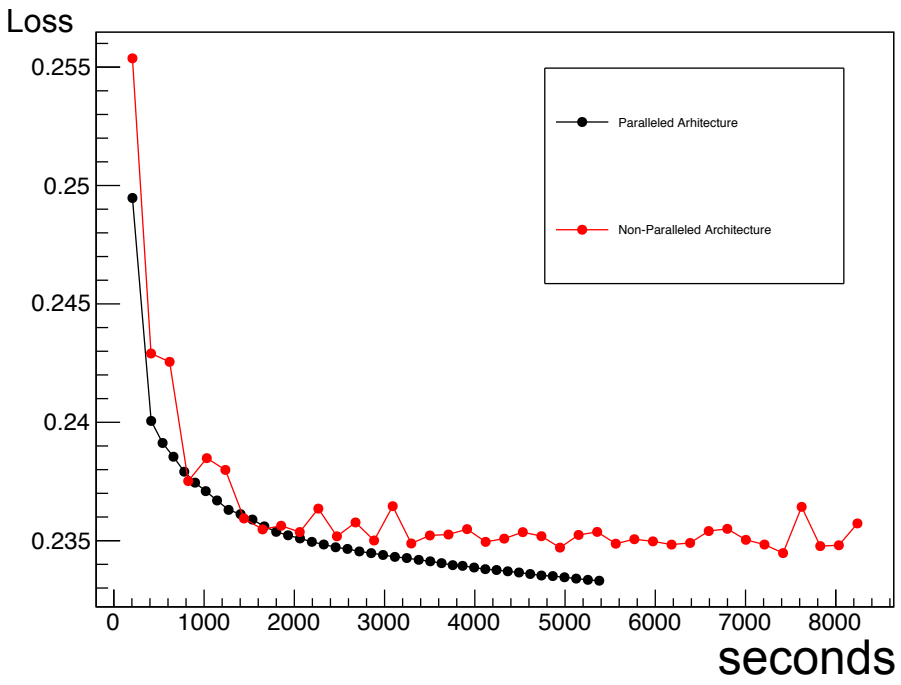
How is training going?



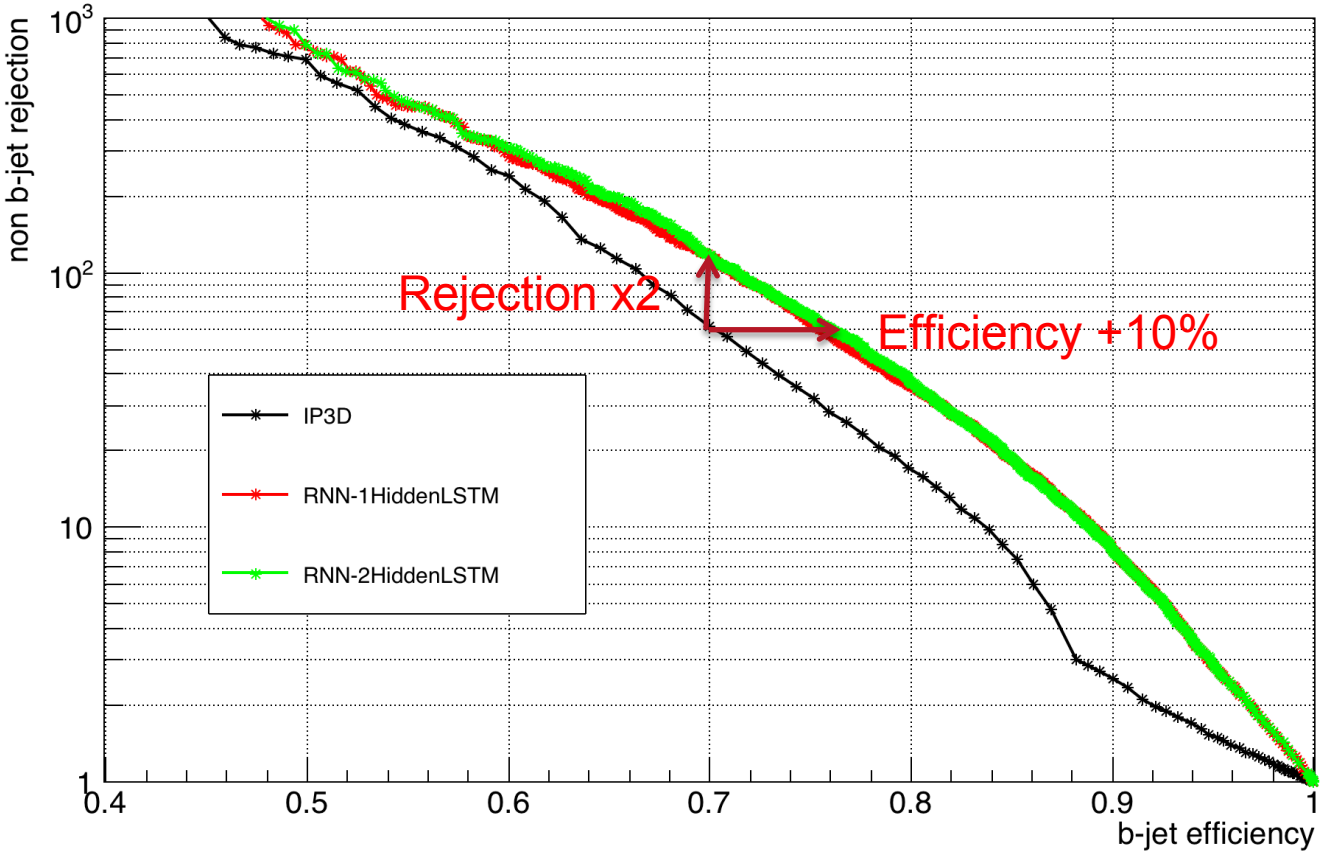
- RNN converges very slow!
- We stop at 40 epoch which takes couple hours
- No over-training is found through cross-validation

Can we make training faster?

- Parallel!
- “Accelerating recurrent neural network training via two stage classes and parallelization”, Speech at Microsoft, Mountain View, CA, USA, by Zhiheng Huang et al.
- Each slave deals with a (overlapping) fraction of total data; Master node wou take the “average” of slave nodes for next iteration
- Speed up, at the cost of derivative accuracy → not necessarily make it optimal
- Empirical studies show that it still wins as number of workers is large enough
- Implemented using MPI based on Keras



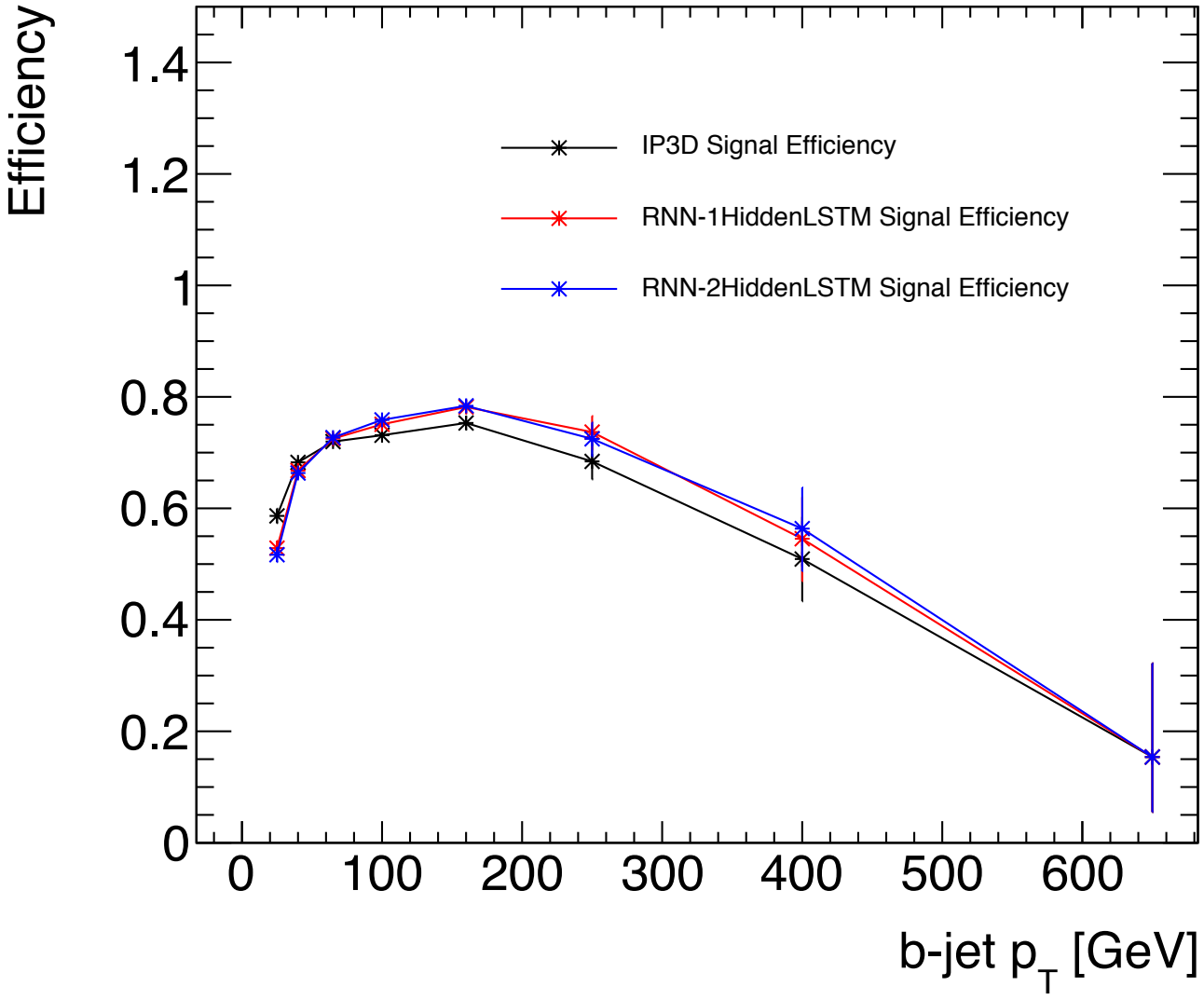
How is the performance?



- We first get distribution of scores for both b-jet and light-jet
- For each cut on the output score, there is a signal efficiency, and background efficiency
- We plot signal efficiency as x-axis, and inverse of bkg efficiency as y-axis

- A significant performance improvement can be seen by using RNN!
- Almost twice rejection power as the baseline at 70% signal efficiency
- No significance difference between 1-layer and 2-layer architecture

Efficiency as function of kinematics



- Besides overall performance, we also need to keep an eye on dependency on kinematics
- Ideally, we want it to be flat
- In practice, very pT dependent
- Efficiency as function of pT, for fixed overall 70% signal efficiency
- Remember: Rejection of RNN is x2 as IP3D here!

Summary

SLAC

- Here we did a preliminary study using RNN for b-quark identification, using same input features as IP3D
- Comparison with baseline IP3D shows significant improvement
- Such improvement mostly come from the usage of track-by-track correlation in a jet
- Potential future works:
  - Final validation of parallel RNN
  - Understand how information is propagated inside RNN
  - Add more track variables (e.g. track pT), which is much more easier in RNN than IP3D approach.