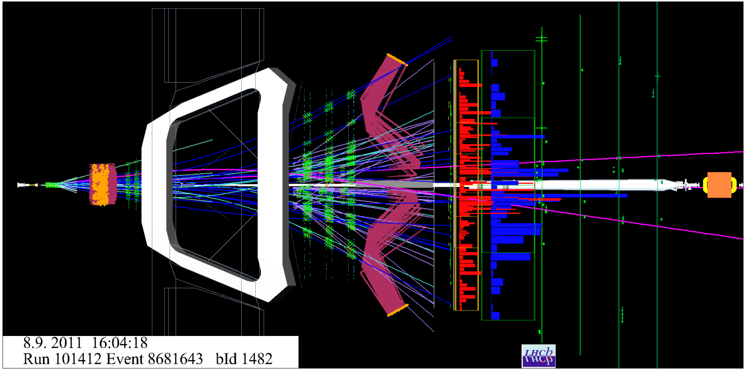
Autoencoders for LHCb

Constantin Weisser, Mike Williams

April 27, 2018

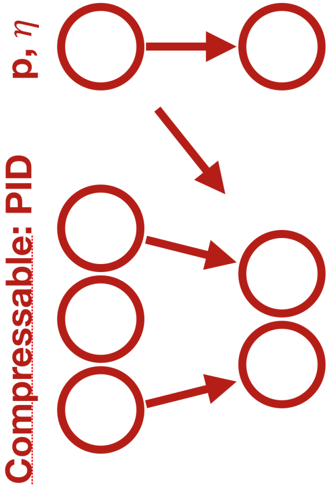
LHCb / MIT

The Problem



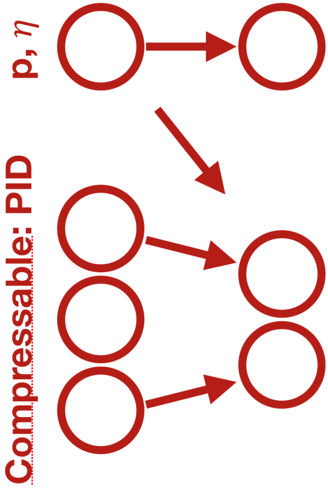
- We collect ∼5 TB/s of data, but our write out is limited
- **The more efficient** we are at storing the data, the more collisions we can keep and **the more physics** we can do

Solution? : Encoding Network



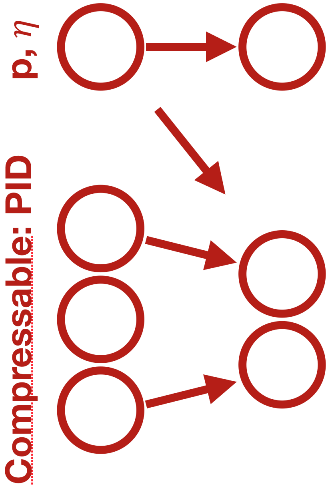
- Features to not compress (in this work)
  - Require very high precision
  - Many use cases $\implies$ undefined loss function (e.g. Tracking variables)

Solution? : Encoding Network



- Features to not compress (in this work)
  - Require very high precision
  - Many use cases $\implies$ undefined loss function (e.g. Tracking variables)
- Features to compress: Particle ID
  - Help classify particle types (e, μ, π, ...)
  - Unimportant by themselves
  - Factorisation: all muons are the same regardless of production (given kinematics and event occupancy)

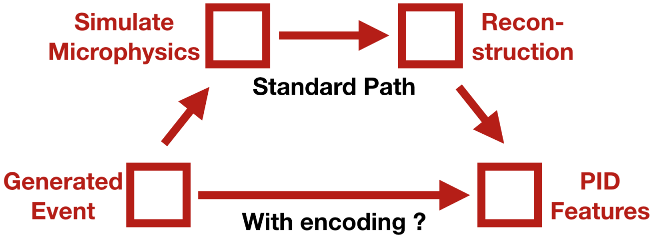
Solution? : Encoding Network



- Features to not compress (in this work)
  - Require very high precision
  - Many use cases $\implies$ undefined loss function (e.g. Tracking variables)
- Features to compress: Particle ID
  - Help classify particle types (e, μ, π, ...)
  - Unimportant by themselves
  - Factorisation: all muons are the same regardless of production (given kinematics and event occupancy)

Find smaller representation of PID features

Implications for MC



MC simulations only need to generate the compressed features

- Faster generation speeds
- Smaller file sizes

Implications for use in HEP

However, cannot get back to original features

- Hard to fix PID if compression goes wrong
- Collaboration needs trust: **Sociological problem**

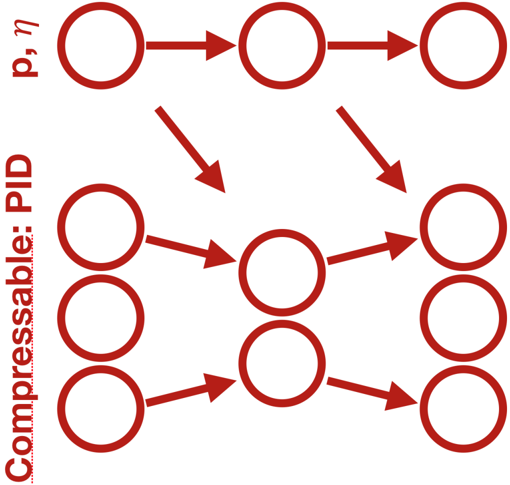
Implications for use in HEP

However, cannot get back to original features

- Hard to fix PID if compression goes wrong
- Collaboration needs trust: **Sociological problem**

Interesting idea, but requires more thought

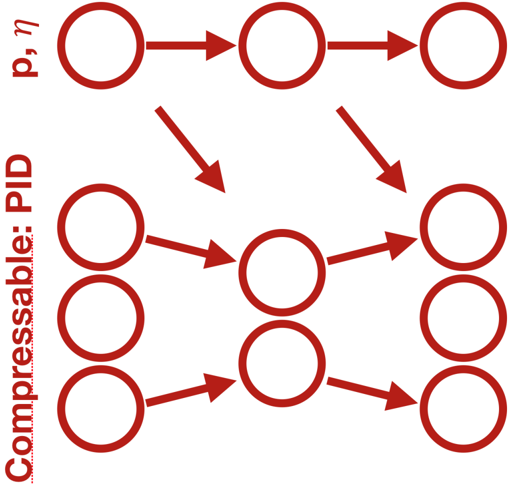
Another Solution? : Autoencoder



Use an autoencoder that

- Compresses information about every particle before writing it out
- Decodes the result when performing the analysis
- Factorization means everything can be characterised offline

Another Solution? : Autoencoder



Use an autoencoder that

- Compresses information about every particle before writing it out
- Decodes the result when performing the analysis
- Factorization means everything can be characterised offline

This brings advantages:

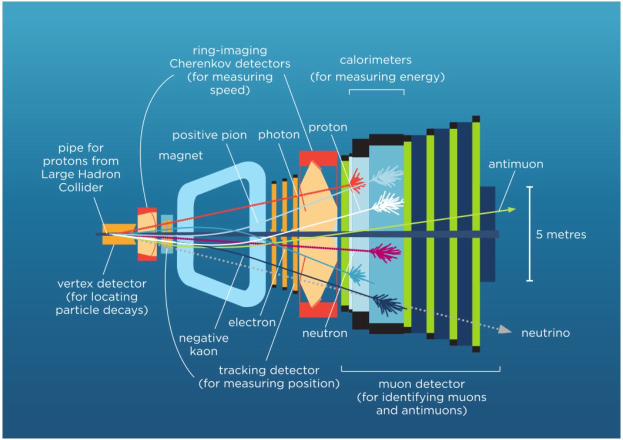
- Can compare input and output for calibration samples
- Less trust needed (input sufficient for $\sim$ 500 papers to date)

Data Set

- Simple proof-of-concept preliminary study
- LHCb Minimum Bias Monte Carlo (Simulation)
- Extracted equal numbers ($\sim$550k) of different charged particle types (e, μ, π, K, p, ghost)
- Can generate more data if required
- Obscured data at `zenodo.org/record/1230552#.WuIbVlMvw6h`
- Code at `github.com/weissercn/LHCb_PID_Compression`
- Get in touch with me at weisser@mit.edu to discuss

Sub-Detectors

- For each particle, information from each of the **5 groups of features** can either be present or missing. **MNAR** (Missing Not At Random)



- RICH
- Muon System
- ECAL General
- ECAL Charged
- ECAL Neutral

Special Case: General ECAL + RICH

- Concentrate on 2.5 out of 3.3 million examples where information from two specific subdetectors (General ECAL, RICH) exist.
- Ignore information from other systems
- Numbers of features to compress: [Always present, ECAL, RICH] = 10,3,5
- Additional auxiliary features present at compression and decompression: [Always present, ECAL, RICH] = 8,0,0
- A different autoencoder like this could be trained for all combinations of subsystems being on or off. ($2^5 = 32$ autoencoders)

Special Case: General ECAL + RICH

- Concentrate on 2.5 out of 3.3 million examples where information from two specific subdetectors (General ECAL, RICH) exist.
- Ignore information from other systems
- Numbers of features to compress: [Always present, ECAL, RICH] = 10,3,5
- Additional auxiliary features present at compression and decompression: [Always present, ECAL, RICH] = 8,0,0
- A different autoencoder like this could be trained for all combinations of subsystems being on or off. ($2^5 = 32$ autoencoders)

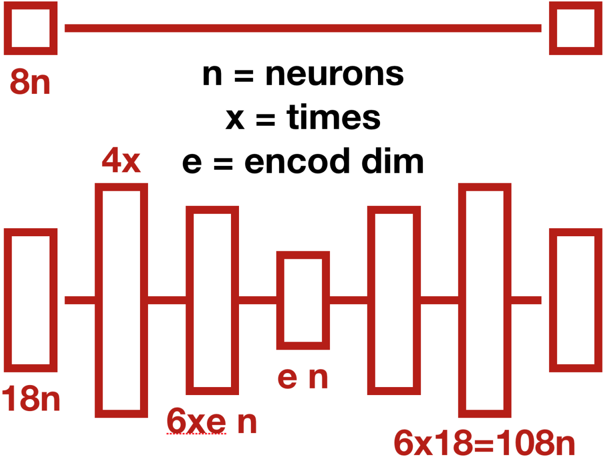
Is there a smarter way to deal with missing data?

Loss Function

There are many different PID classification problems $\Longrightarrow$ there are many possible loss functions

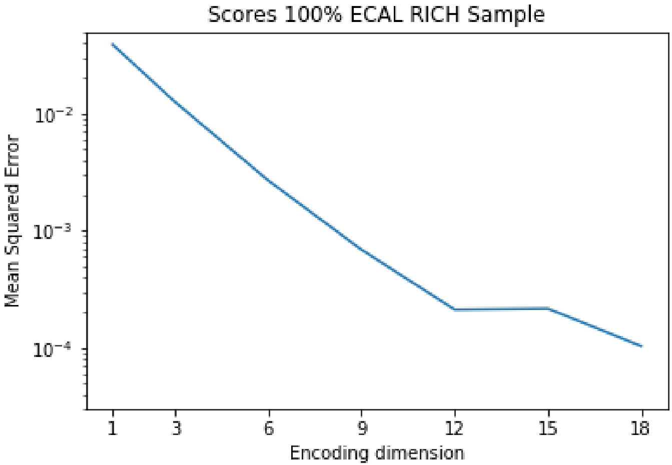
- Training
  - For now want to retain possibility of reconstructing input features
  - Scale and use mean squared error (MSE) loss to train
- Cross-check
  - To test the autoencoder's PID classification ability, use another loss:
  - Train two BDTs on the uncompressed and decompressed features and compare ROC curve for electron selection

Network Architecture



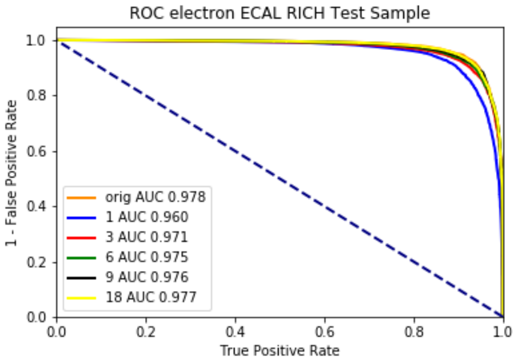
- 5 hidden fully connected layers
- Auxiliary features concatenated after each layer
- Vary the number of encoded features
- Train/Validate/Test = 70/20/10%

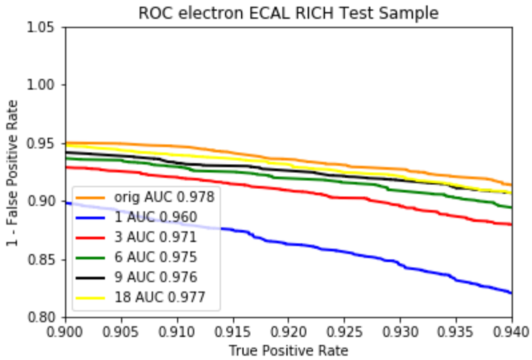
Loss



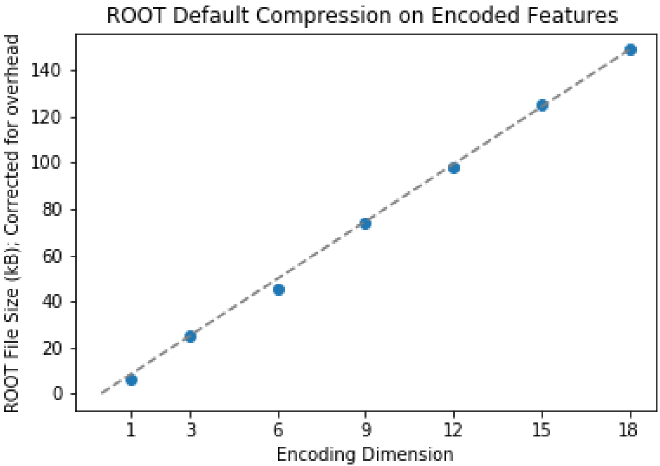
Scores 100% ECAL RICH Sample
Mean Squared Error
10^{-2}
10^{-3}
10^{-4}
1
3
6
9
12
15
18
Encoding dimension

ROC curve for electron selection





Compression on Encoded Features



Compression algorithms like gzip, zero padding are common

- ROOT's default compression (Level 4) yields file size reduction beyond autoencoding
- Need to confirm if this still holds after zero padding
- Straightforward to define algorithm to automatically optimise zero padding

Focus Points

Allows for a critical file size reduction

Potentially a way to make MC Generation in run 5 more feasible

Data at zenodo.org/record/1230552#.WuIbVlMvw6h

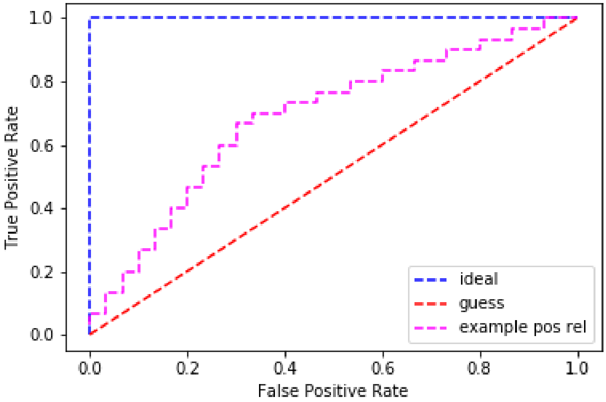
Focus Questions

Is there a smarter way to deal with missing data?

What is a rigorous way to define our optimisation goal?

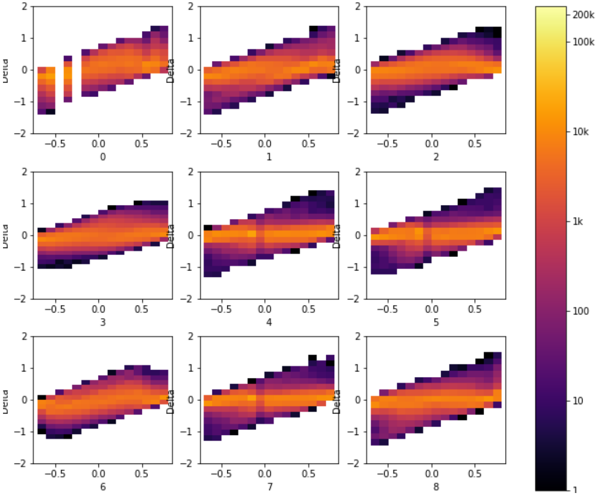
How can the sociological problem best be overcome?

ROC Curves

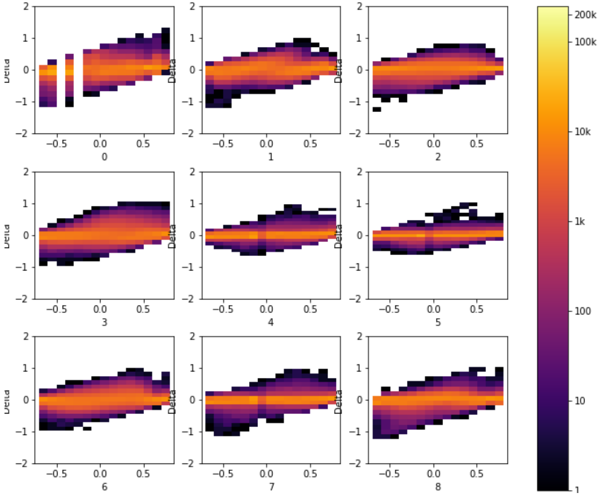


- False Positive Rate: Labelled as a muon, actually something else
- True Positive Rate: Labelled as a muon, actually a muon
- For each variable use ROC curves for distinguishing the relevant particle from all others (*e.g.* for trks_pnn_e distinguish electrons from all other particles)

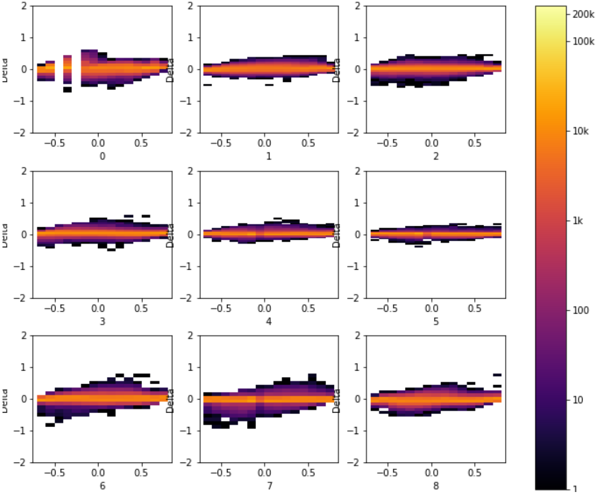
Distribution Deltas Encoding 1/18



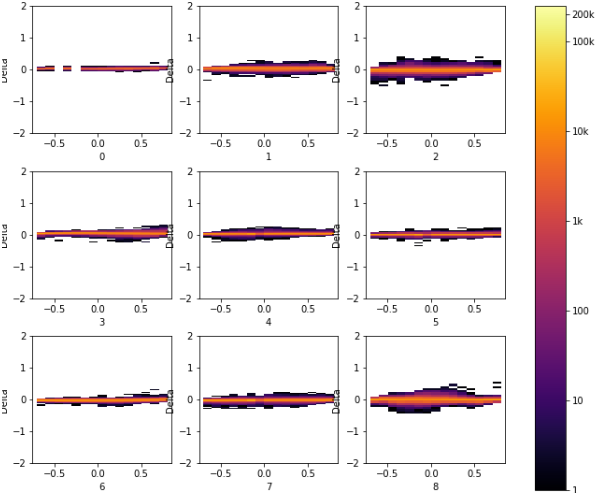
Distribution Deltas Encoding 3/18



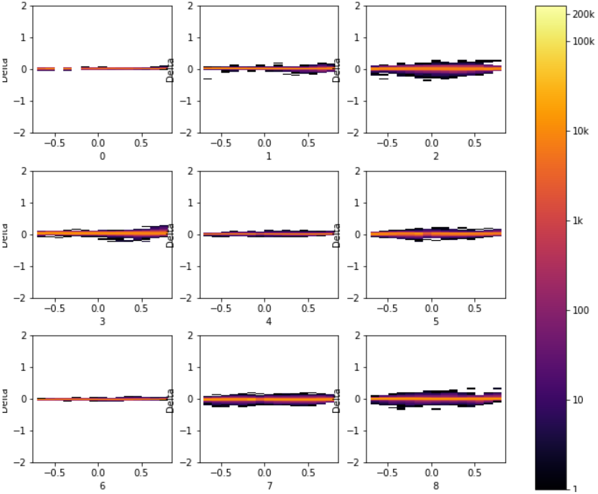
Distribution Deltas Encoding 6/18



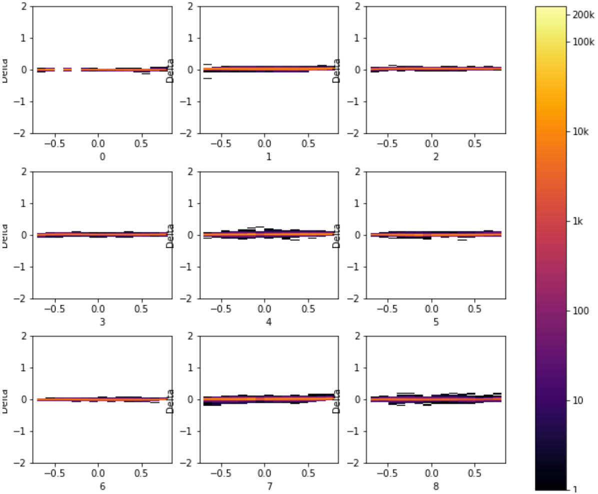
Distribution Deltas Encoding 9/18



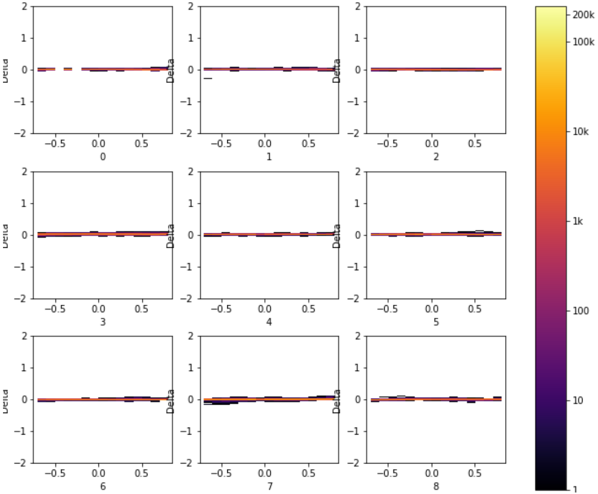
Distribution Deltas Encoding 12/18



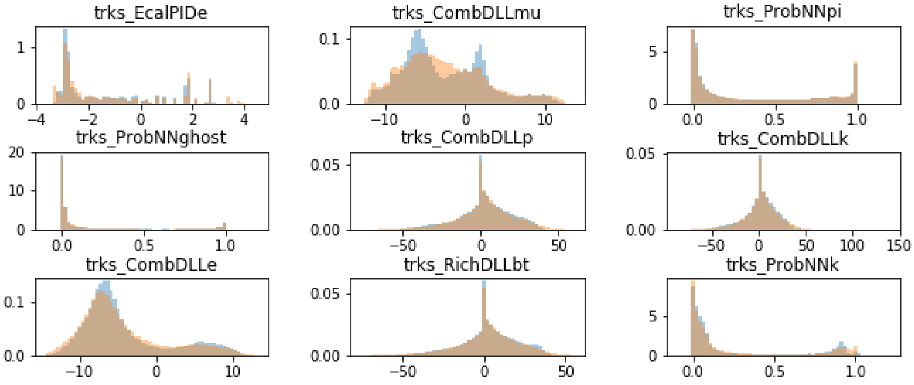
Distribution Deltas Encoding 15/18



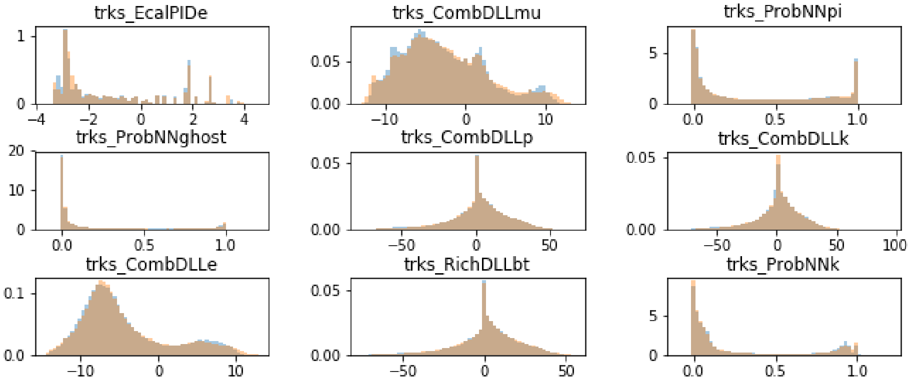
Distribution Deltas Encoding 18/18



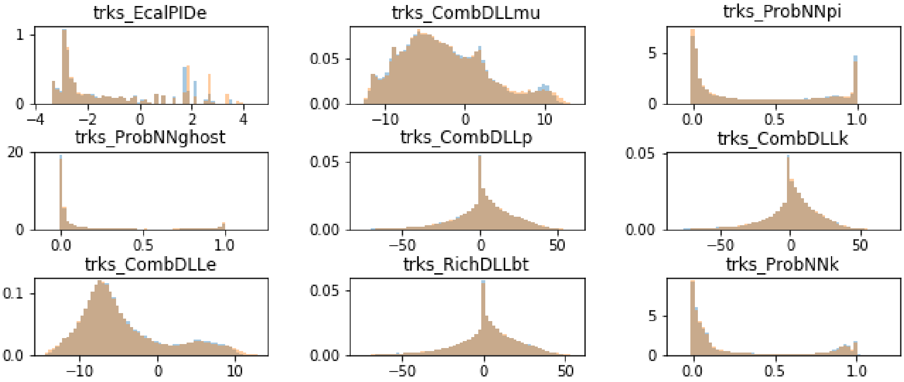
Distributions Encoding 3/18



Distributions Encoding 6/18



Distributions Encoding 9/18



Distributions Encoding 18/18