Foundation models using ATLAS pile-up dataset

Samuel Morand

29.08.2025

Content

- Goal of the project
- Data
- Pre-training : Masked autoencoder (MAE)
  - Deepset
  - Transformer
  - Results
- Downstream task : JVT classification/regression
  - Models
  - Latent space representations
- Conclusion

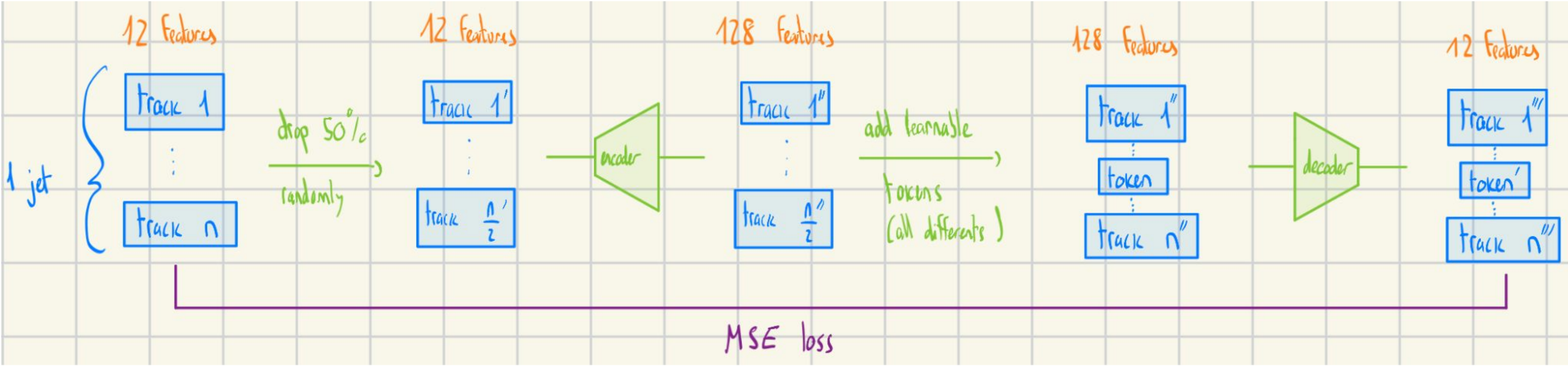
Goal of the project

1. Pre-train a model on a subset of the (huge) ATLAS pile-up dataset
2. Assess the model's ability to perform downstream task in comparison with unpre-trained and simpler models
3. Enable me to familiarise myself with modern ML tools in HEP

Data

- ~ 100 millions jets with Pt > 15 GeV, Eta < 2.0
- Jet features (scalar) : Eta, Phi, Pt, E, # tracks
- Track features (vector) : Pt, Eta, Phi, D0, Z0, Theta, DeltaZ
- Data preparation :
  - Tracks with Pt > 25 GeV removed (spoil normalization o.w.)
  - Normalization of data using mean and std computed on a small subset of data (1 million)
  - Scalar features concatenated at the end of each tracks (5 + 7 = 12 features)
  - Each jet is padded to 35 tracks → improve a lot data loading for the model
  - Masking is used to discard padded tracks
- Prepared data saved in hdf5 files containing 1 million jets each
- 1 file for validation, the rest for training

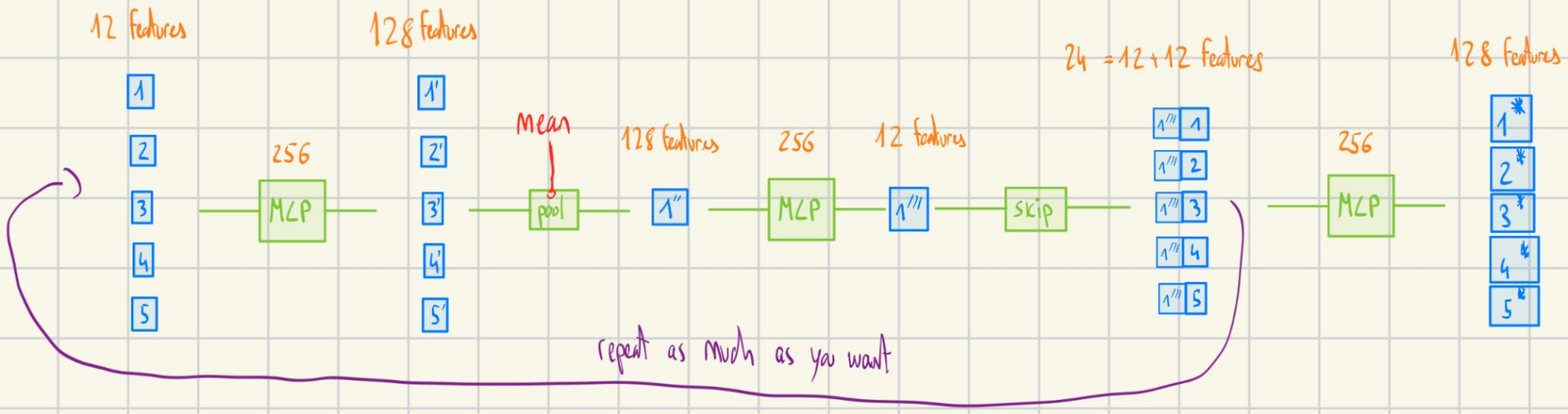
Pre-training : Masked autoencoder (MAE)



- random masking on real tracks is used to drop 50% of them
- better performance with different learnable tokens rather than a single one

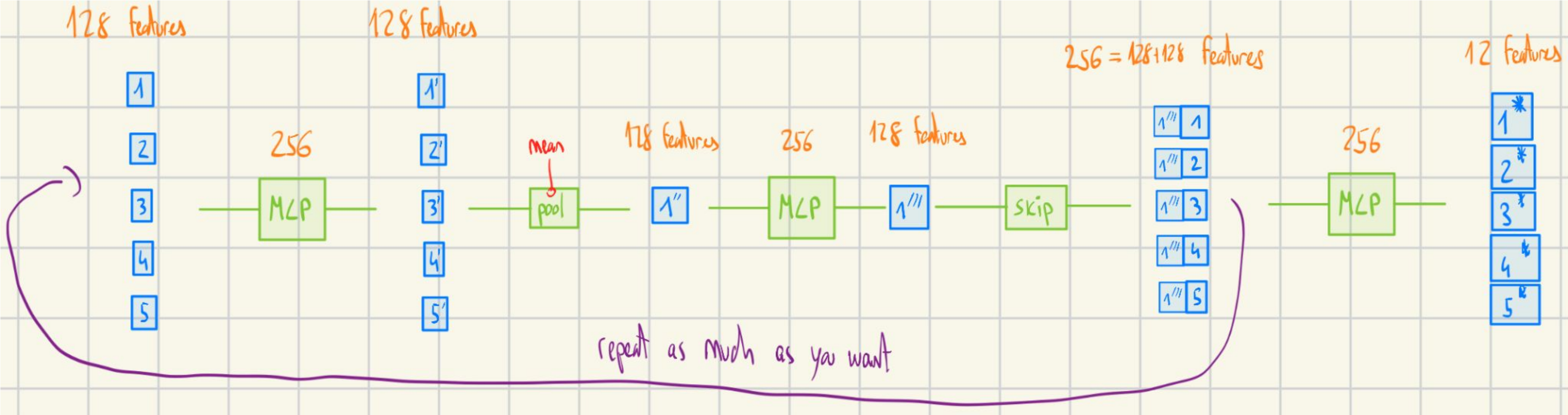
→ for these 2 tasks, need well optimized code to avoid bottlenecks

Deepsets : encoder

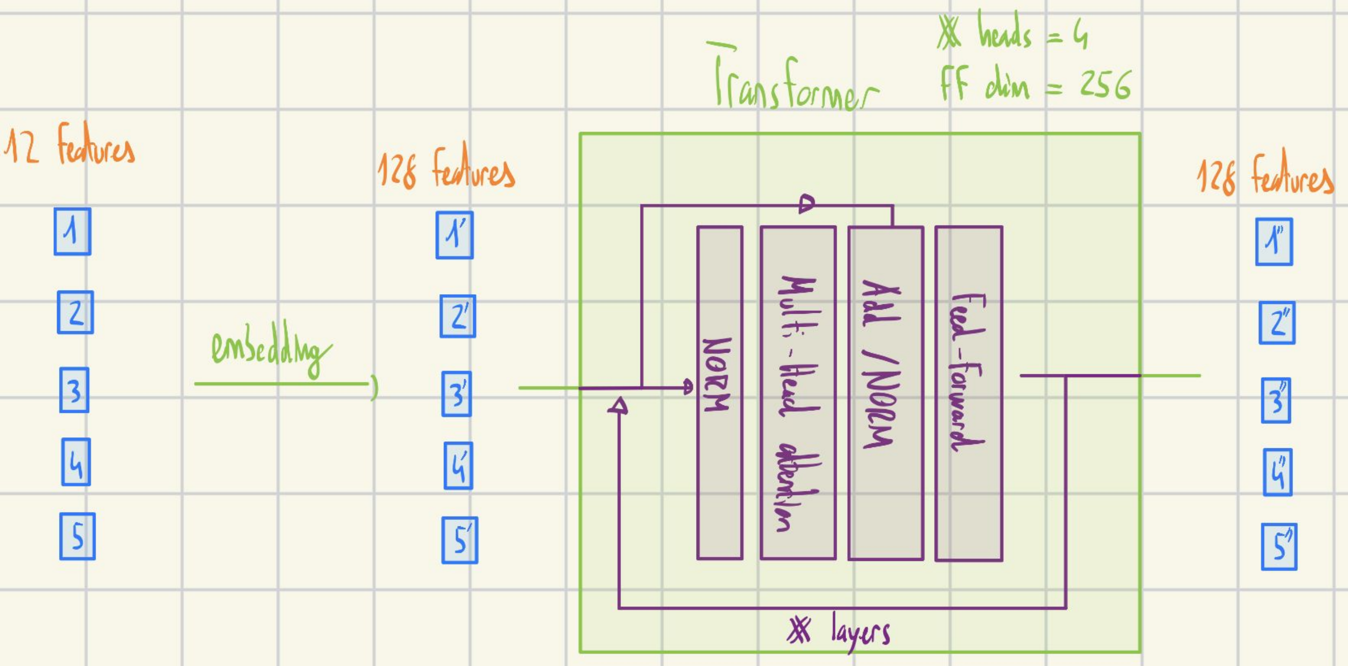


- ReLU activation function

Deepsets : decoder

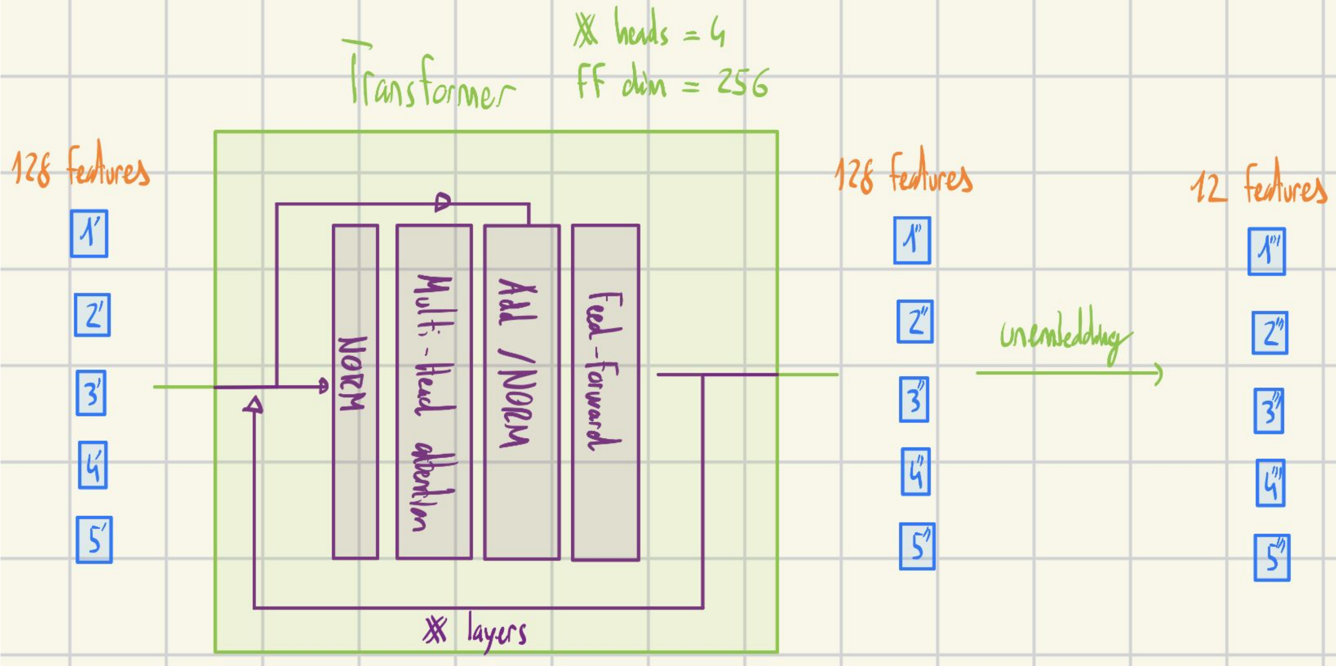


Transformer : encoder



- implemented using Pytorch pre-built class based on “All you need is Attention”

Transformer : decoder

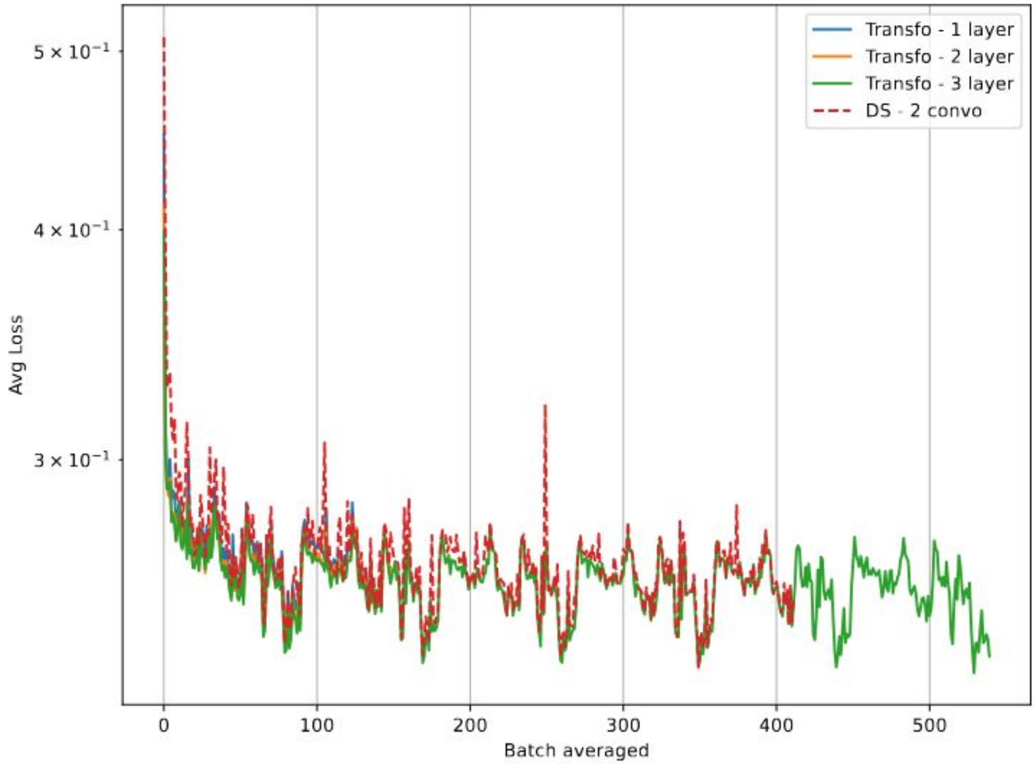


- ReLU activation function

MAE : Results

- trained on lxplus with 1 GPU and 8 CPUs for data loading

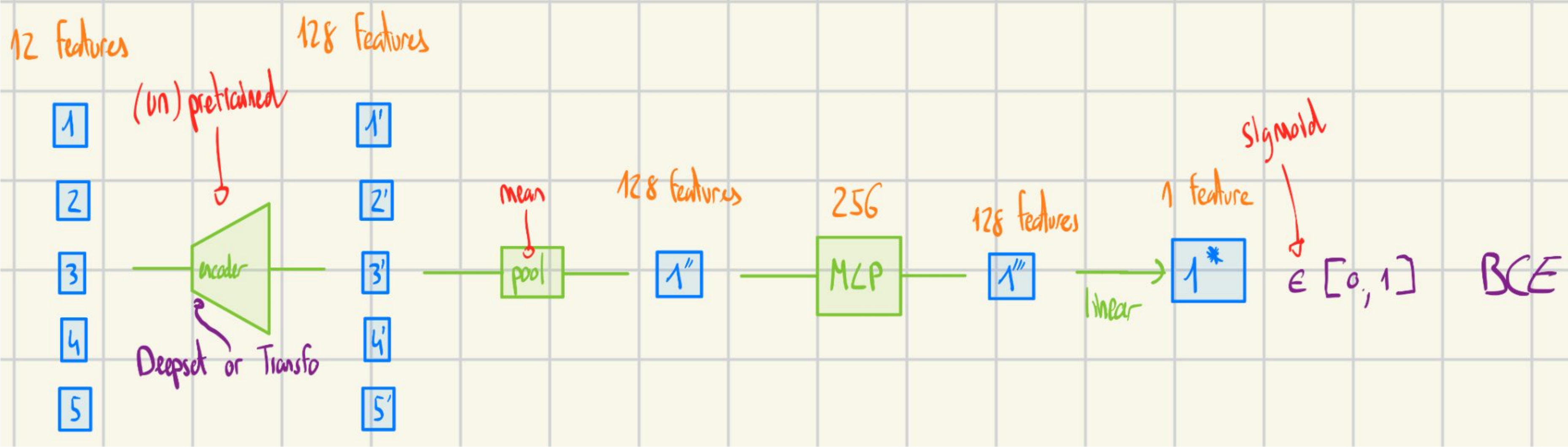
- batch size = 1000
- batch number ~ 100’000
- 6 epochs
- lr = 1e-4
- plateau after 2 epochs



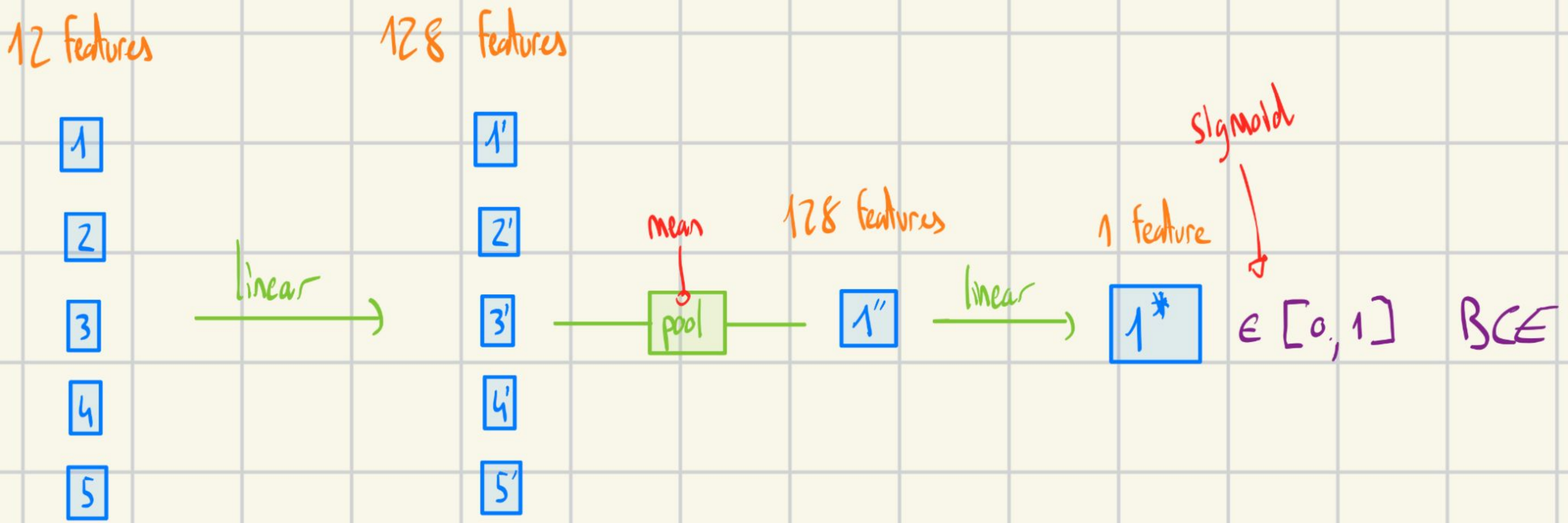
JVT Classification

- Goal : classify whether pile-up jet or not
- Since using real data, no truth labels → best option is to use JVT, o.w MC
- JVT <= 0.1 → 0 , JVT >= 0.9 → 1 and 0.1 < JVT < 0.9 discarded
- perfectly balanced dataset that contains ~ 8 millions jets
- 200’000 jets for validation and the rest for training

Classification Models : (un)pre-trained encoder



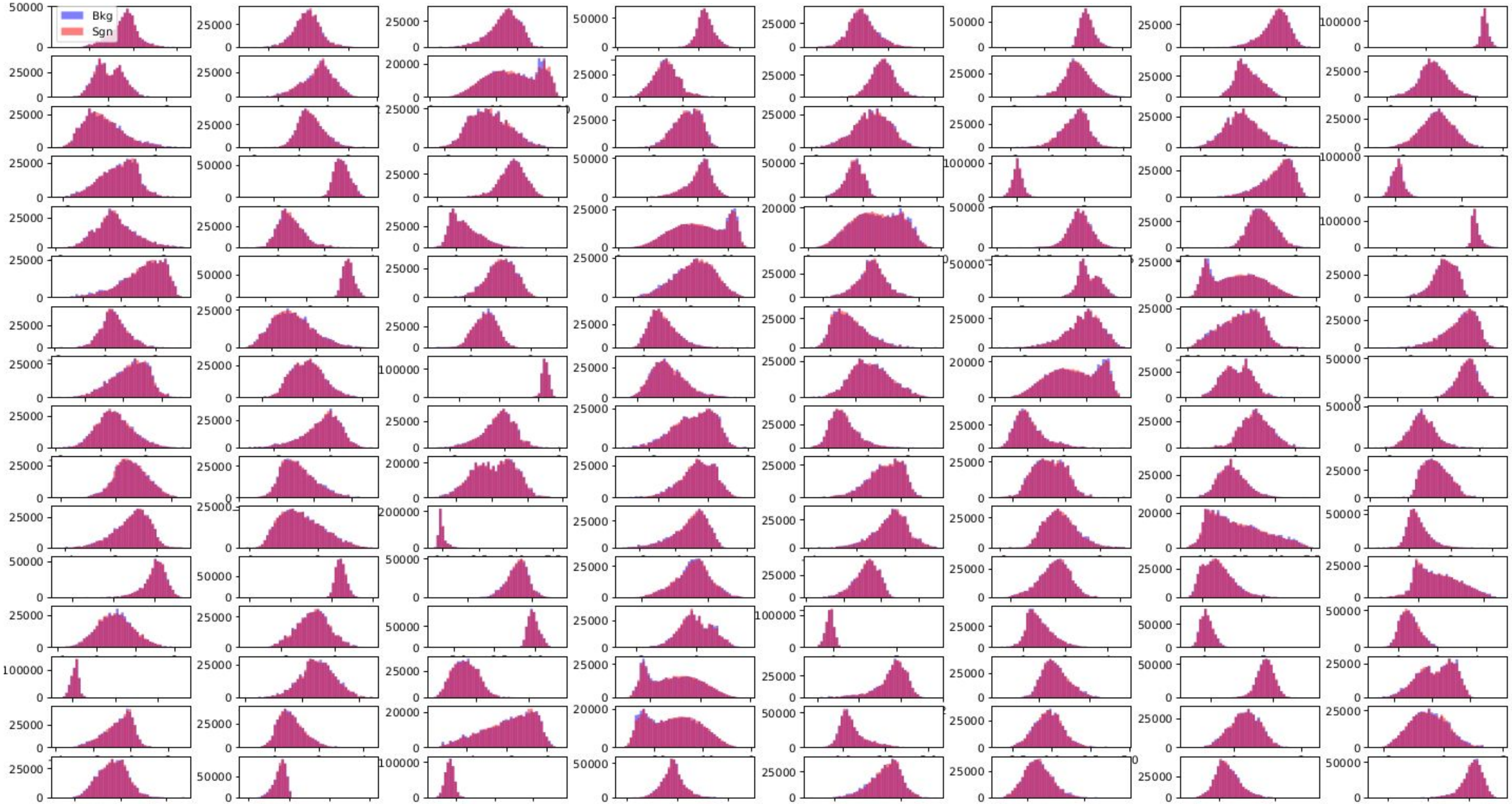
Classification Models : basic MLP



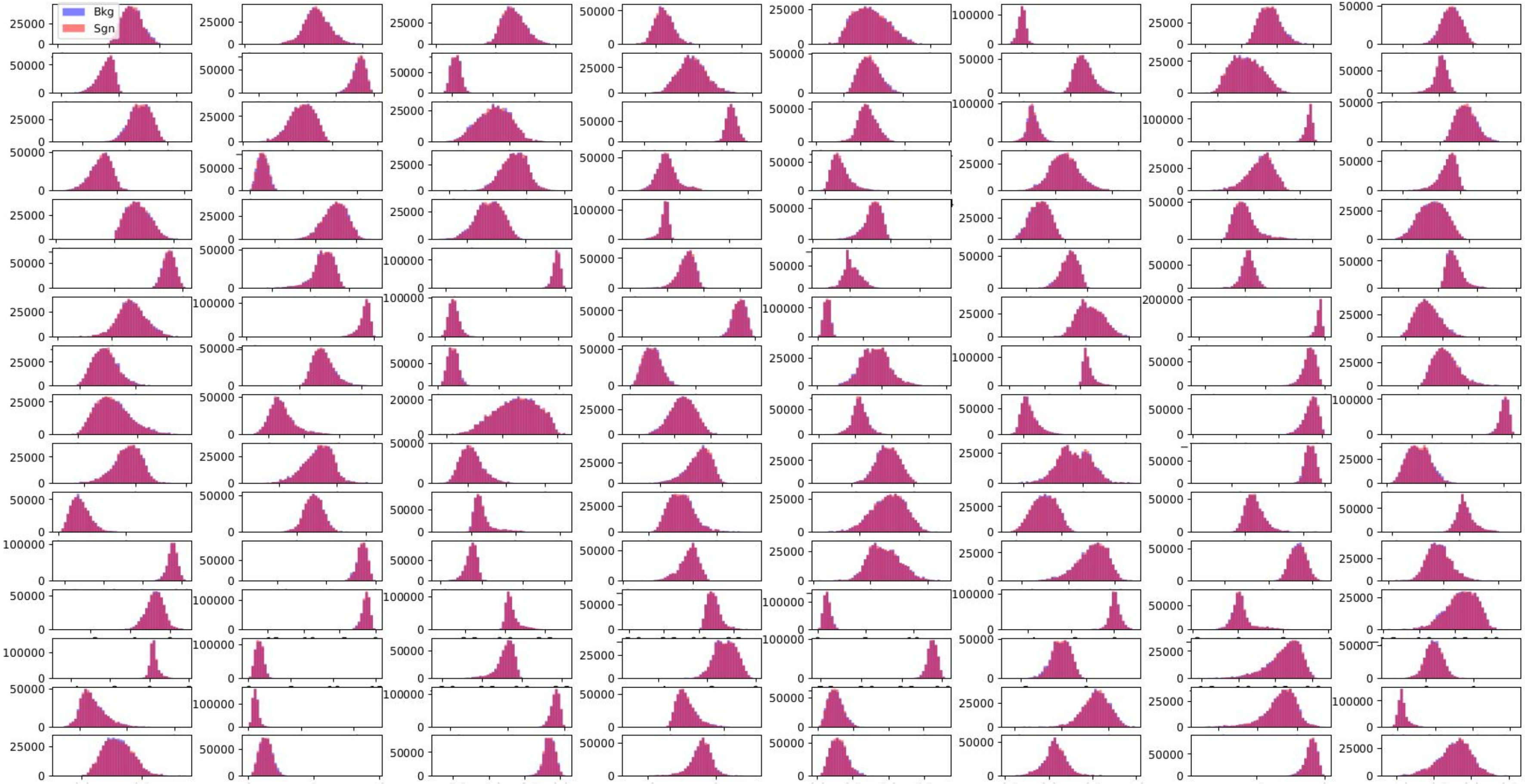
Classification : Results

- Basic MLP achieves AUC > 0.8 after 1 epoch and goes until AUC ~ 0.98
- Encoder models barely reached AUC ~ 0.6 whether with frozen or unfrozen encoder
- Possible explanations :
  - for unfrozen encoder the architecture may be too complicated to fine-tune (~ 300k params)
  - for frozen encoder pooled latent space plots provide good hints (cf. next slides)

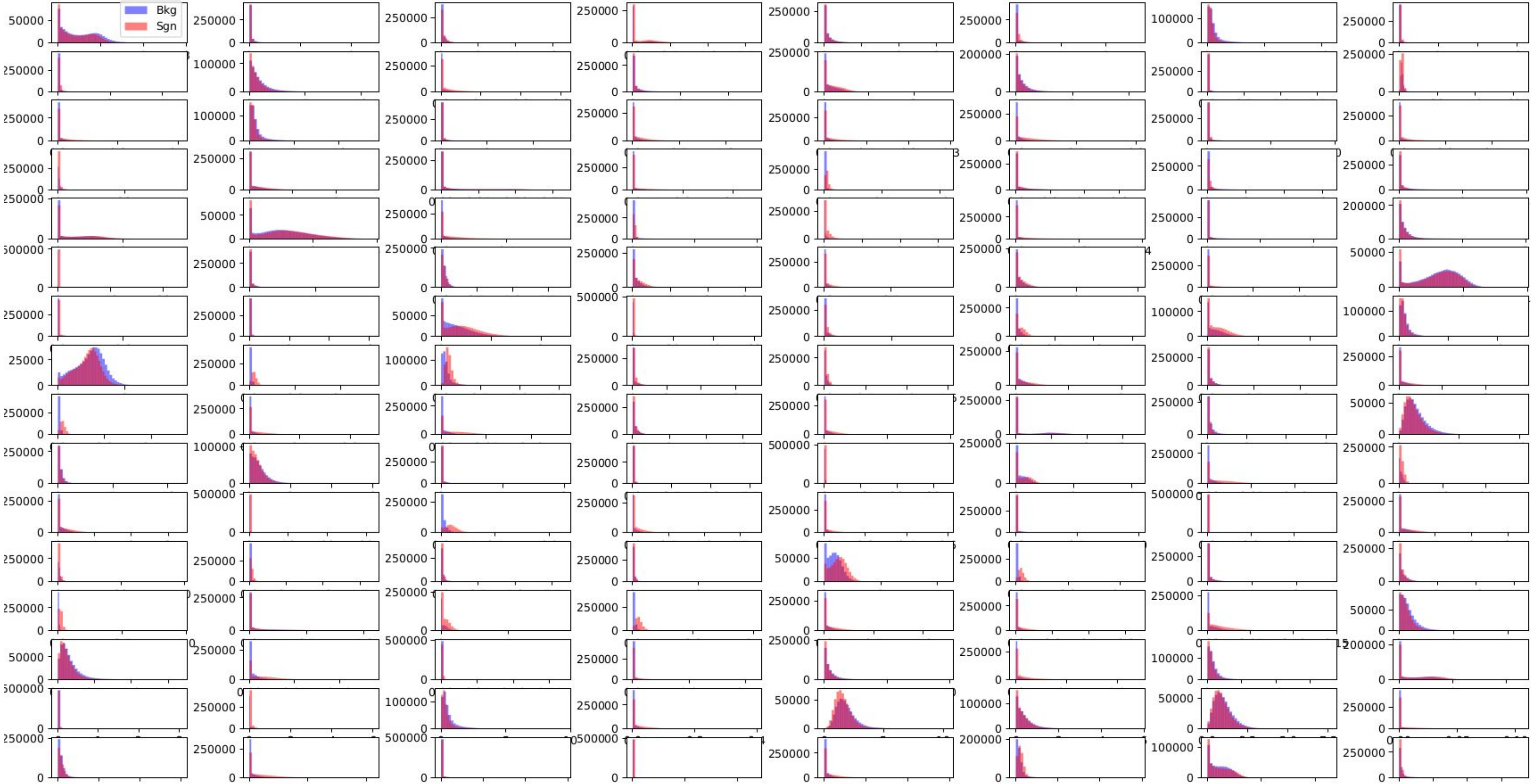
Latent space representation : Transformer pre-trained



Latent space representation : Deepset pre-trained



Latent space representation : basic MLP trained



Conclusion

- Pre-trained models don’t seem to be well-suited for JVT classification or there is a problem in my code
- Would be interesting to try on other downstream tasks if they perform better

THANKS !