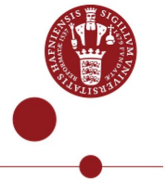
B-jet Momentum Regression Based on Transformer

Zhongqi Zhao, Hengdong Lu, Zhenzhong Tang

UNIVERSITY OF COPENHAGEN



1 Data Background

2 Transformer Architecture

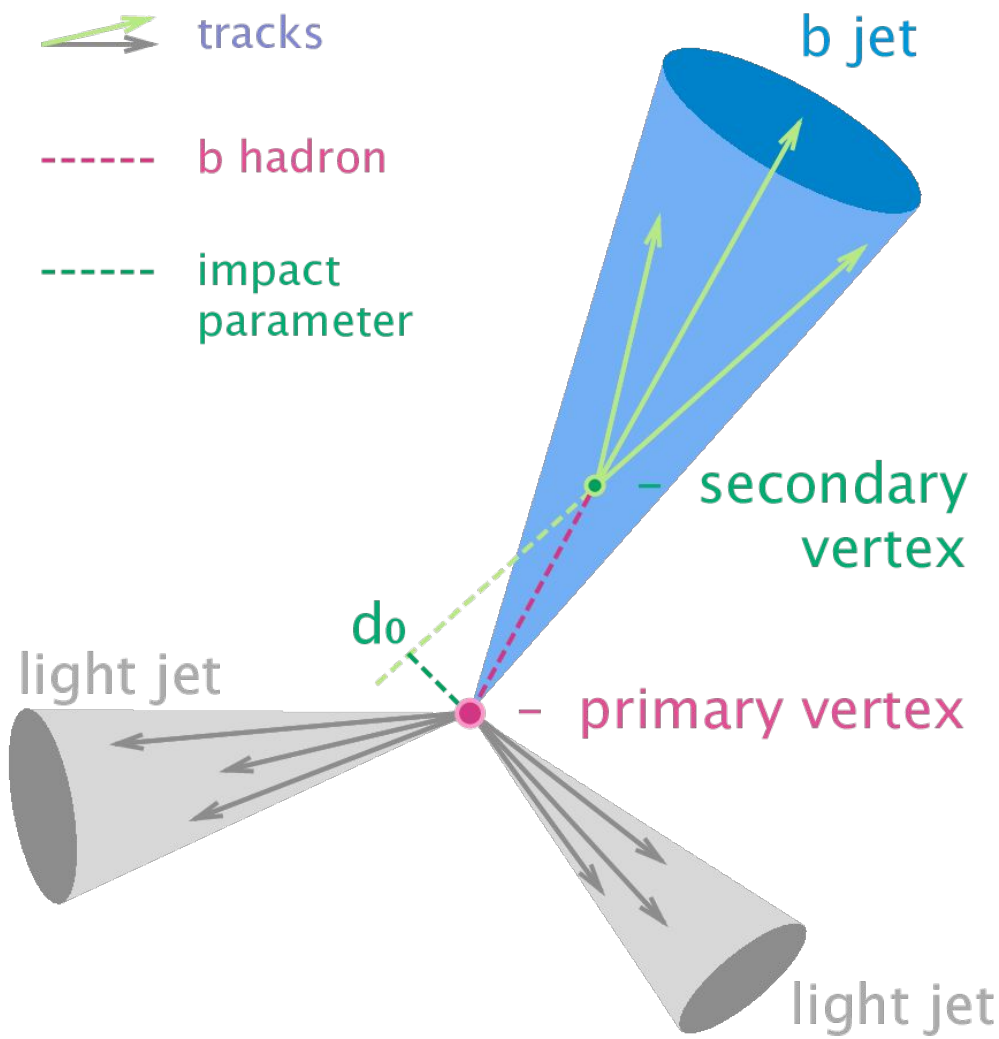
3 Optimization & Techniques

4 Results & Comparison

Data Background

B-jet regression

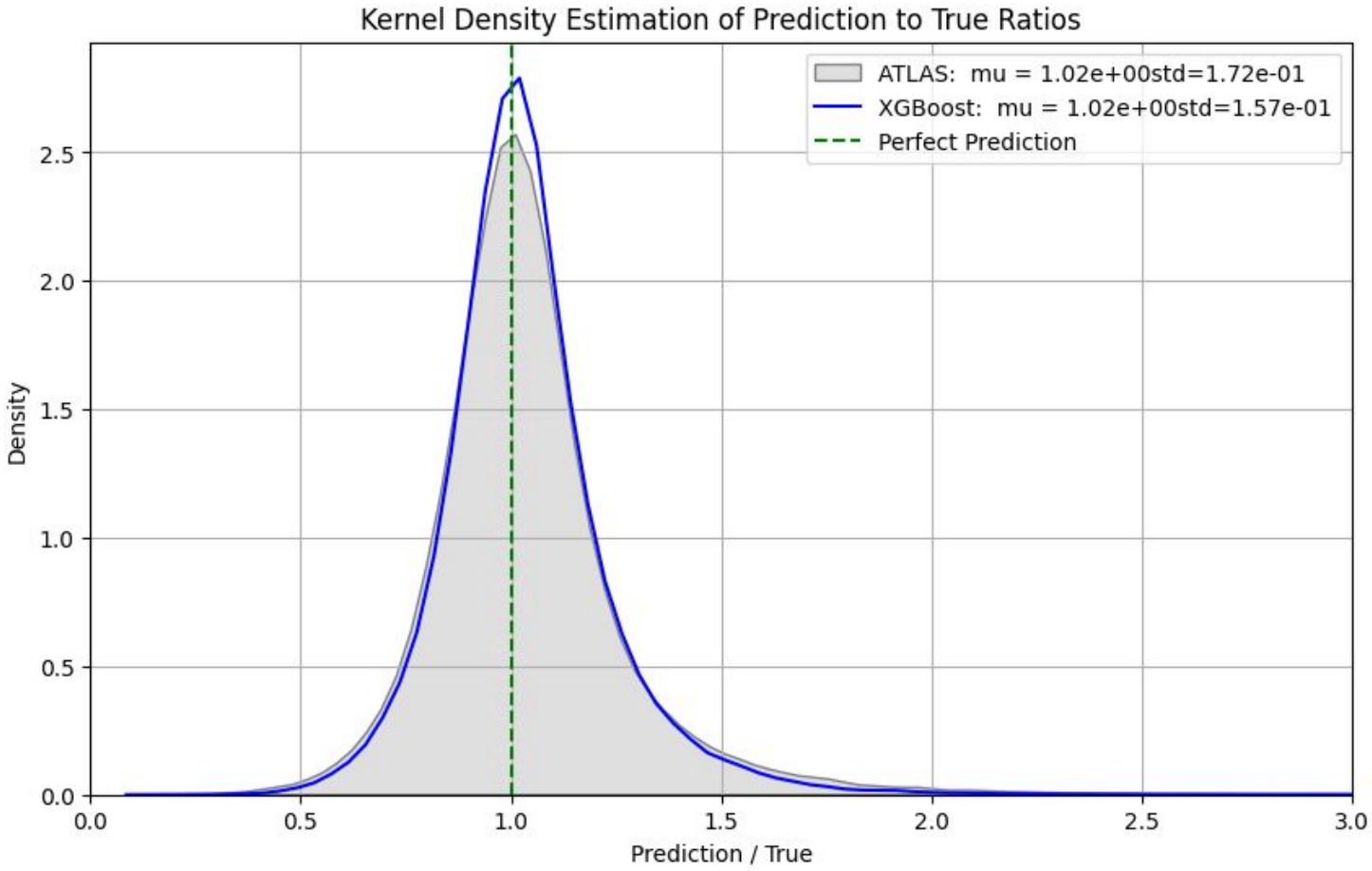
- A b-jet is a beam of particles produced by a collision event between high-energy particles.
- Compare our results with ATLAS prediction.
- Evaluation: average and standard deviation of prediction/truth, mean absolute relative error



Scalar Format Data

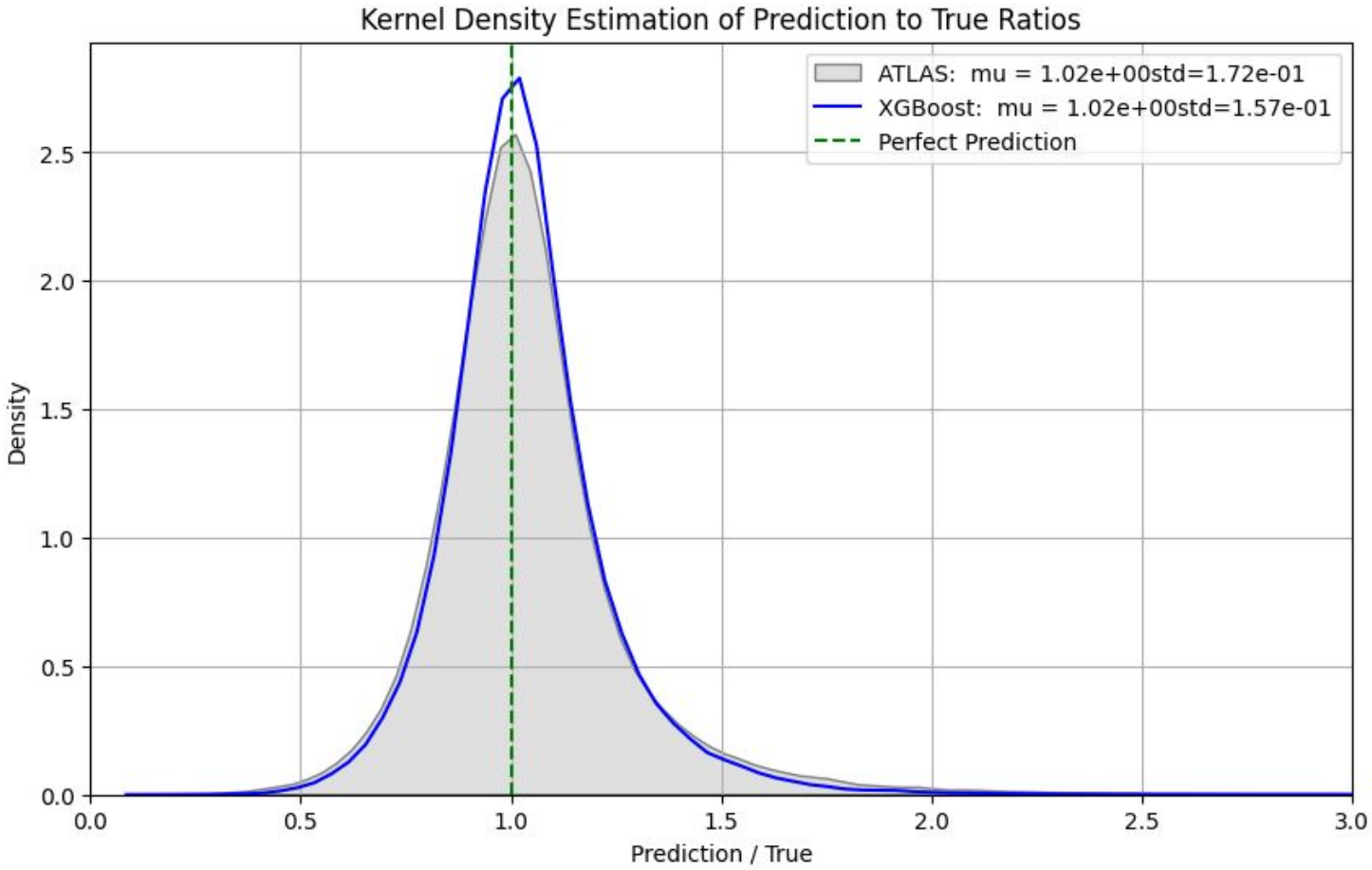
- 780206 data
- XGBoost already better than ALTAS!

How to make that even better?



Scalar Format Data

- 780206 data with 721 characters
- XGBoost already better than ALTAS!

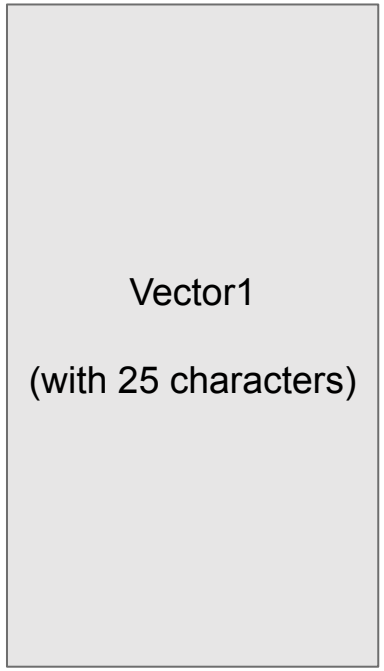


How to make that even better?

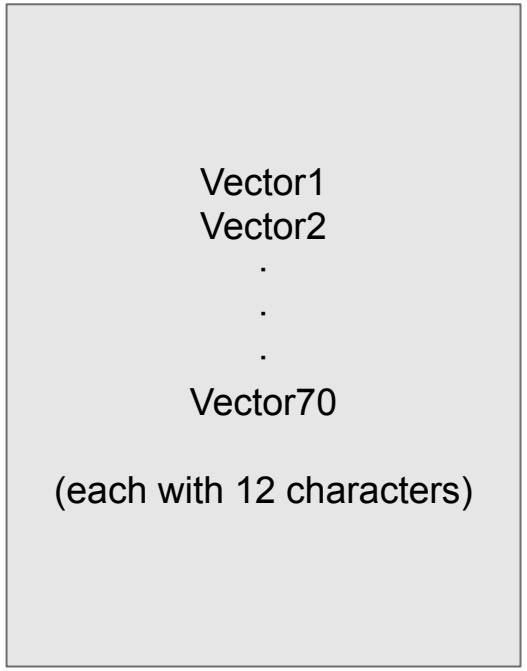
Vector format data contain more information!

Vector Format Data

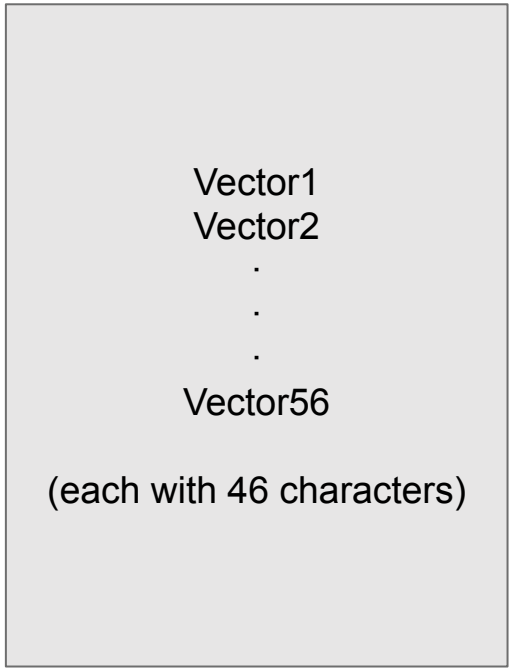
- 3 groups: 'charged_pf', 'neutral_pf', and 'scalar', each containing 780,206 data tensors(highly irregular shape) corresponding to that in scalar format data.
- Scalars in a single vector represent different characters.



Single ‘scalar’ tensor
shape=(1,25)



Single ‘charged_pf’ tensor
shape=(70,12)



Single ‘neutral_pf’ tensor
shape=(56,46)

Vector Format Data

- Lots of invalid data exist in ‘charged_pf’ and ‘neutral_pf’, which would cause huge data noise.

```
{'charged_pf': array([[( 2.2459798,  1.1132307 , -2.8604834 ,  3.7900486, 139.57, -1., 0., 8208., 2.2459797e+03,  28.127182 ,  0.1430246 ,  True),
        ( 1.788123 ,  1.2620555 , -2.649424  ,  3.4143665, 139.57,  1., 0., 8208., 1.7881230e+03,  28.133904 ,  0.01648689,  True),
        ( 1.6688688,  1.1936611 , -3.1198075 ,  3.0090773, 139.57,  1., 0., 8208., 1.6688688e+03, -12.799423 ,  0.00056229,  True),
        ...,
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False)],
       [( 2.6027467,  2.2941675 , -1.4546521 , 13.036634 , 139.57, -1., 0., 8208., 2.6027468e+03,  -2.8193183,  0.06687441,  True),
        ( 1.1454937,  2.1662593 , -1.4887234 ,  5.0651135, 139.57,  1., 0., 8208., 1.1454938e+03, -10.840013 , -0.01959403,  True),
        ( 1.0948113,  2.4059186 , -1.3677806 ,  6.1209264, 139.57,  1., 0., 8208., 1.0948113e+03,  -3.3708525, -0.04104954,  True),
        ...,
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False)],
       [( 4.5994387, -1.7090869 ,  1.790061  , 13.120517 , 139.57, -1., 0., 8208., 4.5994385e+03, -21.33372  ,  0.03466429,  True),
        ( 3.513736 , -1.572401  ,  1.8378757 ,  8.830682 , 139.57, -1., 0., 8208., 3.5137361e+03, -19.407558 , -0.02446163,  True),
        ( 3.0004964, -1.9127998 ,  1.7642748 , 10.382189 , 139.57,  1., 0., 8208., 3.0004963e+03, -21.356228 , -0.05878485,  True),
        ...,
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False)],
       ...,
       [(68.24743  , -0.2006337 ,  2.8248081 , 69.62579  , 139.57,  1., 0., 8208., 6.8247430e-16,  53.318344 , -0.01279807,  True),
        (49.444096 , -0.21111326,  2.8046072 , 50.550217 , 139.57, -1., 0., 8208., 4.8713198e+03,  53.233936 ,  0.03639252,  True),
        (47.147285 , -0.20912382,  2.828629  , 48.18219  , 139.57,  1., 0., 8208., 5.6557241e+03,  53.34078  ,  0.01992808,  True),
        ...,
```

1 tensor in ‘charged_pf’ (totally 780000)

noice

valid

Vector Format Data

- Lots of invalid data exist in ‘charged_pf’ and ‘neutral_pf’, which would cause huge data noise.

```
{'charged_pf': array([[( 2.2459798,  1.1132307 , -2.8604834 ,  3.7900486, 139.57, -1., 0., 8208., 2.2459797e+03,  28.127182 ,  0.1430246 ,  True),
        ( 1.788123 ,  1.2620555 , -2.649424  ,  3.4143665, 139.57,  1., 0., 8208., 1.7881230e+03,  28.133904 ,  0.01648689,  True),
        ( 1.6688688,  1.1936611 , -3.1198075 ,  3.0090773, 139.57,  1., 0., 8208., 1.6688688e+03, -12.799423 ,  0.00056229,  True),
        ...,
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False)],
       [( 2.6027467,  2.2941675 , -1.4546521 , 13.036634 , 139.57, -1., 0., 8208., 2.6027468e+03,  -2.8193183,  0.06687441,  True),
        ( 1.1454937,  2.1662593 , -1.4887234 ,  5.0651135, 139.57,  1., 0., 8208., 1.1454938e+03, -10.840013 , -0.01959403,  True),
        ( 1.0948113,  2.4059186 , -1.3677806 ,  6.1209264, 139.57,  1., 0., 8208., 1.0948113e+03,  -3.3708525, -0.04104954,  True),
        ...,
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False)],
       [( 4.5994387, -1.7090869 ,  1.790061  , 13.120517 , 139.57, -1., 0., 8208., 4.5994385e+03, -21.33372  ,  0.03466429,  True),
        ( 3.513736 , -1.572401  ,  1.8378757 ,  8.830682 , 139.57, -1., 0., 8208., 3.5137361e+03, -19.407558 , -0.02446163,  True),
        ( 3.0004964, -1.9127998 ,  1.7642748 , 10.382189 , 139.57,  1., 0., 8208., 3.0004963e+03, -21.356228 , -0.05878485,  True),
        ...,
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
        ( 0.       ,  0.        ,  0.        ,  0.       ,   0.  ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False)],
       ...,
       [(68.24743  , -0.2006337 ,  2.8248081 , 69.62579  , 139.57,  1., 0., 8208., 6.8247430e-16,  53.318344 , -0.01279807,  True),
        (49.444096 , -0.21111326,  2.8046072 , 50.550217 , 139.57, -1., 0., 8208., 4.8713198e+03,  53.233936 ,  0.03639252,  True),
        (47.147285 , -0.20912382,  2.828629  , 48.18219  , 139.57,  1., 0., 8208., 5.6557241e+03,  53.34078  ,  0.01992808,  True),
        ...,
```

1 tensor in ‘charged_pf’ (totally 780000)

noice

valid

Too difficult for Tree-based Methods !

Transformer

- Self-attention
  - Long distance dependency capture
  - Multi-head attention(Multi self-attention with different weight)

- Inter-layer parallelism

Suitable for such complex and highly irregularly shaped data

Transformer Architecture

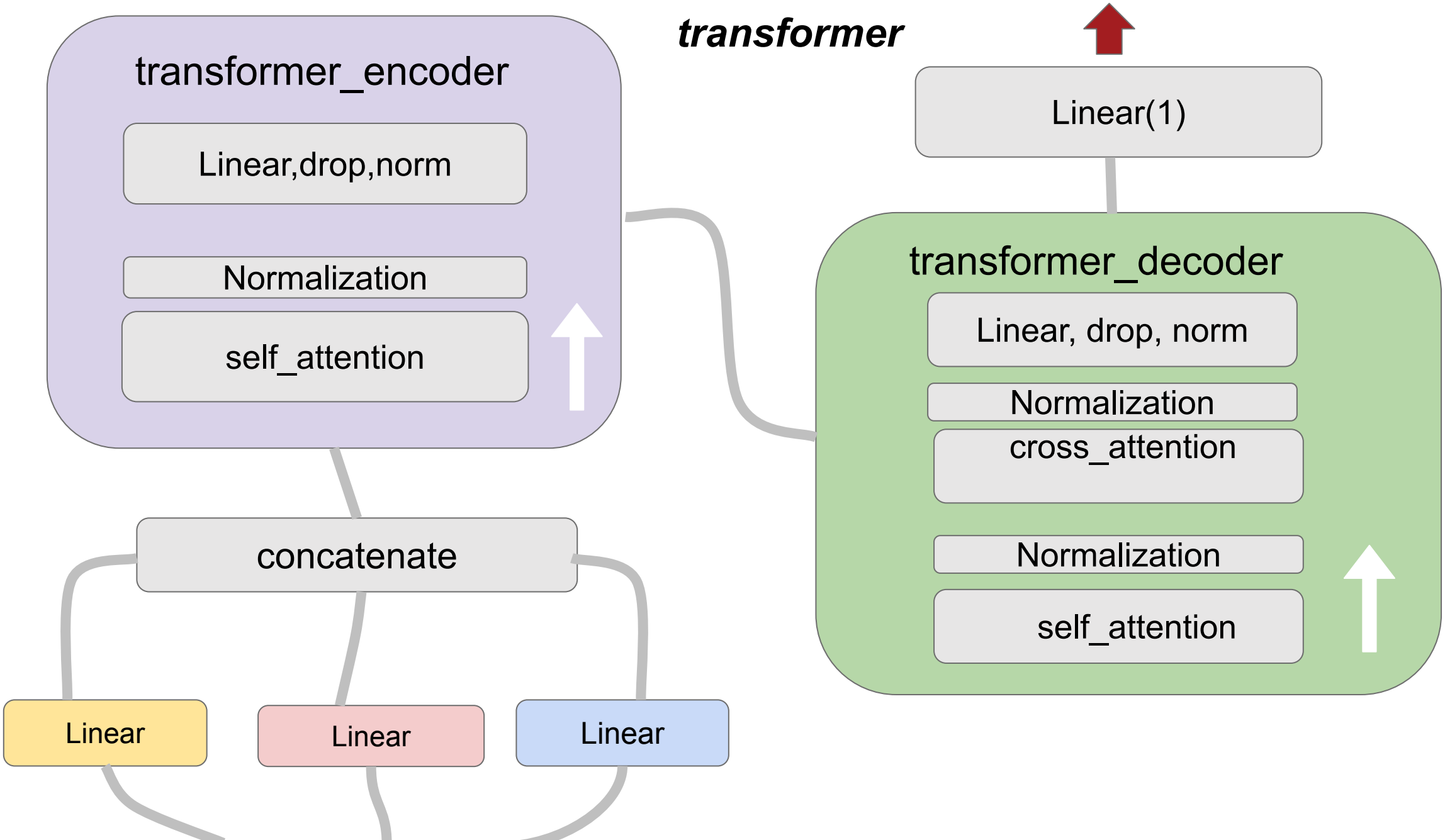
Transformer : Vectors input

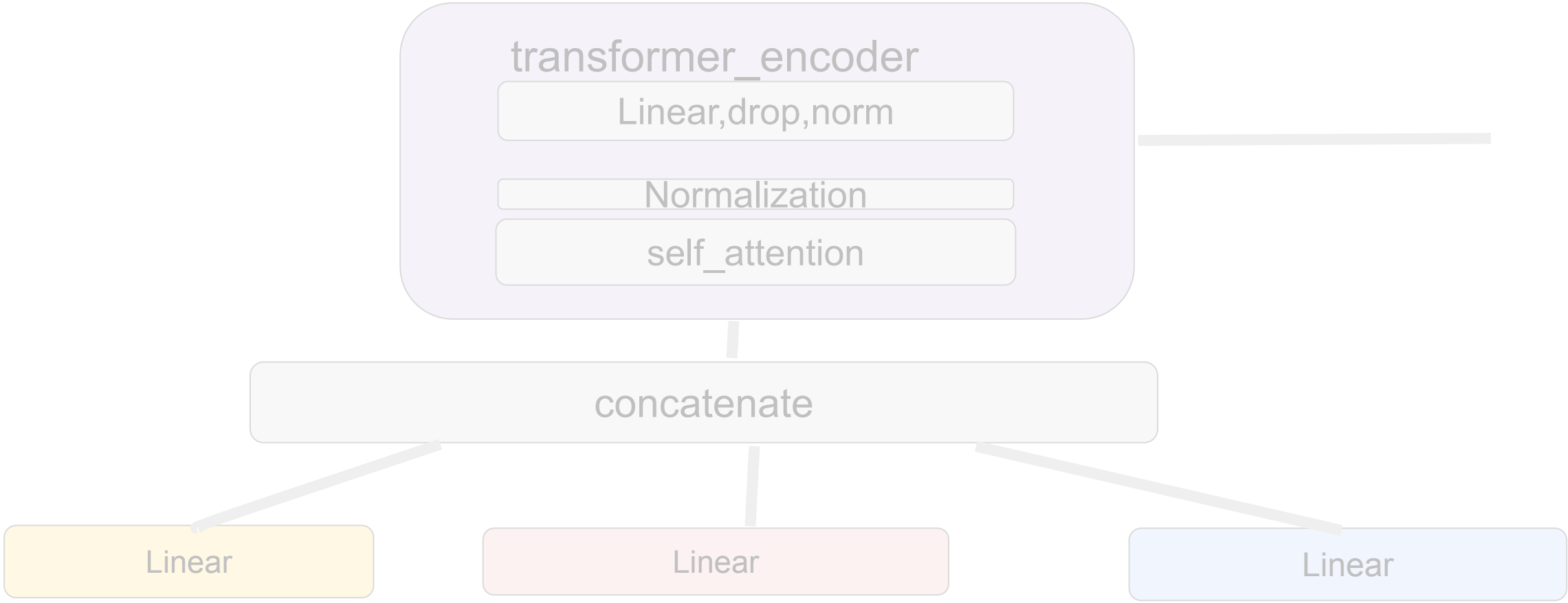
dim = 17	***dim = 11***	***dim = 45***
[Vec1]	[Vec1], [Vec2],..., [Vec70]	[Vec1], [Vec2],..., [Vec45]
Scalar	Charged_pf	Neutral_pf

Sample 1

Sample 2

Sample 780,000

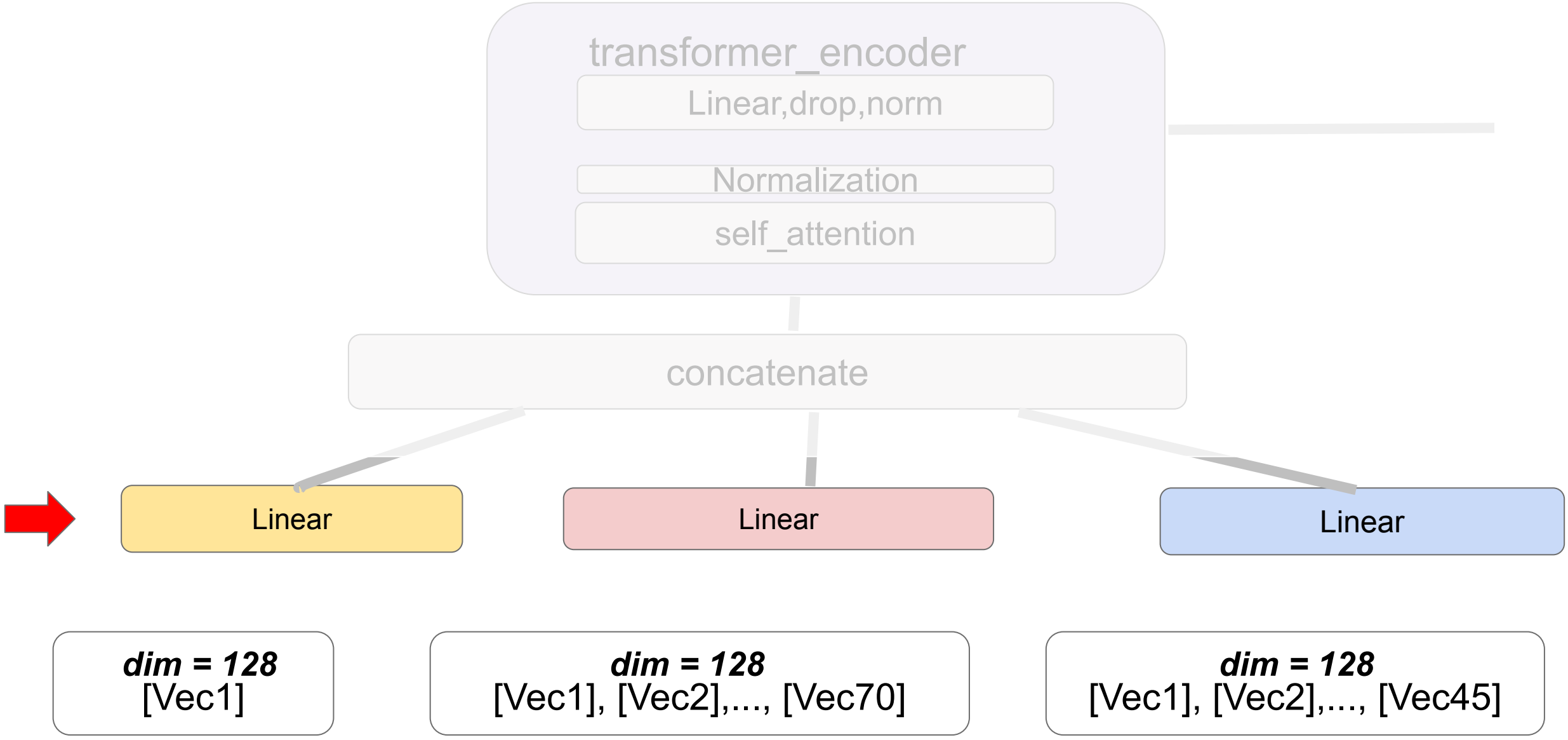

transformer
transformer_encoder
Linear,drop,norm
Normalization
self_attention
concatenate
Linear
Linear
Linear
Linear(1)
transformer_decoder
Linear, drop, norm
Normalization
cross_attention
Normalization
self_attention

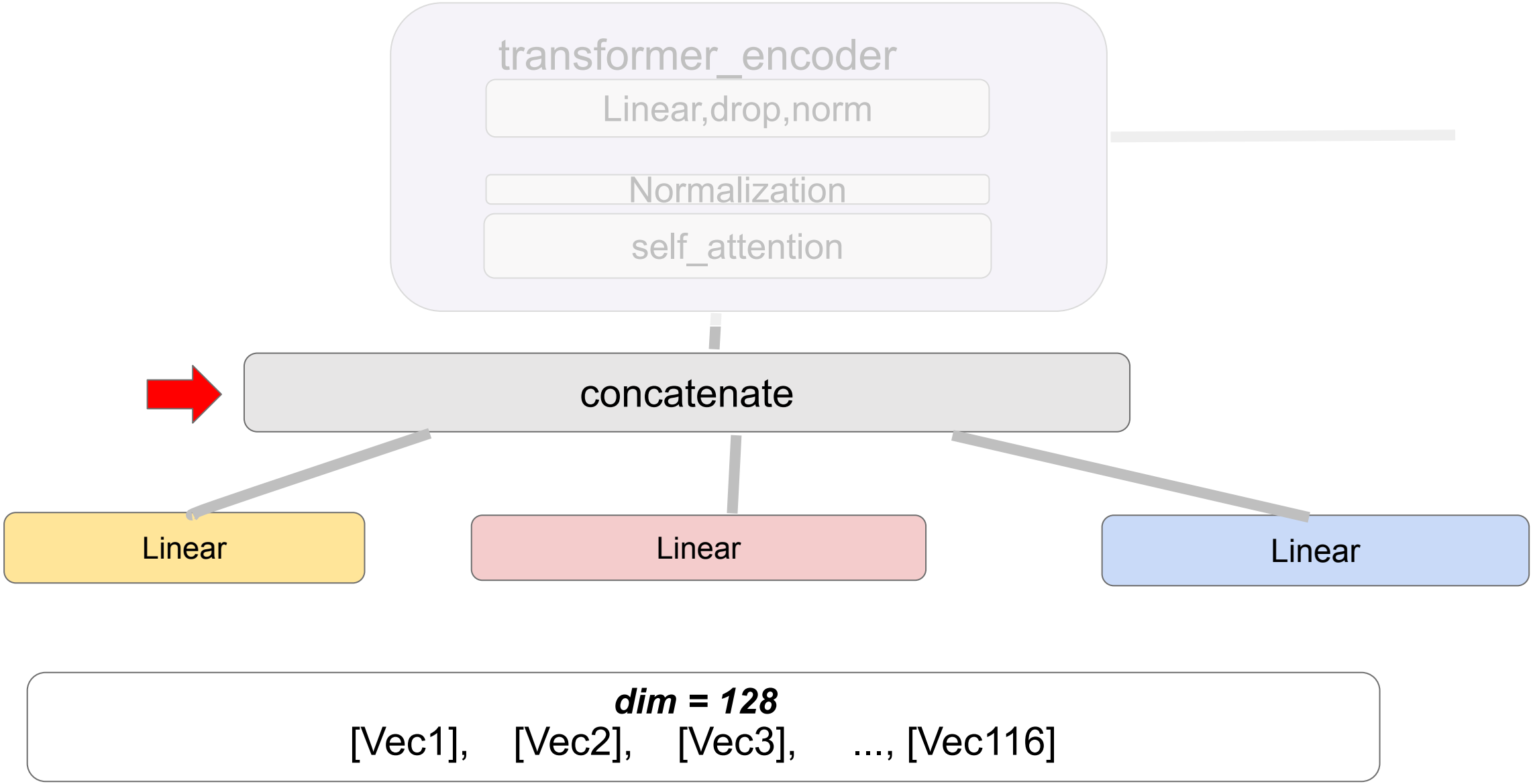


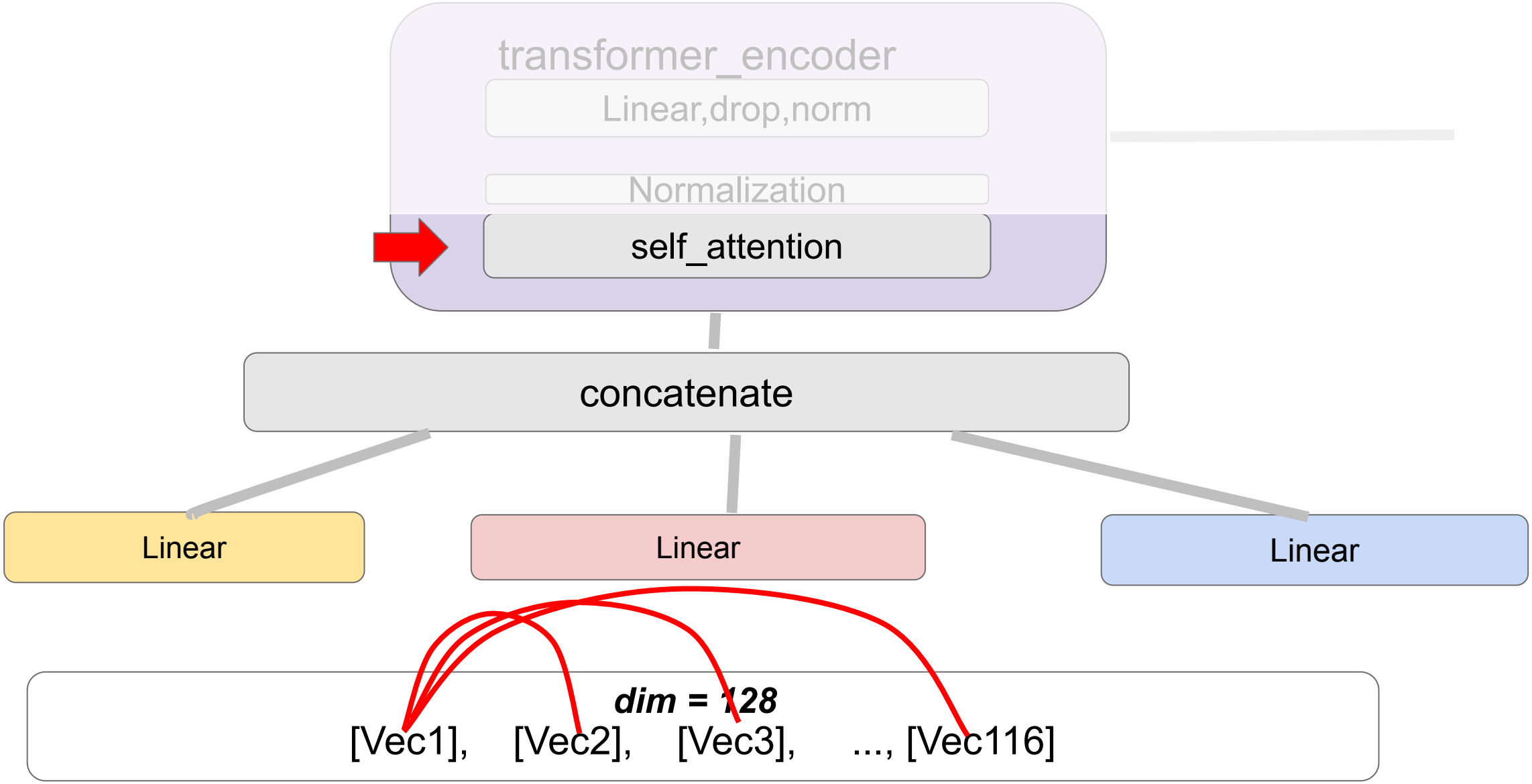
dim = 17
[Vec1]

dim = 11
[Vec1], [Vec2],..., [Vec70]

dim = 45
[Vec1], [Vec2],..., [Vec45]

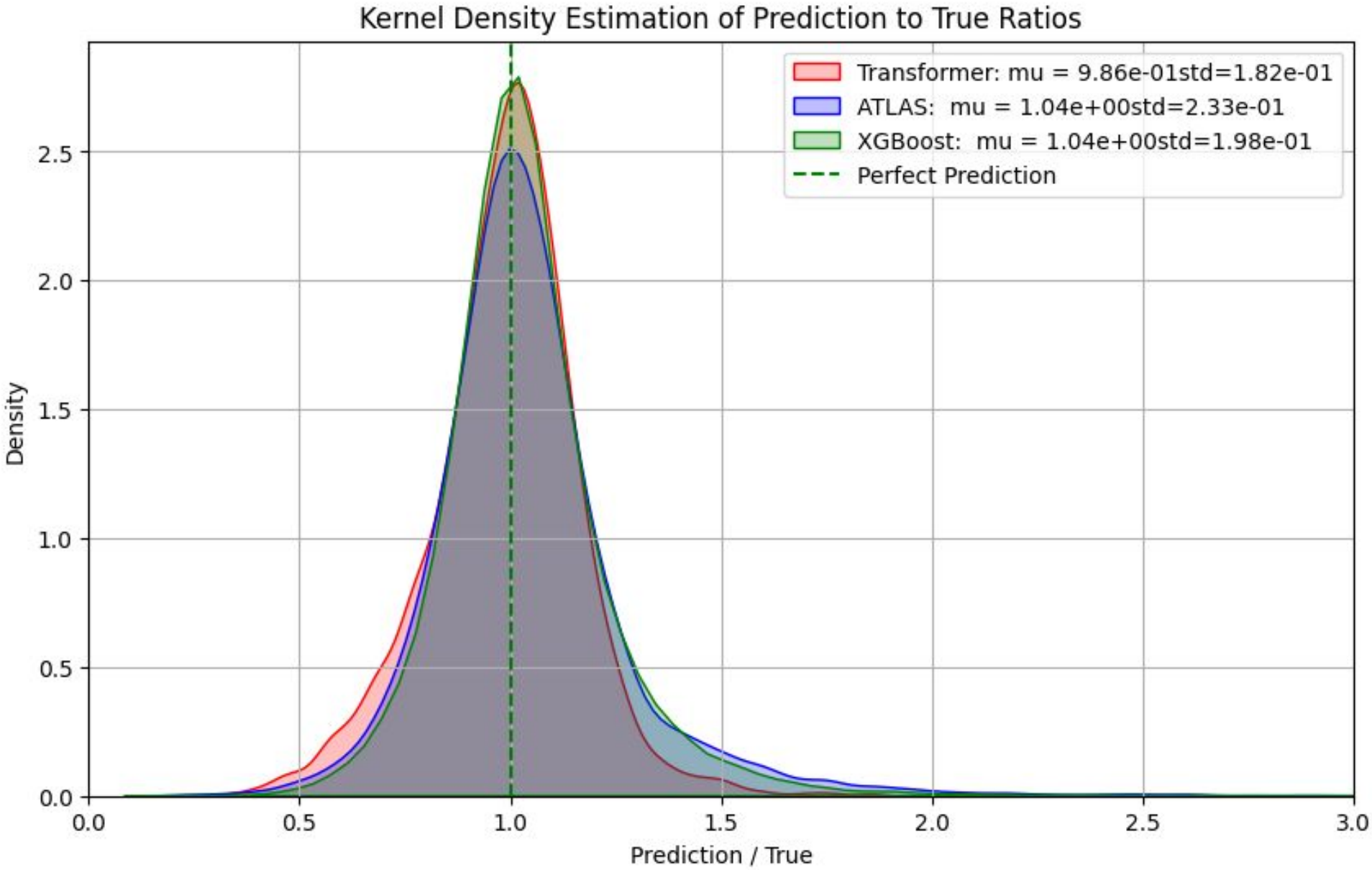

transformer_encoder
Linear,drop,norm
Normalization
self_attention
concatenate
Linear
Linear
Linear
dim = 128
[Vec1]
dim = 128
[Vec1], [Vec2],..., [Vec70]
dim = 128
[Vec1], [Vec2],..., [Vec45]


transformer_encoder
Linear,drop,norm
Normalization
self_attention
concatenate
Linear
Linear
Linear
dim = 128
[Vec1], [Vec2], [Vec3], ..., [Vec116]


transformer_encoder
Linear,drop,norm
Normalization
self_attention
concatenate
Linear
Linear
Linear
dim = 128
[Vec1], [Vec2], [Vec3], ..., [Vec116]

XGBoost
mre:0.140

Transformer
mre:
0.134



How to improve it to beat XGBoost?

dim = 17	***dim = 11***	***dim = 45***
[Vec1]	[Vec1], [Vec2],..., [Vec70]	[Vec1], [Vec2],..., [Vec45]
Scalar	Charged_pf	Neutral_pf

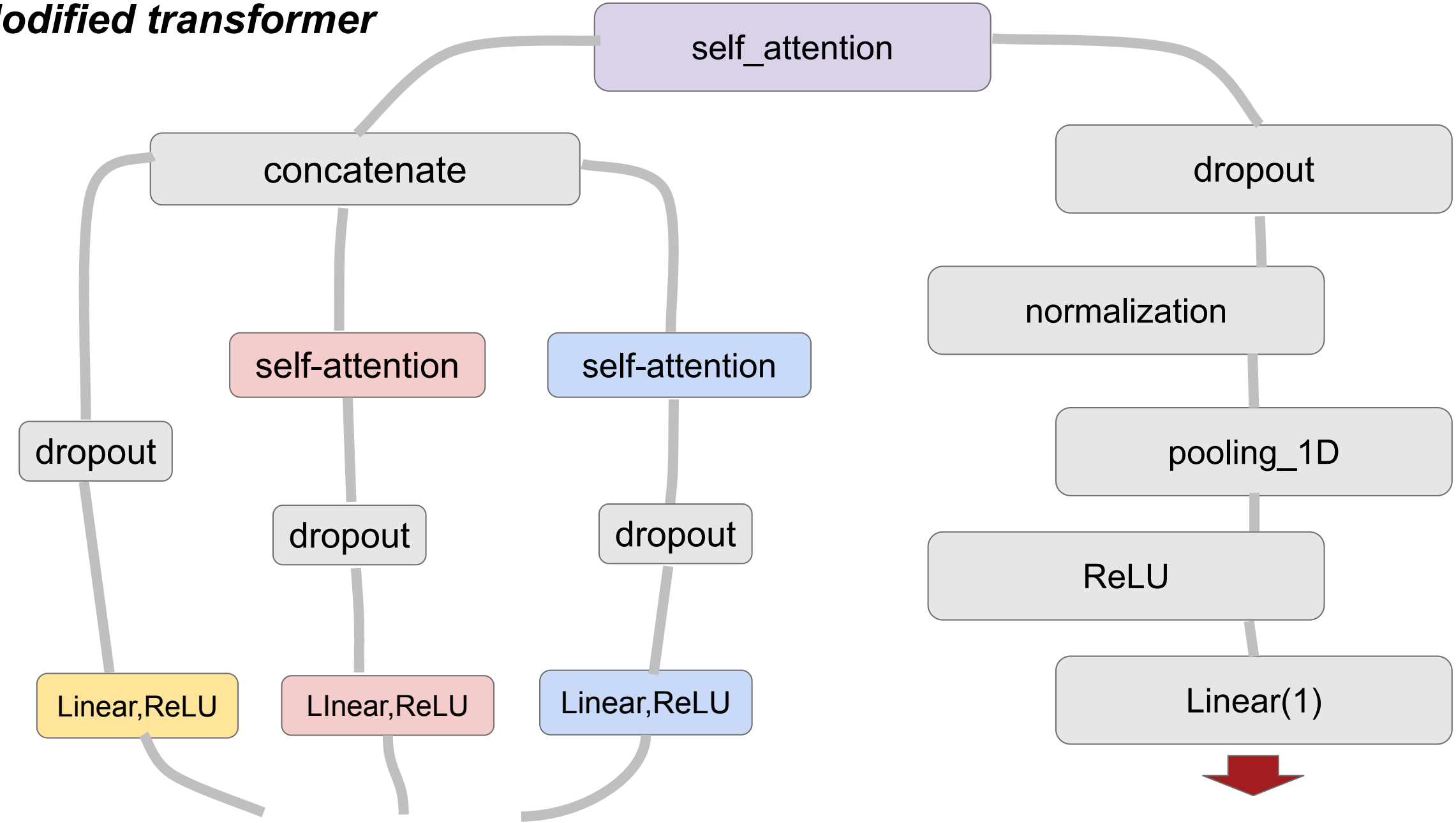
- Point 1: Attention WITHIN each group

Individual attention for charged_pf, neutral_f

- Point 2: Is decoder necessary?

Simplify transformer

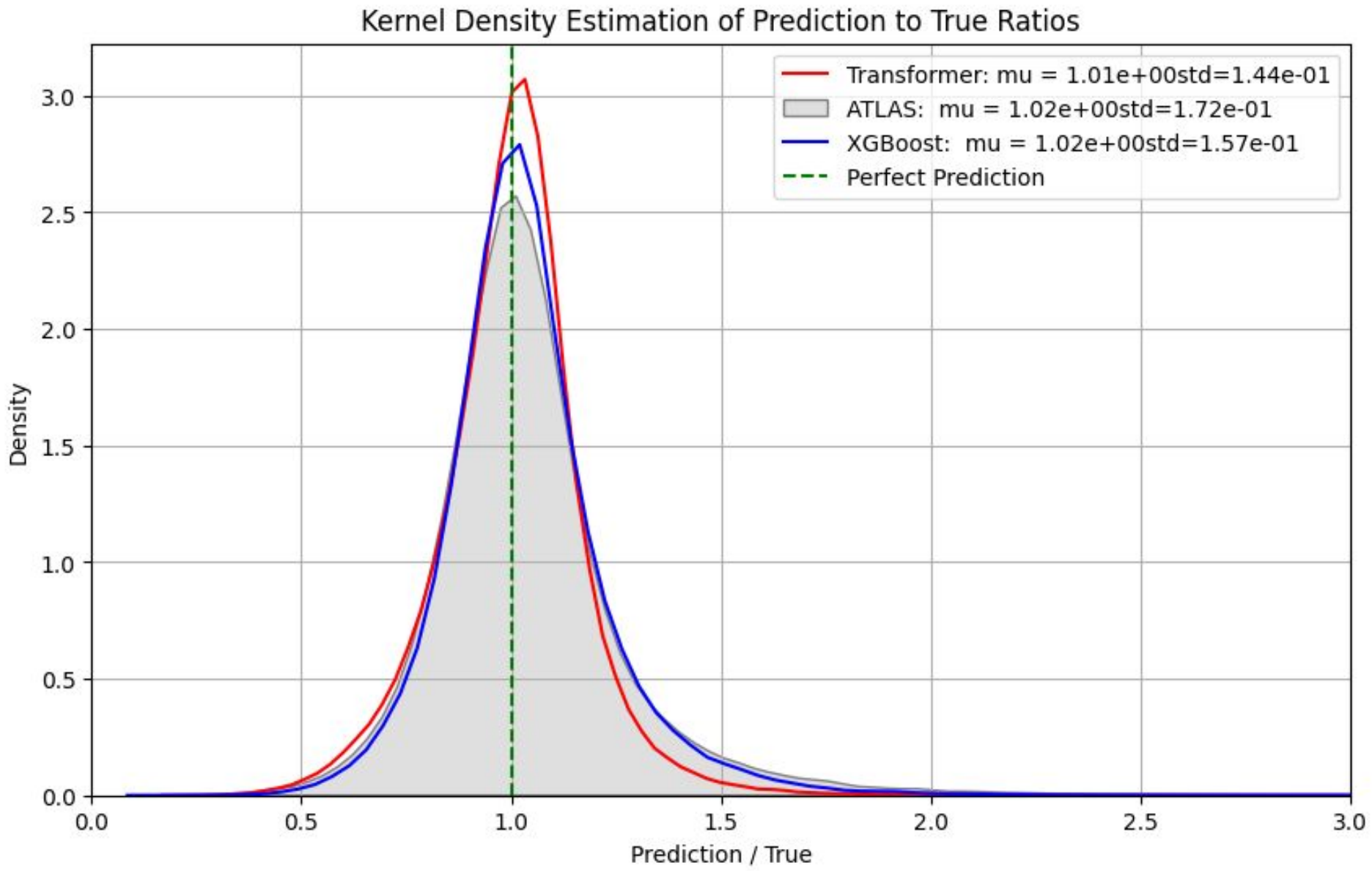
Modified transformer

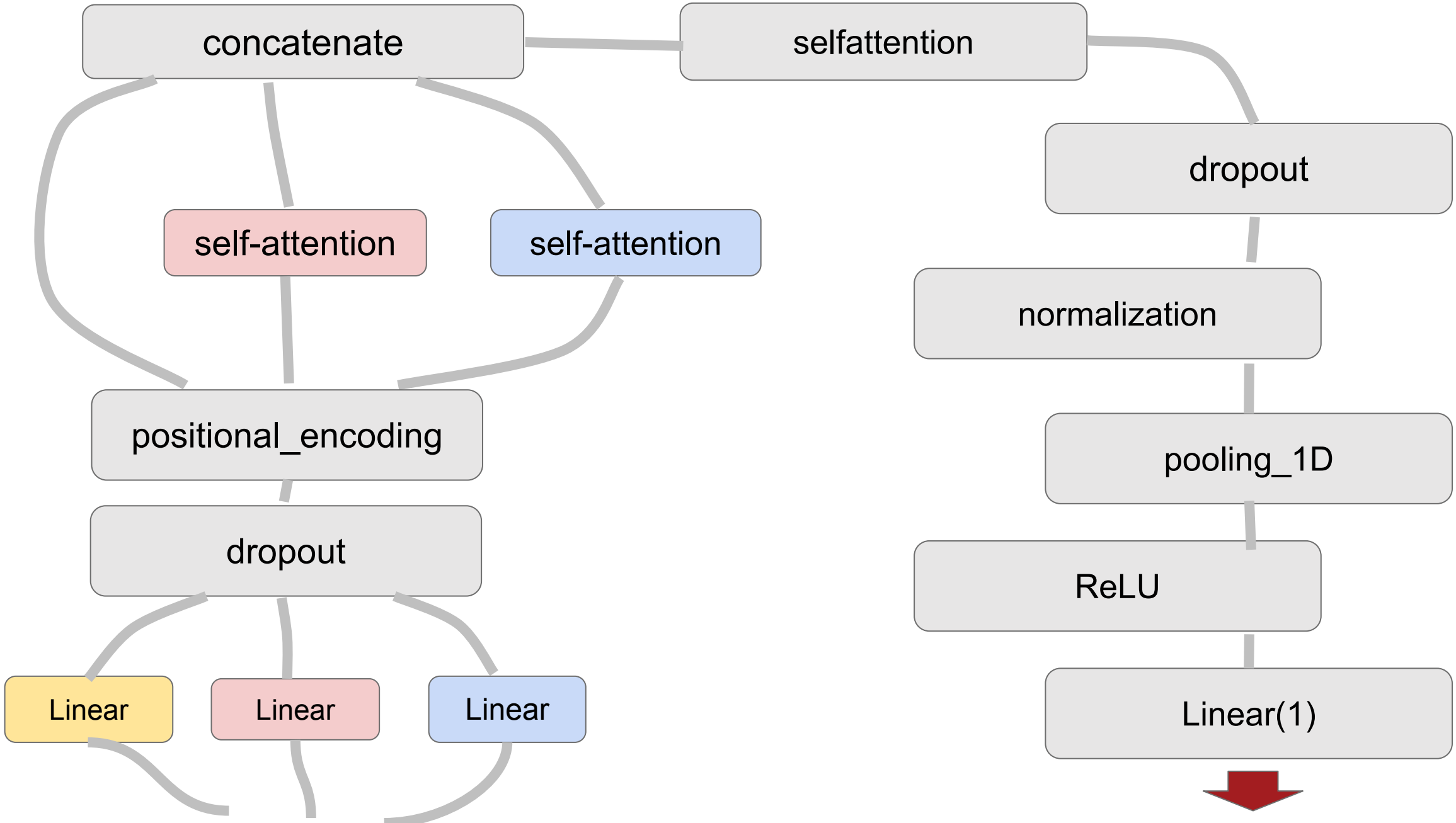


XGBoost
mre:0.140

Transformer
mre:
0.134

Modified
Tranformer
mre:
0.125




concatenate
selfattention
self-attention
self-attention
dropout
normalization
positional_encoding
pooling_1D
dropout
ReLU
Linear
Linear
Linear
Linear(1)

Optimization & Techniques

Experiences

- Layer architecture
- Hyperparameter: Dropout rate, Learning rate, Nheads…
- Loss function: MAE, MRE, Log_response, MSE…

Tensor Processing Unit (TPU)

- Specialization
- High-Efficiency Computing
- Low Power Consumption

MASK

Good for Attention Mechanism

```
[(48.866028 , -1.0349238 , -0.8517365 , 77.45633  , 139.57,  1., 0., 8208., 7.0004043e+03,  12.696693 , -0.07532337,  True),
 (39.858517 , -1.0368809 , -0.85644585, 63.27485  , 139.57, -1., 0., 8208., 1.0329485e+04,  12.742829 ,  0.04961777,  True),
 (31.882801 , -1.0423672 , -0.84905154, 50.830006 , 139.57, -1., 0., 8208., 1.1534389e+04,  12.609667 , -0.2534103 ,  True),
 ...,
 ( 0.       ,  0.        ,  0.        ,  0.       ,   0.   ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
 ( 0.       ,  0.        ,  0.        ,  0.       ,   0.   ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
 ( 0.       ,  0.        ,  0.        ,  0.       ,   0.   ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False)],
[(81.49732  , -0.5086934 , -2.1938472 , 92.27127  , 139.57, -1., 0., 8208., 8.1497317e-16,  20.954634 , -0.04437601,  True),
 (51.169197 , -0.50120354, -2.2005699 , 57.732014 , 139.57,  1., 0., 8208., 4.2174072e+03,  20.835516 ,  0.0185816 ,  True),
 (11.600805 , -0.510175  , -2.191241  , 13.144298 , 139.57,  1., 0., 8208., 5.5604448e+03,  21.237577 , -0.15153563,  True),
 ...,
 ( 0.       ,  0.        ,  0.        ,  0.       ,   0.   ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
 ( 0.       ,  0.        ,  0.        ,  0.       ,   0.   ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False),
 ( 0.       ,  0.        ,  0.        ,  0.       ,   0.   ,  0., 0.,    0., 0.0000000e+00,   0.       ,  0.        , False)]],
```

Learning Rate

"But, there is no reason to consider a fixed value for the learning rate!"

[Troels C. Petersen (NBI)]

```
initial_learning_rate = 0.0001
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate,
    decay_steps=10000,
    decay_rate=0.96,
    staircase=True)
```

Early Stopping and CallBacks

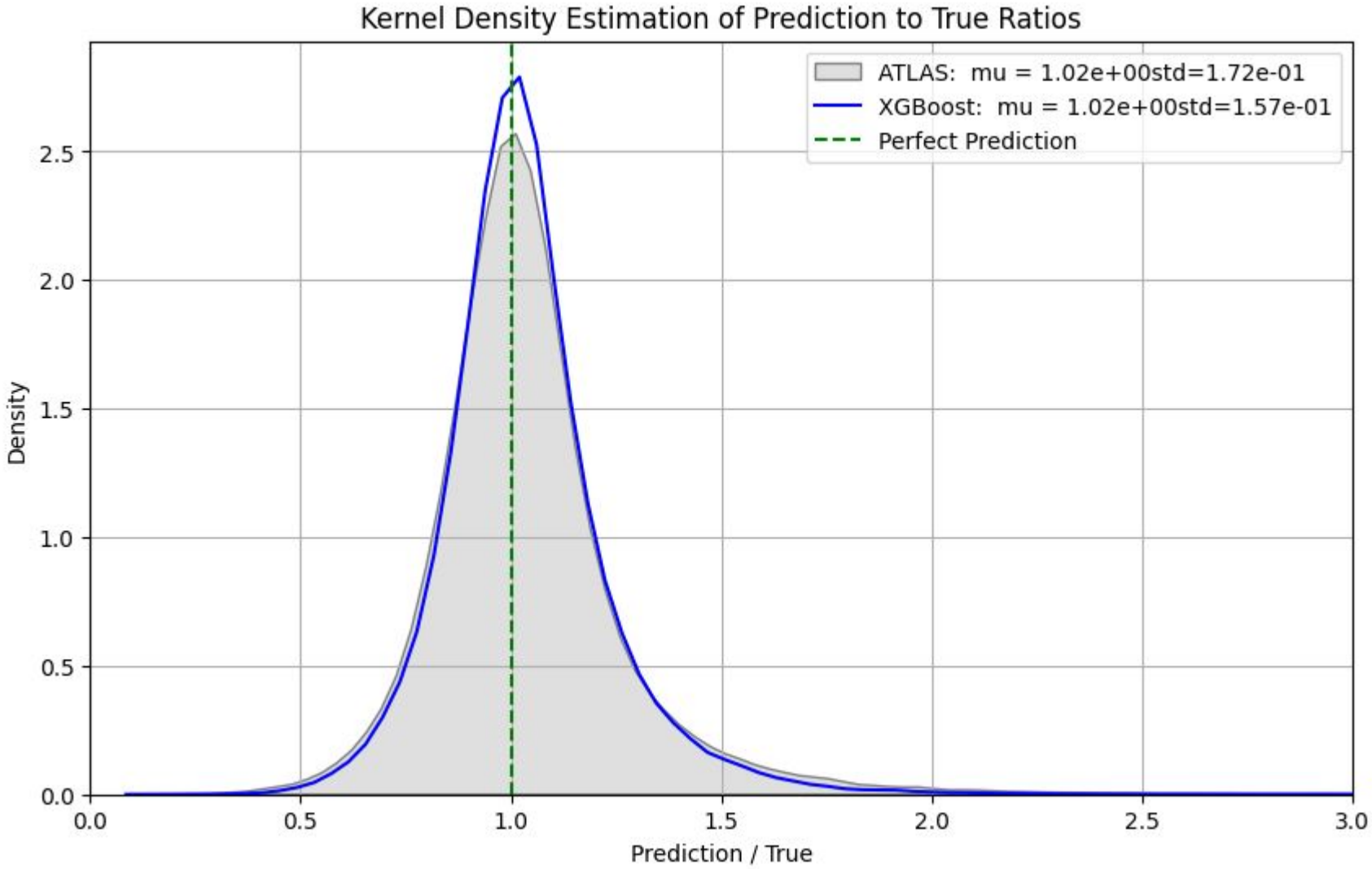
Saves us a lot of time and allows us not to miss the best model.

Results and Comparison

BDT (XGBoost)

ATLAS_mre = 0.1582

XGB_mre = 0.1400

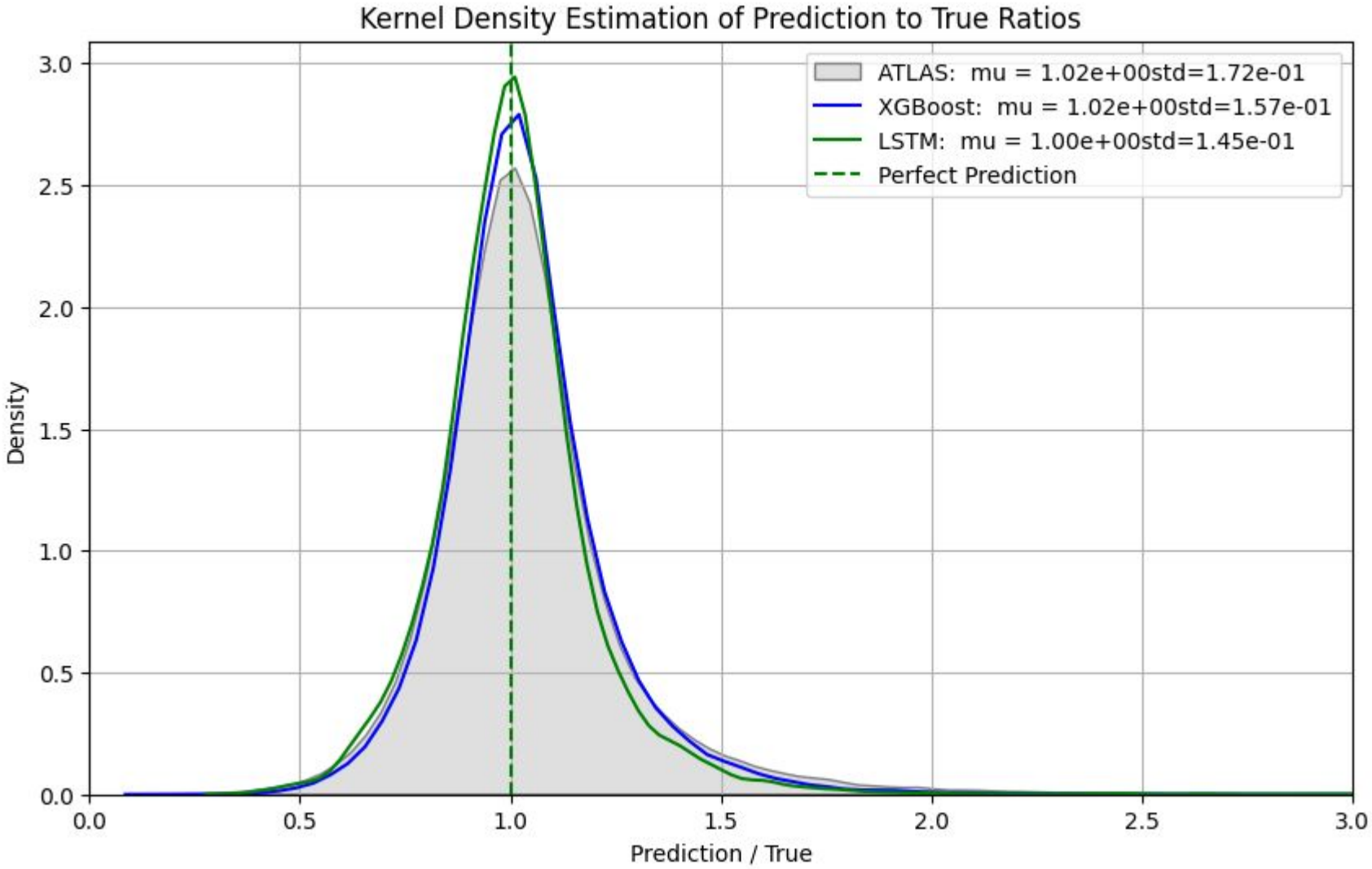


Long Short-Term Memory (LSTM)

ATLAS_mre = 0.1582

XGB_mre = 0.1400

LSTM_mre = 0.1316



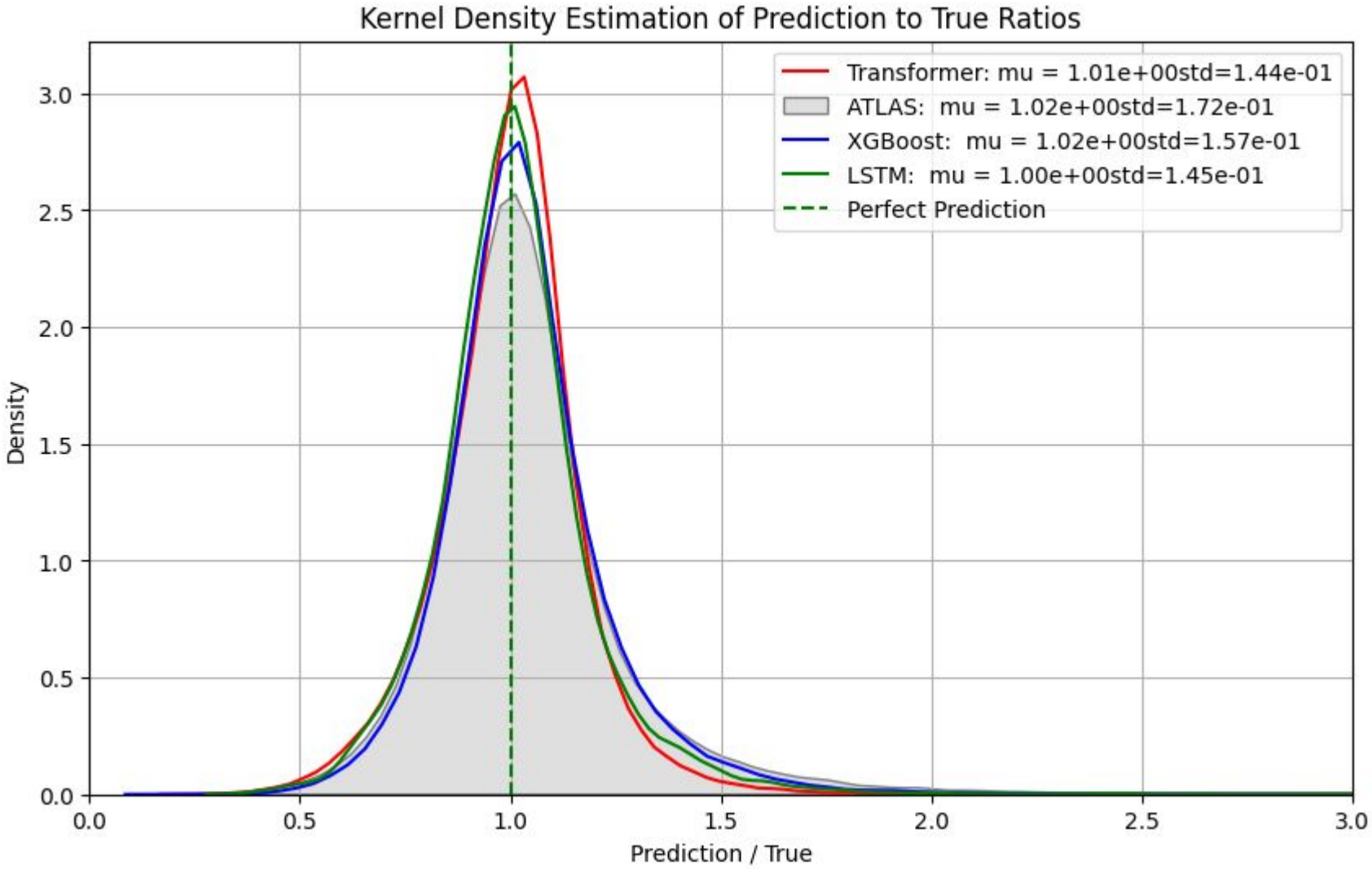
Transformer

ATLAS_mre = 0.1582

XGB_mre = 0.1400

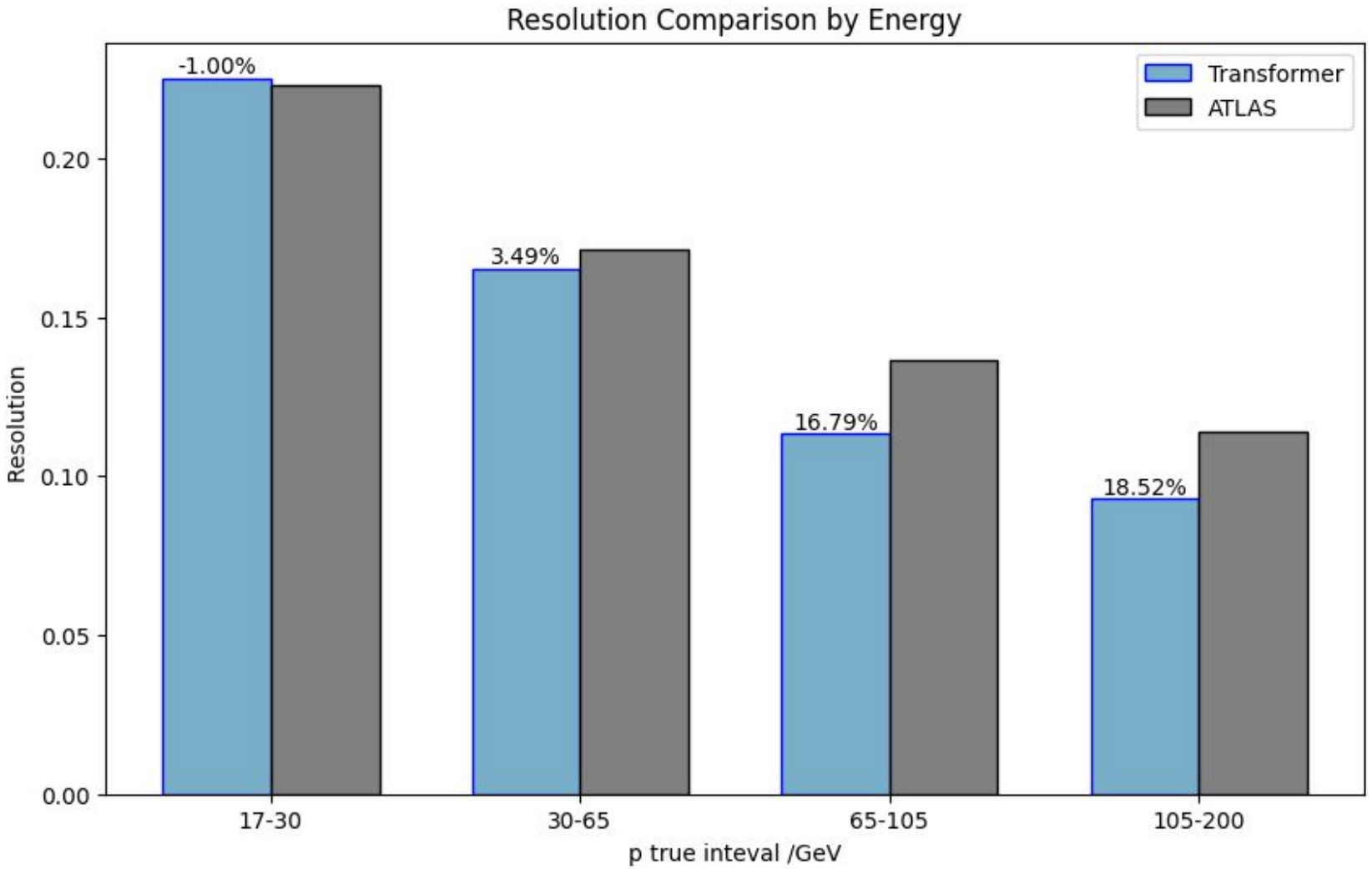
LSTM_mre = 0.1316

Trans_mre = 0.1247

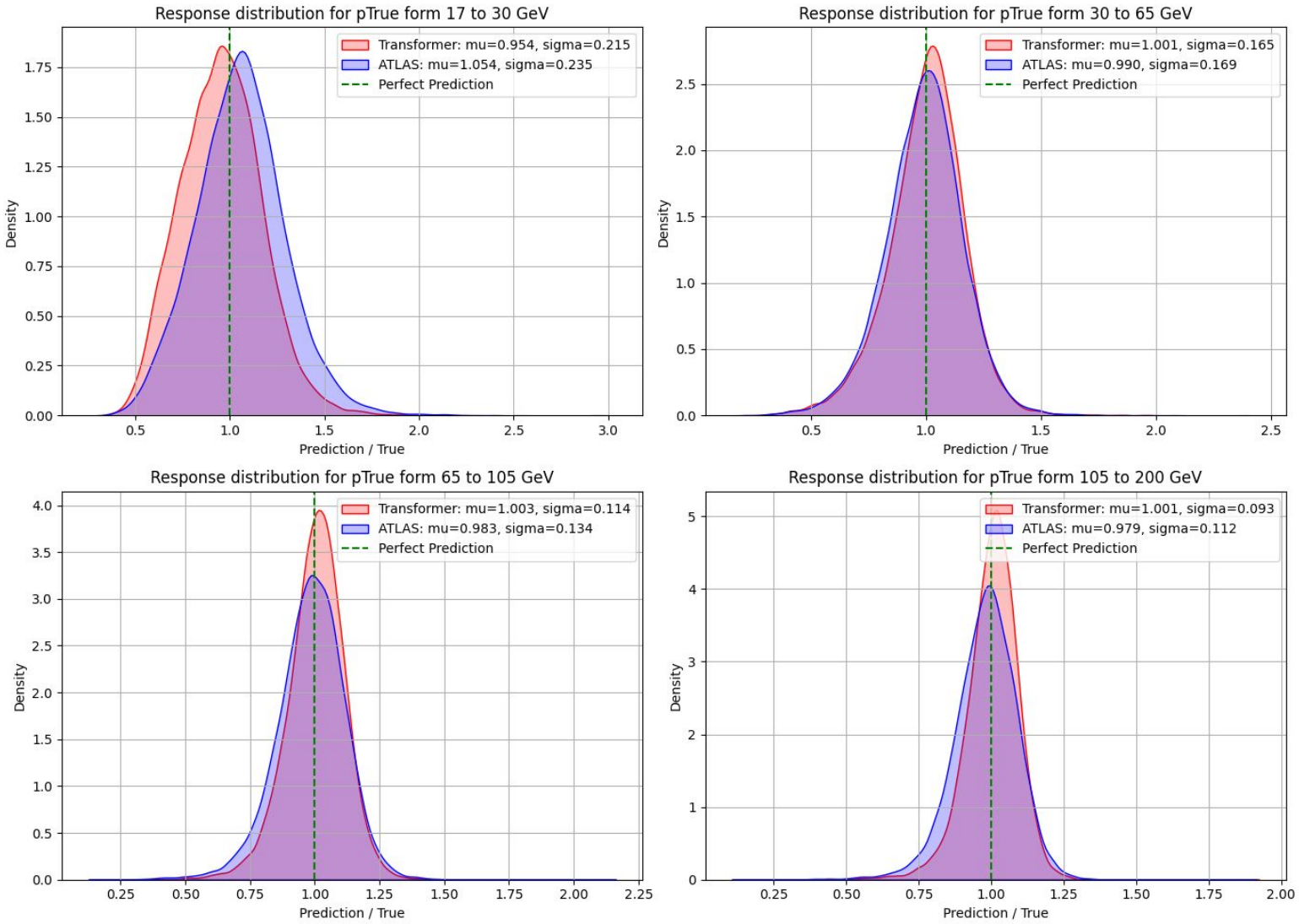


Resolution Comparison (Transformer VS ATLAS)

$$Resolution = \frac{\sigma}{\mu}$$



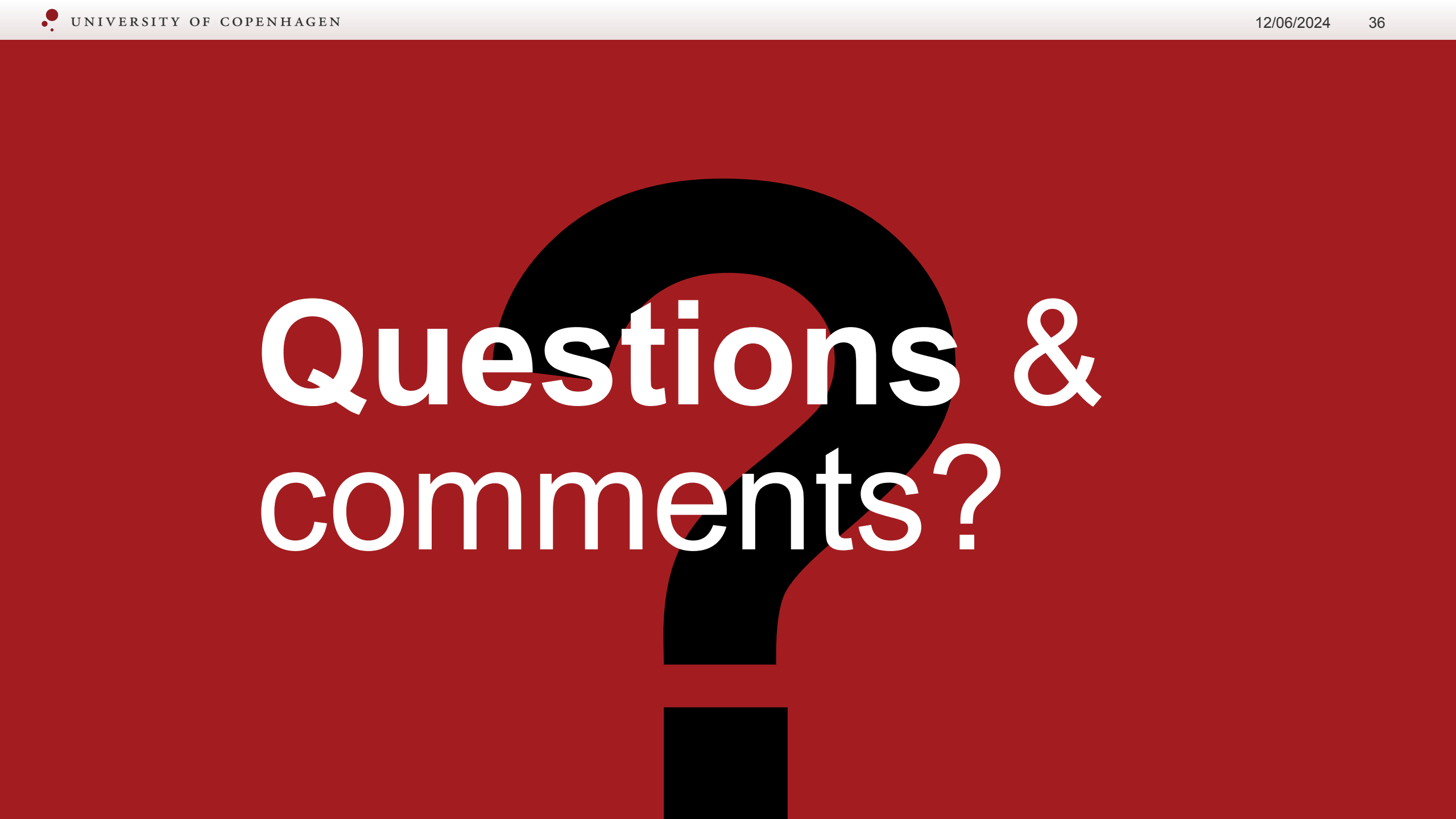
Transformer Result in Different Momentum Intervals



THANKS!

UNIVERSITY OF COPENHAGEN

Questions & comments?



Dropout Layer

e.g.

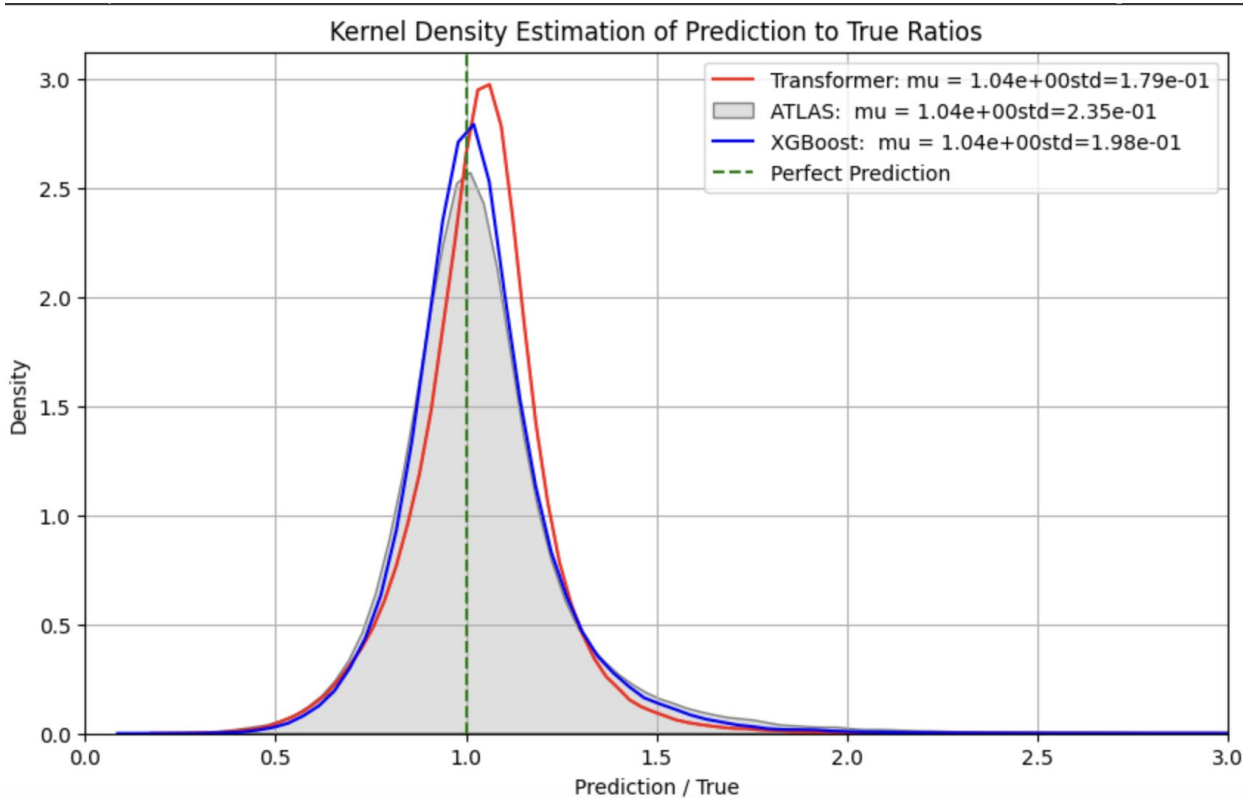
Dropout(0.1)

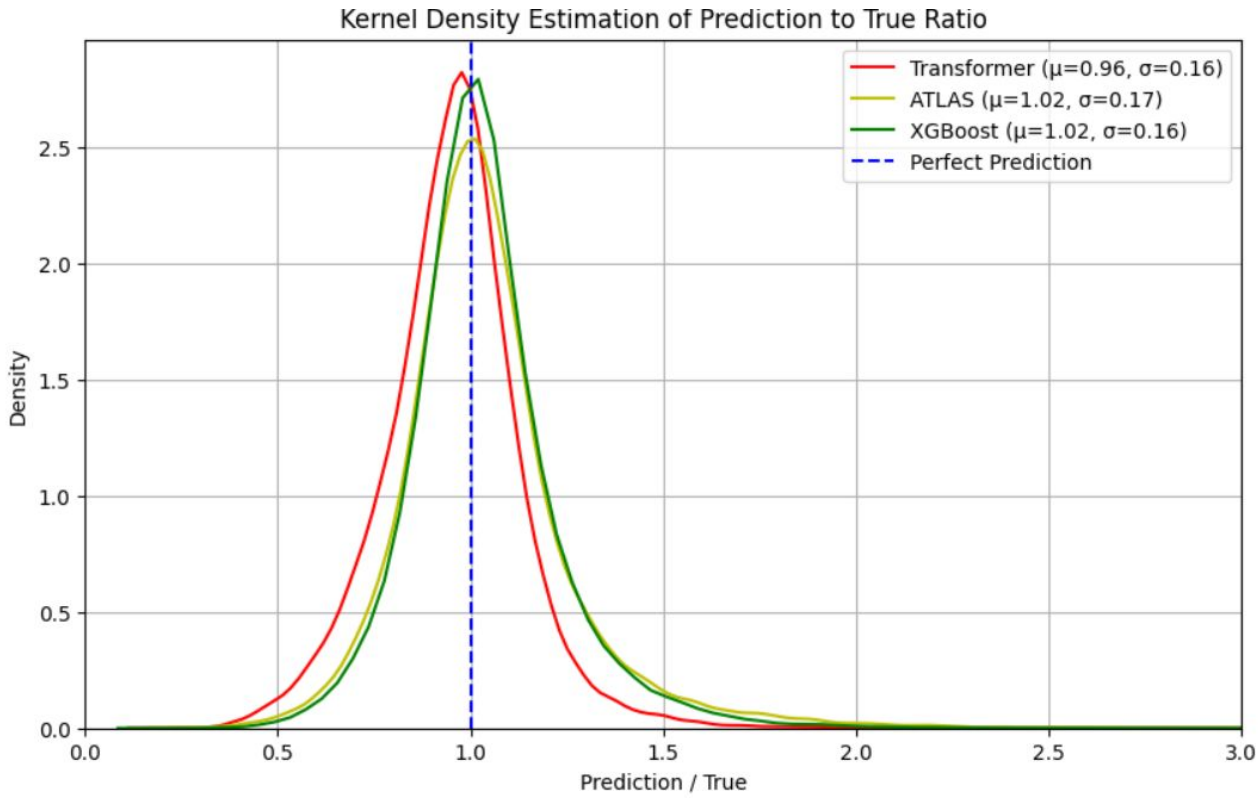
- It means that each neuron in this layer has a *10%* chance of being dropped out at each training step. This operation only occurs during training.

- And during testing or prediction, the *Dropout layer* adjusts its outputs based on the drop rate to ensure the total output remains consistent.

Bias

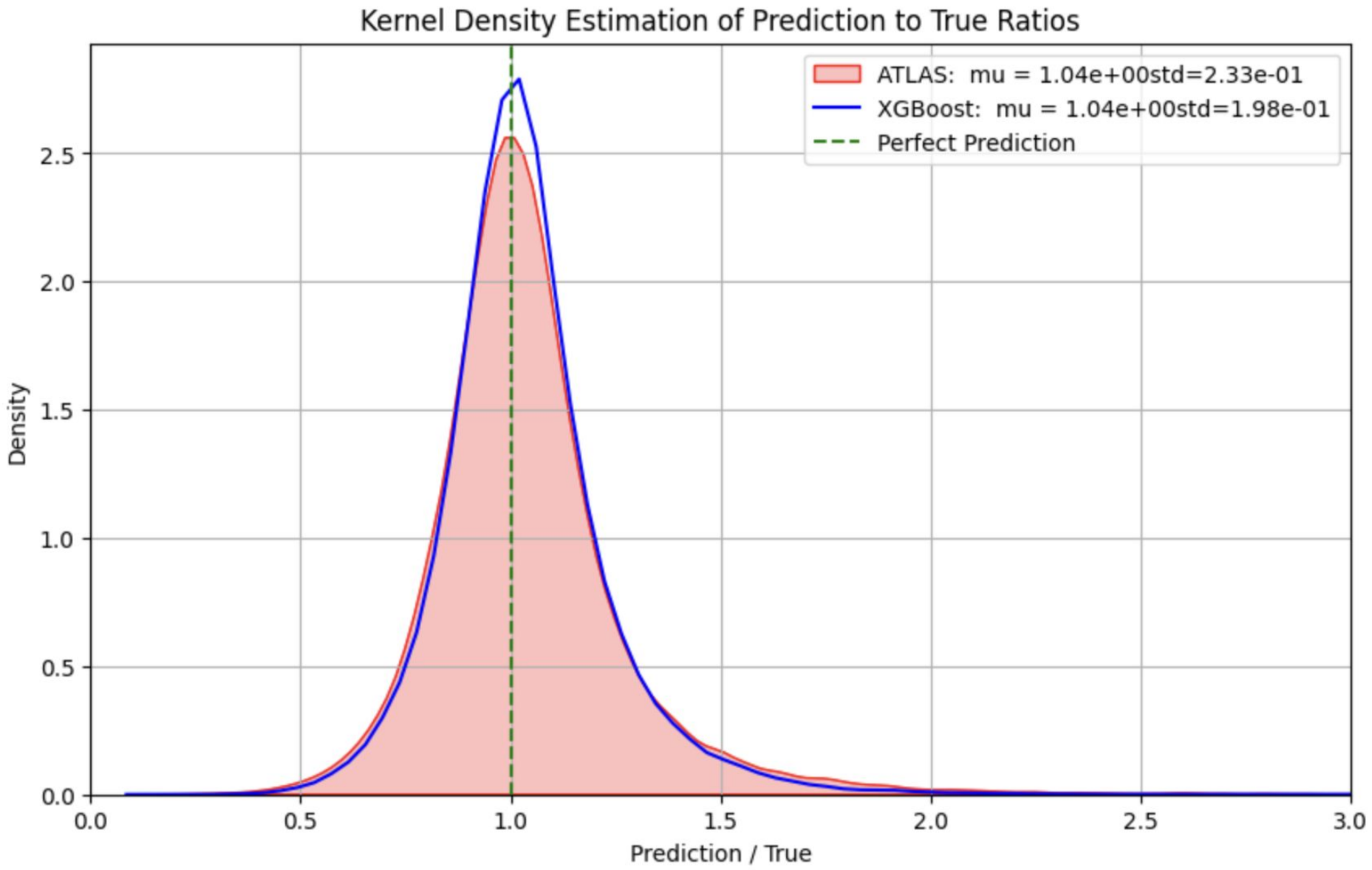
It could be related to loss function, layer structure, data volume and embedding dimension……





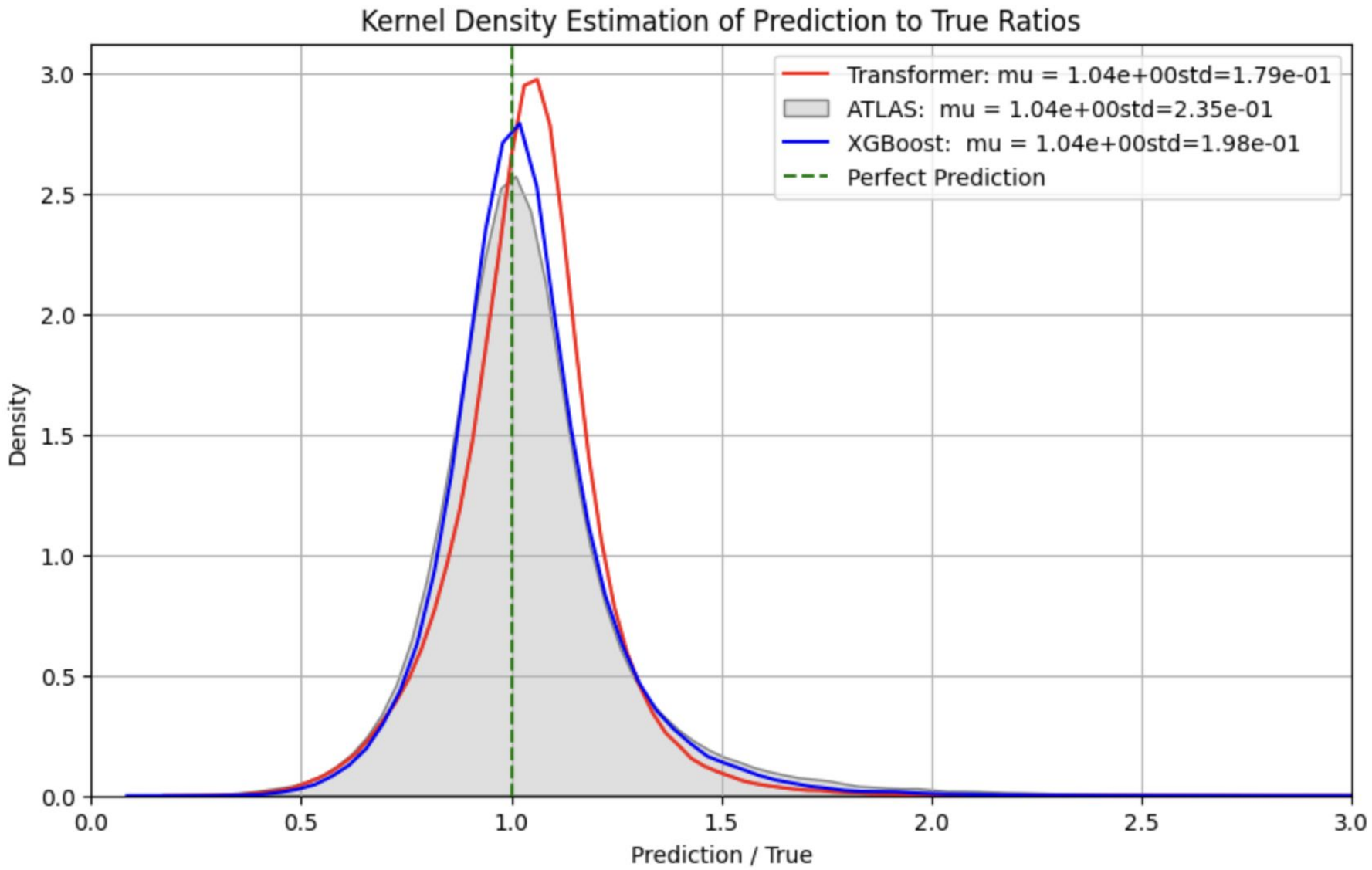
XGBoost

mre:0.140
Runningtime:
21.207 s

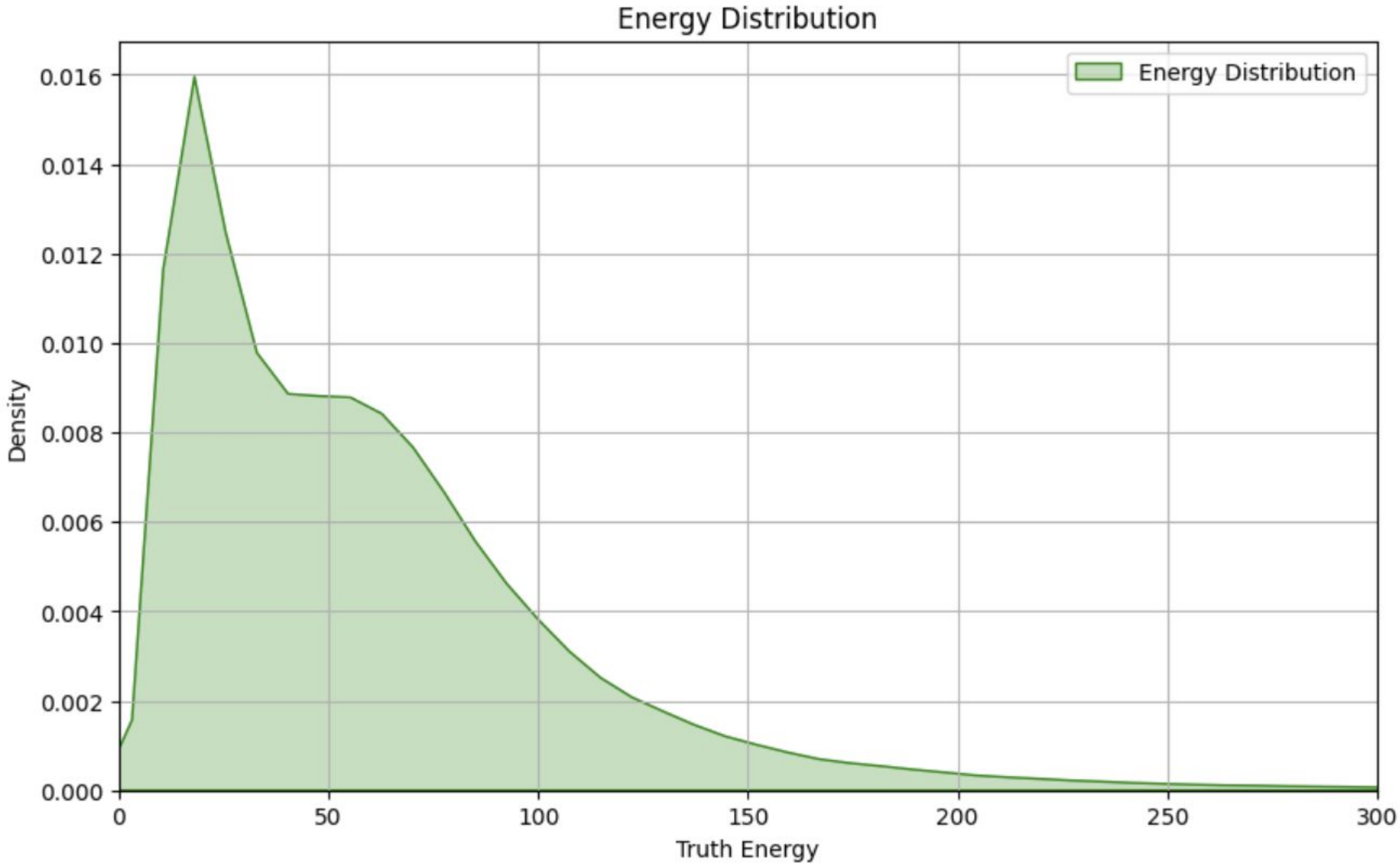


Background

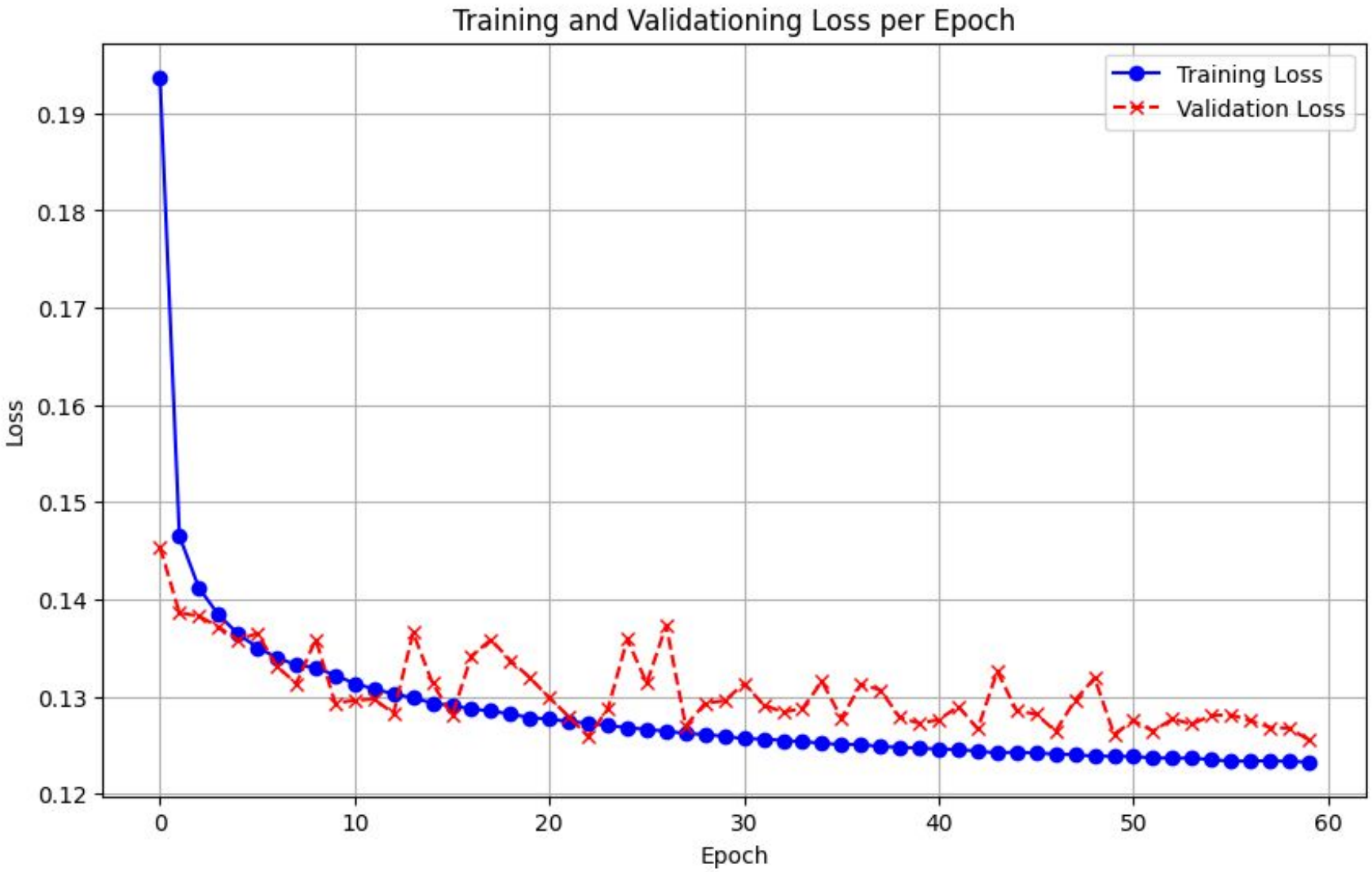
Embedding dim= 256



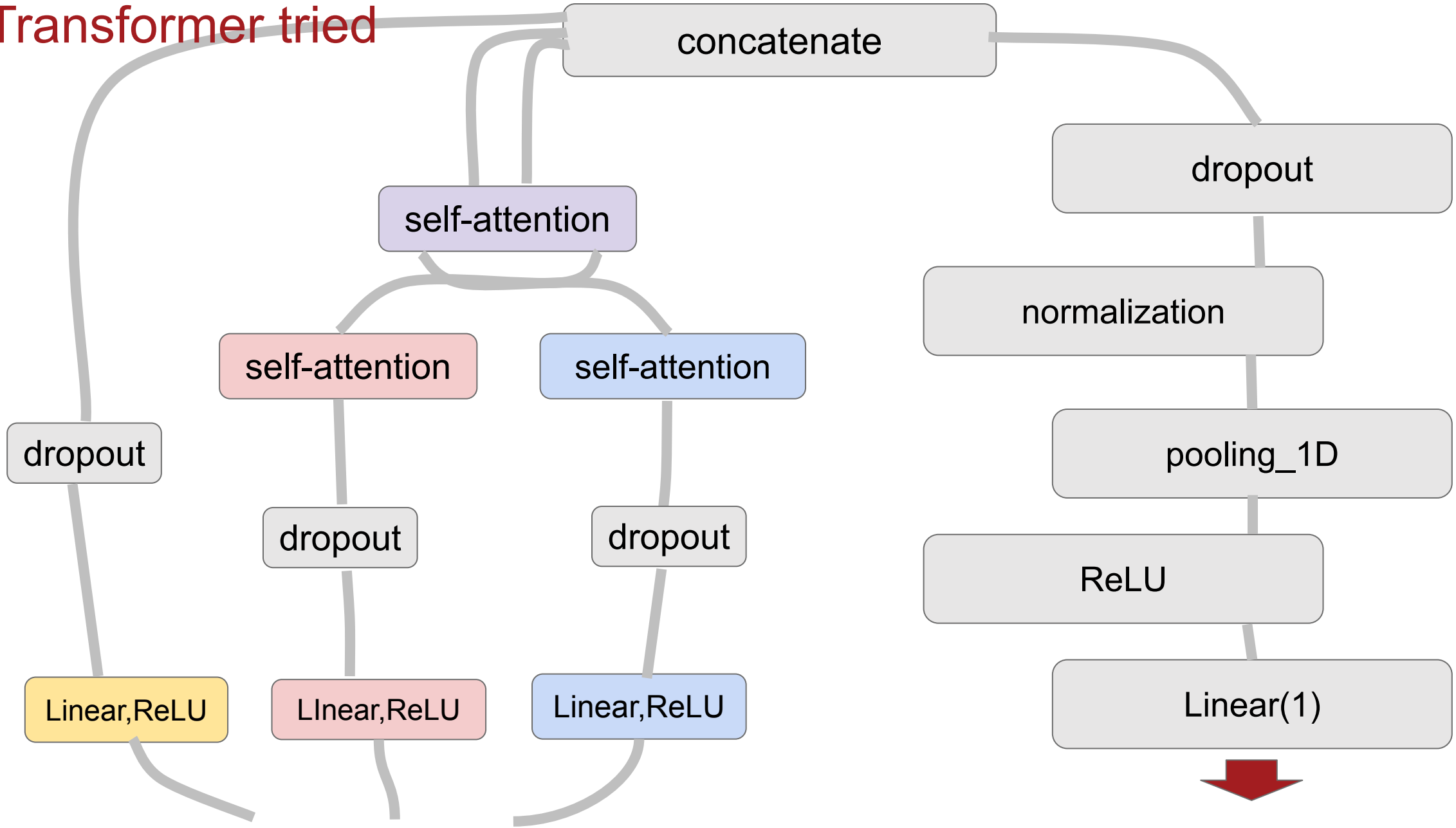
Data Distribution



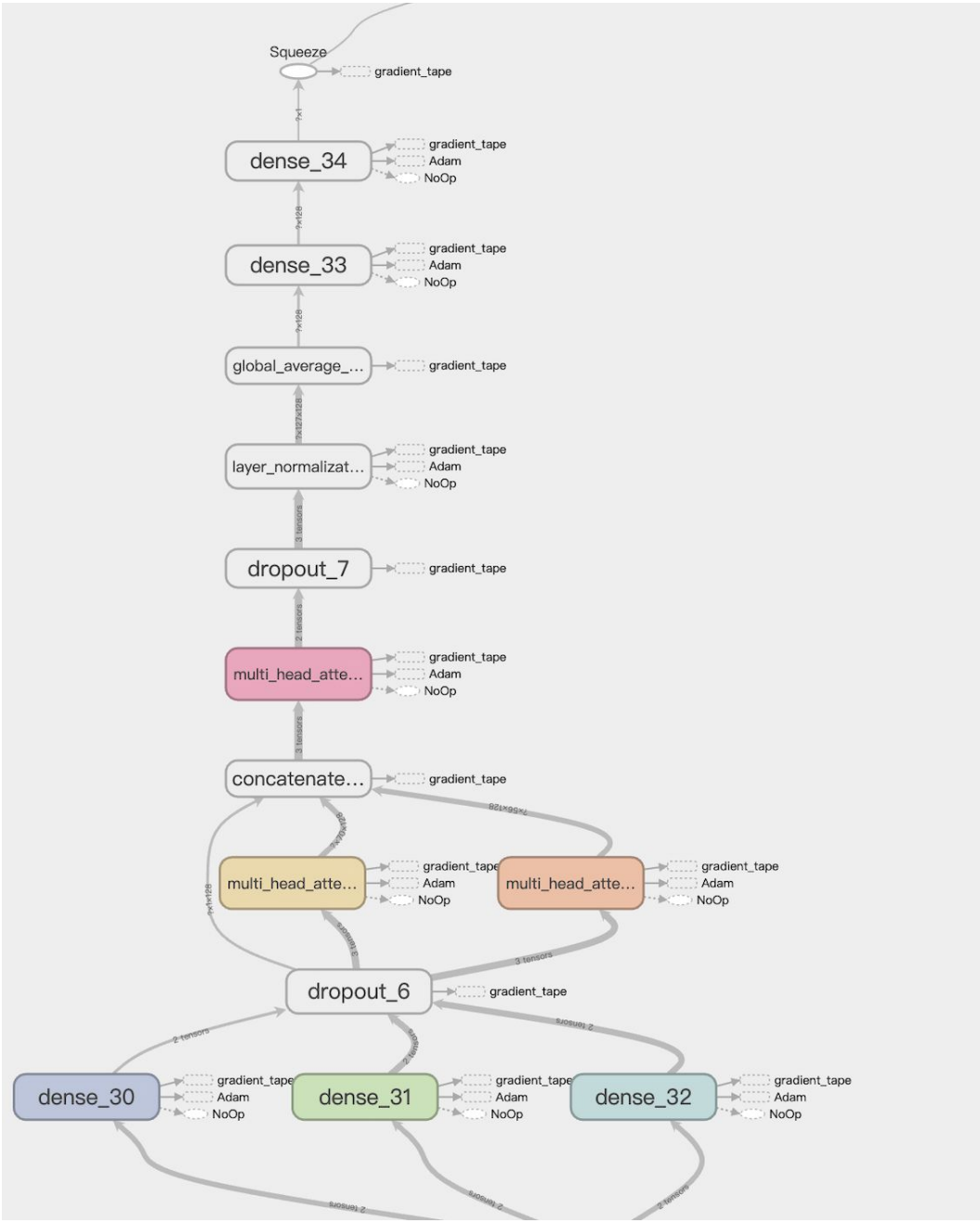
Modified Transformer Loss vs. Epoch



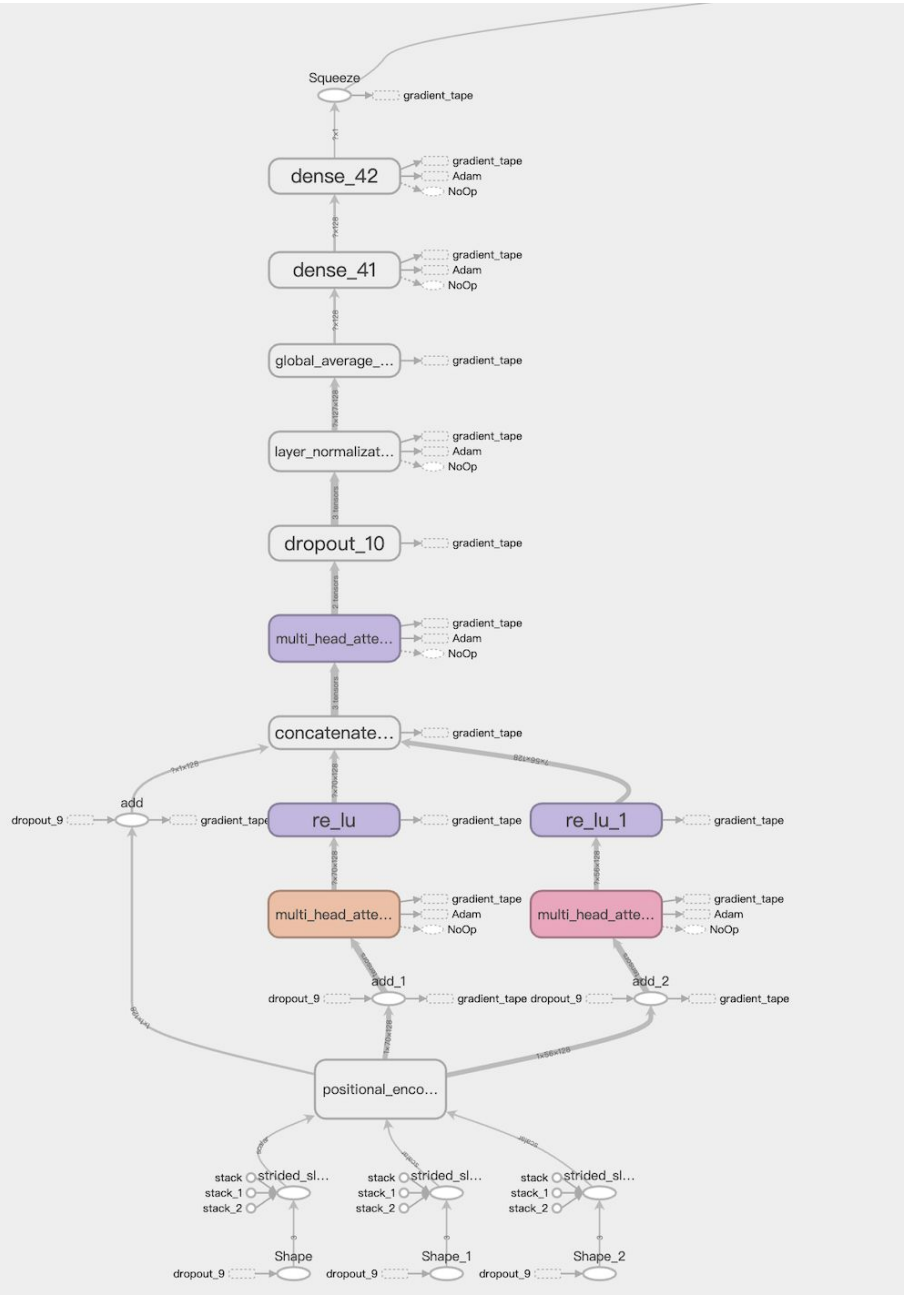
Transformer tried



Transformer tried



Transformer tried



All participants contributed evenly