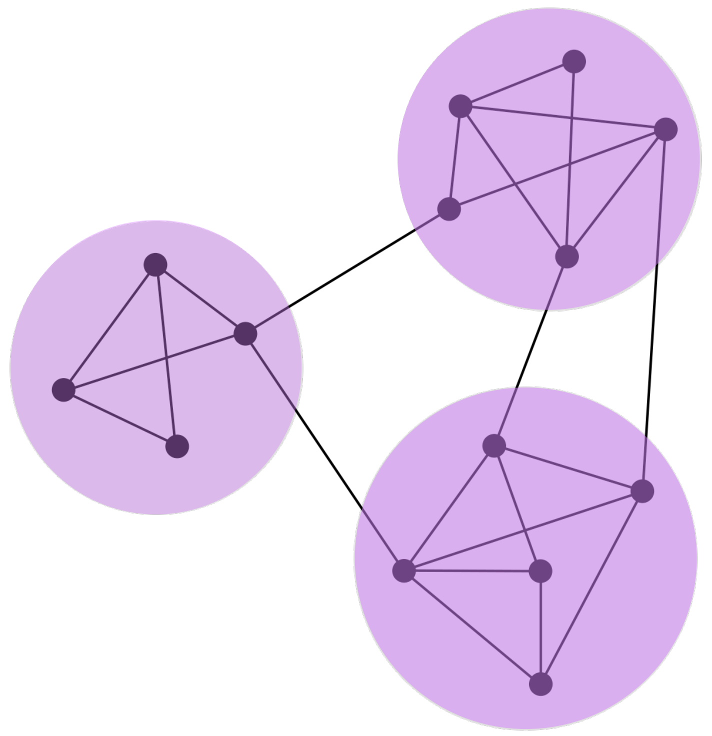
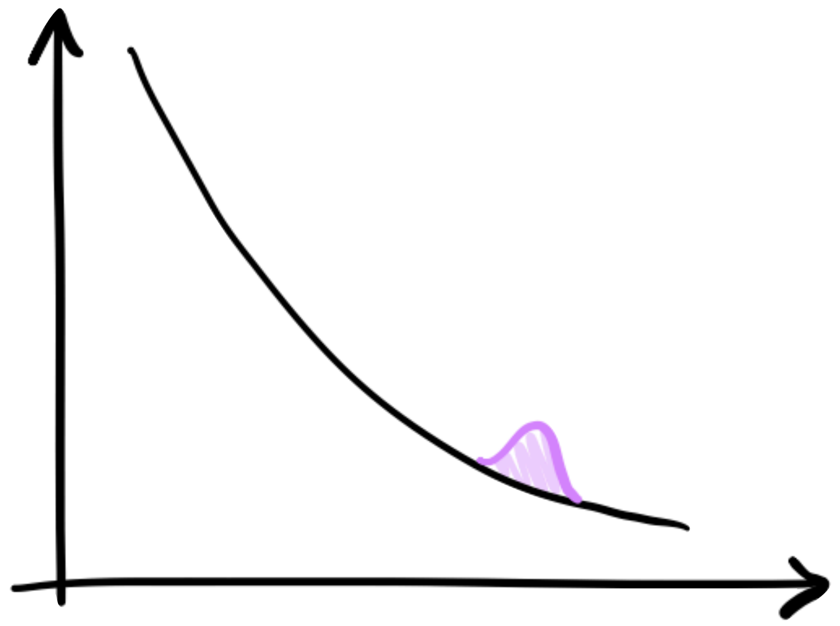
ADJJ - ATLAS mc training



Graziella Russo



ADJJ meeting
09/12/24

Dataset preparation

- Deep understanding of the transformation on the jet and on the constituents
- Implementation of all the functions for the dataset construction: two parameters to be checked for training, `do_ptfrac` and `apply_transform`

```python
    @staticmethod
    def rescale_pt_mass(data, max_constituents, jet_pt, jet_eta, jet_phi, jet_m):
        jet = LorentzVector()
        jet.setptetaphim(jet_pt, jet_eta, jet_phi, jet_m)
        ptrescale = 1.0/jet_pt
        mrescale = 1.0/np.sqrt(jet.e**2 - jet.px**2 - jet.py**2 - jet.pz**2)

        const = torch.Tensor([(data[i,0]*ptrescale, data[i,1], data[i,2], data[i,3]*mrescale, data[i,4])
                              for i in range(0, len(data), 1)])

        return const
```

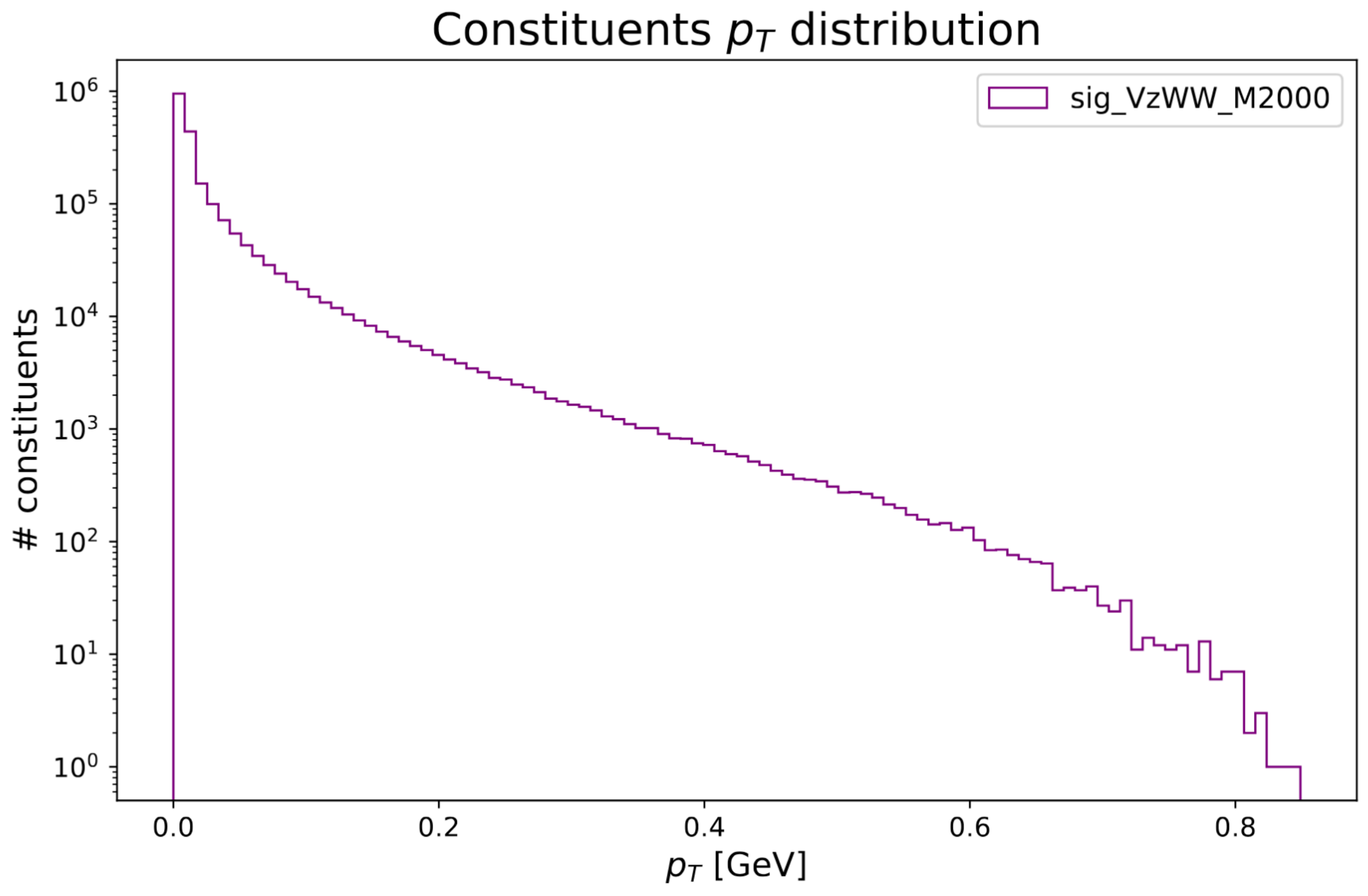
Datasets prepared:

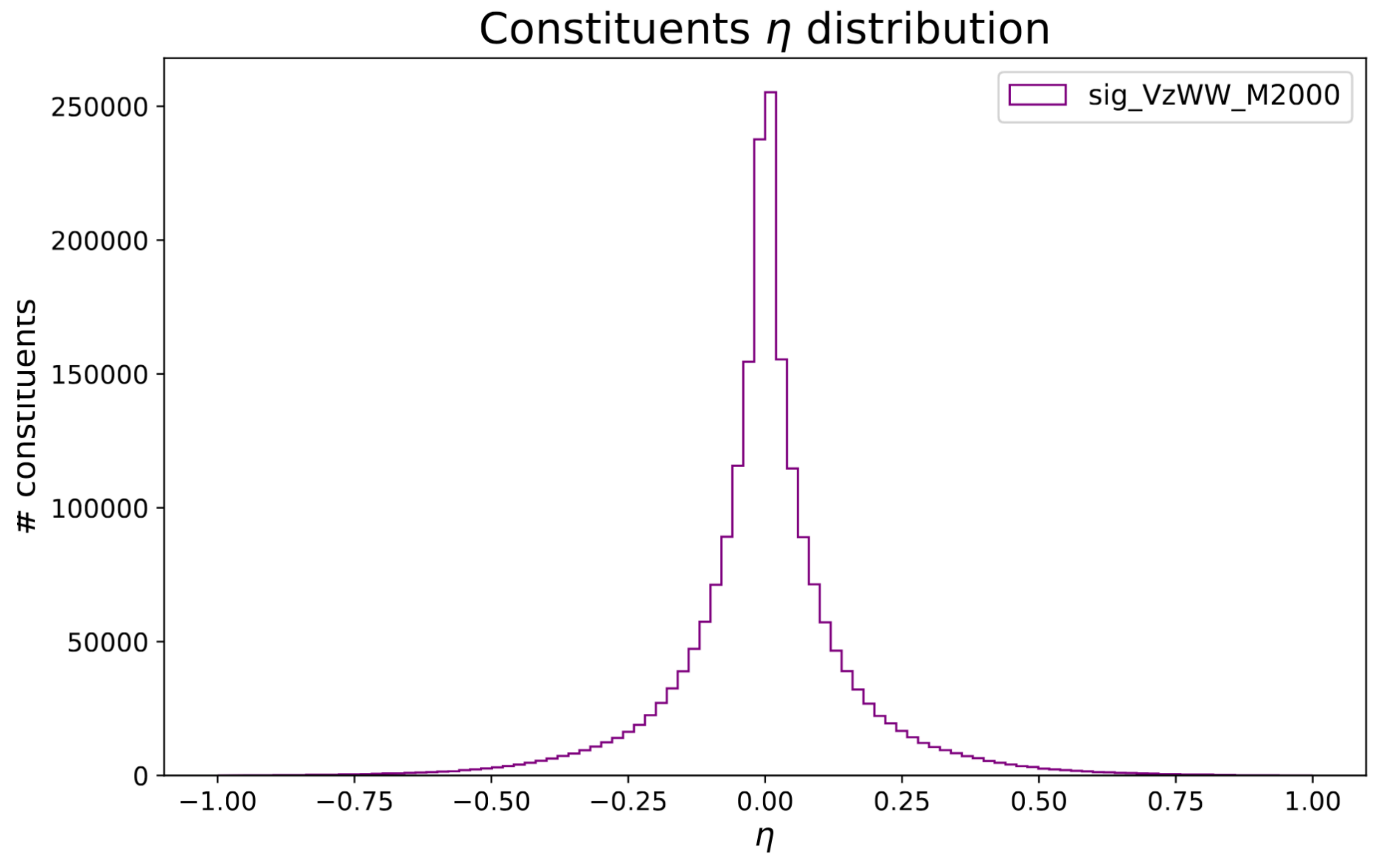
- ✔ QCD dijet slices JZ3-9
- ✔ W + jet
- ✔ Z + jet
- ✔ Z + bb jet
- ✔ ttbar

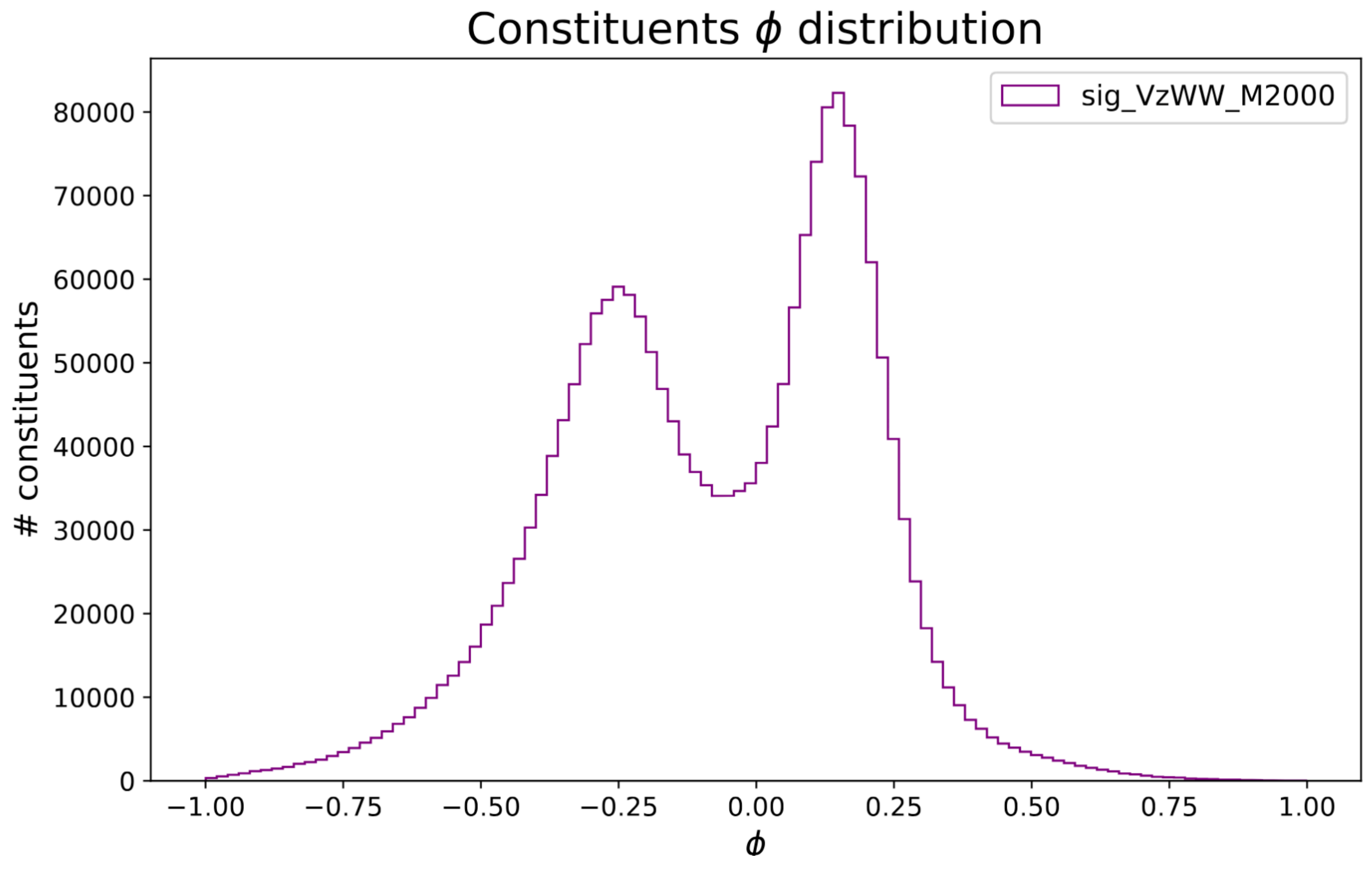
- ✔ dark jet Model 1
- ✔ HVT V+WW (V masses @ 2, 3, 4, 5 TeV)
- ✔ HVT V+XH (V masses @ 2.3, 3.4, 5, 6 TeV)
- ✔ W’ decaying 3prong

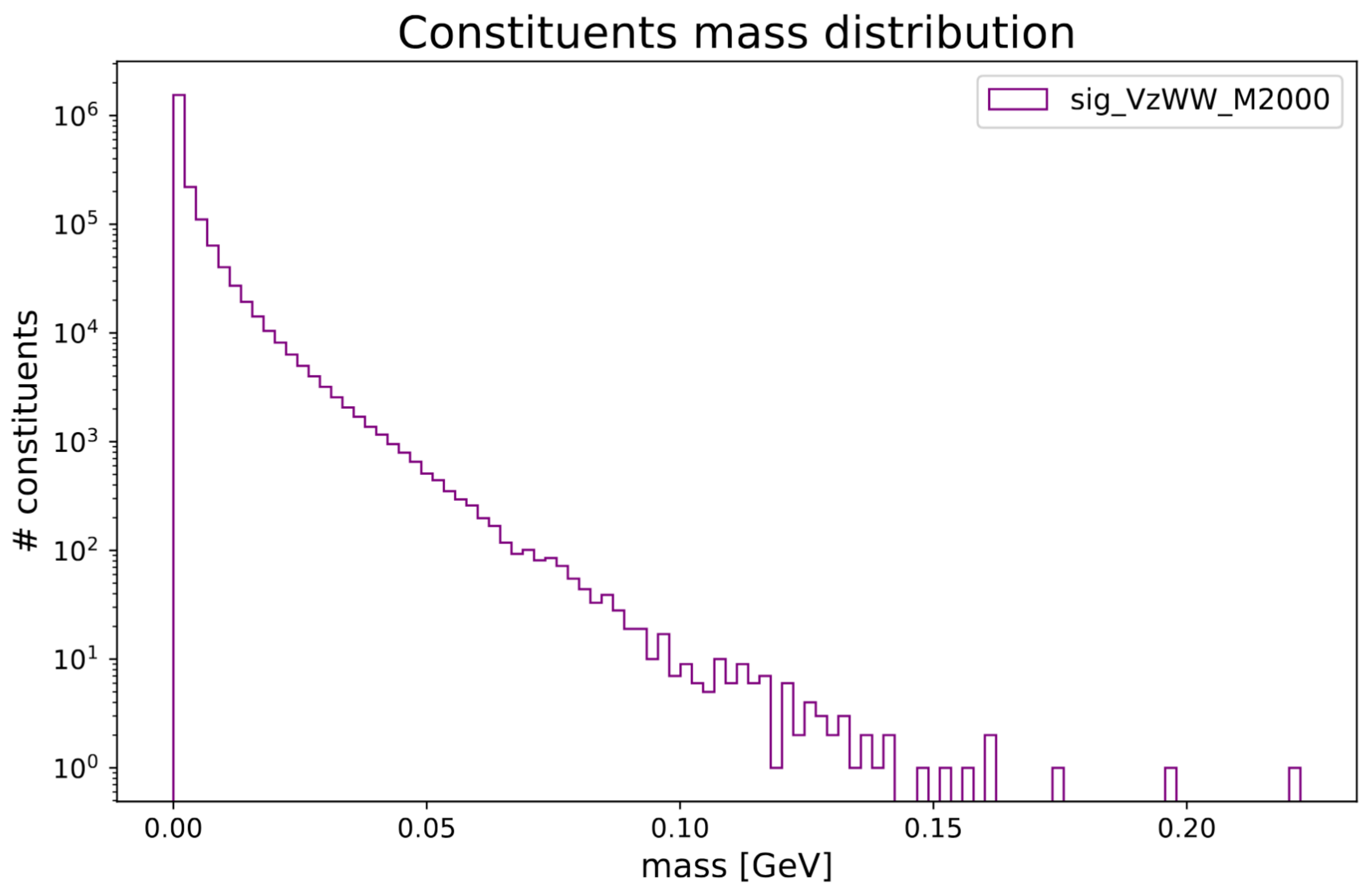
QCD Kinematic distributions

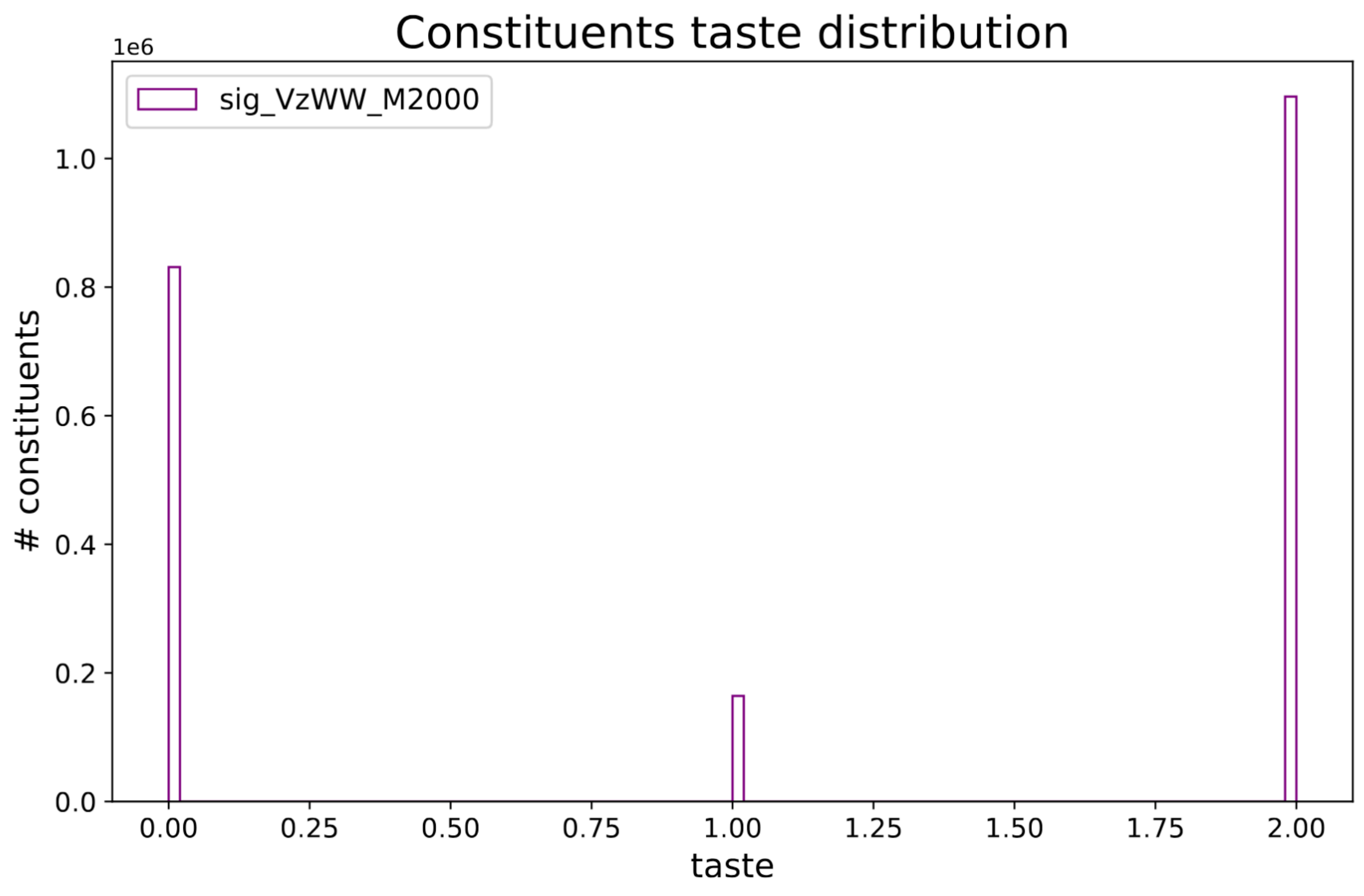
do_ptfrac = False
apply_transformation = True

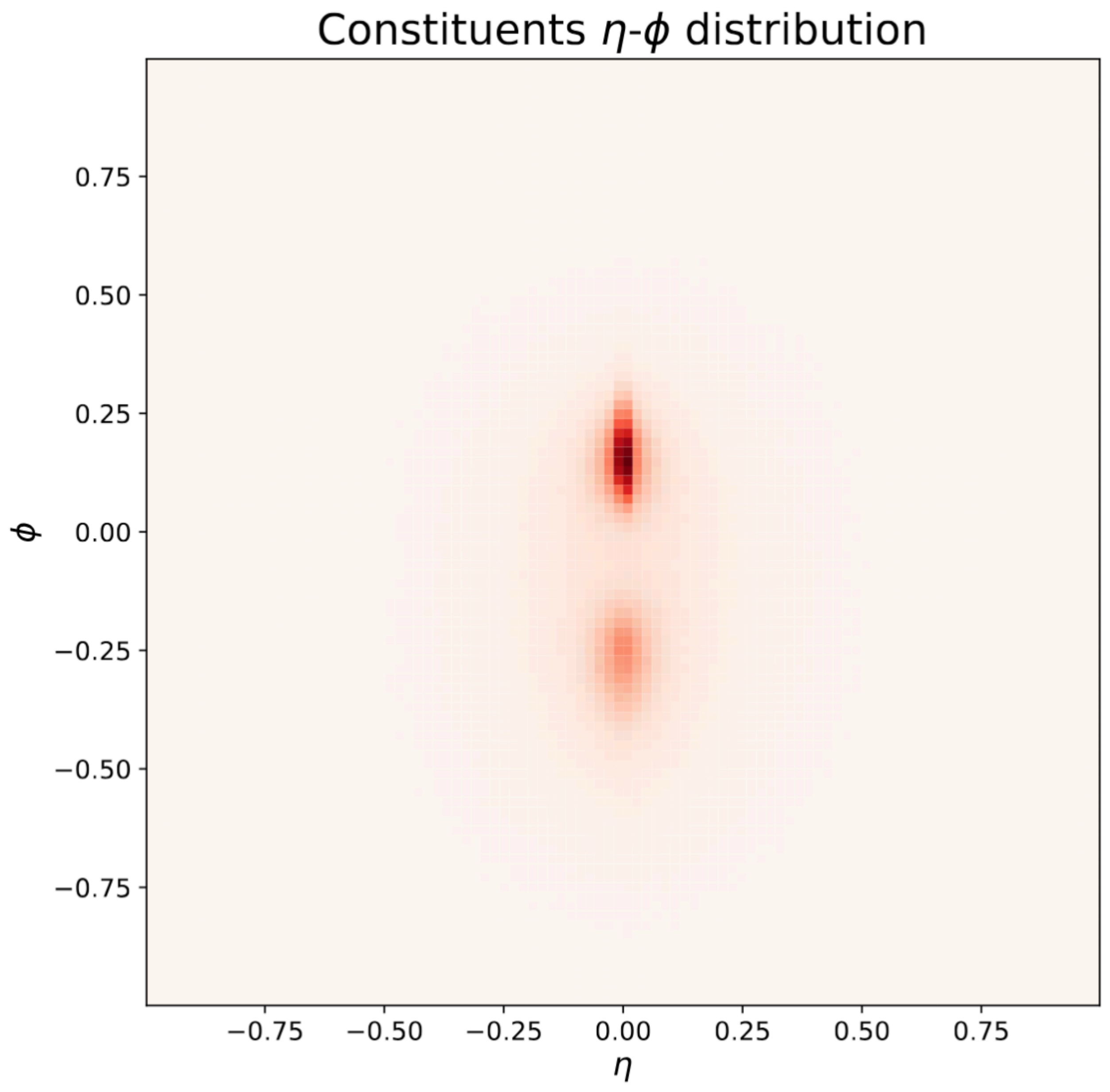












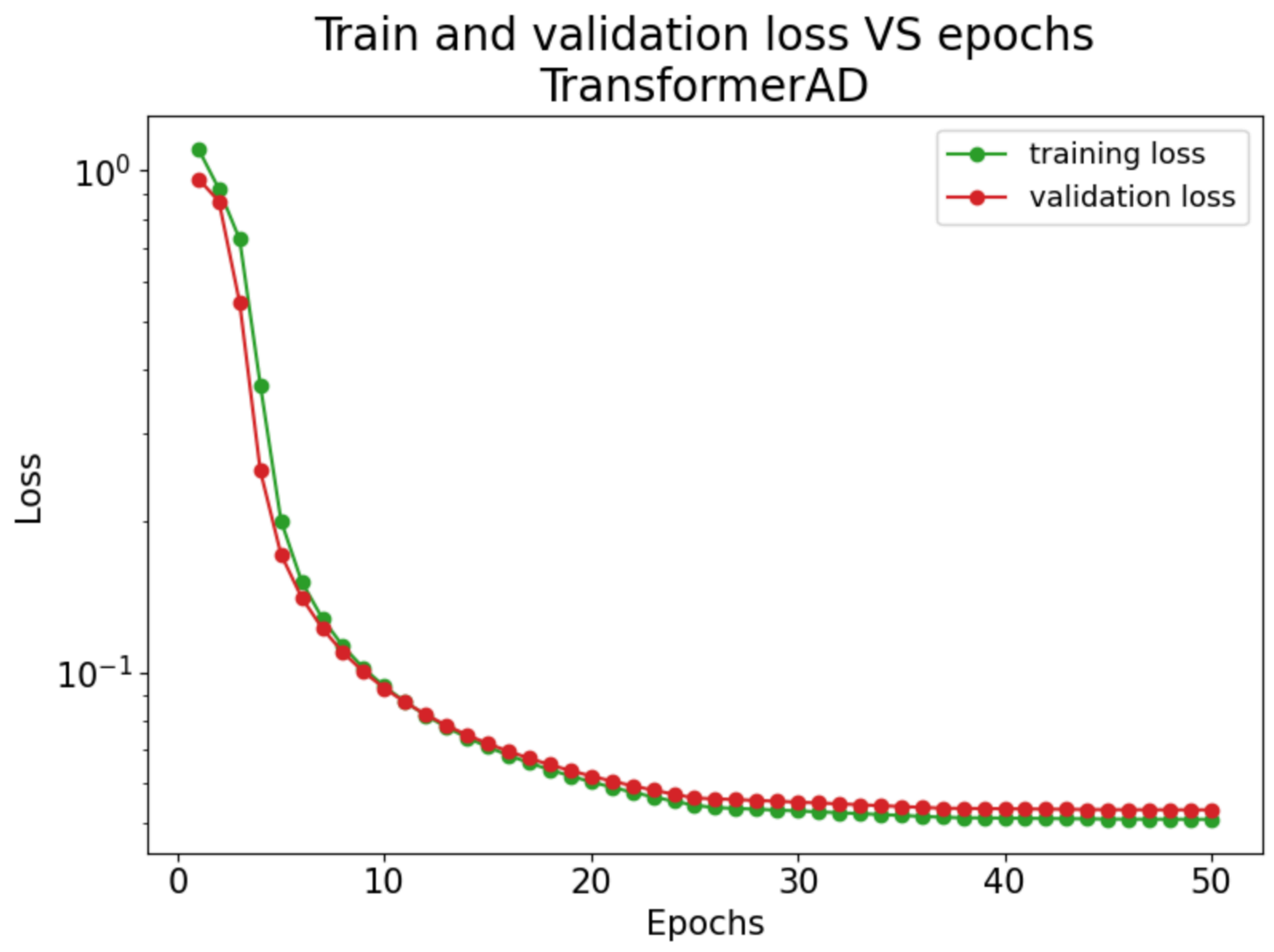
First AD transformer training (I)

- Choose datasets
  - QCD slice JZ3 —> ~350k events
  - QCD slices JZ4-9 —> 2M events
- Dataset features
  - 50 constituents zero-padded, descending p_T ordered
  - p_T, η, ϕ, mass, taste of constituents
  - fully connected graph
- Dataset splitting
  - 80% training set
  - 10% validation set
  - 10% test set

First AD transformer training (II)

- Transformer hyperparameters
  - input and output size: [B, 50, 5]
  - # attention layers: 8
  - # heads: 8

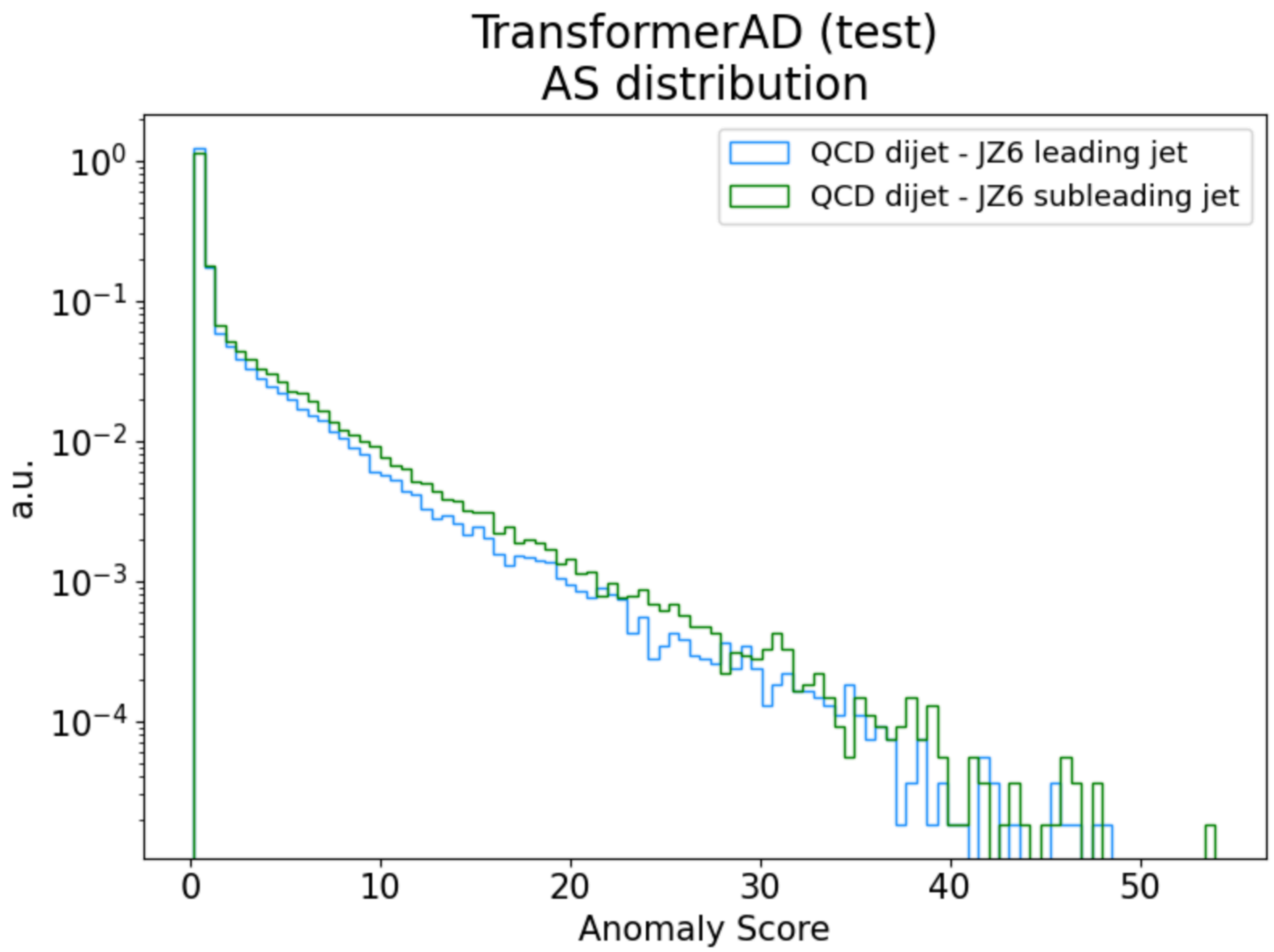
- Training hyperparameters
  - batch size: 1024
  - learning rate: 1e-6 (scheduler with γ=0.2)
  - loss: MSE
  - # epochs: 50
  - dropout: 0.3



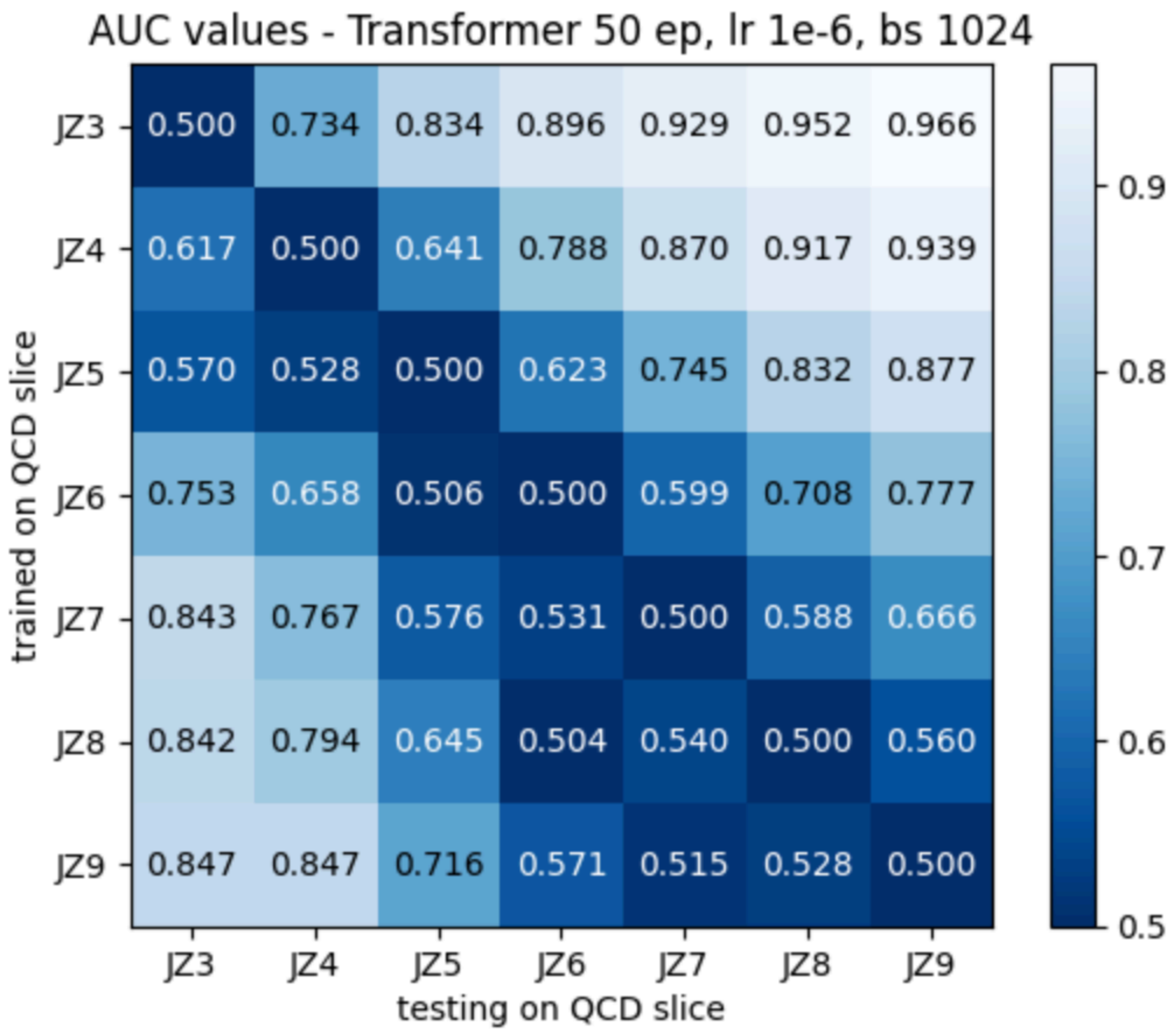
33 s/epoch on JZ3 —> 3 min/epoch on JZ4-9

Results

- Anomaly Score on leading and subleading jets

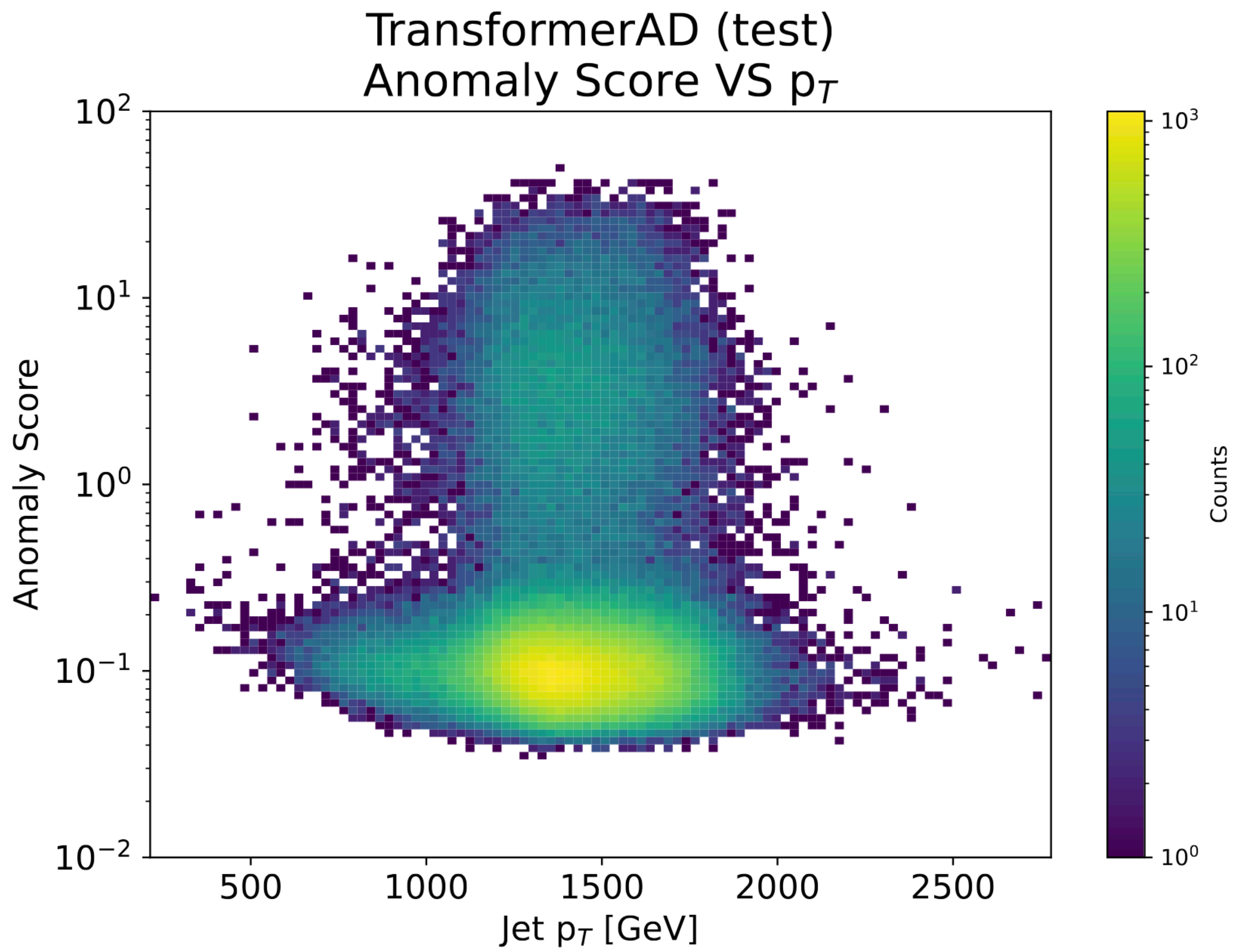


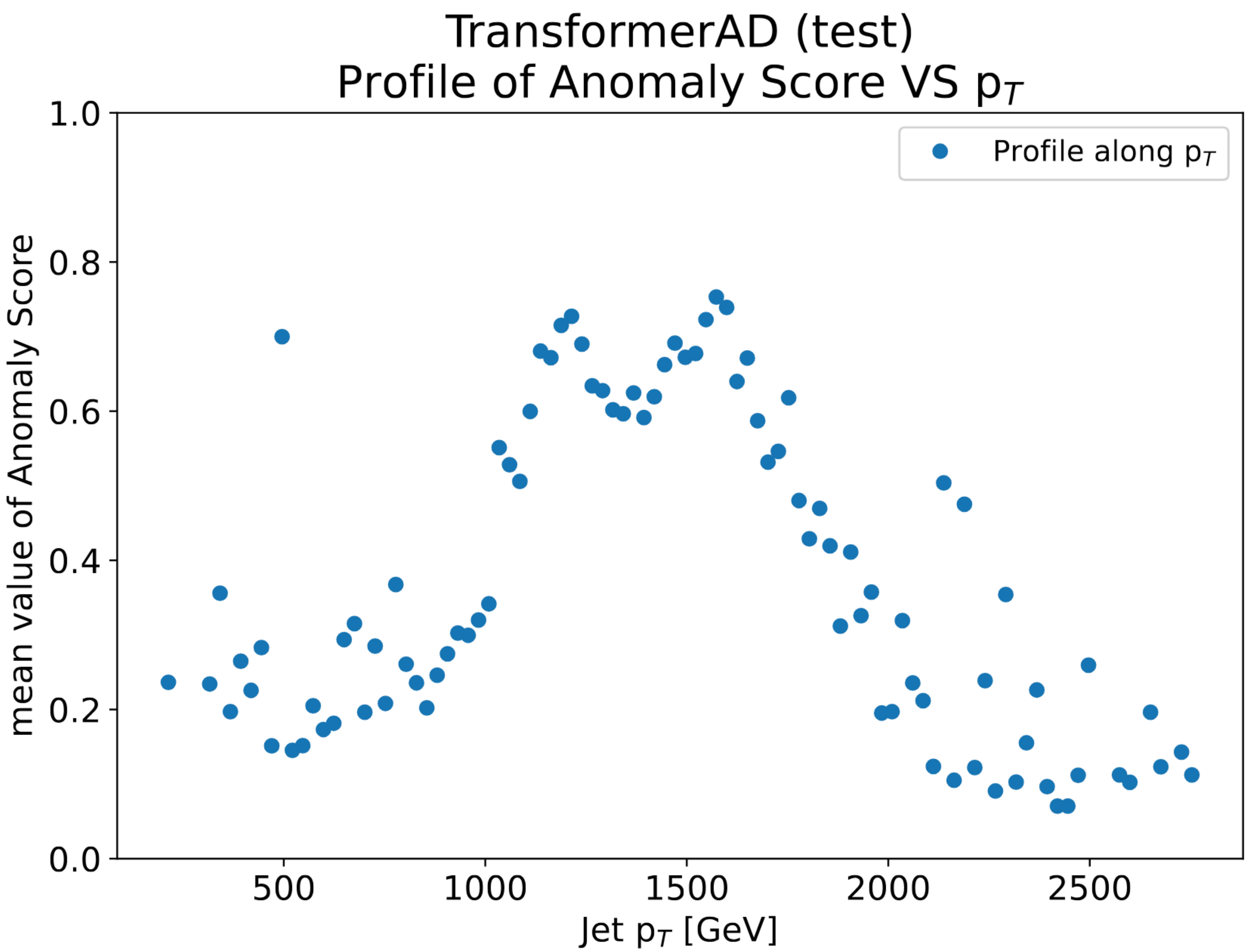
- Testing on the other QCD slices and looking at the AUC



AS dependence on p_T

tested on JZ6

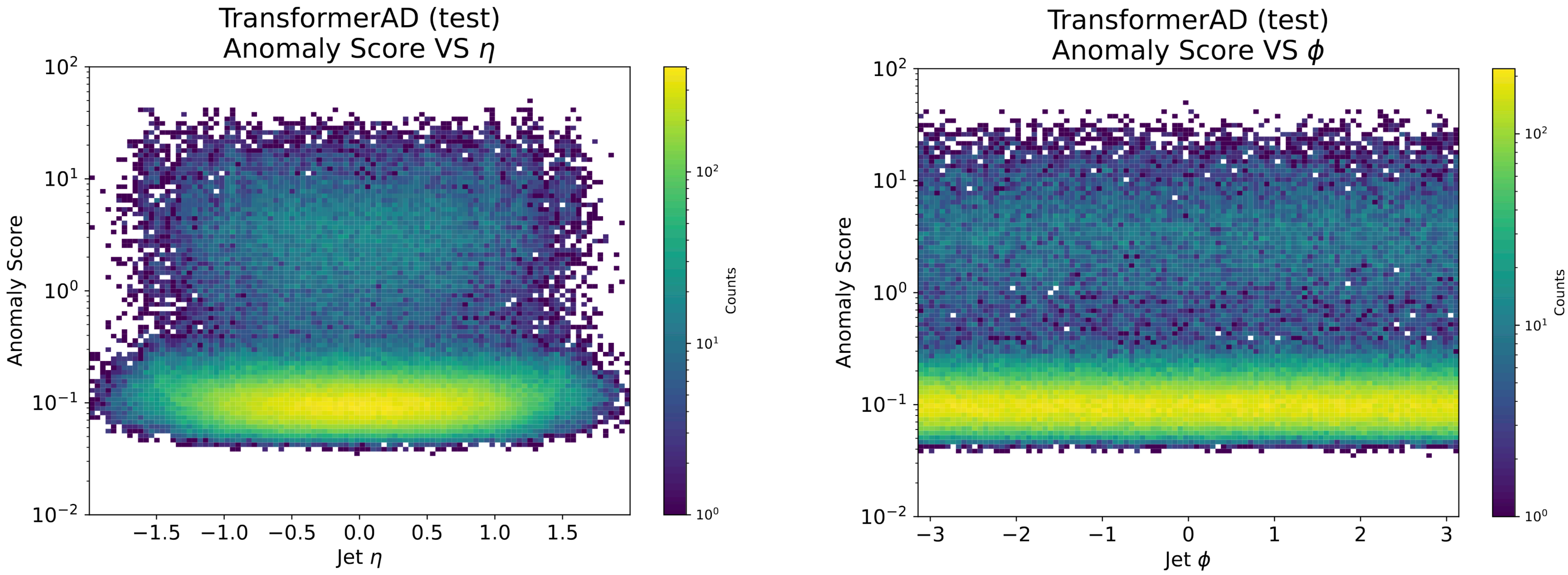




- each dot is the AS mean value in the p_T bin

AS dependence on η and ϕ

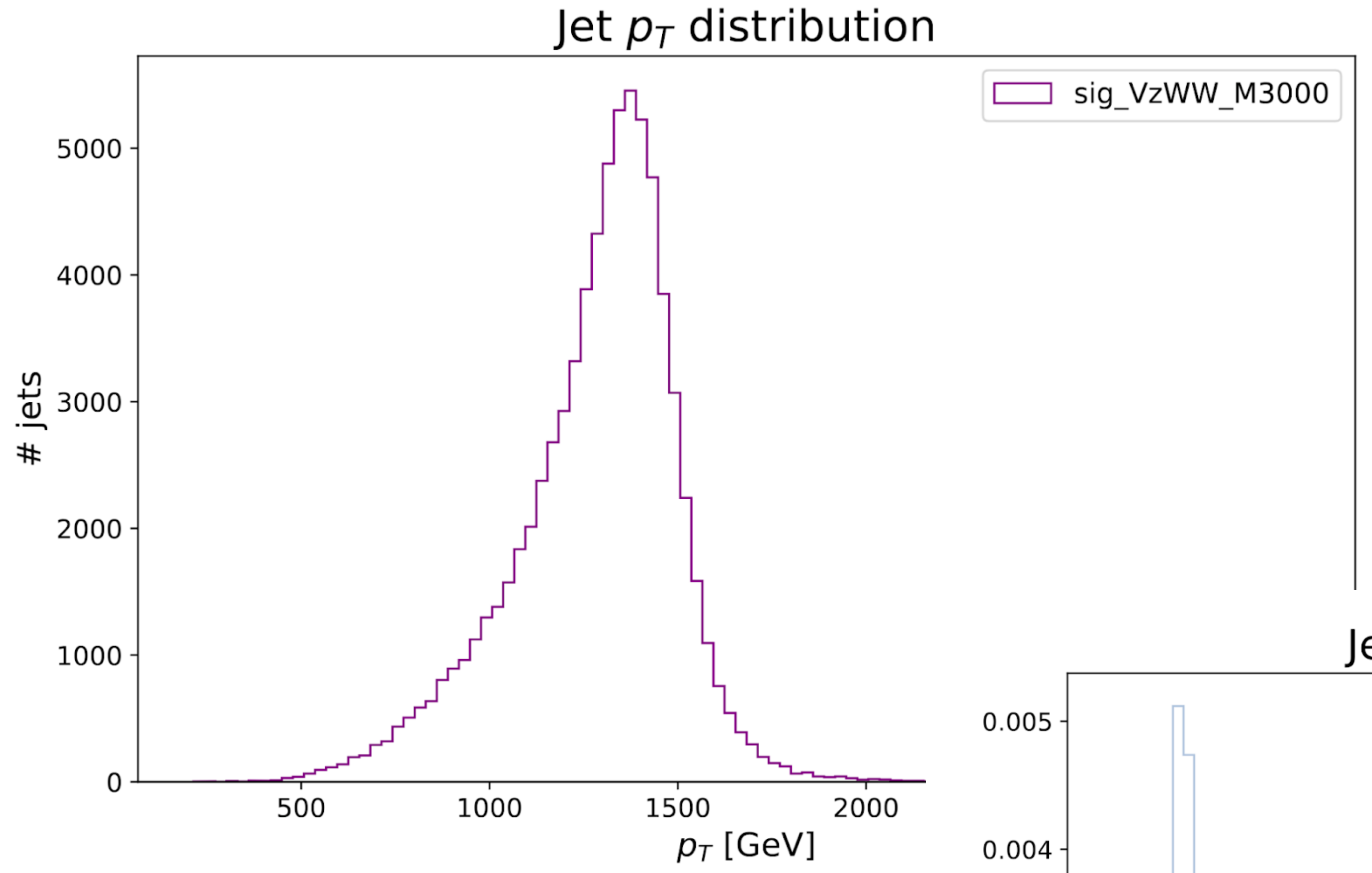
tested on JZ6

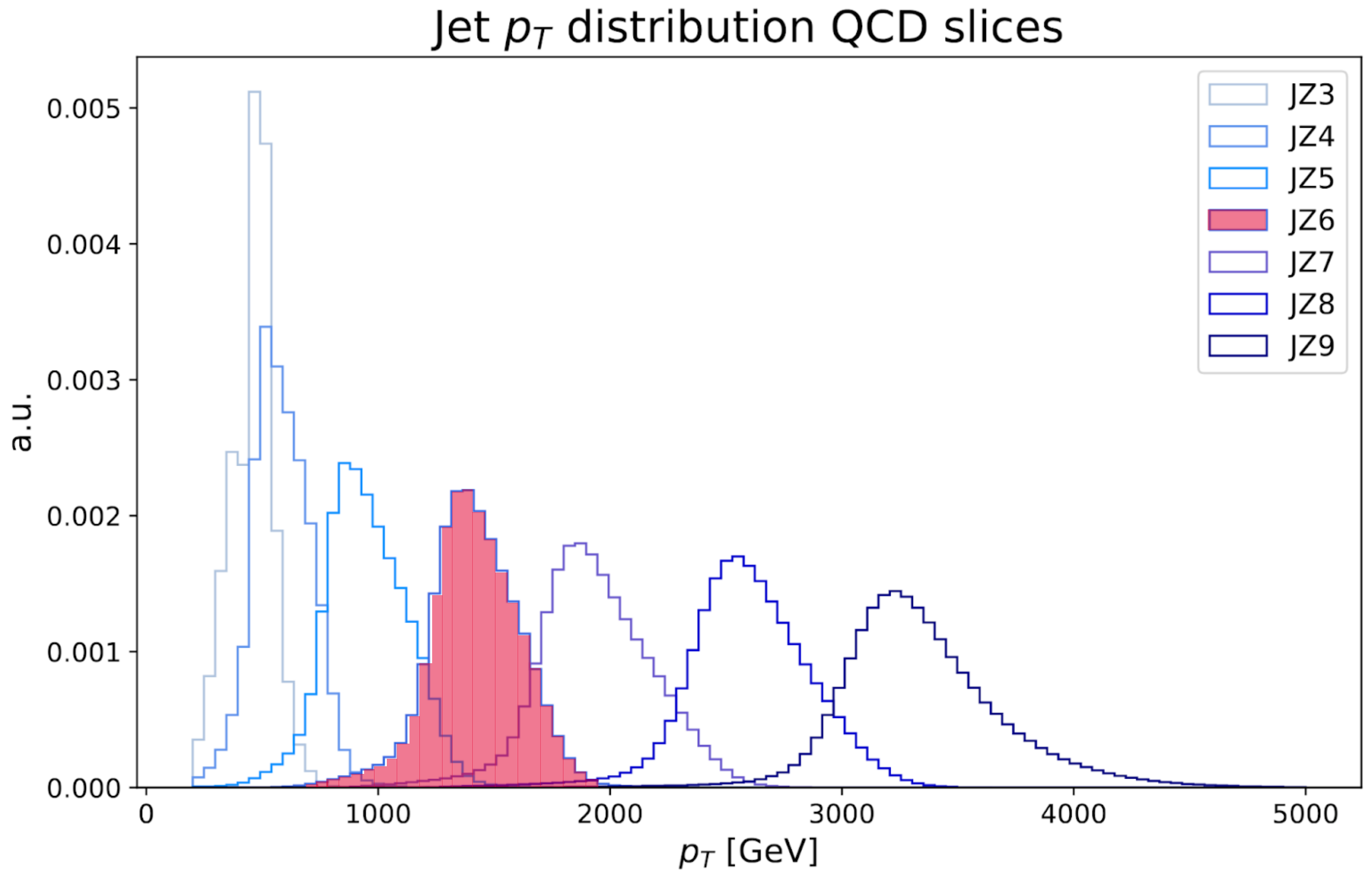


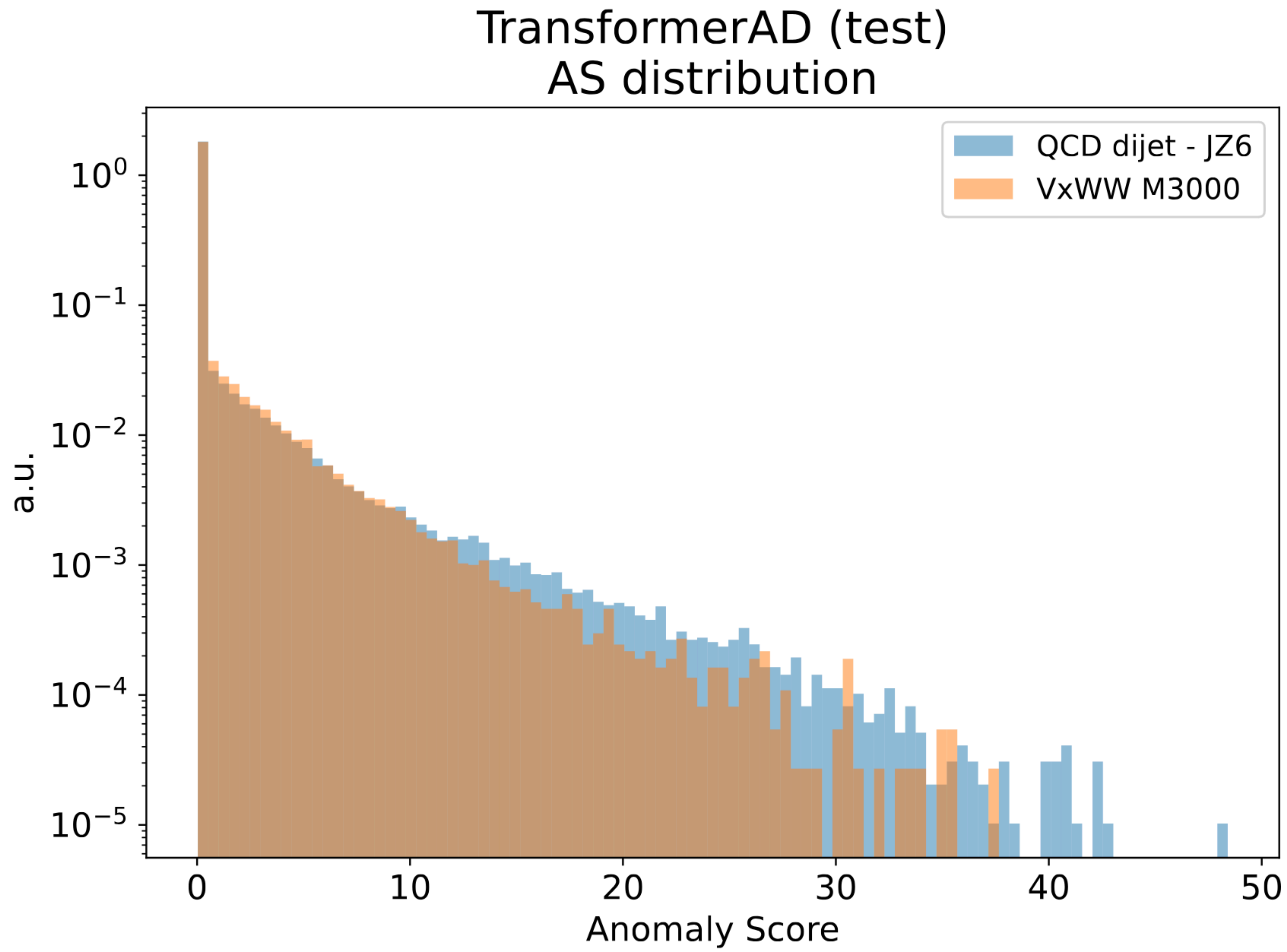
no dependence on η or ϕ

Testing on signal

Since the AS is dependent on the p_T and VxWW p_T distribution is similar to JZ6, training on JZ6 —> testing on VxWW



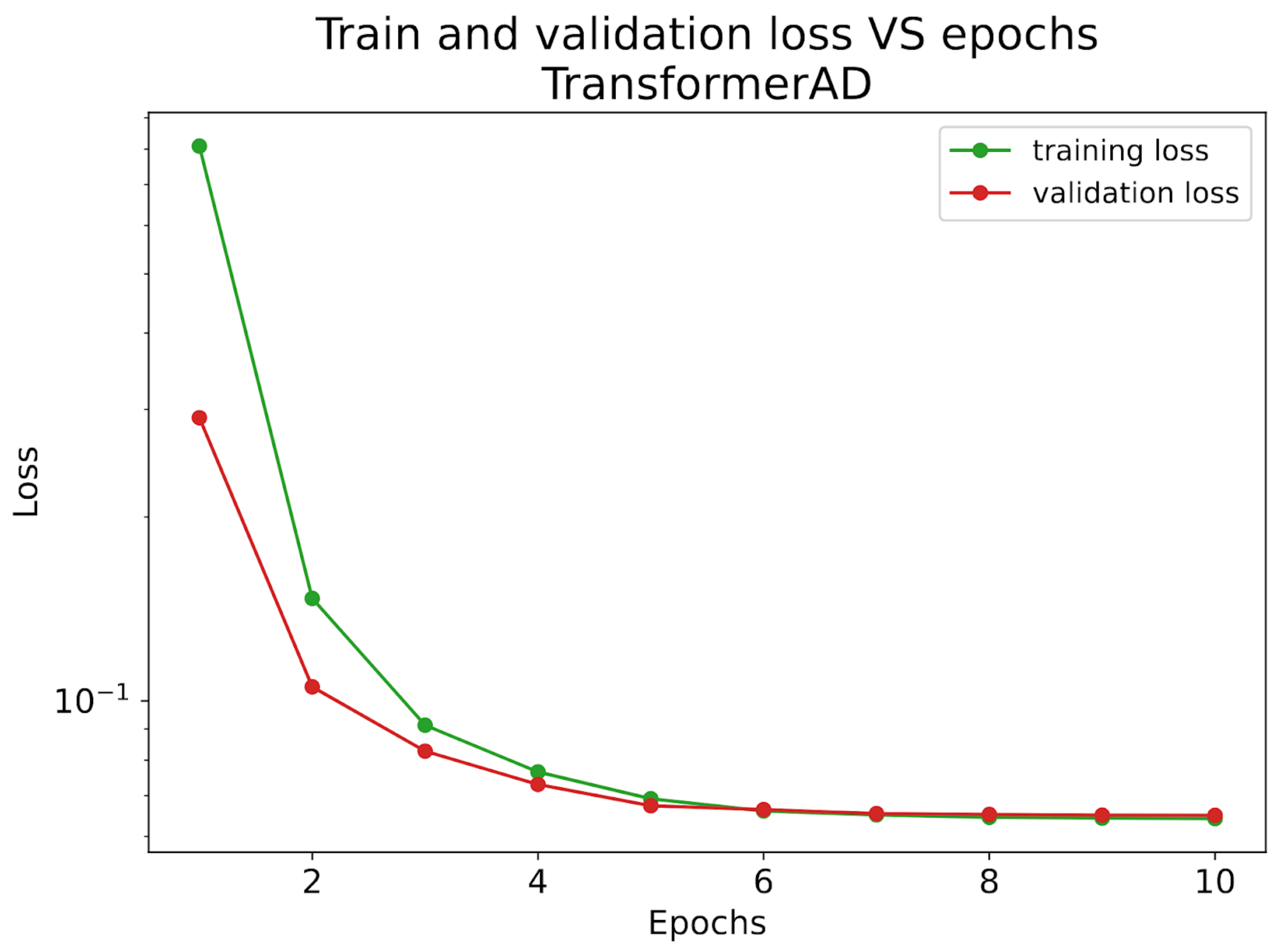




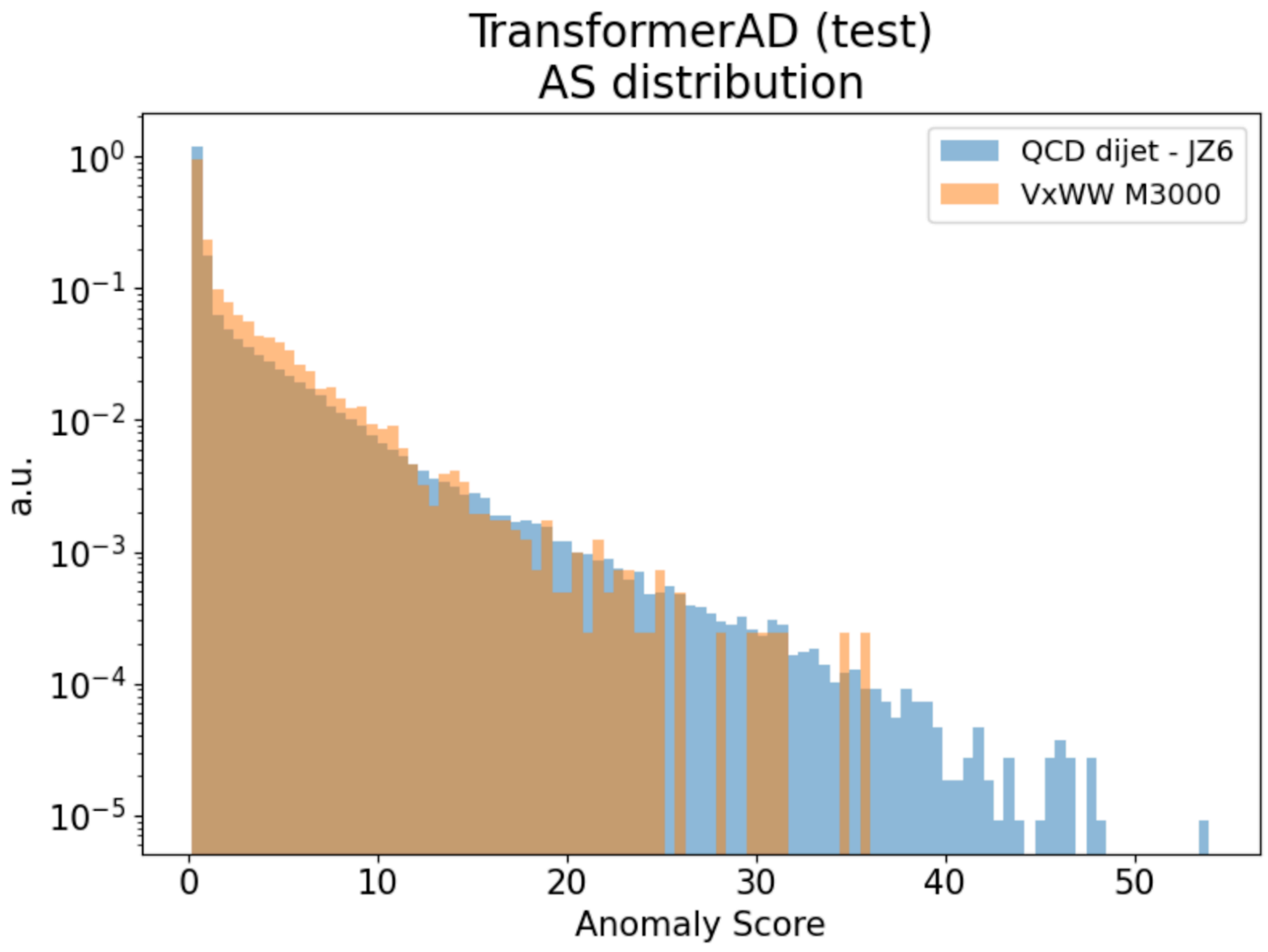
AUC = 60.0%

Testing on signal (fewer epoch)

Maybe the model after 50 epochs is learning too much…



10 epochs training on JZ6

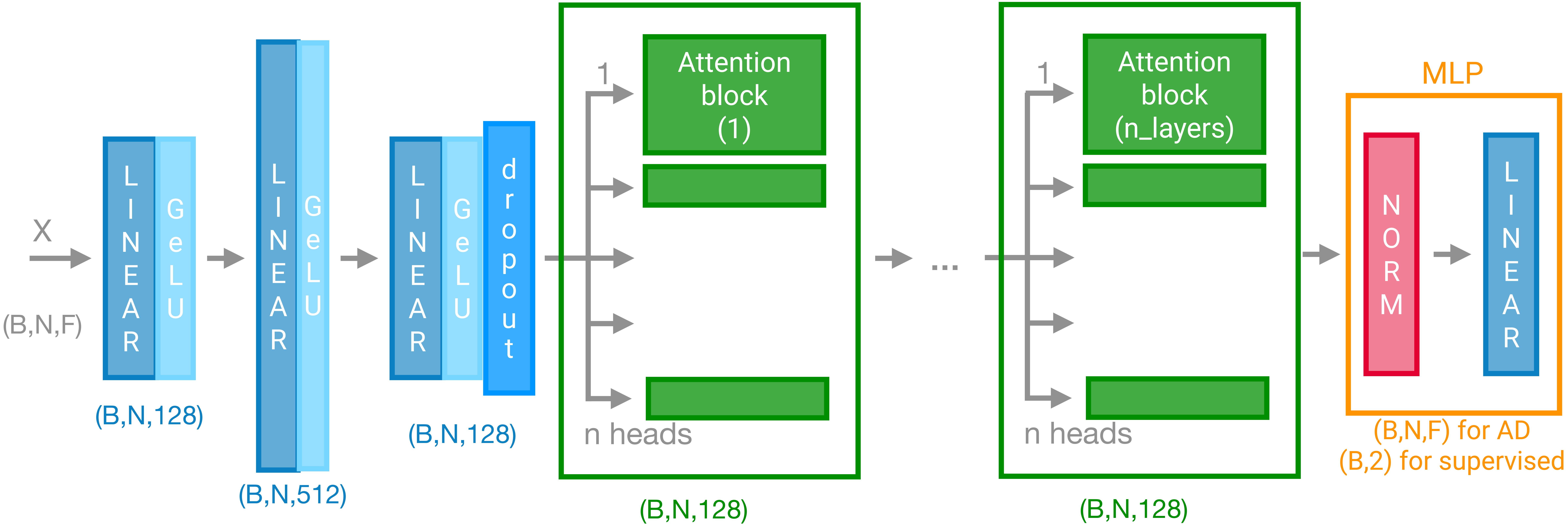


AUC = 60.0%

Back-up

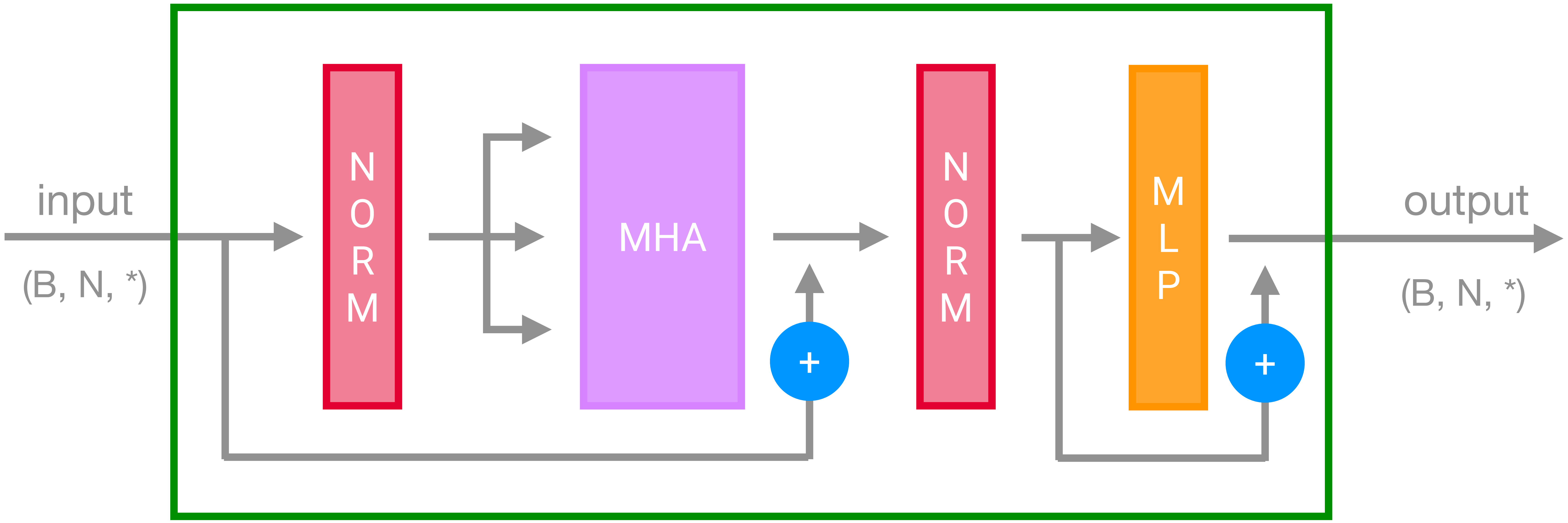
Transformer architecture

B = batch size
N = number of features (50)
F = number of features (3)



Attention block

B = batch size
N = number of features (50)
F = number of features (3)



Transformation - why?

Transformation **applied for data augmentation and model robustness reasons**

- need to decorrelate AD score from S/B discriminants, e.g. mass
- rescaling of the jet four momentum so that $m_{jet} = 0.25\ GeV$
- boost so that the jet energy is $E_{jet} = 1\ GeV$
- further rotation of constituents along jet axis

Apparent differences between paper description and code implementation…

Transformation - how? (paper POV)

0. Definitions:

constituent 4-momentum p_i^μ

jet 4-momentum $P_J^\mu = \sum_{i=1}^{N_J} p_i^\mu$

jet mass $m_J{}^2 \equiv (P_J^0)^2 - (\vec{P}_J)^2$

jet energy $E_J \equiv P_J^0$

1. Mass rescale (m_0): $P_J^\mu \rightarrow P_J'^\mu = \frac{m_0}{m_J} P_J^\mu$ $\quad p_i^\mu \rightarrow p_i'^\mu = \frac{m_0}{m_J} \times p_i^\mu$

2. Lorentz boost so that the final energy is fixed (E_0):

Lorentz boost = direction and Lorentz factor γ

1. $E'_J > E_0$: the boost is along the three-momentum direction of the jet
2. $E'_J < E_0$: the boost is opposite to the three-momentum direction of the jet

we impose $E_0 = \gamma E'_J - \beta\gamma \left|\vec{P}'_J\right|$

and, knowing that $\beta^2 = 1 - \frac{1}{\gamma^2}$

Transformation - how? (paper POV)

2. Lorentz boost so that the final energy is fixed (E_0): Lorentz factor γ

we impose $$E_0 = \gamma E'_J - \beta\gamma \left|\vec{P}'_J\right|$$ and, knowing that $$\beta^2 = 1 - \frac{1}{\gamma^2} ,$$

we end up with this 2nd order equation in γ
$$\left(\frac{m_0^2}{\left|\vec{P}'_J\right|^2}\right)\gamma^2 + \left(-\frac{2E'_J E_0}{\left|\vec{P}'_J\right|^2}\right)\gamma + \left(\frac{{E_0}^2}{\left|\vec{P}'_J\right|^2} + 1\right) = 0$$

Picking the smallest solution
$$\boxed{\gamma = \frac{1}{m_0^2}\left(E'_J E_0 - P_0\left|\vec{P}'_J\right|\right)}$$
where $P_0^2 = E_0^2 - m_0^2$

3. Using Gram-Schmidt procedure to find an orthonormal basis $\{\hat{e}_1, \hat{e}_2, \hat{e}_3\}$ so that the jet has its axis along the $\hat{e}_1$ direction

Transformation - how? (code POV)

1. ϕ rotation and η boost, so that the jet axis is in the origin of the η-ϕ plane:

```
1   def transformation(jet):
2       bv = jet.boostvector
3       bv.x = 0
4       bv.y = 0
5
6       jet_transf = jet.rotatez(-jet.phi())
7
8       jet_transf = jet_transf.boost(bv)
9
10      m_zero = 0.25
11      m_rescale = m_zero/jet_transf.m if jet_transf.m else 1
12      jet_transf = m_rescale * jet_transf
13
14      e_zero = 1.
15      A = (np.sqrt((np.abs(np.square(zero_E) - np.square(jet_1d.m))/
16                  (np.abs(np.square(jet_1d.e) - np.square(jet_1d.m)))))
17      bv2 = A * jet_transf.px / e_zero
18
19      beta = (jet_transf.e - (jet_transf.px/bv2))/(jet_transf.p - (jet_transf.e/bv2))
20      jet_transf = jet_transf.boost(beta, 0, 0)
21
22      return jet_transf
```

Starting with 4-momentum

$$P_J = \begin{cases} (P_x, P_y, P_z, E) \\ (P_T, \eta, \phi, m) \end{cases}$$

From the sk-hep library

```
def boostvector(self):
    """Return the spatial component divided by the time component."""
    return Vector3D(self.x / self.t, self.y / self.t, self.z / self.t)
```

after line 6

$$P'_J = \begin{cases} (P'_x = P_T,\ P'_y = 0,\ P'_z = P_z,\ E' = E) \\ (P'_T = P_T,\ \eta' = \eta,\ \phi' = 0,\ m' = m) \end{cases}$$

boost bv (lines 2-4)

$$bv = \left(0, 0, \frac{P_z}{E}\right)$$

Transformation - how? (code POV)

1. ϕ rotation and η boost, so that the jet axis is in the origin of the η-ϕ plane:

boost bv (lines 2-4) $\quad bv = \left(0,0,\frac{P_z}{E}\right)$

the sk-hep boost function performs the Lorentz boost in a generic direction (z-axis in this case)

$$\begin{bmatrix} ct' \\ x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \gamma & -\gamma\beta_x & -\gamma\beta_y & -\gamma\beta_z \\ -\gamma\beta_x & 1+(\gamma-1)\frac{\beta_x^2}{\beta^2} & (\gamma-1)\frac{\beta_x\beta_y}{\beta^2} & (\gamma-1)\frac{\beta_x\beta_z}{\beta^2} \\ -\gamma\beta_y & (\gamma-1)\frac{\beta_y\beta_x}{\beta^2} & 1+(\gamma-1)\frac{\beta_y^2}{\beta^2} & (\gamma-1)\frac{\beta_y\beta_z}{\beta^2} \\ -\gamma\beta_z & (\gamma-1)\frac{\beta_z\beta_x}{\beta^2} & (\gamma-1)\frac{\beta_z\beta_y}{\beta^2} & 1+(\gamma-1)\frac{\beta_z^2}{\beta^2} \end{bmatrix} \begin{bmatrix} ct \\ x \\ y \\ z \end{bmatrix}$$

after line 8 $\quad P''_J = \begin{cases} (P''_x = P_T, P''_y = 0, P''_z = 0, E'' = \sqrt{E^2 + P_z^2}) \\ (P''_T = P_T, \eta'' = 0, \phi'' = 0, m'' = m) \end{cases}$

```
8   jet_transf = jet_transf.boost(bv)
9
10  m_zero = 0.25
11  m_rescale = m_zero/jet_transf.m if jet_transf.m else 1
12  jet_transf = m_rescale * jet_transf
```

2. Mass rescale (m_0):

after line 12

$$\tilde{P}_J = \begin{cases} (\tilde{P}_x = \hat{m}P_T, \tilde{P}_y = 0, \tilde{P}_z = 0, \tilde{E} = \hat{m}\sqrt{E^2 + P_z^2}) \\ (\tilde{P}_T = \hat{m}P_T, \tilde{\eta} = 0, \tilde{\phi} = 0, \tilde{m} = m_0) \end{cases}$$

where $\hat{m} = \frac{m_0}{m_J}$

Transformation - how? (code POV)

```
14 e_zero = 1.
15 A = (np.sqrt((np.abs(np.square(zero_E) - np.square(jet_1d.m)))/
16               (np.abs(np.square(jet_1d.e) - np.square(jet_1d.m)))))
17 bv2 = A * jet_transf.px / e_zero
18
19 beta = (jet_transf.e - (jet_transf.px/bv2))/(jet_transf.p - (jet_transf.e/bv2))
20 jet_transf = jet_transf.boost(beta, 0, 0)
```

3. Boost along jet direction so that the final energy is fixed (E_0)

A factor (lines 15-16) $$A = \sqrt{\frac{|E_0^2 - m_0^2|}{|\tilde{E}^2 - m_0^2|}} = \frac{P_0}{\tilde{P}_x}$$

bv2 (line 17) $$bv2 = A \cdot \frac{\tilde{P}_x}{E_0} = \frac{P_0}{E_0}$$

beta (line 19) $$\beta = \frac{\tilde{E} - \tilde{P}_x \cdot E_0 \cdot P_0/\tilde{P}_x}{\tilde{P}_x - \tilde{E} \cdot E_0 \cdot P_0/\tilde{P}_x} = \frac{P_0\tilde{E} - \tilde{P}_x E_0}{\tilde{P}_x P_0 - \tilde{E}E_0} \implies \gamma = \frac{|P_0\tilde{P}_x - E_0\tilde{E}|}{m_0^2}$$

after the boost (line 20) $$P_J''' = \begin{cases} (P_x''' = \gamma P_T - \beta\gamma E_0,\ P_y''' = 0,\ P_z''' = 0,\ E''' = E_0) \\ (P_T''' = \gamma P_T - \beta\gamma E_0,\ \eta''' = 0,\ \phi''' = 0,\ m''' = m_0) \end{cases}$$

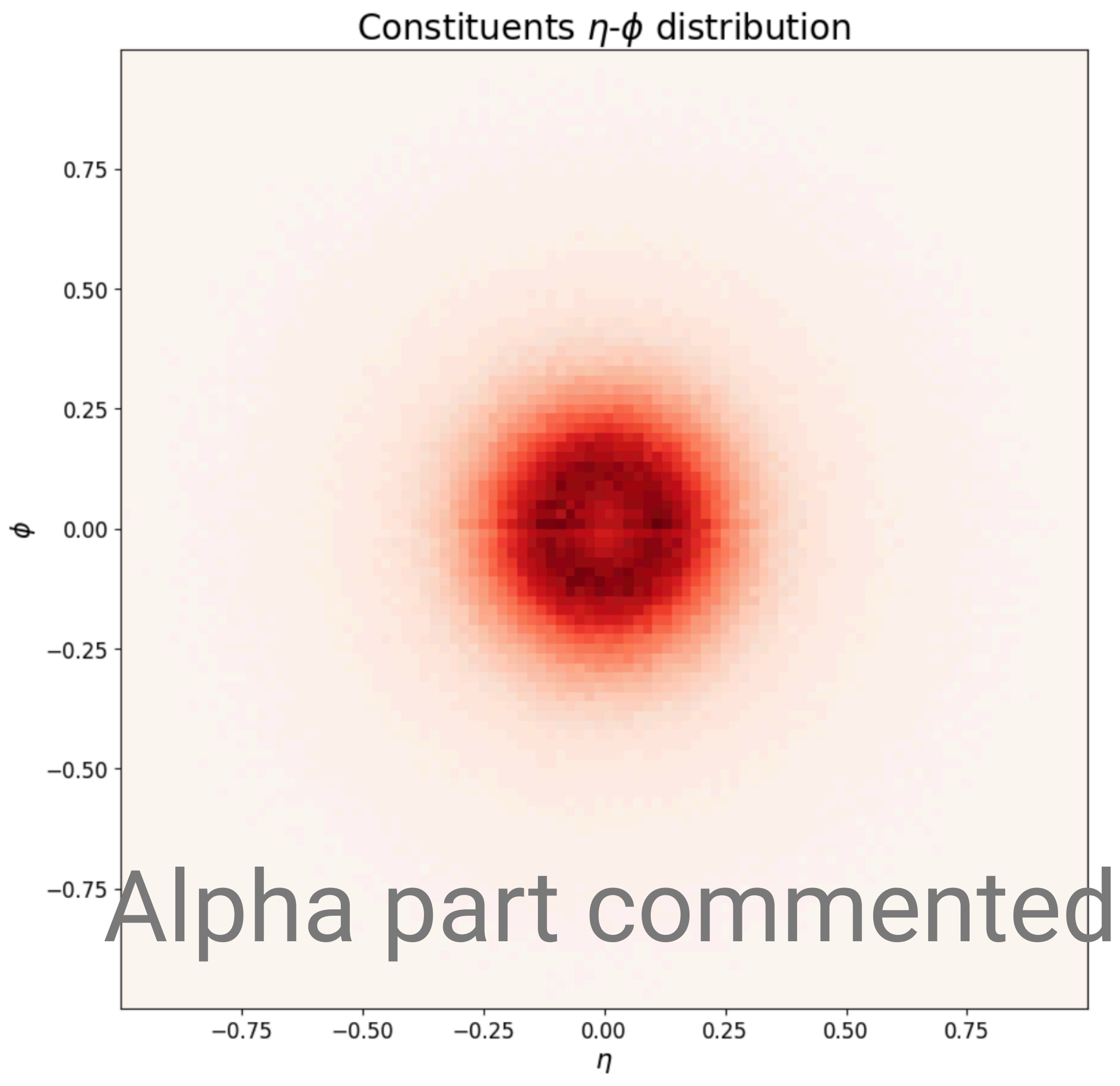
Transformation on constituents

Alpha rotation

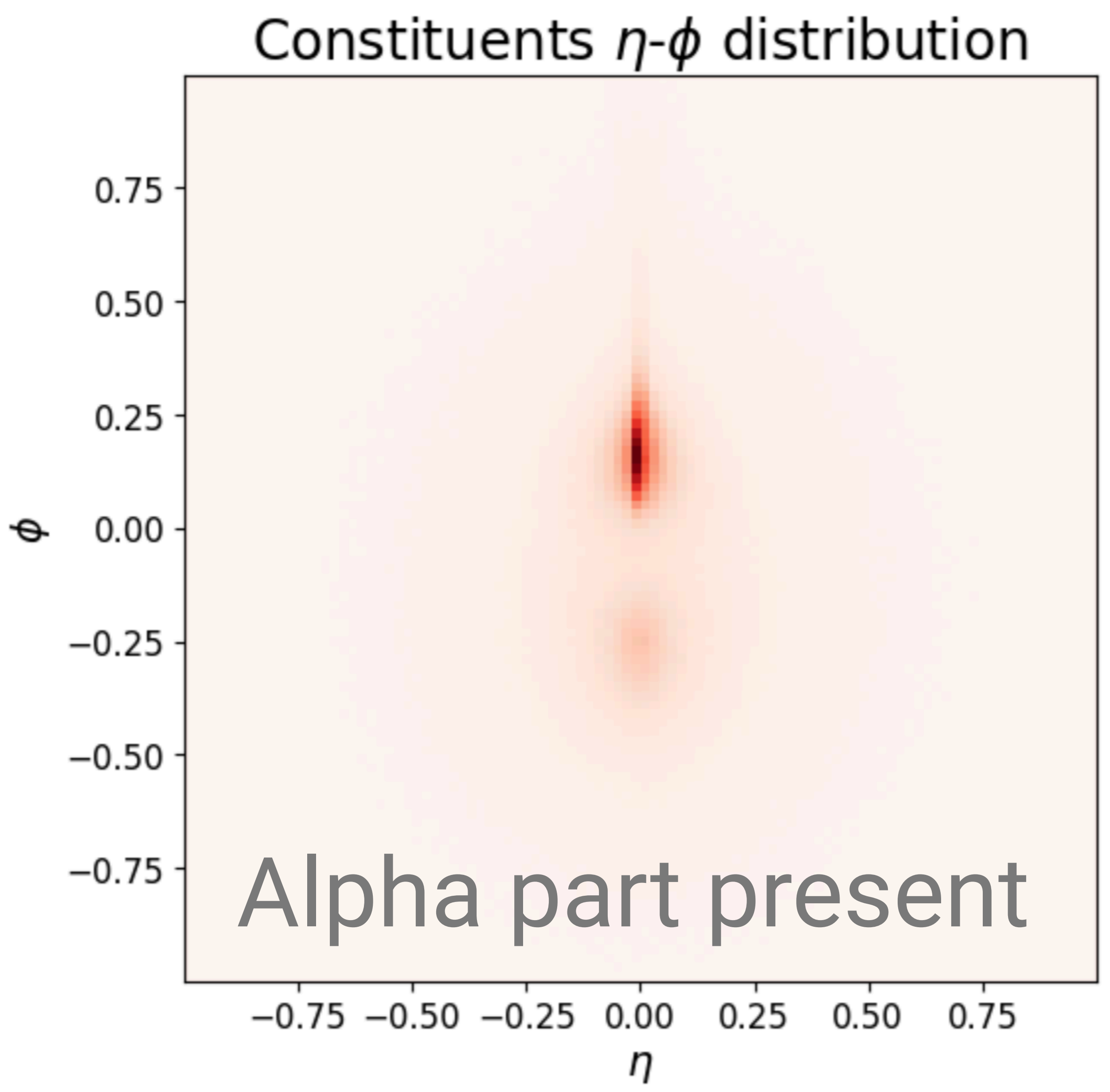
```python
        alpha = None
        new_const = []
        for c in data:
            const = LorentzVector()
            const.setptetaphim(c[0].item(), c[1].item(), c[2].item(), c[3].item())
            const = const.rotatez(-jet_phi)
            const = const.boost(bv)
            const = m_rescale * const
            const = const.boost(beta, 0, 0)
            if alpha is None:
                alpha = np.arctan2(const.phi(), const.eta) #Determine rotation angle in eta-phi plane
            const = const.rotatex(alpha - np.pi/2)
            dr = np.sqrt(np.square(const.eta) + np.square(const.phi()))
            if dr < R: new_const.append([const.pt, const.eta, const.phi(), const.m, c[4].item()])

        while len(new_const) < max_constituents:
            new_const.append([0.0, 0.0, 0.0, 0.0, 0.0])

        # returns the transformed constituents in the form of a torch tensor of shape (Nconst, 5) where the columns
        # are constituents' transformed pt, eta, phi, mass and original taste
        return torch.Tensor(new_const)
```



Alpha part commented



Alpha part present

Transformation on constituents

Alpha rotation

```
    alpha = None
    new_const = []
    for c in data:
        const = LorentzVector()
        const.setptetaphim(c[0].item(), c[1].item(), c[2].item(), c[3].item())
        const = const.rotatez(-jet_phi)
        const = const.boost(bv)
        const = m_rescale * const
        const = const.boost(beta, 0, 0)
        if alpha is None:
            alpha = np.arctan2(const.phi(), const.eta) #Determine rotation angle in eta-phi plane
        const = const.rotatex(alpha - np.pi/2)
        dr = np.sqrt(np.square(const.eta) + np.square(const.phi()))
        if dr < R: new_const.append([const.pt, const.eta, const.phi(), const.m, c[4].item()])

    while len(new_const) < max_constituents:
        new_const.append([0.0, 0.0, 0.0, 0.0, 0.0])

    # returns the transformed constituents in the form of a torch tensor of shape (Nconst, 5) where the columns
    # are constituents' transformed pt, eta, phi, mass and original taste
    return torch.Tensor(new_const)
```

After the transformation the constituents are free to assume every position in the eta-phi plane (accordingly to the transformation obviously)

Alpha rotation to decorrelate from this last rotational symmetry of the constituents

Alpha determined from hardest constituents for IRS reasons

Algorithm 1: Jet Alignment

Start

Boost jet in z direction until $\eta_{Jet} = 0$

Rotate jet about z axis until $\phi_{Jet} = 0$

Rescale jet four-vector such that m_{Jet} =0.25 GeV

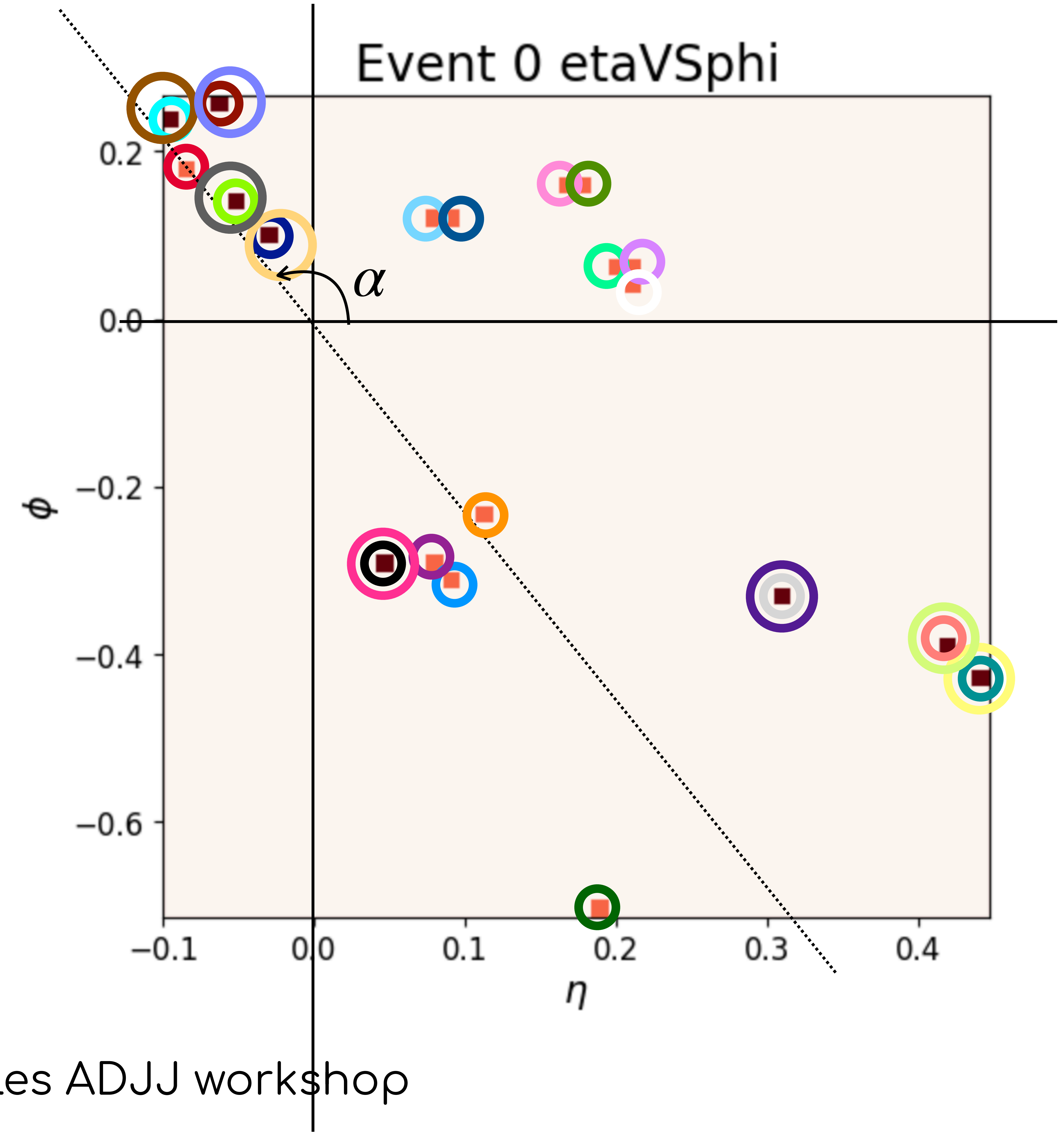
Boost jet along its axis until E_{Jet} =1 GeV

Rotate jet about x axis until hardest constituent has $\eta_1 = 0, \phi_1 > 0$

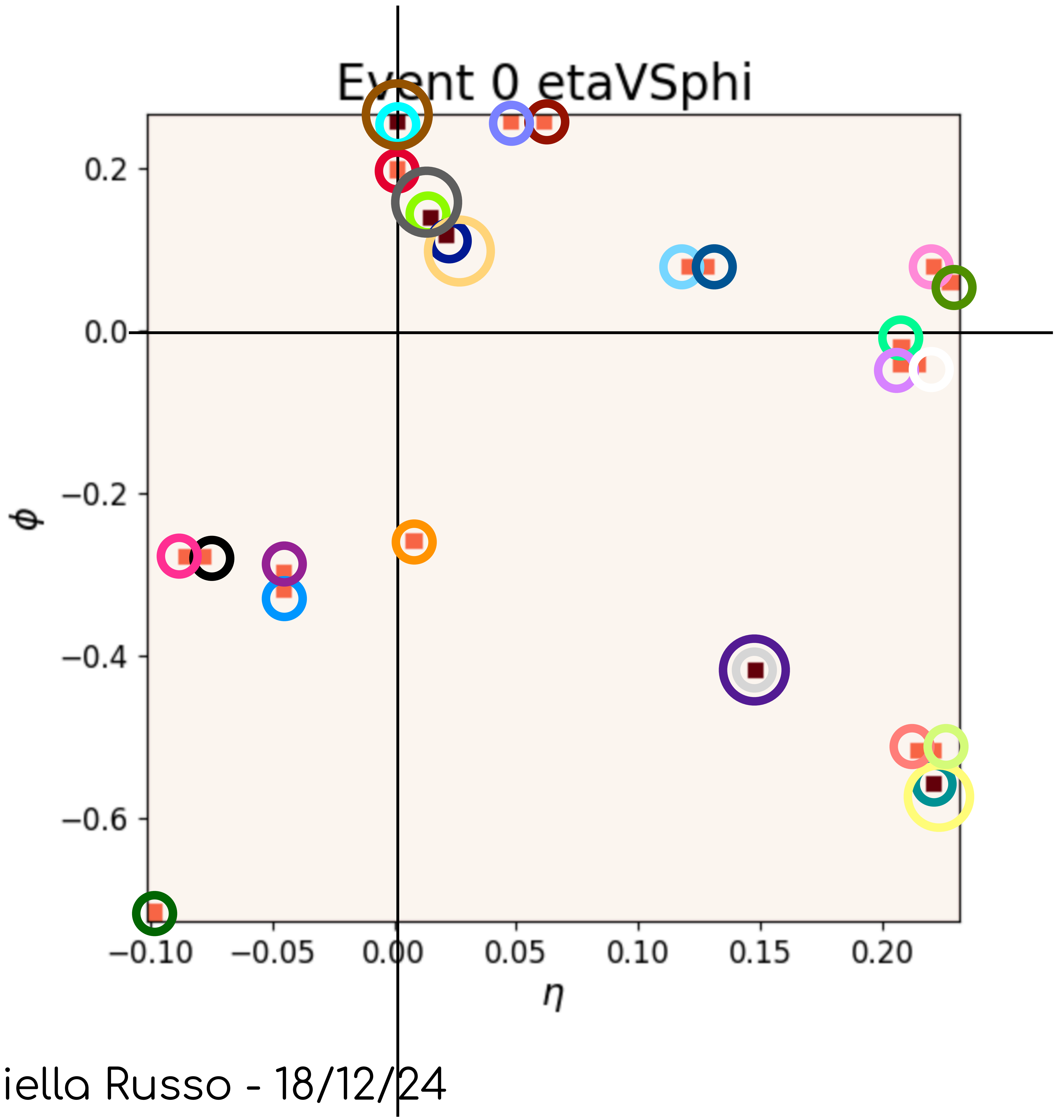
Alpha part present

Transformation on constituents

Alpha part commented



Alpha part present



Transformation on constituents

Alpha part commented

	η	ϕ		
3.3900e-01	-8.4415e-02	1.8026e-01	1.5360e-08	2.0000e+00
1.0997e-01	9.0090e-02	-3.0811e-01	-2.6342e-09	2.0000e+00
7.9118e-02	8.0372e-02	-2.8125e-01	2.9451e-09	2.0000e+00
6.4135e-02	-4.7351e-02	1.3943e-01	-1.8626e-09	2.0000e+00
6.3037e-02	4.3490e-02	-2.9432e-01	2.4640e-09	2.0000e+00
5.5979e-02	-1.0090e-01	2.2770e-01	-2.8705e-09	2.0000e+00
2.1933e-02	3.0831e-01	-3.2111e-01	-1.3576e-09	2.0000e+00
1.9825e-02	-3.0453e-02	1.0885e-01	-1.0413e-09	2.0000e+00
1.3262e-02	1.6976e-01	1.5392e-01	3.6814e-10	2.0000e+00
1.1576e-02	1.1122e-01	-2.2782e-01	9.5999e-10	2.0000e+00
9.0607e-03	-4.7351e-02	1.3943e-01	8.3948e-10	0.0000e+00
8.9918e-03	4.4441e-01	-4.3243e-01	-1.1642e-10	2.0000e+00
8.2963e-03	-1.0090e-01	2.2770e-01	-6.4817e-10	0.0000e+00
6.8075e-03	1.6976e-01	1.5392e-01	-4.4330e-10	0.0000e+00
6.5348e-03	3.0831e-01	-3.2111e-01	3.9478e-10	0.0000e+00
5.4859e-03	4.3490e-02	-2.9432e-01	-1.7462e-10	0.0000e+00
4.9525e-03	2.0138e-01	6.1824e-02	-1.7462e-10	2.0000e+00
4.7419e-03	-3.0453e-02	1.0885e-01	-1.7462e-10	0.0000e+00
3.6187e-03	2.0138e-01	6.1824e-02	-1.8407e-10	0.0000e+00
3.5313e-03	4.4441e-01	-4.3243e-01	7.1290e-11	0.0000e+00
3.2450e-03	7.4793e-02	1.1577e-01	1.2348e-10	2.0000e+00
2.9954e-03	2.0489e-01	4.8106e-02	-1.5400e-10	0.0000e+00
2.2131e-03	-6.6950e-02	2.6102e-01	-9.2034e-11	2.0000e+00
2.0251e-03	-6.6950e-02	2.6102e-01	-7.7001e-11	0.0000e+00
1.9977e-03	7.4793e-02	1.1577e-01	1.3337e-10	0.0000e+00
1.7178e-03	4.1071e-01	-3.8171e-01	-7.4200e-11	2.0000e+00
1.6191e-03	4.1071e-01	-3.8171e-01	-1.1822e-10	0.0000e+00
9.9748e-04	1.8833e-01	-7.1305e-01	1.4552e-11	2.0000e+00

Alpha part present

	η	ϕ		
3.3953e-01	-4.9909e-04	1.9883e-01	3.1534e-02	2.0000e+00
1.0993e-01	-4.4062e-02	-3.1934e-01	1.4005e-02	2.0000e+00
7.9022e-02	-4.2141e-02	-2.9083e-01	1.0109e-02	2.0000e+00
6.4071e-02	1.6281e-02	1.4630e-01	5.9770e-03	2.0000e+00
6.2664e-02	-8.1311e-02	-2.8786e-01	8.0315e-03	2.0000e+00
5.6229e-02	3.4468e-03	2.4865e-01	2.9252e-03	2.0000e+00
2.2692e-02	1.4699e-01	-4.1652e-01	2.3210e-03	2.0000e+00
1.9791e-02	1.8888e-02	1.1142e-01	1.8502e-03	2.0000e+00
1.2891e-02	2.3121e-01	6.5785e-02	3.6876e-03	2.0000e+00
1.1540e-02	1.0331e-02	-2.5730e-01	2.3880e-03	2.0000e+00
9.0693e-03	1.5458e-02	1.4639e-01	1.3705e-04	0.0000e+00
9.6366e-03	2.2356e-01	-5.6503e-01	9.5629e-04	2.0000e+00
8.3379e-03	3.2267e-03	2.4862e-01	1.6049e-04	0.0000e+00
6.7411e-03	2.1970e-01	6.8061e-02	2.0362e-04	0.0000e+00
6.7746e-03	1.4572e-01	-4.1513e-01	2.1008e-04	0.0000e+00
5.4723e-03	-8.2595e-02	-2.8609e-01	5.2071e-05	0.0000e+00
4.8834e-03	2.1646e-01	-3.2034e-02	1.1424e-03	2.0000e+00
4.7407e-03	1.8268e-02	1.1153e-01	2.3204e-04	0.0000e+00
3.6088e-03	2.0949e-01	-2.9508e-02	2.6388e-04	0.0000e+00
3.7878e-03	2.2289e-01	-5.6419e-01	2.7691e-04	0.0000e+00
3.1580e-03	1.2892e-01	7.0973e-02	1.0123e-03	2.0000e+00
2.9801e-03	2.0902e-01	-4.4302e-02	4.3391e-04	0.0000e+00
2.1654e-03	5.8128e-02	2.6689e-01	6.9179e-04	2.0000e+00
2.0134e-03	5.0779e-02	2.6565e-01	3.4904e-04	0.0000e+00
1.9751e-03	1.2057e-01	7.2828e-02	3.5346e-04	0.0000e+00
1.8108e-03	2.1699e-01	-5.1301e-01	3.3849e-04	2.0000e+00
1.7040e-03	2.1820e-01	-5.1440e-01	3.5421e-04	0.0000e+00
1.0100e-03	-1.0168e-01	-7.2727e-01	1.4552e-11	2.0000e+00