Studying the covariance II: LHC $t\bar{t}$ data

(Dated: August 16, 2021)

We want to see whether our quantum GAN has truly learnt the distribution it is supposed to. This includes whether it correctly captures the covariance matrix. In a previous set of tests we attempted to quantify how well the covariance matrix is learnt compared to the exactly specified one in the $3D$ Gaussian setup. In this set of tests we expand towards the LHC dataset.

I. SETUP

The basic setup is exactly as in the previous study with the only difference that now the data passed through the analysis is drawn from the provided $t\bar{t}$ LHC data and generated from the corresponding circuit that trained on it.

This is a big difference with respects to the previous case, as the exact cov.mat. is unknown and the amount of data we can draw is more limited than before. We currently have results for 2 settings:

1. $d_{latent} = 5$, $n_{layers} = 2$
2. $d_{latent} = 5$, $n_{layers} = 4$

these have chosen to be in the "good" parameter regime and trends will be more difficult to make out.

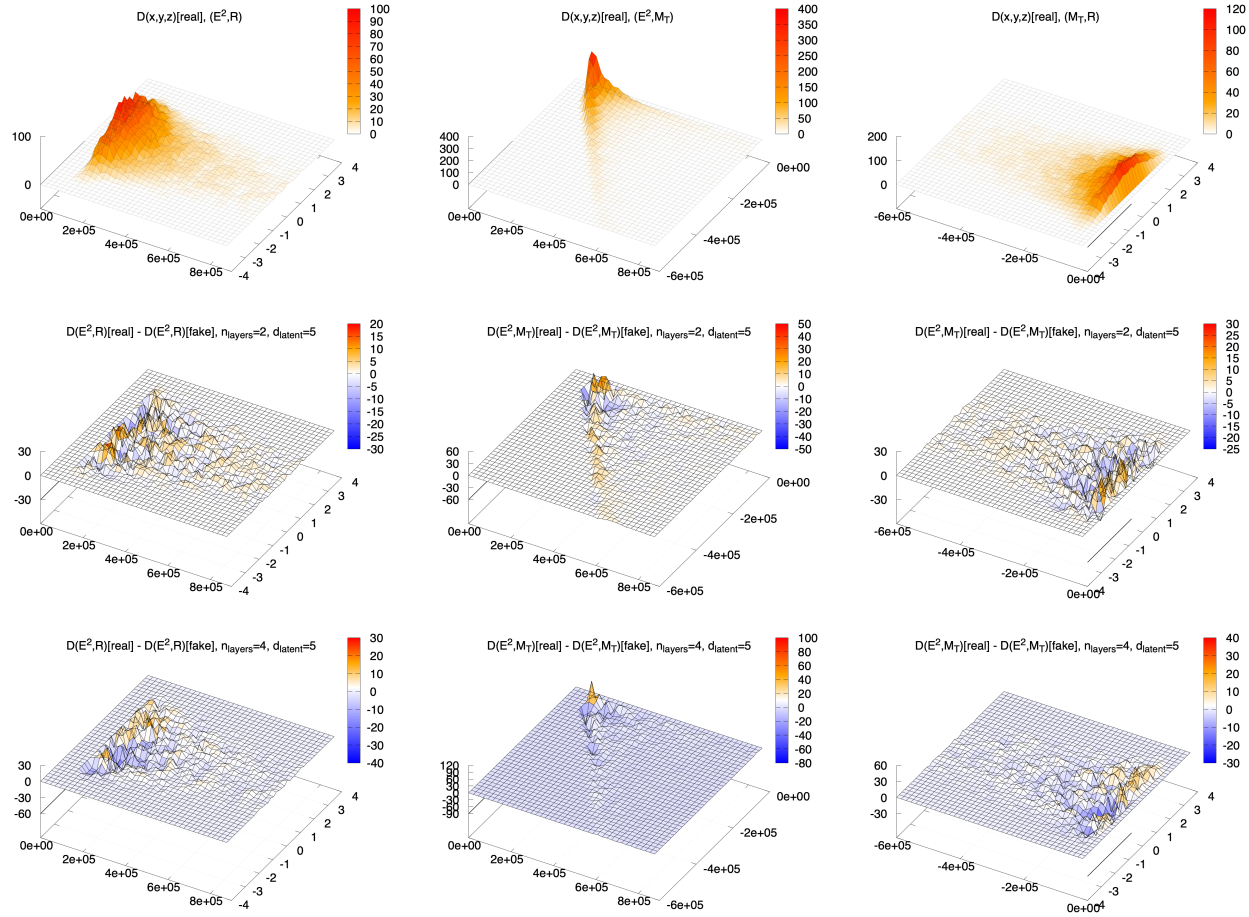

FIG. 1: Estimated densities and differences, please see text for details.

II. APPROACHING THE COVARIANCES

A. 3D histograms

As before we convert the samples $N(real)$ and $N(fake)$ into 3D histograms:

```
# there is an n_bin dependence here
nbin=40 (default)

# data looks like
datareal=(x,y,z)
datafake=(x,y,z)

D_real=np.histogramdd(datareal.T,bins=nbin,range=[[0,9e5],[-6e5,0],[-4,4]])
D_fake=np.histogramdd(datafake.T,bins=nbin,range=[[0,9e5],[-6e5,0],[-4,4]])
```

These define once more the effective densities $D(real)$ and $D(fake)$ and any derived results are dependent on the number of bins set for the 3D histograms (here: $n_{bin} = 40$).

In Fig. 1(top row) the densities for the real data, i.e. the target, are shown. They are projected from $(E^2, M_T, R) \to (E^2, R), (E^2, M_T), (M_T, R)$. The middle and bottom rows show the differences $D(real) - D(fake)$ for the settings 1. and 2. respectively

As before we do not observe a perfect match, but the same considerations hold as to how to quantify this since also the target is only finitely sampled.

B. Towards single number measures: Density deviation and KL divergences

With this in mind we can define a single-number measure based on the difference of the densities as:

```
# define measure as summed difference between the real and fake densities
meas = sum( D(real) - D(fake) )
```

We cannot easily estimate the baseline deviation from sampling the target, since the full dataset is used, and the normalistion is omitted for now. The results we obtain this way are:

```
# d_latent=5
n_layers=2:        meas = 4.639375
n_layers=4:        meas = 4.57875
```

We observe roughly the same value for both settings. Even without the normalisation this would indicate both are very similar and none a priori better.

We can further determine the KL divergences as before:

$$KL_{sum} = \sum_{w=x,y,z} KL[D(real, w), D(fake, w)]$$

by projecting the $real/fake$ combinations to 2D in e.g. x_{real}, x_{fake}. If the distributions match perfectly KL_{sum} should be 0. We find:

```
# d_latent=5
n_layers=2:        KL_sum = 0.007
n_layers=4:        KL_sum = 0.01
```

which is also at the level of what we observed in the 3D Gaussian case.

C. Covariance matrix

Next we look at the covariance matrix, where we have access to:

1. target and learned cov.mat
2. Eigenvalues and Eigenvectors in both cases
3. New fake samples through the cov.mat. (but using a Gaussian as underlying distribution)
4. KL_{sum} as before

The KL divergences are not finite in this case, even with an offset supplied. This is likely to do with how the new fake samples fed into this measurement are drawn and points 3. and 4. are therefore dropped. The full results are:

```
 # ### Checking Nqubits=3, latent_dim=5, layers=2

  Estimated target covmat:
[[ 7.98751959e+10 -4.49449461e+10 -1.60790274e+03]
 [-4.49449461e+10  4.67255376e+10  2.09009246e+03]
 [-1.60790274e+03  2.09009246e+03  1.21827223e+00]]
  Obtained learned covmat:
[[ 5.10165884e+10 -1.72150149e+10 -5.57061936e+03]
 [-1.72150149e+10  1.89905588e+10  1.81668822e+03]
 [-5.57061936e+03  1.81668822e+03  1.26291647e+00]]

  Relative difference target and learned covmat, dmeas= 1.2794729990618734 and matrix:
[[0.63870377 0.38302449 3.46452508]
 [0.38302449 0.40642783 0.86919036]
 [3.46452508 0.86919036 1.03664554]]

  Eigenvalues of target and learned covmat:
  [1.21817431e+00 1.53965763e+10 1.11204157e+11]
  [1.26230790e+00 1.14924405e+10 5.85147068e+10]
 Averaged ratio of Eigenvalues: 1.4017329183943579

  KL divergences in (x,x), (y,y) and (z,z): inf inf 1.045e-10

# ### Checking Nqubits=3, latent_dim=5, layers=4

  Estimated target covmat:
[[ 7.98751959e+10 -4.49449461e+10 -1.60790274e+03]
 [-4.49449461e+10  4.67255376e+10  2.09009246e+03]
 [-1.60790274e+03  2.09009246e+03  1.21827223e+00]]
  Obtained learned covmat:
[[ 6.29555442e+10 -2.65372211e+10  1.02848497e+04]
 [-2.65372211e+10  2.89722222e+10 -3.77743136e+03]
 [ 1.02848497e+04 -3.77743136e+03  1.20328214e+00]]

  Relative difference target and learned covmat, dmeas= -1.4256316636765112 and matrix:
[[ 0.78817389  0.59043838 -6.39643771]
 [ 0.59043838  0.62005113 -1.80730347]
 [-6.39643771 -1.80730347  0.98769562]]

  Eigenvalues of target and learned covmat:
  [1.21817431e+00 1.53965763e+10 1.11204157e+11]
  [1.20158444e+00 1.44529225e+10 7.74748440e+10]
 Averaged ratio of Eigenvalues: 1.171485492593393

  KL divergences in (x,x), (y,y) and (z,z): inf inf 2.833e-10
```

III. OVERVIEW

As before it looks like the qGAN has learned at least part of the covariances. In particular the current setup deployed to the quantum machines with $n_{layers} = 2$ and $d_{latent} = 5$ seems a choice that works. However, it would be interesting to see whether this can be reduced to $d_{latent} = 3$ as in the case of the $3D$ Gaussian. Note that all analyses here are performed on the data as is, i.e. the samples have been scaled back to their original ranges. It is great to see that the qGAN performs well and picks up the correlations across the scaling transformation.

Things to do:

- It would be good to perform a scan in d_{latent} in order to study possible trends similarly to the $3D$ Gaussian case.
- The target data dependence needs to be understood. One way would be to bootstrap the target or to perform a cross-validation analysis.
- Further LHC data sets could be analysed.