Dataset

I am using data from Pythia simulations of e^+e^- collisions at 91 GeV. The .csv file contains a million events, each one containing the 8 most energetic partons in each event. Each column of the .csv file corresponds to a component of the particle's four momentum (E,px,py,pz). In events which contain less than 8 particles I have used zero-padding during the data extraction stage. Therefore no zero-padding is needed during the training of the GAN. I have turned off ISR as well as electro-weak showers in FSR and I have only used 5 quarks (all quarks excluding the top quark) in the parton shower. It is also worth mentioning that I have taken the 8 most energetic partons of each event without utilizing any jet algorithm, which means that each event contains particles from both jets that result from the initial collision.

End Goal

By increasing the number of dimensions (gen_dim) from three to four (and subsequently changing the noise_dim as well) it can be seen that the four-momenta distributions of the generated particles are reproduced relatively well. The issue, of course, is that the EEC distribution is not accurately reproduced. Furthermore, I've also noticed that the on-shell condition is completely absent from my generated data, resulting in unphysical particles whose invariant mass squared is negative. As we've already discussed in the last meeting, I've tried to input some physical constraints in the model by introducing a penalty term in the generator's loss function. The end goal would be to introduce a penalty term that forces the GAN to learn correctly the invariant mass squared of each parton and terms like the opening angles or the energy flow polynomials in each event. By doing that, we would be able to see if this approach could help in producing a better EEC distribution (or any IRC safe observable's distribution for that matter). Unfortunately, I am facing some issues regarding the implementation of such a regularizing term. In the code that I've provided I am simply trying to implement the on-shell condition by constructing a penalty term that aims to force the GAN model to accurately produce data that have the same invariant mass as the true data.

Penalty term and issues related to scaling

The true and generated tensors have a shape [BATCH_SIZE,8,4] in which each event contains a 2d matrix of the form:

$$\begin{pmatrix} E^1 & p_x^1 & p_y^1 & p_z^1 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

In order to construct the term that's going to impose the on-shell condition, I element-wise multiply each tensor (X_real and G_fake) first with itself and then with a constant tensor tf.constant([1.0,-1.0,-1.0,-1.0]) which (from what I understand) is broadcasted automatically to fit the dimensions of the tensors containing the result of the element-wise multiplication of the true or generated data in each batch. I then sum all the elements in each row of tensor by using tf.reduce_sum(...,axis=-1) in order to obtain a rank-2 tensor of shape [BATCH_SIZE,8] in which each of the 8 columns (corresponding to the 8 partons in each event) is going to contain the following relation: $(E^i)^2 - (p_x^i)^2 - (p_y^i)^2 - (p_z^i)^2 = (m^i)^2$, where $i = 1, 2, ..., 8$. I then proceed to subtract this rank-2 tensor of the generated data from the rank-2 tensor of the true data. I then construct a regularization term which is going to be the mean of the absolute value (or square) of the difference between the constructed rank-2 tensor of the generated data and the true data (you can easily see that in the code). Finally, I insert this term, multiplied by a constant in an effort to force the loss function to minimize this penalty term and thus to force every generated particle to have the same relation between the squares of their-four momenta as the particles from MC simulations, which essentially means that I want them to have the same invariant mass as the MC generated particles.

Two issues that I have faced are the following: a) my regularization term is initially minimized and then reaches a point that cannot be minimized any further (which is not really close to zero either), regardless of the value of the constant in the penalty term that I input, b) I am not sure if that relation of the on-shell condition is correct in the case of the scaled data (using min-max scaling as in the original code).

Regarding the first issue, I have tried tweaking some parameters like the learning rate of the optimization algorithms in both networks, or the value of the regularization constant (even taking the values to extreme levels which spoil the resulting four-momenta distributions), changing the ranges or completely ignoring gradient clipping, introducing Batch Normalization layers in my model (no particular reason why, just trying different things), without seeing any improvements on the minimization of the regularization term. I have also tried changing the value of the penalty term which contains the norm of the discriminator, or even getting rid of this penalty term completely (inspired by some papers regarding the regularization of GANs) because I was suspecting that this term was severely restricting the form of my discriminator. Unfortunately, I did not see an improvements on the minimization of the regularization term in both cases. It is worth point out that whenever I got rid of the penalty term in the discriminator without inputting my constructed penalty term in the generator I usually got NaN values in my loss function or values that remain unaltered in each epoch. If I understand correctly, such values stem from the issue of exploding or vanishing gradients that GAN models usually come along with. When I got rid of the penalty term in the discriminator but also inserted my penalty term in the generator,then my loss functions yielded reasonable results without, however, minimizing the regularization term after a certain point. In my code I also plot how the original term in the generator loss function and how my regularization term change in each epoch.

When it comes to the actual formula of the on-shell condition, I am worried that it is a bit more intricate than what is described above, due to the presence of the min-max scaling. To be more specific, by implementing this form of scaling, each value is consequently scaled as: $X \to X' = \frac{X - X_{min}}{X_{max} - X_{min}}$. When constructing my rank-2 tensor of shape [BATCH_SIZE,8] and inserting this term in the loss function, I basically tell the GAN to create generated data that will adhere to the formula: $(E'^{i})^2 - (p_x'^{i})^2 - (p_y'^{i})^2 - (p_z'^{i})^2$. The thing is that $p_x'^{i2} = (\frac{p_x^i - p_{x,min}}{p_{x,max} - p_{x,min}})^2$,which apart from quadratic terms in the momentum, includes also a constant term and a term linear in p_x^i. So the actual condition that I am enforcing is not,in the end, the on-shell condition that I was looking for, but just some random relation between the components of the four-momenta. I could, theoretically overcome this issue by expressing the on-shell condition in the scaled variables and impose that specific relation (which will also include linear terms in the momenta) in my loss function instead of the relation that I presented above, during the construction of the rank-2 tensor. The problem with this approach is that in order to do that, I would inadvertently impose the inverse scaling transformation (since I am basically expressing the on-shell condition in terms of the p'_μ variables by using the relation between p_μ and p'_μ) and I will thus have problems regarding the minimization of the loss function. In my code I plot the histograms of the invariant mass squared for the generated and true data both in the original and the scaled variables, in order to better explain the concerns I am having regarding the on-shell condition. I have also tried getting rid of the scaling completely which didn't really go well (as expected). Additionally, I have also tried scaling the data (without shifting them by X_{min} as is the case in the min-max scaling) by simply dividing them with a constant (e.g. the energy in the C.o.M or just half of that), which is something that you also suggested, if I remember correctly, in the last meeting. By doing that, I would essentially get rid of all those extra linear and constant terms that stem from shifting the data with X_min. The problem is that the Energy only takes positive values, whereas the three-momenta take also negative values. By scaling my data with a simple constant I would manage to get the energy values to range in a region approximately from 0 to 1 but the three-momenta to range from values approximately close to -1 and all the way up to 1. Therefore, I am not able to bring all the variables of my input data on the same range (as is the case with the min-max scaling which makes all the input variables to range from 0 to 1) and I thus face optimization related issues. I have tried implementing that specific type of scaling without any success but I admittedly haven't put enough time in this specific case, as there are so many strategies and parameters that I have tried in order to get the result that I want. It could be the case that this is the correct course of action but I would need to pay more attention to the details.

Next Steps

In case you're wondering what all of these things have to do with the EEC, which was the original goal after all, it's because I believe that the on-shell condition is, first of all, essential in order to get meaningful generated data, and secondly, it is more straightforward to implement. I am convinced that if I manage to do that, I could also force (using the same strategy) the GAN model to learn the correct opening angles or/and energy flow polynomials between the particles in each event and see if that improves the EEC distribution (which I think it will). What I am trying to do now is try to start from simpler constraints and see what I am doing wrong. Therefore, I am currently trying to implement the on-shell condition using the same form (the usual formula that has the energy squared minus the norm of the three-momentum squared) as is the case of the original, unscaled variables. Even if this condition is not really physical (since from what I suspect the on-shell condition has a different form in the min-max scaled space) it still corresponds to some distribution (with not particular physical meaning) that the GAN cannot accurately reproduce (although it comes close compared to the case of the unscaled data). By doing that, I want to test if I can get my regularization term to be minimized enough so as to get the correct distributions in the scaled space. I believe that if I manage to do that, it would be easier to then go ahead and find a way to implement the actual on-shell condition expressed as a function of the scaled variables.

Additional things I have tried

I was also thinking of ditching the idea of adding an extra regularization term completely and instead make some modifications either in my data or in the architecture of the GAN. To this end, I tried introducing a second discriminator in my model. One discriminator is going to be trained on the data (as is already the case) and the other is going to instead take as input the constructed rank-2 tensor that I mentioned above. Basically the second discriminator will be trained on the relations between the square of the energies and the three-momenta for each particle and see which relation is true and which is generated. It is not really evident to me what the form of the loss function is going to be in this case. I have tried different loss functions (e.g. just adding the expected value of the output of the second discriminator in the generator loss function) with "interesting", although incorrect, results. I have also thought about keeping the original architecture with one discriminator and just augmenting my data by introducing another column for each particle which will be the on-shell condition itself,i.e., the square of the 0th column (energy) minus the square of the rest of the columns (three momenta components). Unfortunately, that did not work out, although I didn't bother too much with approach. Someone could ask why not include an extra column in the beginning (in the .csv file) which will include the invariant mass of each particle, if the goal is to actually get the correct masses of the quarks. However, that would result into adding an extra, unnecessary degree of freedom which wouldn't also enforce the on-shell condition between the four-momenta. The goal is to generate the correct correlations between the four-momenta of each particle (mass) and between different particles in each event (opening angles). The invariant mass and the opening angles are encoded in these correlations. I was also thinking (as I've already told you) that perhaps a conditional GAN could actually help impose the on-shell condition or help enforce the GAN to learn the energy flow polynomials but I haven't really looked into that case yet.

One last thing

I find it interesting how the GAN actually manages to produce the distribution of the EEC and the invariant mass squared in the case of the scaled variables (not perfectly but certainly better than the distributions of the unscaled variables). It is as if the GAN is actually learning the correlations between the data in the scaled space but these correlations do not translate well once we take the inverse scaling transformation. I would also like to know if you can also see that fact using your data and kinematic variables. To be more specific, I would also like to know if the discrepancy in the EEC distribution between the real and generated data is smaller in the scaled data (before you perform the inverse transformation) than the unscaled data. The form of the EEC, of course, is going to look different in the case of the scaled data.