Towards physically consistent data-driven weather forecasting: Integrating data assimilation with geometric deep learning in a case study with ERA5

Third Round of Revisions

Thank you again for thoroughly responding to the reviewers' comments: At this point, the manuscript is clear and the authors' framework reproducible. However, the manuscript's first key point remains to be proven because (1) U-NETx and U-STNx are not compared in a systematic fashion (1.1); and (2) U-STN is still misleadingly presented as equivariance-preserving in parts of the manuscript (1.2). Additionally, WeatherBench models may be unfairly compared to a version of U-STN including data assimilation cycles (1.3), and U-STN is still framed as a physics-informed machine learning (ML) framework (1.4). Finally, there may still be discrepancies between the shared code and what is reported in the manuscript (2.1). I continue to think that this manuscript will be a welcome contribution to GMD once revised so as to not mislead readers about equivariance preservation. Overall, it would be helpful to clarify whether the improved test performance of U-STNx reported by the authors without data assimilation (e.g., Fig 6) is truly the result of some form of equivariance preservation, or whether using a bottleneck and a spatial transformer simply prevents overfitting the WeatherBench training set.

Contents

1 Major Issues

1.1 Comparison between U-NET and U-STN

The first key point of the manuscript relies comparing U-STN with U-NET to establish U-STN's improved performance compared to U-NET ("improving the forecast skill by 45%"). This requires being as precise as possible when comparing U-STN and U-NET:

- According to Table 2, U-STN has three additional layers (10/11/12) of 200/100/50 neurons. Would it be possible to transparently list the number of trainable parameters (e.g., the output of model.summary()) for both the U-STN and U-NET? If they are different, what would suggest that this is a fair comparison?
- For example, how is it possible to know whether U-STN improved performance over the test set comes from the spatial transformer or from the bottleneck created by layers 10/11/12. Would it be possible to introduce a U-NET with the exact same architecture as U-STN (minus the spatial transformer) to test whether the improved performance simply comes from not overfitting the WeatherBench training set?
- "the optimal set of hyperparameters that have been obtained after extensive trial and error": This is a vague statement in the age of formal hyperparameter optimization (e.g., Hyperas, Hyperopt, the Keras tuner, SHERPA, etc.) that does not establish in any way that the architecture of U-STN is comparable to that of U-NET. Would it be possible to:

- List the hyperparameters and their considered range during the manual search?
- Systematically report the performance of U-NET and U-STN over the training/validation/test set (as opposed to just the test set) to identify whether U-STN is a better fit, whether it generalizes better, etc.?
- This could also help better understand the role of the spatial transformer (e.g., does it help because it prevents overfitting the training set?) as equivariance is not generally preserved by the U-STN framework.

1.2 U-STN is not equivariance-preserving

U-STN is still framed as a framework preserving equivariance:
[L4 in the abstract] "to preserve a property called equivariance"
[L44] "based on building physical properties called equivariance"
[L102] "Introducing the equivariance-preserving"
[L338] "to preserve equivariances".
While the authors added a discussion at the end of subsection 3.1.2, the above statements would mislead most readers (especially those focusing on the abstract/introduction/conclusion) into thinking that this framework ensures equivariance preservation. Assuming that the above claims relate to $\mathrm{SO}(3)$ equivariance, which if I am not mistaken would here be defined along the lines of:

$$\forall \boldsymbol{\theta}, \forall \mathbf{Inputs},\ \mathrm{U-STN}\left[\mathcal{T}_{\boldsymbol{\theta}}\left(\mathbf{Inputs}\right)\right] = \mathcal{T}_{\boldsymbol{\theta}}\left(\mathrm{U-STN}\left[\mathbf{Inputs}\right]\right),$$

would it be possible to clarify that:

- The framework is not equivariant because it learns one set of $\boldsymbol{\theta}$ parameters per sample, and that these parameters are not physically interpretable according to the authors (in their response to reviews),
- The U-STN architecture includes **skip-connections**, which prevent equivariance preservation at multiple levels of the architecture (up6 and up7 in the code use conv2 and conv1).

Note that other manuscripts using equivariance preservation (e.g., [1]) typically define equivariance mathematically and exactly stipulate whether the property is satisfied or not upfront.

1.3 Misleading comparison with WeatherBench

[Table3, Sec 6.2]

1. Does the U-STN12 compared to WeatherBench models assimilate data from the test set? In the affirmative, I recommend comparing WeatherBench models to the version of U-STN12 that does not use data assimilation (e.g., the version presented in Fig 6) and clarifying the text accordingly.
2. Are the authors using the same test set (their paper uses 2018) as the test set used to report the WeatherBench models' performance (2017 and 2018 according to [2])? This would also prevent a fair comparison and I recommend using the same test set as in WeatherBench while explicitly acknowledging that samples from the validation set will be used for this particular comparison.

1.4 U-STN is still framed as a physics-informed ML framework

U-STN is still framed as a framework to improve physical consistency:
[Title] Towards physically-consistent
[L3 in the abstract] improve their physical consistency
[L44] provide a framework for [L41-42] incorporating physical constraints into the often physics-agnostic ML models

However, given that U-STN does not preserve equivariance (see 1.2), it is unclear why U-STN would make the predictions more physically consistent (other than the qualitative description in Sec 4.1). If it is impossible to quantitatively establish the improved physical consistency of U-STN predictions, I suggest removing the claims listed above.

2 Minor Comments

2.1 Possible discrepancies between the code and the manuscript

In the accompanying code for the baseline U-NETx, it seems that the authors are still training a U-STN ("stn()") for most of the training files instead of a U-NET ("unet_baseline()"). Would it be possible to explain this apparent discrepancy, which appears in all of the U-NETx ($x = 1\mathrm{hr}, 12\mathrm{hr}$, etc.) scripts?

```
model.compile(loss='mse', optimizer='adam')
model.summary()


batch_size = 10     ### This has undergone HPO. Don't change
num_epochs = 8      #### This has undergone HPO. But less sensitive to change
lead = 1            #### See paper for details on this variable. This lead refers to "x" in U-NETx
count=0
for loop in fileList_train:
    print('******************* counter*************',count)
    File=nc.Dataset(loop)
    Z=np.asarray(File['z'])
    trainN=np.size(Z,0)-300
    Z=(Z-M)/sdev

    x_train=Z[0:trainN,:,:]
    x_train=x_train.reshape([np.size(x_train,0),32,64,1])
    y_train=Z[lead:trainN+lead,:,:]
    y_train=y_train.reshape([np.size(y_train,0),32,64,1])

    x_val= Z[trainN+lead:np.size(Z,0)-lead,:,:]
    x_val=x_val.reshape([np.size(x_val,0),32,64,1])

    y_val= Z[trainN+lead*2:np.size(Z,0),:,:]
    y_val=y_val.reshape([np.size(y_val,0),32,64,1])

    if (count>0):

        model = stn()
        model.compile(loss='mse', optimizer='adam')
        model.load_weights('best_weights_lead1.h5')
        hist = model.fit(x_train, y_train,
                        batch_size = batch_size,
             verbose=1,
             epochs = 20,
             validation_data=(x_val,y_val),shuffle=True,
             callbacks=[keras.callbacks.EarlyStopping(monitor='val_loss',
                                      min_delta=0,
                                      patience=5, # just to make sure we use a lot of patience before stopping
                                      verbose=0, mode='auto'),
                        keras.callbacks.ModelCheckpoint('best_weights_lead'+str(lead)+'.h5', monitor='val_loss',
                                                verbose=1, save_best_only=True,
                                                save_weights_only=True, mode='auto', period=1),history]
             )

    else:
```

Figure 1: The U-NET ("Unet_noSTN") scripts appear to use the stn() model as soon as the model is trained on the second file of the training set.
From `https://github.com/ashesh6810/DDWP-DA/blob/master/Unet_noSTN_lead1.py`

2.2 Typos

[L26-27] "e.g." should be moved to the beginning of the parentheses
[L131] "indicates the Δt that is used": Should e.g., "in units hours" be added to specify the units of Δt in the U-STNx acronym?
[L352] "data-drivenly": Is this grammatically correct or should "in a data-driven fashion" be used instead here?
[L387] "have shown" → "show"

References

[1] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis. Learning SO(3) Equivariant Representations with Spherical CNNs. *International Journal of Computer Vision*, 128(3):588–600, 2020.

[2] S. Rasp, P. D. Dueben, S. Scher, J. A. Weyn, S. Mouatadid, and N. Thuerey. WeatherBench: A benchmark dataset for data-driven weather forecasting. feb 2020.