Reply to reviewer 2: DINCAE 1: multivariate convolutional neural network with error estimates to reconstruct sea surface temperature satellite and altimetry observations

We thank the reviewer for the careful reading of the manuscript and the constructive criticisms. We copied the reviewer's comments below and our responses are in bold.

The paper presents an update on the previous version of DINCAE, a convolutional autoencoder method for in-painting of sparse satellite data. DINACE2 presents some improvements over the previous version, most notably in performance (vs DINCAE1), speed (presumably due to being rewritten to Julia from Python) and an option to treat ungridded data like satellite altimetry observations. It also introduces an extra refinement step in the cost function to increase its depth, and an intermediate loss term is included in the total loss to compensate for the vanishing gradients of the deep network. When treating sparse data, the error variance estimation of DINCAE2 is more reliable than that from variational interpolation method DIVAnd. The results are solid, the paper is well written, the figures are clear. I recommend publication after minor revision. I do have some comments which might be worth discussing further.

Specific comments

- page 4, eq4: In the comment to equation 4, the authors state that CAE refinement leads to a deeper network and thus to potential worsening of the vanishing gradient problem. They attempt to mitigate this by including intermediate loss term into the total loss function. They state that by doing so, the vanishing gradient problem is reduced. Can the authors perhaps illustrate more clearly that this is indeed the case? Is there a way to say a bit more about this, so that the reader does not have to take the author's word for this?

  We added the following to the manuscript to clarify this question about the vanishing gradient:

  With a refinement step, the neural network becomes essentially twice as deep and the number of parameters (approximately) doubles. The increased depth would make it prone to the vanishing gradient problem. However, by including the intermediate results in the cost function this problem is reduced. In fact, the information from the observations is injected during back-propagation by the loss function. Due to the refinement step and the loss function which also depends on the intermediate result, the information from the observation is injected at the last layer and at the middle layer of the combined neural network (Szegedy et al., 2015). The relationship between the first layers of the neural network and the cost function is therefore more direct, which helps in the training of these first layers.

- Also, wouldn't an arbitrary number of refinements further exacerbate the vanishing gradient problem? Which term would be dominant in this case – adding further refinement steps versus including further intermediate losses to the total loss?

  It is important to note that for every refinement step an additional term must be added to the total loss. The number of refinement steps and terms in the loss function cannot be varied independently. In simple terms, the vanishing gradient problem can be expressed by the distance (counted as the minimal number of intermediate layers) between each layer and the loss function. For regular deep neural networks (without refinement), this distance would increase in average with the number of layers. However, this distance would be kept constant if additional refinement steps were added. In practice, adding

a refinement step would double the amount of required GPU memory (two additional refinements, triple the amount of GPU memory,...). Regarding in-painting applications, we are only aware of applications with a single additional refinement step (Liu et al., 2019) while for image classification the loss function of the Inception network provides class labels at three different steps of the network, which is conceptually similar to the refinement approach used in our work (Szegedy et al., 2015).

- Page5: when handling missing data there is an interpretation throughout the text that setting the missing value to zero corresponds to an infinitely large error. This is undoubtedly true for variables which are normalized by their variance, as those in this paper. However there are a number of other scalings where variables are not normalized by their variance. In these cases, it seems to me, the authors interpretation is not the most appropriate. I would propose an independent interpretation that setting the missing values to zero simply numerically means that there will be no back-propagation of error from those missing data – thus the training can continue without any impact from the missing data. This interpretation does not have anything with the specific variable normalization at hand.

  For the output value of the network, one can indeed ignore the missing values e.g. on land grid cells (this is also done here) and their associated value (0 or anything else) would not be used to update the network when computing the gradients via back-propagation. However, the more difficult aspect is how to treat missing values in the *input*. If the inputs are normalized, then a zero input value could either be the average or a missing value. To disambiguate between the two cases, we think it is important to provide an additional field which could either be a binary mask or the inverse of the error variance (both would be equivalent if the error variance of present data is assumed constant, as it is the case here). Other scales might be possible leading to a different interpretation and handling of missing value, but we found that the present one is the most natural from a data assimilation perspective when using Gaussian distributed errors (as discussed in Barth et al. (2021), equation 1 and 2). Once the scaling is chosen, the interpretation of missing values follows from the scaling. We clarified in the manuscript that our interpretation is specific to the chosen scaling.

- A cosmetic remark. The hyperparameters were tuned using Bayesian optimization, which seems adequate. Let's say hyperparameter optimization gives you an optimal network. Separate instances of training this same optimal network (with the same fixed optimal hyperparameters) would provide separately trained versions of this same network. We can use these set of the same network to create a set of predictions. What is the error variance of this set of predictions? I would expect that this error variance is on the order of a 5

  We agree with this comment and reduced the number of decimals in Table 2. The revised table now has only 2 decimals. This way we could also more easily add more information in this table as asked by the other reviewer.

References

Barth, A., Troupin, C., Reyes, E., Alvera-Azcárate, A., Beckers, J.-M., and Tintoré, J.: Variational interpolation of high-frequency radar surface currents using DIVAnd, Ocean Dynamics, 71, 293–308, doi: 10.1007/s10236-020-01432-x, 2021.

Liu, H., Jiang, B., Xiao, Y., and Yang, C.: Coherent Semantic Attention for Image Inpainting, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 4169–4178, doi:10.1109/ICCV.2019.00427, URL http://openaccess.thecvf.com/content_ICCV_2019/html/Liu_Coherent_Semantic_Attention_for_Image_Inpainting_ICCV_2019_paper.html, 2019.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A.: Going Deeper with Convolutions, in: Computer Vision and Pattern Recognition (CVPR), URL http://arxiv.org/abs/1409.4842, 2015.