Review of JAMES: 2020MS002324

Spherical convolution and other forms of informed machine learning for deep neural network based weather forecasts (Scher and Messori)

Summary: This paper joins the growing literature on development of purely data-driven modeling solutions for medium-range global weather prediction, specifically tackling the problem laid out in WeatherBench (Rasp et al. 2020). It seeks to fill an important gap in the current literature by using modified convolutional architectures adapted specifically to data on a sphere. The results suggest that improvements can be made by using these adapted deep learning methods relative to vanilla CNNs. The analysis of forecast bust events is also an interesting approach towards user confidence in machine-learning weather forecasts.

While with some significant modifications and improvements this manuscript has potential to contribute to ML-based weather forecasting and be of interest to JAMES readers, unfortunately in its current form this paper has several fundamental flaws that this reviewer believes will require more time and work to address than a major revision decision. These issues are detailed in the major comments below. In short, I believe that the authors need to fix their algorithm implementation and train their models for much longer. These changes would make this work much more convincing and impactful.

Recommendation: Reject now, but invite to re-submit in the future.

Major comments:

1. The main contribution of this paper is to apply sphere-aware adaptations within convolutional neural networks to improve weather prediction performance. However, the algorithm used in the paper, modeled after the SphereNet developed by Coors et al. 2018, is based on a fundamental misinterpretation of the latter. The key in SphereNet is that, for each convolutional filter location, the data are interpolated onto a local gnomonic grid, ensuring that each 3x3 box of data the filter sees covers an equal area of the globe. The authors apply only a cosine transformation to correct for the increased spread in longitude at high latitudes, justifying this different transformation accordingly:

   > "The original method was tailored for the gnomic projection. Since our data is on a regular grid, we changed the interpolation accordingly."

   This is incorrect, as the SphereNet also used equirectangular data as input, but *mapped to* a gnomonic projection. The authors also state that they "allow only multiples of the gridpoint distance at the equator as points in the latitude direction." This is also not a correct assumption, as while lines of constant longitude are great circles, once the longitude is interpolated to different points the edges of the convolutional filter area no longer are along great circles without also interpolating the latitude. Fundamentally then, by not using the correct approach in SphereNet, the implementation in the manuscript does not actually solve the problem of the violation of translational invariance when applying convolution operations on equirectangular grids.

2. The key results are highly questionable at best, and possibly incorrect at worst. How could a model which corrects the boundary conditions at the poles degrade in performance over lead time relative to the basic model which does not? This suggests either there is an error in implementation or the models are not sufficiently trained (more on the latter in the next point). Why is there still such a large deficit relative to prior work of Weyn et al. 2020? In order to justify that the "spherical" nature of the authors' algorithm is responsible for improved performance, particularly near the poles, I think a more appropriate comparison could be made against a "base" model which implements correct padding at the poles (i.e., the values 180º across). (This should not be difficult to implement, and it is a shame that Weyn et al. 2019 did not consider this approach.) Based solely on the fact that the spherical models should have much better treatment of the poles (which is not specific to their spherical nature) I would expect to see them performing quite a bit better at longer lead times.
3. The models presented in the paper simply do not appear to be sufficiently trained, especially for the high resolution data. A few points of evidence to support this hypothesis:
    a. Comparable models such as Weyn et al. 2020 trained for a minimum of 100 or more epochs, whereas this paper uses a maximum of 54 (30) in the low (high) resolution case.
    b. As noted before, a model with better geometry simply should not be performing worse at longer lead times.
    c. One consistent observation is that between nearly-identical architectures, the ones with shared weights do better than their non-shared counterparts. Can the authors rule out the simple explanation that there are fewer learnable parameters in the "shared" model and therefore it is easier to train? The extreme example is the dense-shared model. This model doesn't change the geometry but increases the number of parameters by a factor of >2. It does considerably worse, to be expected with insufficient training.
    d. The networks quickly diverge from reality when run for longer time. Weyn et al. showed that it should be possible to get realistic performance.

    I do understand that the experiments can be computationally expensive. But it is very important in any deep learning problem to be able to demonstrate that the neural network architecture has saturated and that a suitable loss minimum has been found. The authors should investigate training much longer, hopefully encountering an over-fitting regime, before drawing conclusions about the architectures. And incidentally, since the authors claim that they have trained on higher-resolution models than prior studies, it's doubly important to show that the model has sufficiently learned to use this higher-resolution information. I think it's important to address the problems listed above on low-resolution data before moving to higher resolution.

Other comments: these are comments I would like to see addressed in a future submission.

1. There is a rich literature on adaptation of CNNs to spherical (omnidirectional, 360º) data. A few papers are listed below. The authors should acknowledge these methods and justify why they chose their particular approach. I would suggest that “spherical convolution” more correctly refers to convolutions on data that have actually been transformed to spherical harmonics (Cohen at al. 2018), and therefore the terminology should be clarified in this paper.
    a. T. S. Cohen, M. Geiger, J. Koehler, and M. Welling. Spherical CNNs, 2018.
    b. W. Boomsma and J. Frellsen. Spherical convolutions and their application in molecular modelling, 2017.
    c. M. Eder, T. Price, T. Vu, A. Bapat, and J.-M. Frahm. Mapped Convolutions, 2019.
2. The authors misinterpret the algorithm of Weyn et al. 2020, namely the sharing of weights across cube faces. There are two sets of weights: one for the equatorial faces and one for the polar faces:

    > “DLWP learns separate weights and biases for the four faces centered on the equator and the two polar faces. Using one set of CNN weights for the equatorial faces and another for the poles...”

    The first sentence is vague but the second makes it clear the weights are shared across the equator.
3. Adding angle transformations of cyclic inputs is a great idea, but I don’t understand why both sin and cos are used. Since they are just phase shifted, no additional information is given to the algorithm by including both. And since the importance of the hour is for the model to capture the diurnal cycle, wouldn’t it be much better to use local solar time to standardize across the globe?
4. Please improve clarity of the figures in general. Complete and detailed captions which do not require reading the text, labeled axes with units (e.g. time), uniform colorbar intervals, larger text, etc.