Deep Climate
Using CNNs to Fine-Tune Climate Ensembles

Brian Reed

***Abstract*—While climate models have significant predictive power over multidecadal, regional scales, they become less accurate when trying to make predictions over smaller temporal or spatial scales. Here we explore using a convolutional approach to combine the predicted temperatures from an ensemble of climate models in a way that better captures observed spatial variability. Our model is based on the u-net architecture, and in this first pass, we use daily hindcast and observed temperature from 1900-2005, though we explore ways to increase the temporal scale. We find that the convolutional approach greatly reduces the mean-squared prediction error when compared to a more traditional approach, averaging the predictions of the models in the ensemble.**

I. Introduction

The scientific community is in agreement that human emissions of greenhouse gases are forcing the Earth out of the historical range of climate variability, into a generally hotter and wetter world. These results are robust when averaged over large spatial and temporal scales, but the global, multidecadal trends mask a certain amount of noise. It is one thing to robustly forecast global trends, and another to predict how the changing climate will impact conditions in specific locations, or how it will impact specific Earth system processes, or even how it will change intra- and interannual temperature and precipitation patterns in the short term. For this reason, two main areas of research in the earth systems modeling (ESM) community are near-term and regional climate prediction. These are both notoriously difficult, yet crucially important for policymaking.

In this paper, we explore a convolutional ensemble approach to making short-term, regional climate projections more spatially accurate. We take as input the predicted daily maximum temperature from three climate models [2], normalize the input data for each day, feed this into a u-net architecture, and try to predict the normalized climate temperature in each pixel of the image[1]. In this initial application, we focus on a region that includes most of the landmass of the continental United States, as well as parts of Northern Mexico and Southern Canada. North America.

While the results from these climate models are typically averaged over longer periods of time, on the order of years to decades, we focus on daily images in order to get the most information on the spatial biases of the model. However, we begin to explore an approach to look across longer time horizons as well.

We find that our approach greatly reduces the error relative to the common practice in ensembling, which involves taking an average over different model inputs. It is worth further exploring the extent to which this approach can be used to fine-tune climate predictions.

II. Related Work

The classic ESM approach involves using fluid and thermodynamic equations to predict future states of the atmosphere, which can be translated into such variables as expected surface temperature and precipitation. In order to gain a sense of the range over different initializations and parametrizations, climate scientists create an ensemble of results from different model runs. One such type of ensemble, which we draw from here, is a multi-model ensemble, which compares results from a collection of different models, often from different modeling centers, in order to get a sense of the variation in results among the modeling community [5]. There is no standard procedure for using these ensembles to generate a single forecast, though researchers have experimented with calculating weighted averages and selecting the model that most accurately replicates the past.

In contrast to the classic ESM approach, individuals with expertise in statistics, computational math, and computer science have begun to harness the growing body of observational data on the Earth system to gain insights into how the Earth system functions. Due to the rise of remote sensing technologies, there is an ever-increasing amount of said data. Researchers have used it to investigate such tasks as long-term forecasting [11] and statistical downscaling [9], [10]. There has even been discussion of creating a global ESM strictly from data [7], although in order for such a model to be accepted in the ESM community, it would likely have to meet several boundary conditions prescribed by laws of thermodynamics and fluid mechanics.

In this work we seek to bridge these two approaches, using a u-net architecture to improve the spatial accuracy of an ensemble of existing climate models.

III. Overview of the Dataset

A. Data Sources

The historical model hindcasts come from global circulation models compiled for the Coupled Model Intercomparison Project (CMIP5), while the historical observational data comes from Berkeley Earth[2][1]. The CMIP models are the gold standard for climate models, as they form the foundation for the projections in the Assessment Reports put out by the UN-founded Intergovernmental Panel on Climate Change (IPCC). We use the CMIP models developed by NASA GISS, Environment Canada, and the Community Climate System. While the Earth Systems Grid Federation (ESGF) provides a centralized repository for the model output, different data centers still put their data in slightly different formats, so some standardization is needed to compare results across models. We make use of wget scripts provided by the ESGF to download the data for a collection of models from 1900-2005, while we download the observational data directly from Berkeley Earth's website. The files are in a .nc format and can amount to several gigabytes of data from each data source over the entire time period.

B. Data Processing & Input Architecture

After downloading the relevant .nc files, we crop the data spatially and temporally to focus on the continental US from 1900-2005. For each day, for each data source, we generate a map of the estimated daily maximum temperature over the continental US and save this as an image. For each day, we demean the data and divide by the standard deviation of the observed temperatures within the given window. This should help to remove some aspects of seasonal variations, as well as year-to-year changes in temperature. In future work, it would be interesting to directly predict temperature. For now, though, the standard deviation approach provides insight into how well the u-net approach can help reduce spatial biases.

We convert each image to grayscale and stack the results from different models as channels in one image. Here we experiment with two different architectures as pictured in Figure 1.

Architecture 1: We stack the results from the three models for a given day into one image and try to predict the one day observed temperature.

Architecture 2: We stack the results from the three models for 5 days and try to predict the 5 day moving average temperature.

After processing and stacking the data, the resulting ground truth images are $102 \times 52 \times 1$, while the model images are $102 \times 52 \times (m \times d)$, where m is the number of models (here, 3) and d is the number of days. The images are this small in order to facilitate running a model with more than 30,000 of them. While the pixel size may seem relatively small, the model outputs themselves are 1 degree cell by 1 degree cell, so we do not cut down on the resolution of the models. Sample GCM outputs, which are the inputs to our u-net model, are below.

C. Train, Validation, and Test Sets

We start with 1 image for 365 days across 105 years.[1] We shuffle the data from 1900 to the end of 1999 across all time periods before splitting into a training set, consisting of 32,850 images, and a validation set, consisting of 3,650 images. We set aside the data from 2000-2005, consisting of 2,190 images, for the test set. This split is captured in Figure 2.

[1]The climate models do not include leap years, so we remove the leap days from the observational data set as appropriate.

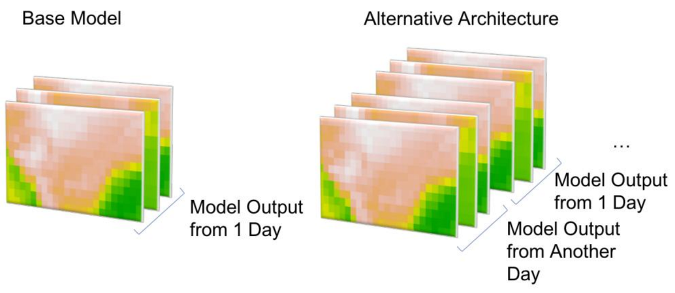

Fig. 1. Sample of different architectures explored in the model.

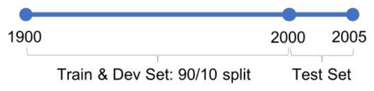

Fig. 2. Train, validation, and test set split.

Although this set-up means that that our test set is from a different distribution than our training and validation sets, it allows us to mimic the classic goal in climate research: prediction. In this set-up, the data from 1900-1999 reflect what we know about past climate, while the data from 2000-2005 serves as our prediction period.

IV. Methods

To combine the model inputs, we use a simplified version of a u-net convolutional architecture. The architecture was originally produced for biomedical image segmentation [6]. The classic u-net architecture consists of a sequence of convolutions followed by a pooling layer, repeated several times, followed by a sequence of up-convolutions and convolutions to expand the output and recover a segmentation map. For our purposes, the key is that its output can be visualized as an image that is on a similar scale as the input images. This will allow us to map the model output and visually compare the results across with both the ground truth and our input images.

Our implementation relies on the Unet-ants repo created by Nate Cullen [8].[2] The general architecture of the unet implementation I run is below in Figure 3 below, showing sample inputs and output images as well. This is a take on the representation in the original paper by Ronneberger, [6], where here we represent each layer as the layer type.

We try different model architectures by altering the number of days that we include in an input image, as discussed previously in Section III-B.

The loss function we use is the MSE, given below. In

[2]This repo relies on keras[3] and uses a tensorflow backend [4].

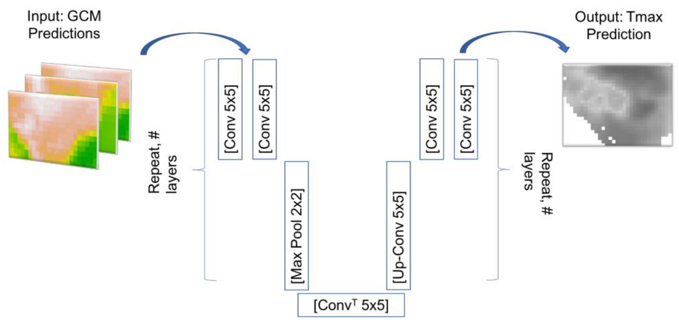

Fig. 3. Simplified representation of our model, with the middle representation of u-net based loosely on the depiction in [6].

general, it sums over every pixel $j \in J$ the difference between the predicted pixel, $\hat{y}_j$, and the ground truth pixel, y_j. Here we have a number of samples m as well, over whatever set we are training.

$$\mathcal{L} = \sum_{i=1}^{m} \sum_{j=1}^{J} (y_{ij} - \hat{y}_{ij})^2$$

This function allows us to minimise the pixelwise squared difference between images, so that we penalize over- and under-predictions equally. We will also look maps of the sum of the straight difference between the ground truth and the prediction over our test set, to look at whether the model routinely over- or under-predicts values in given regions.

We train the model using an Adam optimizer and we experiment with several learning rates, as discussed below.

V. Experiments

In order to expedite the training and tuning process, we simultaneously run the model on three p2.xlarge instances in Amazon Web Services, and we conduct several runs on a p3.2xlarge as well.

In our initial search, we experiment with 3 learning rates across what we will refer to as a "1 layer" model and, separately, a "2 layer" model. In this particular case, the "number of layers" determines how many times the model iterates through the layers as pictured above in Figure 1.

When we set the learning rate at 0.1, we find that the model converges to an MSE within 1 epoch and then stay there. Specifically, the model settled on a training loss of 1.0202 in the 1-layer model and approximately 0.8 for the 2-layer model. If we wanted to keep an initial learning rate of 0.1, perhaps to speed up the learning process at the very beginning, we could experiment with learning rate decay, but we find promising results with other hyperparameter settings.

We present the training and validation loss curves for the remaining learning rate - layer combinations in Figure 4 below, making a slight modification to the 2 layer model with a learning rate of 0.001. For this model, within just a couple of epochs we see that the training rate error is decreasing dramatically while the validation error is increasing, so we added a regularization to prevent overfitting. In the preliminary run pictured in Figure 4, we added L2 regularization with a small penalty of 0.001.

While the training loss in the 1 layer models looks to

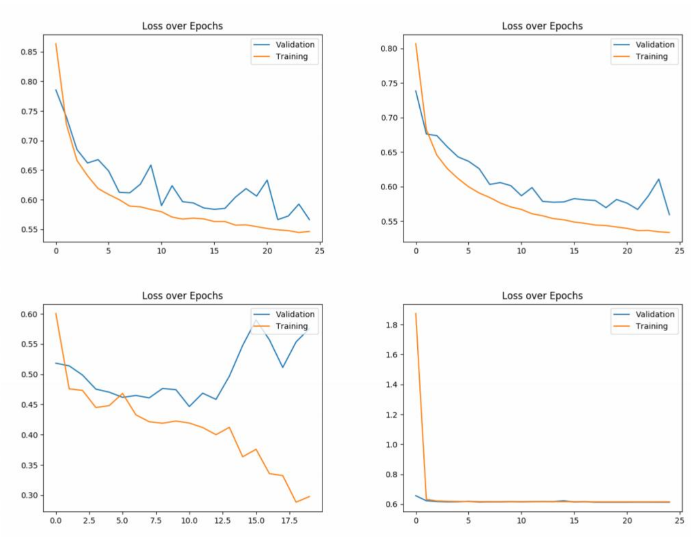

Fig. 4. Clockwise from top left: MSE in 1 layer model with learning rate of 0.01, MSE in 1 layer model with learning rate of 0.001, MSE in 2 layer model with learning rate of 0.01, and MSE in 2 layer model with learning rate of 0.001 and L2 regularization, penalty 0.001.

be on a decreasing trend even through 25 epochs, the validation loss has itself more or less stabilized by around the 15th epoch for both learning rates, at values between 0.55 and 0.6. If we look at the losses of the 2 layer model with a learning rate of 0.01, we see a sharp initial decrease in the training loss from about 1.8 after the first epoch to 0.632 after the second. Both the training and the validation losses stay roughly constant over the next 23 epochs, with the final validation loss being 0.616 after the 25th epoch.

Overall, we see that the lowest training and validation loss was achieved in the 2 layer model with a learning rate of 0.01 and L2 regularization with a penalty of 0.001. The validation MSE of this model reached a minimum of 0.47 after the 10th epoch. If we needed a model that is relatively simple and quick to run, though, the validation MSE of the 1 layer model with a learning rate of 0.001 reached a minimum of about 0.57.

Given that the 2 layer model with a learning rate of 0.001 has the lowest validation error rate, we focus on this model and experiment with different settings for the penalty function. In addition to the penalty weighting of 0.001 pictured above, we test penalty weights of 0.005 and 0.001. Below, we provide the loss curves over 4 epochs for the penalty of 0.005 and 20 epochs for the penalty of 0.001. We can see a leveling off in the validation loss over runs of these lengths, though in a less-time constrained world, we may have let the models run for additional epochs to verify these were the minima.

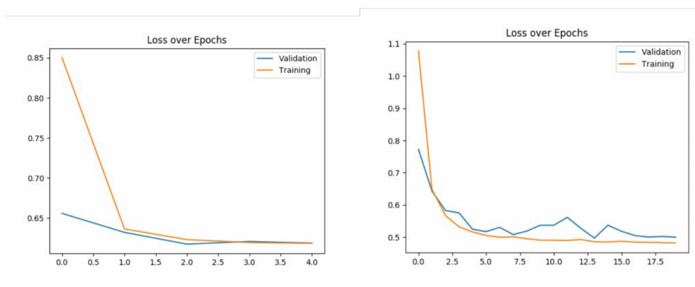

Fig. 5. At left: MSE in 2 layer model with learning rate of 0.001 and L2 regularization, penalty 0.005. At right: MSE in 2 layer model with learning rate of 0.001 and L2 regularization, penalty 0.005.

After this hyperparameter tuning, we proceed to analyze the results for our 2 layer model with a learning rate of 0.001 and a penalty of 0.01 in the L2 regularization.

In implementing our 5 day model, we use the same hyperparameter settings as in the 1 day model. With additional time, we would further tune the hyperparameters on this alternative architecture. The loss curve for the 5 day model is below.

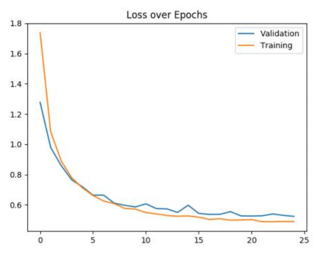

Fig. 6. Using the 5-day model, MSE in 2 layer model with rate of 0.001 and L2 regularization, penalty 0.01.

VI. Results

We run the 1 day and 5 day models using the learning rate of 0.001, the L2 penalty of 0.01, and 2 "layers". We run the model for 7 epochs, as our validation loss reached its minimum at this number of iterations. We then use these models to predict the daily normalized temperature for every day over our test set, and we find our training and test errors. We compare these results with one practice in climate ensembling, which is to take the average over a span of models.

Results - Total MSE

The MSE for our u-net implementation is considerably lower than the MSE for the average of the ensemble. It is interesting to note that the ensemble approach performs better in its 5 day setting, while the CNN approach does better in the 1 day approach. It makes sense that the ensemble approach would do better in this set-up, as we are taking the average over a few days and reducing the noise in the predictions. One potential reason why the u-net approach may be faring worse is that in this current implementation, our sample size is effectively smaller in the 5 day approach as compared to the 1 day approach. Specifically, we run the model using discrete 5 day moving averages. This means that for every 5 days of data that we start with, we get 1 data point in the 5 day model and 5 data points in the 1 day model. In the future, it would be possible to retrain the model with a sliding 5 day moving average to preserve the size of the dataset. Regardless of the differences in performance in the 5 day and the 1 day set ups, it is still the case that the u-net model outperforms the simple ensemble average.

Model	Training MSE	Test MSE
Ensemble, 1 Day	1.85	1.72
CNN, 1 Day	0.434	0.402
Ensemble, 5 Days	1.78	1.65
CNN, 5 Days	0.495	0.418

Further, it is interesting to note that the model performs slightly better on the test set than on the training set, for both the ensemble and our u-net implementations. Though it is worth further exploring the reasons behind this, it is possible that there was less observed variability over the first 6 years of the 2000s, and that the climate hewed closely to its long-term average.

Results, Spatially

In addition to looking at the overall MSE, we can learn more about what, exactly, the model is doing if we plot the results spatially. Here we will look at the pixelwise loss in the form given below, which allows additional insight into whether the model systematically over or under predicts temperature a given region. We find this loss over our out-of-sample test set, from the start of 2000 through 2005.

$$\mathcal{L} = \sum_{j}^{J} \sum_{\text{t = 01/01/00}}^{\text{12/31/05}} \hat{y}_{jt}^{\text{model}} - y_{jt}$$

In Figure 7, we plot the error for our models over the test set. We also provide for a baseline the ensemble averages.

In these images, red indicates that the models overpredict the observed temperature, blue indicates that they underpredict the observed temperature, and white indicates that the error, on net, was minimal. In terms of the land-based temperatures, which are more of interest to policymakers and the general public, it is clear to see that u-net model on net reduced the biases of the ensemble. However, we see that the patterns of errors are carried over from the ensemble approach to our u-net approach: there are visible edges that carry over from the ensemble approaches into the u-net approach. This is the case even in cases where the sign of the error was different in the u-net approach as compared with the ensemble approach, for instance in the Northwest corner of the US.

It is clear that the error of the ensemble is particularly

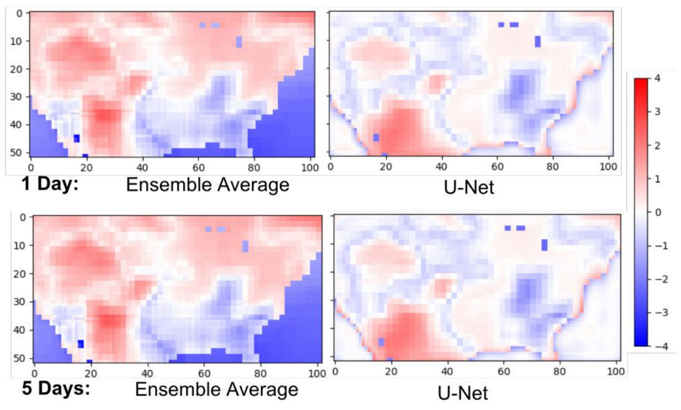

Fig. 7. Pixelwise error for the ensemble average and our u-net approach, summing over the errors each day from 2000 through 2005. The units are in normalized temperature.

extreme over the oceans. For our purposes, this is not of much concern, as we are mostly interested in predicting temperatures that will impact human populations. In future iterations, we may want to remove over-water predictions from our training and test sets to focus more specifically on our regions of interest.

Interestingly, the regional distribution of errors seems to track roughly with ecological zones. For instance, the area that is most "over-predicted" in the ensemble approach, as well as in our u-net approach, centers around the southwestern US and stretches into Northern Mexico. This is a region that is largely comprised of desert. Similarly, the area that is most under-predicted in the ensemble as well as in the u-net is in the southeastern United States, which historically has been comprised of forests. Land cover does impact climate processes, so it is possible that the climate models, in focusing on atmospheric effects, do not fully account for the effects of different ecosystems.

Finally, in comparing the 1 and 5 day models, there appear to be only slight differences in the shades of individual pixels. It may be the case that in order to see more dramatic differences, we need to expand the temporal horizon and include data from additional days.

VII. Conclusion and Next Steps

We find that there is significant promise for using a convolutional approach to climate ensembling in place of computing an average. In our implementation, our best performing model features 2 passes through the layers in our simplified u-net architecture, with a learning rate of 0.001, and L2 regularization with a penalty of 0.01. The loss function is MSE, and we use an Adam optimizer to train our results. We experiment with two architectures, one that uses 5 days of input to predict a five day moving average, and one that uses 1 day of input to predict a single days' average. We find that there was minimal difference between our 1 day and our 5 day models, though both models have significantly lower errors that than the ensemble averages over, respectively, 1 and 5 days.

When we plot our errors spatially, we find that the u-net and the traditional ensemble are both particularly poor at predicting temperature in the Southwest and the Southeast of the United States. Although the u-net approach clearly reduces overall error, we see that the regions that are mispredicted bear similar patterns in the u-net approach as in the ensemble, though the sign of the error may be different.

Given the performance improvements that the convolutional approach seems to make when compared to a more classic average, it is worth considering directions for future research. These include the following.

Direct Prediction of Temperature

In this first pass, we aim to try to minimize the effects of seasonal variation and long-term climate trends by demeaning the data and looking at spatial variation. However, it would be relatively straightforward to convert this model into one that predicts temperature directly. The results from this would be more interpretable and more directly relevant to policy discussions.

Experimentation with Different Time Horizons

While climate models are usually run over periods of a few years to a few decades, our approach has used daily data in an attempt to gain insight into spatial biases in the model. It is worth further expanding the number of days of data, beyond 5, that are included within one stack of input data, to see how well this approach works with longer-term trends.

Data from Additional Models

In this first pass, we have included data from 3 separate runs of different climate models. However, there are tens of these centers all over the world, and each publishes results from model runs with slightly different parametrizations and initializations. This means that we could dramatically expand the number of models that we are including as inputs to our convolutions. This should only improve the accuracy of our predictions, hopefully allowing us to further decrease the errors over the Southeastern and Southwestern United States.

Looking at Different Filters for Different Times of Year

There is undoubtedly significant seasonal variation in climate. It would be interesting to see if we could further reduce errors by splitting our dataset and trying to run separate models for separate times of the year, for instance running separate models for summer and for winter. The main downside to this would be that we would have only a fraction of the data with which to train the model for a given season.

Increasing the Complexity of the Model

This implementation represented a simplification of the standard u-net approach, so it may be worth experimenting with an architecture that is more directly aligned with [6].

REFERENCES

[1] Berkeley Earth, 2018. http://berkeleyearth.org/data/

[2] https://esgf-node.llnl.gov/search/cmip5/. Unfortunately the site was down for maintenance when I went back to pull citations, but I used the r5i1p3 scenario run from CanESM2, the r6i1p1 run from GISS-E2H, and the r1i1p10 run from CCSM4. These are all historical runs, focusing on tasmax.

[3] Kotikalapudi, Raghavendra and contributors, 2017. GitHub. https://github.com/raghakot/keras-text.

[4] Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vigas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[5] Parker, W. S. (2013). Ensemble modeling, uncertainty and robust predictions. Wiley Interdisciplinary Reviews: Climate Change, 4(3), 213223. http://doi.org/10.1002/wcc.220.

[6] Ronneberger, O.,Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. Retrieved from http://arxiv.org/abs/1505.04597.

[7] Schneider, T., Lan, S., Stuart, A., & Teixeira, J. (2017). Earth System Modeling 2.0: A Blueprint for Models That Learn From Observations and Targeted High-Resolution Simulations. Geophysical Research Letters, 44(24), 12,396-12,417. http://doi.org/10.1002/2017GL076101

[8] U-net ants, GitHub Repo. https://github.com/ncullen93/Unet-ants

[9] Vandal, T., Kodra, E., & Ganguly, A. R. (2017). Intercomparison of Machine Learning Methods for Statistical Downscaling: The Case of Daily and Extreme Precipitation, 120. Retrieved from http://arxiv.org/abs/1702.04018

[10] Vandal, T., Kodra, E., Ganguly, S., Michaelis, A., Nemani, R., & Ganguly, A. R. (2017). DeepSD: Generating High Resolution Climate Change Projections through Single Image Super-Resolution, 4. http://doi.org/10.1145/3097983.3098004

[11] Yu, R., Zheng, S., Anandkumar, A., & Yue, Y. (2017). Long-term Forecasting using Tensor-Train RNNs. Retrieved from http://arxiv.org/abs/1711.00073