Manuscript Title

This manuscript (permalink) was automatically generated from ccherry2/earthsystemsmodel2@f019252 on December 6, 2020.

Authors

- **Charlotte Cherry**
  · ccherry2
  Department of Civil Engineering, University of Illinois Urbana-Champaign

- **Joyce Yang**

  Department of Civil Engineering, University of Illinois Urbana-Champaign

- **Yiwen Zhang**
  · Yiwen-Zhang97
  Department of Civil Engineering, University of Illinois Urbana-Champaign

Abstract

II. Introduction

Urban areas take up a relatively small percentage of the Earth's land cover but have disproportionately large impacts on the climate and on humans. They are major drivers of emissions and climate change, serve as economic and social centers around the world, and house most of the human population. However, the very nature of their small physical footprint makes it challenging to study their impact on humans and the environment accurately and comprehensively.

To grapple with uncertainty and develop climate change mitigation and/or adaptation strategies, it is crucial for policymakers and planners for urban areas to understand different climate projections and scenarios, as well as urban-specific dynamics. Earth Systems Models (ESMs) are complex mathematical models that produce climate projections. They represent physical processes in the atmosphere, ocean, cryosphere, biogeochemical cycling in terrestrial and marine ecosystems, and interactions and feedbacks between these domains. These models are highly computationally demanding – taking long amounts of time and storage capacity to run over timescales of hundreds of years. This can make ESMs impractical for many uses, particularly policy analyses. Interested users may not have the capacity to utilize GCMs on a typical computer or reasonable budget.

Currently, most state-of-the-art ESMs used today for climate change projections do not explicitly parameterize urban areas, largely due to their small area. While this does not significantly impact the quality of regular or large-scale studies, this lack of explicit parameterization limits our ability to adequately capture unique urban characteristics and dynamics. The Community Earth Systems Model (CESM) is one of the few state-of-the-art ESMs that explicitly parameterizes urban areas, with the latest version even distinguishing between three separate urban density classes. Most quantitative attributions have been typically done for non-urban surfaces, but effective development decisions and local actions to manage risks rely on robust urban climate projections. This is the motivation for us to use CESM, which has a representation of urban areas, to build a location dependent emulator, and apply it to other ESMs in order to get their urban temperature responses.

While CESM provides the advantage of explicit urban parameterization, it does still require significant supercomputing resources, which may limit its usefulness. Therefore, there is a desire to use artificial intelligence to reduce this load. A climate emulator could achieve this by statistically replicating the nonlinear behavior of ESMs more quickly and with less computing power. This project will use machine learning methods to develop a model that can emulate urban temperatures (using other atmospheric forcing variables), where the risk of heat waves in the future could have the greatest negative impacts on human health. This model will be loosely based off of the conceptual framework presented by Zhao et al (2020) - "Global multi-model projections of local urban climates" (currently in press). Urban areas in this dataset refers loosely to areas where people live (i.e., not oceans or uninhabitable areas) but they do not exclusively correspond to cities.

III. Methods

A. Exploratory Data Analysis Findings

B. Preprocessing Data for Model

C. Model Testing

1. Neural Network

2. Random Forest (Selection of Variables)

3. Random Forest (Minimized Variables)

IV. Results

A. Overview

Overall, the random forest models tended to do better than the neural networks that we tried, however, model performance (quantified as mean absolute error) was highly dependent on feature selection. We had to keep in mind two main goals as we developed our model. On one hand, we wanted to reduce the mean absolute error, so the model could predict urban temperature with the greatest accuracy possible. This would improve the usefulness of the model for urban planners, policymakers, and other stakeholders who could use results from this emulator. However, the main motivation behind this project was to build a model that would be easily adapted to run on other earth systems models participating in CMIP6. For this emulator to be useful for that, it would need to be easily adaptable to a wide variety of models. During model development, this manifests in decisions such as variable selection – using uncommon atmospheric variables would reduce the number of models this emulator could be used on, thus decreasing the adaptability. Considering the balance of these two goals (accuracy and adaptability), we drew from each of our individual models to synthesize the "best model." An outline of this model will be described below.

B. Preprocessing

We individually arrived at many common preprocessing steps, which will be incorporated into the "best model." We found that it was necessary to drop rows where TSA values were "NaN," in order to build the model at all. The time was converted into datetime format and processed in some way: either making a new column for the year or converting day and year information using trigonometric functions (to better represent their cyclical nature). The training data was split into consecutive training and validation samples, without random shuffling. This improves the robustness of the model by testing it on the last time intervals of the dataset. We also found that it was important to fill in NaN values in the feature columns – if we dropped all rows with any NaN values, this would result in a greatly diminished training dataset. We found that sklearn's "Simple Imputer" worked relatively well by replacing missing values with the mean value for each feature. Finally, we normalized the data using the mean and standard deviation of the training data, since the features had a wide range of values and magnitudes. Further model development could investigate the reason for these TSA = NaN values. These values could be NaN due to issues in the conversion between netcdf and csv, an issue with the original file itself, issues in the extraction of urban grid cells, a characteristic of the simulation itself, etc. Based on the reason, these rows could either be dropped, or the missing values in the training data could be imputed.

C. Model Development

For our best model, we would recommend using a random forest machine learning model. Our hyperparameter tuning revealed that the number of trees (n_estimators) had the greatest impact on model error. We were able to achieve optimal results with 300 estimators. The model was not as sensitive to other hyperparameters, such as the maximum leaf nodes or minimum number of samples to split. Not bootstrapping did lead to poorer model performance. With greater computing resources, we would recommend trying a K-fold cross validation along with grid search to truly optimize hyperparameters. However, we found that most of the default hyperparameter settings

worked relatively well for our model. Another aspect of model development that strongly influenced performance was variable selections. Our approaches ranged from using only seven common atmospheric variables to using over 40 of the feature columns. In order to improve the adaptability of the model, we did not use all of the features provided, since some of those were outputs of the model (similar to TSA, and likely having the same atmospheric drivers) and others are not commonly used in other earth systems models. In order to maximize the adaptability of the model while minimizing the error, we would recommend using 7-10 common atmospheric variables as features.

Future Development

In summary, our best model was a random forest model with 300 trees. We cleaned the data by dropping TSA values that were "NaN," converting time into datetime and extracting the year or transforming the year/month using trigonometric functions, splitting the training and validation data without shuffling (for hyperparameter tuning), filling in NaN values for the feature columns with their mean value, and normalizing the data by mean and standard deviation. We found that most of the default hyperparameter settings worked well for our model, with the exception of the number of trees. Finally, we experimented with a variety of feature column combinations. In the interest of balancing the accuracy and adaptability of the model, we would recommend using some variation of these variables: lat, lon, FSDS (atmospheric incident solar radiation), FLDS (atmospheric longwave radiation), RAIN (atmospheric rain), TBOT (atmospheric air temperature), PBOT (atmospheric pressure at surface), QBOT (atmospheric specific humidity), and U10 (10-m wind), and time feature(s) (year, or trigonometric transformations of year or month, to better represent the cyclical nature of time). By preprocessing and using this model architecture, we were able to achieve a RMSE of less than 0.20 degrees Kelvin. Further development of this model could focus on feature processing/extraction, or general model architecture. It would be interesting to incorporate more spatially explicit features, such as distance to coast, climate zone, etc. Lat/lon information could also be a more predictive feature with further manipulation, such as binning. We would recommend trying different methods of imputing NaN values, which could vary by the reason for the NaN values, and vary by the feature. Different selections and combinations of features could also be explored. After adjusting feature processing/extraction, model architecture and hyperparameters could be further developed. While there is room for further model improvement, our combined findings and recommended model provide a strong starting point for future work.

V. Discussion and Conclusion

A. Interpreting Results

B. What We Learned

C. Use of Machine Learning for Earth Systems Model Emulation

References