Spatio-Temporal Machine Learning Models for Emulation of Global Atmospheric Composition

Mohammad H. Erfani*
Center for Climate Systems Research
Columbia University
New York, NY 10025

Kara D. Lamb
Department of Earth and Environmental Engineering
Columbia University
New York, NY 10027

Susanne E. Bauer
NASA Goddard Institute for Space Studies
New York, NY 10025

Kostas Tsigaridis
Center for Climate Systems Research
Columbia University
New York, NY 10025

Marcus van Lier-Walqui
Center for Climate Systems Research
Columbia University
New York, NY 10025

Gavin Schmidt
NASA Goddard Institute for Space Studies
New York, NY 10025

Abstract

Interactive atmospheric composition simulations are among the most computationally expensive components in Earth System Models (ESMs) due to the need to transport a large number of gaseous and aerosol tracers at every model timestep. This poses a significant limitation for higher-resolution transient climate simulations with current computational resources. In ESMs such as NASA GISS-E2.1 (hereafter referred to as ModelE), pre-computed monthly-averaged atmospheric composition concentrations are often used to reduce computational expenses. This approach is referred to as Non-Interactive Tracers (NINT). In this study, we extend the NINT version of the ModelE using machine learning to emulate the effects of interactive emissions on climate forcing. We use data from a fully interactive composition climate model with surface-driven emissions to develop an ML-based NINT climate model. This version accounts for instantaneous atmospheric conditions, enabling the tracers to respond dynamically to meteorology without the need for explicit calculation of tracer transport. This approach could be applied to any aerosol species and integrated into ESMs to simulate aerosol concentrations interactively. The proposed framework emulates the advection term at the surface pressure level, with a focus on predicting surface-level concentrations of Black Carbon (BC) from biomass burning, which is a contributor to elevated levels of PM2.5 concentrations. Two consecutive years of ModelE simulated data were used as training data. To capture both temporal and spatial dependencies, a Convolutional Long Short-Term Memory (ConvLSTM) model was used. Results show the ConvLSTM achieved an average R^2 of 0.85 ($\sigma = 0.08$) on the test set. In contrast, using monthly-averaged atmospheric composition concentrations resulted in an average R^2 of 0.42 ($\sigma = 0.73$) for the same period.

*Corresponding author (se2639@columbia.edu)

Tackling Climate Change with Machine Learning: workshop at NeurIPS 2024.

1 Introduction

Earth System Models (ESMs) are foundational tools in climate science, simulating atmospheric dynamics by numerically solving fluid flow equations on a 3D grid box. However, the complexity of the atmosphere requires that sub-grid scale processes–such as turbulence, convection, and radiation–be represented through semi-empirical parameterizations [1]. Among these, representing atmospheric composition and chemistry remains one of the greatest challenges, as uncertainties in this area are considerably higher compared to other climate system forcings. For instance, modeling aerosol radiative forcing is particularly difficult due to the complex and transient nature of aerosols, their interactions with radiation, and their influence on cloud processes. Further complicating this task are the uncertainties in both natural and anthropogenic emissions, which affect the spatiotemporal distribution of aerosols in the atmosphere [2].

A specific case of this challenge can be observed in the NASA GISS-E2.1 climate model used for Coupled Model Intercomparison Project v.6 (CMIP6) simulations. This model does not include online interactive composition; instead, it relies on prescribed monthly aerosol and ozone concentration fields. These fields are calculated prognostically using the One-Moment Aerosol (OMA) version of the model [2, 3]. While this approach significantly reduces computational demands, it introduces notable drawbacks. By depending on pre-calculated fields, the model may fail to accurately capture real-time feedback between aerosol (source, transport, and sink) and other climate variables, potentially leading to inaccuracies in the simulation of climate dynamics. This approach critically constrains our capacity to model the interactions between a changing climate and surface-level air quality, the contributions of transient emission sources such as forest fires to direct radiative forcing, and the role of aerosols in modulating cloud processes through indirect climate forcing.

Recently, machine learning (ML) has emerged as a promising tool to address these and other challenges in climate modeling. For example, ML techniques such as super-resolution adversarial deep learning have been used to downscale GCM data to resolutions sufficient for climate impact studies [4]. Additionally, ML can emulate existing parameterizations to reduce computational costs [5], or even replace traditional methods by capturing latent relationships between climate subcomponents that are difficult to model through physical process parameterizations [6, 7, 8]. However, despite these advancements, significant challenges remain, particularly in achieving stable prognostic inference within fully coupled simulations. The primary issue here is stability, which is closely tied to generalization–the ability of ML models to perform well under conditions not encountered during training [9]. To mitigate this challenge, selecting an appropriate architecture with a suitable inductive bias is crucial, as it enhances the model's ability to generalize beyond the training data [10].

Predicting aerosol composition interactively presents several significant challenges for ML models. Due to their short-lived nature in the atmosphere, BC concentrations can vary by several orders of magnitude, from high levels near emission sources to much lower levels in remote regions, such as over oceans where aerosols have largely dissipated. So far, most efforts in applying ML to atmospheric composition have concentrated on either emulating local physical processes or replicating the entire atmospheric portion of the model [11, 12]. However, these approaches have not fully captured the complexity of both local and non-local influences on aerosol tracers. To address this limitation, we propose a Convolutional Long Short-Term Memory (ConvLSTM) model designed to project the effects of interactive emission information on climate forcing. The model dynamically calculates concentrations from emission sources, precipitation, and velocity fields for different aerosol species. By providing real-time, responsive aerosol concentration data, the model aims to enhance the accuracy and stability of climate simulations under varying conditions, leading to a better understanding of both direct and indirect aerosol climate effects. Our focus here is on carbonaceous aerosols—particularly BC, which is recognized as the most impactful absorbing aerosol in terms of its climate effects.

2 Methodology

2.1 Training Dataset

To train the ML model, the NASA GISS ModelE3 was run with prescribed sea surface temperature (SST) and sea ice fraction during the historical period for two consecutive years, 1950 and 1951. The atmospheric component of ModelE3 operates at a horizontal resolution of 2° latitude by 2.5°

longitude, with 62 vertical layers. The model's temporal resolution is set to half-hour intervals. Model diagnostics, including velocity field components (u, v, and ω), tracer concentrations across all 62 vertical levels, precipitation, and surface-level emissions for various aerosol species (e.g., biomass burning BC), are prognostically calculated using the model's OMA scheme, resulting in 35,040 timesteps over the entire 2-year period. The first year, comprising 17,520 timesteps, is used as the training set to ensure that the ML model learns from diurnal patterns as well as from a range of scenarios driven by seasonality, meteorological variations, aerosol emission changes, and differing atmospheric conditions. The second year is dedicated to validation and testing, with 10% of the data allocated for validation and 90% for testing.

2.2 Model Architecture

To capture the inherent high spatial and temporal dependencies in the data, we used ConvLSTM, which combines Convolutional and Long Short-Term Memory (LSTM) operations. By extending the fully connected LSTM to include convolutional structures in both the input-to-state and state-to-state transitions, ConvLSTM enables the construction of an end-to-end trainable model for problems involving spatiotemporal sequences, such as video data, where both spatial and temporal dependencies must be captured [13]. The ConvLSTM model received inputs at each timestep from ModelE diagnostics, including emission sources at the surface level, precipitation, and velocity field components. These inputs are the primary drivers for the source, sink (wet deposition), and transportation of the tracers at each timestep. Additionally, at each timestep, the tracer concentration at the surface pressure level was considered the target variable. To model this temporal dependency, the ConvLSTM received as input the concentration value at a given timestep, along with the ModelE diagnostics from the 48 preceding timesteps [14]. In total, we used five inputs at each half-hour timestep: emission source, precipitation, and three velocity field components. More details are provided in Appendix A.

3 Results

The R^2 values for the test set are used to compare each model output at every timestep with the corresponding ground truth values, which represent the BC concentrations interactively calculated by ModelE. Figure 1-a displays the range of R^2 values for the test period using a boxplot. The first quartile is 0.81, and the third quartile is 0.91, with a mean of 0.85 and a median of 0.87. These results suggest that the model performs acceptably during the inference phase. Figure 1-b shows the R^2 score for each timestep throughout the test period.

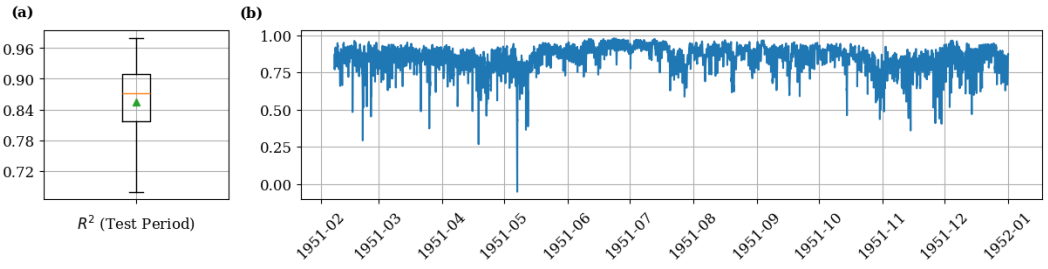

Figure 1: (a) Boxplot of R^2 values for the test period, showing the range of performance. (b) R^2 scores for each timestep throughout the test period.

The global averages of the model results for the test set were compared with the corresponding ground truth values. Figure 2-a displays a scatter plot of these comparisons. It shows that the model performs better for lower values but tends to underestimate higher values. This could be crucial for predicting extreme events, such as air quality during severe pollution episodes. Figure 2-b compares the global averages of model outputs and ground truth values over the test period. It highlights that the model struggles to capture peak values (on a global average) accurately between July and the end of September. This is likely due to the significant contributions of carbonaceous aerosols to the atmosphere during wildfire season in the Northern Hemisphere, which are associated with the highest BC concentrations during this time.

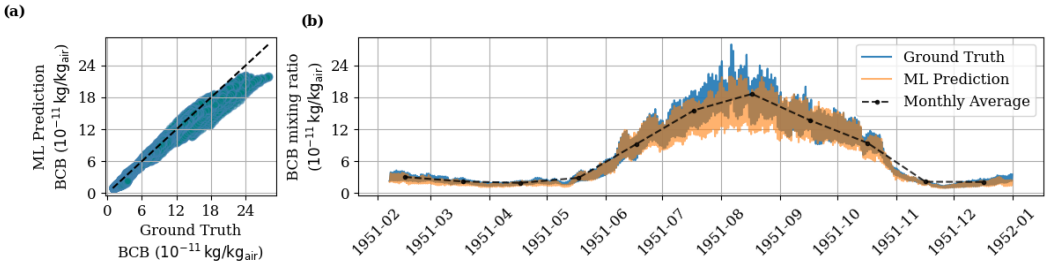

Figure 2: (a) Scatter plot comparing the global averages of model results with ground truth values for the test set. (b) Comparison of global average values of model outputs and ground truth over the test period.

We compared 10,000 randomly selected grid value instances during the test period with their corresponding model outputs. This analysis reveals that while the global average shows underestimation for high values, the model performs better for individual instances with high concentrations compared to low concentrations. This is because the model tends to treat very low values (e.g., BCB concentrations in the South Pole or middle of the ocean) as zero, prioritizing high concentration values. This bias is a common issue when using the mean-squared error (MSE) loss function, which overemphasizes larger concentration values [14]. To mitigate this, we included the mean absolute error (MAE) and used the average of MSE and MAE as the final loss function [15]. However, due to the scale difference where high concentrations are several orders of magnitude larger than low values, some biases persist.

Figure 3-a shows a scatter plot comparing the grid instances with their model outputs on a linear scale. Since low concentrations dominate the random sampling, we also present the results on a log scale (Figure 3-b), which better highlights the model's performance across concentration ranges. As shown, the model output deviates from the ground truth by orders of magnitude for low concentrations. To further investigate extreme events, we performed an exceedance probability analysis on grid values (Figure 3-c). Using the Weibull formula, a common method in Earth sciences for estimating the return period of extreme events, we assessed high-concentration BCB instances. The results indicate that the model begins underestimating events with BCB concentrations equal to or exceeding $630 \times 10^{-11}\ kg/kg_{air}$, with the probability of such events being less than 0.1 %.

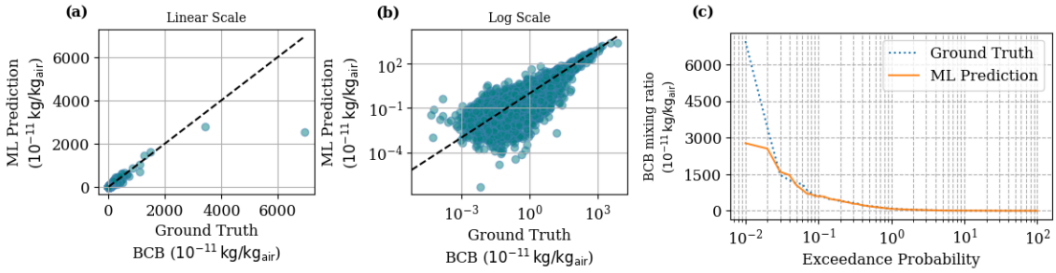

Figure 3: (a) Scatter plot comparing the randomly selected grid value instances of model outputs with ground truth values for the test set on a linear scale. (b) The same comparison on a logarithmic scale. (c) Exceedance probability plot, illustrating the return period of extreme BCB concentration events.

To better understand the model's performance across the spatial domain, the R^2 values for each grid during the test period were calculated and plotted in Figure 4-a. Regions with high R^2 values are shown in red, while regions with low and negative R^2 values are represented in blue. Figures 4-b, -c, and -d compare the model outputs and ground truth for three distinct grids where located in red and blue regions. These colors represent regions where the model performs well, or worse than a baseline model that predicts the mean of the observed values. The model performs well in areas with high tracer concentration (Figure 4-b) but struggles to capture spikes in regions with lower concentration values (blue regions, Figures 4-c and -d). In the blue regions, the tracer concentration is several orders of magnitude lower compared to regions with higher concentrations.

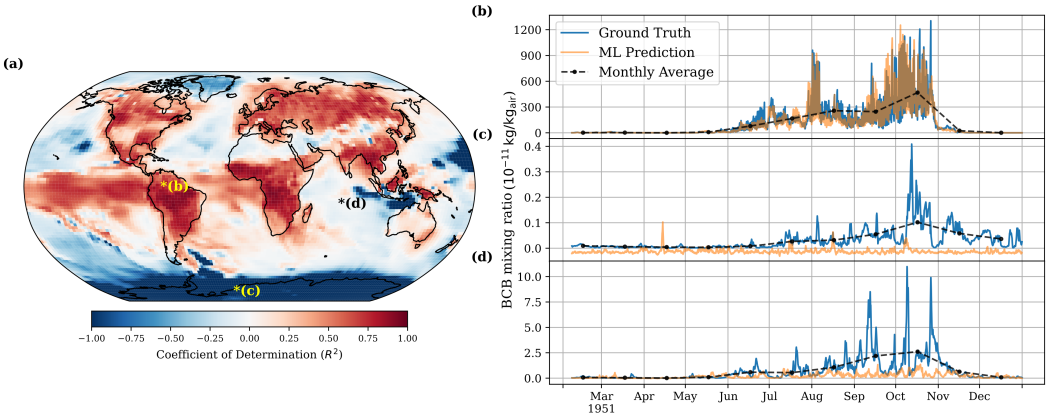

Figure 4: (a) Spatial distribution of R^2 values during the test period. (b) The grid point is located at lat: -3.0, lon: -71.25, with R^2 of 0.87 (c) The grid point is located at lat: -81.0, lon: -21.25, with R^2 equal to -1.08 (d) The grid box is located at lat: -15.0, lon: 68.75, with R^2 of -0.07.

We analyze spatial results zonally across latitude bands: tropical (-21° to +21°), northern mid-latitudes (+21° to +61°), northern polar (+61° to +90°), southern mid-latitudes (-21° to -61°), and southern polar (-61° to -90°). Weighted zonal averages of R^2 values, based on mean grid-level concentrations over time, are 0.75, 0.56, 0.43, 0.51, and -1.52, respectively. The model performs best in the tropics where the BCB emissions were high in the test period (1951) due to several factors such as extensive deforestation, and prevalent slash-and-burn agricultural practices. These activities led to significant biomass burning, releasing large amounts of black carbon into the atmosphere. This indicates that the model is more accurate in regions with higher emission levels, likely because the signal-to-noise ratio is higher, making it easier for the model to capture the underlying patterns and dynamics.

4 Conclusions

In this study, we have introduced an ML approach for interactively calculating atmospheric composition, with a particular focus on biomass-burning black carbon (BC), which are critical short-lived climate forcers due to their strong solar radiation absorption. As forest fires become more severe and frequent due to climate change, the air quality impacts of biomass burning events are becoming increasingly significant [16]. Therefore, improving predictions of surface-level concentrations of carbonaceous aerosols is essential for accurately forecasting air quality effects under future climate scenarios. Our results indicate the feasibility of using ML-based approaches for interactive calculations of atmospheric composition across a range of atmospheric conditions, offering a significantly lower computational cost compared to current fully interactive methods.

In future studies, we plan to add Self-Attention Memory (SAM) to memorize features with long-range dependencies in both spatial and temporal domains [17]. Preliminary results show that adding such a module improves the performance of the model. However, using attention modules on spatial data significantly increases GPU memory usage. To address this, we are working on adding encoder/decoder steps before and after the SAM module to control VRAM occupation. Another important step is configuring the model to estimate BCB concentration at higher vertical levels, starting from the surface level. For this purpose, we may combine Recurrent Neural Network (RNN) architecture with Graph Convolutional Networks (GCNs) to better simulate the behavior of aerosols in the atmosphere in our machine learning models.

We will integrate our ML model with NASA GISS-E2.1, which is primarily written in Fortran. To achieve this, we are using FTorch, which is written and maintained by the Institute of Computing for Climate Science (ICCS).

References

[1] Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Griffin Mooers, Milan Klöwer, James Lottes, Stephan Rasp, Peter Düben, et al. Neural general circulation models for weather and climate. *Nature*, pages 1–7, 2024.

[2] Susanne E Bauer, Kostas Tsigaridis, Greg Faluvegi, Maxwell Kelley, Ken K Lo, Ron L Miller, Larissa Nazarenko, Gavin A Schmidt, and Jingbo Wu. Historical (1850–2014) aerosol evolution and role on climate forcing using the giss modele2. 1 contribution to cmip6. *Journal of Advances in Modeling Earth Systems*, 12(8):e2019MS001978, 2020.

[3] Maxwell Kelley, Gavin A Schmidt, Larissa S Nazarenko, Susanne E Bauer, Reto Ruedy, Gary L Russell, Andrew S Ackerman, Igor Aleinov, Michael Bauer, Rainer Bleck, et al. Giss-e2. 1: Configurations and climatology. *Journal of Advances in Modeling Earth Systems*, 12(8):e2019MS002025, 2020.

[4] Karen Stengel, Andrew Glaws, Dylan Hettinger, and Ryan N King. Adversarial super-resolution of climatological wind and solar data. *Proceedings of the National Academy of Sciences*, 117(29):16805–16815, 2020.

[5] Lizao Li, Robert Carver, Ignacio Lopez-Gomez, Fei Sha, and John Anderson. Generative emulation of weather forecast ensembles with diffusion models. *Science Advances*, 10(13):eadk4489, 2024.

[6] Frederik Kratzert, Daniel Klotz, Claire Brenner, Karsten Schulz, and Mathew Herrnegger. Rainfall–runoff modelling using long short-term memory (lstm) networks. *Hydrology and Earth System Sciences*, 22(11):6005–6022, 2018.

[7] Jatan Buch, A Park Williams, Caroline S Juang, Winslow D Hansen, and Pierre Gentine. Smlfire1. 0: a stochastic machine learning (sml) model for wildfire activity in the western united states. *Geoscientific Model Development*, 16(12):3407–3433, 2023.

[8] Donifan Barahona, Katherine H Breen, Heike Kalesse-Los, and Johannes Röttenbacher. Deep learning parameterization of vertical wind velocity variability via constrained adversarial training. *Artificial Intelligence for the Earth Systems*, 3(1):e230025, 2024.

[9] Pierre Gentine, Mike Pritchard, Stephan Rasp, Gael Reinaudi, and Galen Yacalis. Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters*, 45(11):5742–5751, 2018.

[10] Pieter-Jan Hoedt, Frederik Kratzert, Daniel Klotz, Christina Halmich, Markus Holzleitner, Grey S Nearing, Sepp Hochreiter, and Günter Klambauer. Mc-lstm: Mass-conserving lstm. In *International conference on machine learning*, pages 4275–4286. PMLR, 2021.

[11] Paula Harder, Duncan Watson-Parris, Philip Stier, Dominik Strassel, Nicolas R Gauger, and Janis Keuper. Physics-informed learning of aerosol microphysics. *Environmental Data Science*, 1:e20, 2022.

[12] Kyleen Liao, Jatan Buch, Kara Lamb, and Pierre Gentine. Simulating the air quality impact of prescribed fires using a graph neural network-based pm _{2.5} emissions forecasting system. *arXiv preprint arXiv:2312.04291*, 2023.

[13] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

[14] Frederik Kratzert, Daniel Klotz, Mathew Herrnegger, Alden K Sampson, Sepp Hochreiter, and Grey S Nearing. Toward improved predictions in ungauged basins: Exploiting the power of machine learning. *Water Resources Research*, 55(12):11344–11354, 2019.

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[16] Daniel A Jaffe, Susan M O'Neill, Narasimhan K Larkin, Amara L Holder, David L Peterson, Jessica E Halofsky, and Ana G Rappold. Wildfire and prescribed burning impacts on air quality in the united states. *Journal of the Air & Waste Management Association*, 70(6):583–615, 2020.

[17] Zhihui Lin, Maomao Li, Zhuobin Zheng, Yangyang Cheng, and Chun Yuan. Self-attention convlstm for spatiotemporal prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11531–11538, 2020.

[18] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

A ConvLSTM Architecture

The ConvLSTM was configured with three cells: the first cell applied a convolutional operation to map the five inputs into 64 cell states using a kernel size of 5; the second cell mapped the 64 cell states into 32 using a kernel size of 3; and the final cell reduced the 32 cell states to 16 with a kernel size of 3 before a single convolutional layer mapping 16 cell states to 1 output layer with a kernel size of 1. The last convolution operation acted as a regression layer. Figure 5 illustrates the different components of the model architecture. The hidden state of the last sequence of the third cell, $H^3_{t_{48}}$, with dimensions Batch Size (BS) $\times$ 16 $\times$ 100 $\times$ 154, is fed into the last convolution layer with a kernel size of one to project it to the target values.

Stacking additional convolutional layers generally helps to capture spatial dependencies more effectively, as deeper networks can exponentially increase their capacity to model both spatial and temporal dependencies [17]. However, because this model is intended to perform emulation in a computationally efficient manner, we aimed to keep it as lightweight as possible. While we have not conducted a formal ablation study, we tested different numbers of cells and hidden-state sizes for each cell to avoid inefficient architectures. In terms of temporal dependencies, we also experimented with various time-step lengths, both longer and shorter than 48, based on the longevity of aerosol species considered in the main model, but using 48 preceding timesteps showed the best performance.

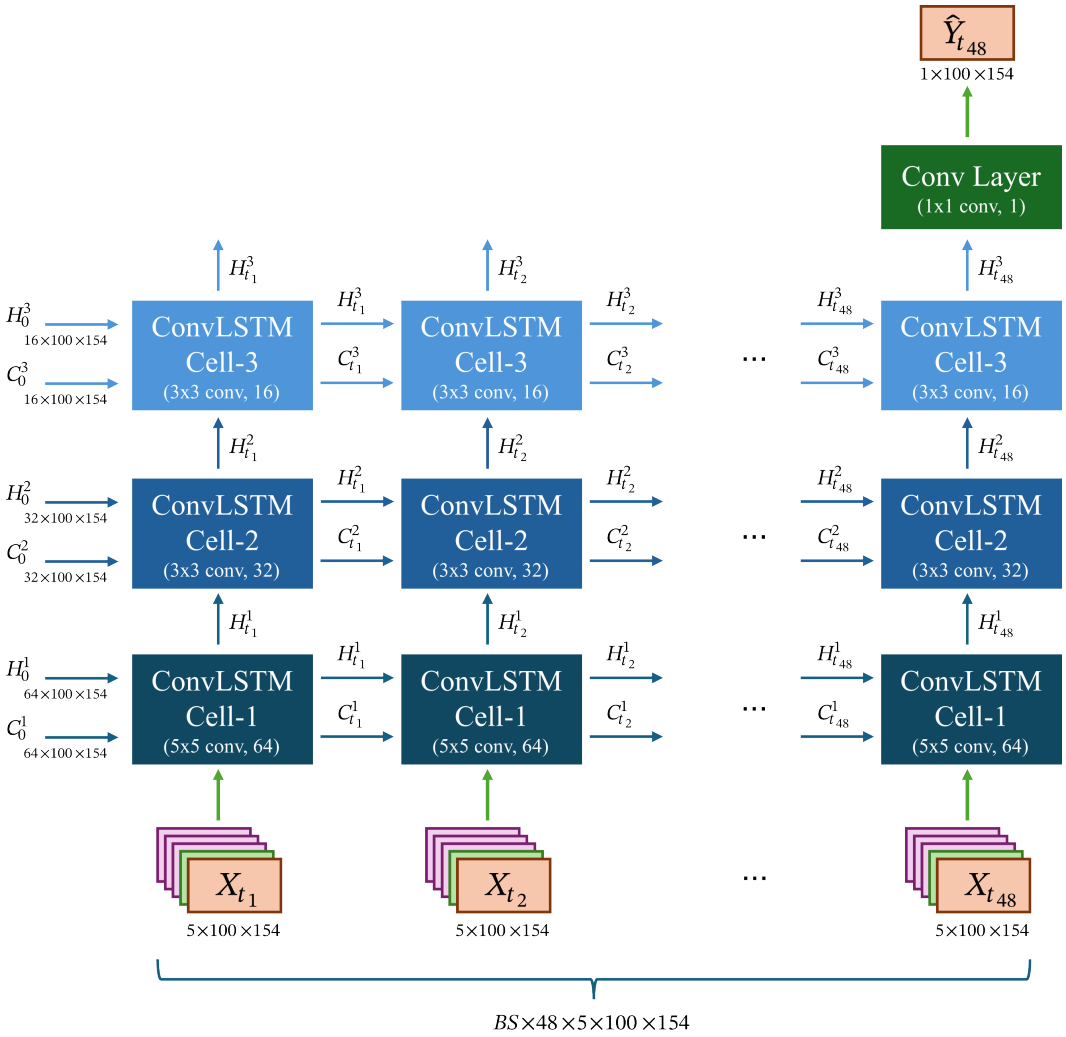

Figure 5: ConvLSTM model architecture. To capture temporal dependency in each timestep, the ConvLSTM received as input the concentration value at a given timestep, along with the ModelE diagnostics from the 48 preceding timesteps.

Figure 5 illustrates how the ConvLSTM architecture processes spatiotemporal data over a sequence of 48 timesteps, denoted by $X_{t1}, X_{t2}, \ldots, X_{t48}$, each with dimensions $5 \times 100 \times 154$. These five inputs include the velocity field in 3 dimensions, precipitation, and emissions. The overall input shape, $BS \times 48 \times 5 \times 100 \times 154$, represents the batch size (BS), the sequence length (48), the number of channels (5), and the spatial dimensions (lat = 100, lon = 154) considering 5 pixels of cyclic padding along the longitude on each side and 5 pixels of reflective padding along the latitude on each side. The model consists of three stacked ConvLSTM cells, labeled 'Cell-1,' 'Cell-2,' and 'Cell-3,' each operating across the entire sequence of 48 timesteps. 'Cell-1' applies a 5×5 convolutional filter with 64 channels, producing hidden states H^1 and cell states C^1 at each timestep. 'Cell-2' utilizes a 3×3 convolutional filter with 32 channels, outputting states H^2 and C^2, while 'Cell-3' applies a 3×3 convolutional filter with 16 channels, outputting states H^3 and C^3. At each timestep t, the hidden and cell states from each ConvLSTM cell are updated and passed to the next timestep. After processing all timesteps, the final output H^3_{48} represents the hidden states at the last timestep t_{48}. This is then fed into a 1×1 convolutional layer with a single output channel, producing the final output $\hat{Y}_{t_{48}}$ with dimensions $1 \times 100 \times 154$. This output represents the model's prediction for the given timestep.

As mentioned in Section 2.2, the 48 preceding timesteps were used to predict the BCB concentration for the current timestep. According to Figure 5, the initial hidden state is set to zero at the first timestep. The model then recurrently calculates the concentration over the past 47 timesteps to arrive at the current timestep. A portion of these 48 timesteps serves as a "spin-up" period, allowing the model to adjust the initial zero values and stabilize its solution. To understand the length of the spin-up period, we collected the hidden state output after each timestep update during the inference phase. To compare these outputs with their corresponding ground truth concentrations, we scaled them based on the ground truth values. We then averaged the hidden states along the spatial domains (latitude and longitude) to convert each hidden state into a vector of length 48. These vectors were then plotted against their corresponding ground truth values. Results for four randomly selected instances are shown in Figure 6. This figure shows that it takes approximately 10 timesteps for the model to stabilize the initial hidden state.

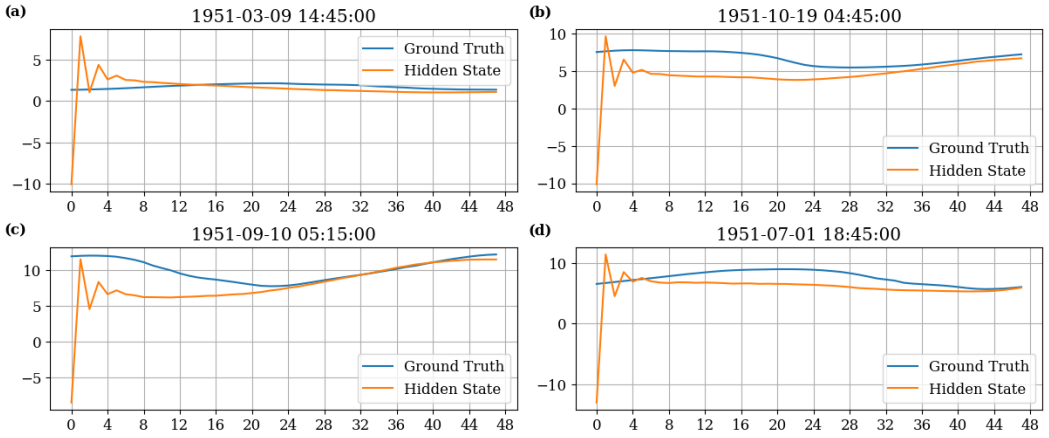

Figure 6: The number of timesteps required for the model's hidden state to stabilize.

This model was implemented using PyTorch [18]. During training, the base learning rate was set to 0.001 and was decayed by a factor of 0.5 every 10 epochs. The network was optimized using Adam with beta values of 0.5 and 0.999. The network was trained for 30 epochs with a batch size of 8. Because atmospheric data on a global scale is continuous at the edges (i.e., longitude 0° and 360° are the same), cyclic padding was applied along the longitude axis. Moreover, reflection padding was used along the latitude axis to reflect the data at the boundaries thus mirroring data near the poles across the pole.

B Model Parameter Summary and Performance Analysis

This section provides a detailed summary of the model's parameters, memory requirements, and computational performance. Table 1 outlines key metrics, including the total number of parameters,

estimated memory usage, and execution times on both GPU and CPU, offering insight into the model's efficiency during training and inference.

Table 1: Model Specifications and Performance

Description	**Value**
Total Parameters	580,305
Trainable Parameters	580,305
Non-Trainable Parameters	0
Total Mult-Adds (G)	428.65
Input Size (MB)	14.10 [1]
Forward/Backward Pass Size (MB)	52.75
Params Size (MB)	2.21
Estimated Total Size (MB)	69.07
GPU Execution Time [2]	38.7 ms ± 356 µs
CPU Execution Time	668 ms ± 19.2 ms

[1] Input size estimated for a batch size of 1
[2] GPU model: NVIDIA A100 80GB PCIe