Machine Learning Approaches for NO_2 Forecasting: Direct Model Predictions vs. Foundation Model with Residual Correction

REU Site: Online Interdisciplinary Big Data Analytics in Science and Engineering

Ray Chen[1],Oscar Tuten[2], Michael Togbe[3], Aryan Poshtiwala[4], Adeline Barden[5]
Pavan Raj Ravi[6], Ruoyan Gong[7], Kyo Hugo Lee[8]
Nicholas LaHaye[9], Xiaohua Pan[10], Hazem Mahmoud[11], and Jianwu Wang[6]

[1]Department of Computer Science, University of Maryland, College Park
[2]Department of Computer and Data Science, Goucher College
[3]Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County
[4]Clarksburg High School, Montgomery County, Maryland
[5]Department of Education, Troy University
[6]Department of Information Systems, University of Maryland, Baltimore County
[7]P.C. Rossin College of Engineering and Applied Science, Lehigh University
[8]Jet Propulsion Laboratory
[9]Spatial Informatics Group
[10]NASA Goddard Space Flight Center
[11]NASA Atmospheric Science Data Center

Technical Report HPCF–2026–4, hpcf.umbc.edu > Publications

Abstract

Accurate nitrogen dioxide (NO_2) forecasting is critical for air quality management, yet training models from scratch is computationally expensive. Pretrained geo foundation models such as Microsoft's Aurora can be applied to forecasting tasks without retraining. However, it remains unclear whether using a pretrained foundation model improves NO_2 forecasting compared with direct machine learning approaches. This work compares two approaches for forecasting total column NO_2($tcNO_2$) over the west coast of North America. The first post-processes Aurora forecasts through residual correction using Copernicus Atmosphere Monitoring Service (CAMS) atmospheric as input features and NASA TEMPO satellite observations as ground truth. The second trains machine learning models to predict TEMPO tcNO2 directly from CAMS atmospheric inputs. Random Forest, Extreme Gradient Boosting (XGBoost), and Normalizing Flow models are trained and evaluated under both approaches to determine whether incorporating a pretrained foundation model produces forecasts that more closely match TEMPO observations than direct prediction from CAMS inputs.

1 Introduction

Nitrogen dioxide (NO_2) is a key indicator of air pollution and a significant public health concern, with concentrations known to spike sharply during events such as wildfires. Reliable NO_2 Forecasting could support earlier warnings and better-informed public health responses in affected regions. However, training a dedicated forecasting model from scratch is computationally expensive and requires large volumes of data and substantial compute resources.

Recent advances in pretrained foundation models offer a potential alternative. Models such as Microsoft's Aurora are trained once on large, heterogeneous Earth system datasets and can be applied to forecasting tasks without retraining. This raises a natural question: does leveraging a pretrained foundation model improve NO_2 forecasting compared to training a model directly for the task, or does the added complexity of incorporating a foundation model fail to justify itself?

This work directly investigates this question. We evaluate two forecasting approaches over the western coast of North America: a residual correction approach, which uses Aurora's forecast as a baseline and trains machine learning models to correct its forecast error, and a direct prediction approach, which trains the same class of models to predict NO_2 independently from Aurora, using the same atmospheric input data. Both approaches are evaluated against NASA's TEMPO satellite observations, which serve as ground truth throughout this study. Random Forest, XGBoost, and Normalizing Flows are trained and compared under both approaches to assess whether incorporating Aurora provides a measurable advantage over direct NO_2 prediction.

2 Related Works

There have been many foundation models used for weather and climate forecasting. Different models are optimized uniquely, making them better suited for specific tasks or types of forecasting. For example, the Aurora foundation model we use for this research is best suited for weather forecasting [1].

Machine learning models including XGBoost and random forest have been used previously for extreme temperature prediction [4]. Extreme temperature often changes seasonally and is affected by a variety of atmospheric conditions. This is similar to our direct approach with the XGBoost and random forest models. Our models take in different meteorological and air-pollution-related inputs to predict total column NO_2 ($tcNO_2$).

Previous research has already been done on air pollution including air quality predictions. One research team used a random forest model to predict urban air quality in Shenyang, China [13]. The model ended up with 81% precision for the air quality index (AQI) prediction, reaching a higher precision than most of the other models used in comparison, clearly getting better precision than some other simple models [13]. Like this research, we use a random forest model to make a prediction about the atmosphere. In our case, we are predicting $tcNO_2$ instead of AQI.

Another research team compared AQI predictions by classical machine learning models, like XGBoost, to a deep learning algorithm called Long-Short Term Memory [11]. Our research is a bit different in that we are not considering the same types of models for comparison, but we are using an XGBoost model.

There are other models that have been used for direct prediction of NO2 as well. Land use regression models have previously been used to directly predict NO_2 and PM_{10} in Changsha, China with some success [8]. Convolutional neural networks (CNNs) have also been used to predict NO_2 in the United States, excluding Hawaii and Alaska [3].

Like other works, we implemented a bias correction method. These usually involve statistical methods of identifying and reducing bias [12]. Our work varies from this approach by directly predicting the residual between another prediction and the predicted ground truth and adding the residual to the prediction. Machine learning bias correction has previously been applied to forecasting spatial distribution of hourly precipitation in a limited area [12]. Our research was also done in a limited area, allowing the models to better pick up on more regional biases.

Traffic patterns are based on sudden increases in traffic due to time of day, location, and random events. This is similar to NO_2 patterns. One such random event that causes a sudden fluctuation in NO_2 is a wildfire. Cities also experience increased NO_2. Kim et al. created an add-on module they called ResCAL to predict the residuals of the traffic forecasts of different models versus METR-LA data from Los Angeles, California and the PEMS-BAY dataset from California [7]. These researchers found that their ResCAL add-on module improved the performance of the existing forecasting models [7]. Like Kim et al., we create bias correction in our research, though ours are

residual correction models that are not directly attached to the prediction model.

Our work uses a similar, if somewhat simplified, process for residual correction and direct prediction as previous research. Using even a simple bias correction method can improve predictions to some extent. We incorporate direct predictions as well to compare the results, allowing us to draw more reliable conclusions.

3 Background

3.1 Foundation Models

Foundation models (FMs) are learning models that are able to generalize patterns to predict information based on the data they learn from, able to be adjusted for further down-stream tasks [2]. FMs learn from a wide variety of sources. The more sources a FM learns from, the more likely it is that the FM will be able to identify a pattern to predict the variable or variables of interest. Although FMs use a lot of resources to be pre-trained, mostly due to the large volume of data that is being inputted into the model for it to learn, the process of pre-training usually only needs to happen once.

Earth FMs in particular tend to be trained on data from satellites, Earth simulations, Earth reanalysis, and already-existing observational data, among other sources. Like other FMs, Earth FMs take in the data and identify patterns and trends based on different factors. Earth FMs are important for weather forecasting and air quality forecasting. The Aurora foundation model, more concisely named Aurora, is an example of an Earth FM.

3.2 Aurora

Aurora, developed by Microsoft Research is a 1.3 billion parameter foundation model designed to forecast a broad range of Earth system variables [1]. Aurora is pretrained on heterogeneous datasets that contain different physical variables, pressure levels, and spatial resolutions. This allows the model to be adapted to several applications, including weather prediction, ocean wave forecasting, and air quality forecasting.

Aurora uses an encoder, a neural simulator, and a decoder to transform its input into a forecast. The encoder converts atmospheric inputs from the current and previous time steps into a common three dimensional representation. The neural simulator then updates this representation using a 3D Swin transformer U-Net. The decoder converts the updated representation into physical variables on the target grid. For forecasts that extend beyond one time step, Aurora uses an auto regressive approach in which each prediction becomes part of the input for the following step [1].

In this work, we use Aurora to provide air quality forecasts that serve as a baseline for our NO_2 prediction experiments.

3.3 TEMPO

The Tropospheric Emissions: Monitoring of Pollution (TEMPO) instrument is a satellite in geostationary orbit over North America, gathering light data over wavelengths of 290-490 nm and 540-740 nm. The TEMPO satellite performs hourly east-west scans over North America during daylight hours. During early morning hours for the East Coast, data for the West Coast is not available. Likewise, during late afternoon hours for the West Coast, data for the East Coast is not available. TEMPO data products were generated by the Science Data Processing Center (SDPC) at the Smithsonian Astrophysical Observatory (SAO) from the raw data [5]. One such data product

is Trace Gas (NO_2 and HCHO) and Cloud Level 3.

TEMPO Trace Gas (NO_2 and HCHO) and Cloud Level 3 data is in a $0.02^\circ \times 0.02^\circ$ resolution grid over 168°W to 13°W and 14°N to 73°N calculated using weighted gridding from level 2 data. There are sections of the grid area outside of the scan range of the instrument. Version 03 of this data product covers a temporal range from 08/02/2023 to 09/16/2025. The instrument was not in operation on some of the days, with a notable section missing from 08/09/2024 to 08/21/2024. The data is stored in netCDF-4 (nc4) format. The data points are stored in a coordinate system of latitude, longitude, and time. Data variables in the product include Stratosphere NO2, Troposphere NO2, main data quality flag, and effective cloud fraction. High cloud cover is known to interfere with data quality and should be filtered using effective cloud fraction. Other factors for suspect data are aggregated in main data quality flag [5].

3.4 CAMS

Copernicus Atmospheric Monitoring Service (CAMS) reanalysis data is a dataset of atmospheric composition created by the European Centre for Medium-Range Weather Forecasts (ECMWF) [6]. CAMS reanalysis data includes aerosols, chemical species, and, separately, greenhouse gasses [6]. CAMS data was used to fine-tune Aurora Pre-Trained, creating Aurora Air Pollution [1]. Aurora Air Pollution has a 0.4° resolution since the CAMS data used has a 0.4° resolution [1]. CAMS data is also used as the inputs for the models we present in this research. This is due to the large number of variables that CAMS provides. The specific variables we use as inputs are the surface-level variables. This includes air pollution variables and other related variables corresponding to the wind, ocean, and temperature. There are 12 total variables from CAMS that go into our models. Since there are so many variables, our models are more likely to notice underlying patterns. This should help the models create better overall predictions.

3.5 Residual Correction

Residual correction is a technique of taking the residual from two different groups of data points by calculating their difference. Residual correction based models are designed to calculate the difference between the predicted outcome and the ground truth to allow for more precise forecasting [7]. After taking this series of residual values, they can be added back to the original prediction to obtain the ground truth data points. This process involves at least two datasets or points that can be compared on a one to one ratio with values of the same unit. It is used in a variety of applications like traffic forecasting [7].

4 Problem Definition

This work addresses the task of forecasting the $tcNO_2$ over the western United States at one, two, and three days ahead. These lead times are referred to as T+1, T+3 and T+5, where each time step represents 12 hours.

In designing the models, the goal was to include variables with potential correlation to our target of NO_2, and those with similar concentrations in the air. This would allow the models to have the best chance of success after training by allowing them adequate data to draw from for their decisions. Correlation gives the models more context across time steps. For example, if the total column (meaning the count from stratosphere to troposphere) of Nitrogen Oxide increases alongside Nitrogen Dioxide from 12 UTC to 00 UTC in the training data, then the models are

helped by the strong predictive signal. Following this logic, the physics that affects these surface level variables is also an indicator of some relationship, even if very small.

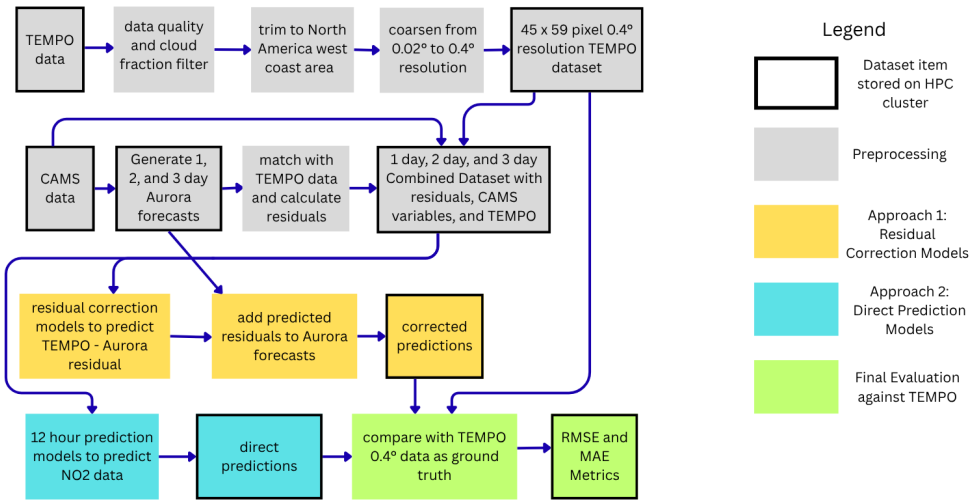
Not all possible CAMS features were used due to the possibility of overfitting with high dimensionality. The majority of the surface level variables were chosen due to their total column units being similar and easy to compare. Many of the atmospheric CAMS variables were stored across pressure levels which made them tedious to use, and the surface level variables included already covered potentially useful variables. The next steps were to integrate these variables along with latitude and longitude to place each pixel in a data set to train the models. Below is the table of included variables:

Raj pls add a table of the input variables we used to train the ML models.

Two forecasting approaches are compared. The first uses Aurora as a baseline forecast and applies machine learning to predict and correct its forecast error, referred to as residual correction. The second directly predicts $tcNO_2$ from CAMS atmospheric variables without using Aurora. For residual correction, the target is the residual which is the difference between the TEMPO ground truth and Aurora's forecast. The predicted residual is then added back into Aurora's forecast to produce the final prediction. For direct prediction, the target is TEMPO's observed $tcNO_2$ itself. Both approaches use the same CAMS input variables and are evaluated against TEMPO observations.

The primary research question is whether incorporating a pretrained foundation model through residual correction provides more accurate $tcNO_2$ forecasts than directly predicting $tcNO_2$ from atmospheric data. Model performance is compared using mean absolute error(MAE) and root mean squared error (RMSE).

5 Pipeline



5.1 Data Acquisition

5.1.1 Downloading TEMPO

An account was created for urs.earthdata.nasa.gov to be able to access TEMPO Trace Gas (NO_2 and HCHO) data. The login information was then stored in a '.netrc' file on UMBC's chip

cluster. NASA's Harmony system was used in a Python script to download each day of data. The collection ID for TEMPO Trace Gas (NO_2 and HCHO) and Cloud Level 3 data V03 is C2930763263-LARC_CLOUD. All data files from the collection were downloaded with the variables: 'product/vertical_column_troposphere', 'product/vertical_column_stratosphere', 'product/main_data_quality_flag', 'support_data/
eff_cloud_fraction', 'latitude', and 'longitude'. A Slurm job array setup was used to download all of the days of data.

5.1.2 Downloading CAMS

An account was created on the Copernicus Atmosphere Data Store (ADS) to access CAMS global atmospheric composition forecast data. The account credentials were stored in a '.cdsapi' file on UMBC's chip cluster. The cdsapi Python client was used in a Python script to download one file per day for the full study period. The dataset named used to request this data is 'cams-global-atmospheric-composition-forecasts'. For each day, both 00 UTC and 12 UTC timestamps were requested with a lead time hour of 0, corresponding to CAMS analysis data rather than a forecast. Each file includes 12 surface-level variables including wind, temperature, pressure, particulate matter, and pollutant concentrations, and 10 atmospheric variables spanning 13 pressure levels from 50hPa to 1000hPa. The script was run within a dedicated conda environment, with all required packages installed via the accompanying setup script, and checked for already downloaded files before each request, allowing an interrupted download to be resumed without re-downloading previously completed days.

5.2 TEMPO Preprocessing

The preprocessing for TEMPO data consisted of three steps: pixel quality filtering, area trimming, and reducing resolution. Pixel quality filtering was performed using two criteria: main data quality flag and effective cloud fraction. Pixels with main data quality flag > 0, signifying suspect data points, were removed. Pixels with effective cloud fraction > 0.2 were also removed. Removed pixels were replaced with NaN values. An area slice was then taken from each file within the bounds: latitude from 31.31°N to 49.30°N, and longitude from 125.45°W to 101.86°W. This focuses on a target area in the west coast of North America.

Finally, the resolution of the dataset was reduced using an aggregation function from 0.02° resolution to 0.4° resolution. The aggregation function takes a block of 20×20 0.02° resolution pixels and performs mean aggregation. If all the values in the block are NaN, the final aggregated pixel is NaN. Otherwise, a mean is calculated from the remaining valid pixels in the block. The preprocessing produces 0.4° resolution TEMPO files of a North American west coast focus area that can be compared to 0.4° resolution CAMS data and Aurora predictions.

5.3 Aurora Forecast Generation

Microsoft Aurora's team created the Aurora Air Pollution model to predict CAMS air pollution variables. The code to run Aurora Air Pollution was created by Bodnar et al. [1] and is hosted on is publicly available [10]. When run, the original code creates four maps of $tcNO_2$ and PM_{10}, using four timesteps. Each day, there is a map for midnight and noon in coordinated universal time (UTC), times 00:00 and 12:00 respectively. The original code creates these predictions for June 12, 2022 and June 13, 2022 from the data on June 11, 2022 at midnight and noon UTC.

For this research, the original code was used but changed to reflect what we needed for our purposes. Since we needed a year's worth of predictions saved differently based on if they were one-day, two-day, or three-day predictions, three different versions of the code were created. Each version starts predicting based on the data from CAMS on July 24, 2024 at 00 and 12 UTC. As such, the first predictions of the one-day, two-day, and three-day predictions were July 25, 2024 at 00 UTC, July 26, 2024 at 00 UTC, and July 27, 2024 at 00 UTC respectively. We used Slurm jobs to run the code.

The changes made to the original code include saving the date as its own variable and all of the strings involving the date to have f-strings. Also, the predictions for all variables are saved into a file with each file representing a different day. The code automatically runs through the specified number of days, which was another change to the original code. This part was possible with a range function. For the two-day and three-day predictions, the number of steps are tracked and only saved if the number of steps equals 2 or 3 for the two-day predictions and 4 or 5 for the three-day predictions. The steps follow the ordering of normal objects in Python, so the first step is called step 0. There were six total steps, two for each day. These changes are what allowed us to use the Microsoft Aurora Air Pollution predictions for our own research.

5.4 Matching Aurora Forecasts with TEMPO Observations

Aurora forecasts are produced for a single fixed time each day: 00UTC and 12UTC. TEMPO scans North America hourly throughout daylight hours and produces multiple scans per day at irregular times as the satellite moves from east to west across the continent. Since TEMPO retrieves NO_2 using reflected sunlight, it can only produce usable data during daylight hours over a given region.

Over the western United States, 00UTC corresponds to late afternoon local time. In contrast, 12 UTC corresponds to the pre-dawn hours before sunrise. Because TEMPO does not have usable daylight scans before sunrise, matching Aurora's 12 UTC forecasts would have required comparing them with TEMPO scans that had little to no valid coverage over the western study region. For this reason, Aurora's 00 UTC forecast was used for all comparisons in this study. This forecast aligns with a time when TEMPO reliably has full daylight coverage over the western United States.

For each day, the TEMPO scan closest to 00UTC was selected within a 180-minute window on either side of the target time. A scan was only accepted if at least 70% of the pixels within the study region contained valid, non-missing data. If no scan within the time window met this coverage threshold, that day was excluded. This prevented lower-quality scans with insufficient coverage from being used as ground truth data.

Using this matching process, 328 of 367 days had a TEMPO scan that met both the timing and coverage requirements. These days were used for residual correction and direct prediction approaches described in this study.

6 Forecasting Approaches

6.1 Residual Correction

The residual correction approach was taken in order to use the high quality data from TEMPO as a ground truth for the Aurora generated forecasts. The idea was to run the foundation model on CAMS data at a lower resolution to generate approximate NO_2 forecasts that can be corrected by the high resolution ground truth of TEMPO. After calculating the residual, the model was trained on these values for each data point at each longitude and latitude for all numerical variables. The model was then tested to predict the residual at the following time step. The predicted residual is

added back to the original values, which results in a full prediction value for each data point at the time step. In this case, we tested at time T+1, T+3, and T+5, where each time step is 12 hours. Each machine/deep learning model was attempted with the residual correction approach to have consistent comparison across models.

6.2 Direct Prediction

The direct prediction approach was done secondarily as a way to compare the specific task of forecasting NO_2 from the large multipurpose foundation model, Aurora, to simpler machine learning models designed for the task. This approach involved the same steps as residual correction, but without the inclusion of Aurora forecasts. The models were developed to receive the same time series data set that was split for two months of testing dates. The models were tasked with directly predicting the TEMPO NO_2 values at the subsequent time steps of T+1, T+3, and T+5 where T is a 12 hour time step. The models were trained without the Aurora values entirely. The RMSE and MAE were obtained from predicted values versus the TEMPO ground truth values. Each of the three models were trained and tested at each time step for the direct prediction approach.

7 Machine Learning Models

7.1 Random Forest

Random Forest is an ensemble machine learning algorithm that combines predictions from many decision trees. Each tree is trained using randomized samples of the training data and subsets of the input features. For regression, the predictions from the individual trees are averaged to produce the final output. This approach can model nonlinear relationships while reducing the variance and overfitting associated with a single decision tree. Random Forest has also been used successfully in previous air-quality prediction research [13].

In this study, Random Forest used the same CAMS atmospheric, spatial, and date-based input features for both forecasting approaches. For residual correction, the target was the difference between TEMPO $tcNO_2$ and Aurora $tcNO_2$, and the predicted residual was added back to Aurora to generate the corrected forecast. For direct prediction, the target was TEMPO $tcNO_2$ itself, with Aurora excluded from the model inputs. This kept the input information consistent between approaches while changing only the prediction target.

The Random Forest model used 100 trees, a maximum depth of 20, a minimum of 5 samples per leaf, and a random seed of 42. The same configuration was used for T+1, T+3, and T+5 and for both residual correction and direct prediction. Performance was evaluated using MAE, RMSE, and bias against TEMPO observations.

7.2 Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting (or XGBoost for short) is an open source machine learning algorithm that uses decision trees. It leverages the loss minimizing of gradient descent to make decisions on data sets. Because of its scalability and ability to capture non-linear relationships, it is advantageous for running on air pollution forecasting problems [9].

In this study, XGBoost is trained on CAMS data over the year as long as a TEMPO scan and CAMS scan exists for the date. For both approaches, the model targets NO_2 in regards to a residual (the difference of TEMPO NO_2 - Aurora forecast NO_2) or the ground truth itself (TEMPO NO_2

values). The model utilized the same hyperparameter configuration for both approaches along with the same input variables.

In terms of hyperparameters, the model was designed with lambda regularization to one to prevent serious overfitting due to the high dimensionality. The estimators were 1000, and the learning rate was 0.01 to allow the model plenty of time for adjustment. The evaluation metrics used were Mean Absolute Error and Root Mean Squared Error as well as bias.

7.3 Normalizing Flow

A normalizing flow is a deep learning model that represents a probability distribution by transforming a simple distribution into a more complex one through a series of reversible transformations. Unlike Random Forest and XGBoost, which produce a single point estimate, a normalizing flow models a full probability distribution over possible target values. This allows the model to capture more complex patterns in the target, including asymmetric or irregular distributions.

In this study, the normalizing flow is conditioned on the CAMS input features described in the Problem Definition (**4**) section. The target variable depends on the forecasting approach. In both cases, the model learns a conditional distribution for the target given the CAMS inputs.

Before training, both the input features and the target variable were standardized using the mean and standard decision calculated from the training data. This was done because the input and target values span a wide range of magnitudes. Standardization places the variables on a similar scale, which helps stabilize neural network training. After prediction, the standardized outputs were converted back to their original units of molecules per centimeter squared before calculating the evaluation metrics. The same training set statistics were used to standardize the test data.

The normalizing flow used in this study is a Neural Spline Flow (NSF), implemented using Zuko, a PyTorch-based library for normalizing flows. The NSF uses three transformation layers, with hidden layers containing 64 units each. The model was trained using the Adam optimizer with a learning rate of 0.001, a batch size of 4096, and 50 epochs. The training data was shuffled at the beginning of each epoch.

To improve prediction stability, an ensemble of five independently trained normalizing flow models were used. Each model used a different random initialization seed, with the final prediction being calculated by averaging the predictions from all five models. The same ensemble procedure was used for both residual correction and direct $tcNO_2$ prediction.

8 Experimental Setup

8.1 Training and Test Split

The training and testing split is crucial to the development of machine learning models. The inclusion of certain variables, the percentage of the split, and how the data is pulled from the data set is integral to the best predictive results for the models. The data set was comprised of individual pixels as rows and columns of the input features at both 12 UTC and 00 UTC respectively. Each pixel is located and kept separate by their latitude and longitude over the bounding box of the western United States. Due to the file matching of available satellite data, and subsequent quality and cloud coverage filtering, the amount of usable days for the data set was lowered from the preliminary full year (365 days) to 328 days of data.

The split was organized to be as close to 80 percent training and 20 percent testing as possible while still maintaining each single day as a complete set of pixels. The issue with splitting randomly is that each pixel row would be mixed across days which would remove the spacial and temporal

aspect of the data. Instead, the split was made to keep the testing consistent by only taking the last two months as the testing dates. This allows the models to potentially learn how the variables and target variable change over seasons and long periods of time. Then, it is tested to see how well it can adjust its predictions to two new months of data.

8.2 Evaluation Metrics

The evaluation metrics are used to understand and compare the results received from the models after testing. Since each model is being tested on the same data set and with the same input parameters, they also share the evaluation metrics as a standard for comparison. The metrics chosen for the three models were mean absolute error and root mean squared error. MAE was picked due to its stability and robustness to outliers which is beneficial for this data with a large range and magnitude. RMSE was included because of its recommendation for Aurora, as well as being a counter metric by penalizing large errors or deviations. Together, they give a simple point of comparison to see which one is predicting closest to TEMPO over all of the pixels per day.

9 Results

This section presents the performance of the three machine learning models under both residual correction and direct prediction, evaluated at each of the three forecast lead times. Results are organized by context type. The Table subsection presents the numerical comparison across all models and lead times. The Distribution Plots subsection shows the spread of daily bias and RMSE across test days. The Maps subsection shows spatial comparisons for a representative test day at each lead time.

9.1 Numerical Results

Each of Tables 9.1, 9.2, 9.3 below reports six metrics for a given lead time: Aurora's baseline MAE and RMSE, the residual-corrected MAE and RMSE, and the direct prediction MAE and RMSE. The aurora baseline values are identical across all three models at a given lead time, since they represent Aurora's own uncorrected forecast against TEMPO, independent of which correction or direct model is applied afterward. In each table, the best metrics of each ML model are in bold.

9.1.1 T+1

Table 9.1: T+1 (12 h Lead) Forecast Error Against TEMPO

	Aurora Baseline		**Residual Corrected**		**Direct Prediction**	
Model	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
XGBoost	7.82	9.45	4.17	5.92	**4.00**	**5.69**
Random Forest	7.82	9.45	4.30	6.08	**3.97**	**5.70**
Normalizing Flows	7.82	9.45	**4.38**	**6.14**	4.58	6.31

At T+1, direct prediction achieves lower RMSE than residual correction for both XGBoost and Random Forest. Normalizing Flows shows the opposite pattern, with residual correction slightly outperforming direct prediction.

Table 9.2: T+3 (36 h Lead) Forecast Error Against TEMPO

	Aurora Baseline		Residual Corrected		Direct Prediction	
Model	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
XGBoost	12.70	13.94	4.14	5.90	4.14	**5.83**
Random Forest	12.70	13.94	4.32	6.08	**4.10**	**5.82**
Normalizing Flows	12.70	13.94	4.57	6.32	**4.54**	**6.29**

At T+3, direct prediction achieves lower RMSE than residual correction for all three models, though the margin for XGBoost and Normalizing Flows is small.

Table 9.3: T+5 (60 h Lead) Forecast Error Against TEMPO

	Aurora Baseline		Residual Corrected		Direct Prediction	
Model	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
XGBoost	15.42	16.47	**4.03**	**5.78**	4.30	5.97
Random Forest	15.42	16.47	4.25	6.01	**4.19**	**5.88**
Normalizing Flows	15.42	16.47	**4.41**	**6.15**	5.20	6.95

At T+5, the pattern reverses for two of the three models. Residual correction achieves lower RMSE than direct prediction for both XGBoost and Normalizing Flows, while Random Forest continues to favor direct prediction.

9.2 Distribution Plots

In Figure 1a, each distribution plot shows two panels side by side for a given lead time and model: the distribution of daily mean bias as a percentage, and the distribution of daily RMSE. Each panel compares three variants, Aurora's baseline forecast, the residual-corrected prediction, and the direct prediction, across all test days.

A clear trend emerges across lead times. Aurora's baseline bias shifts substantially further negative as lead time increases, clustering mostly between -20% and -5% at T+1, and shifting to mostly between -40% and -25% by T+5. This visually confirms Aurora's own forecast accuracy degrading with increasing lead time, consistent with the pattern discussed in the Conclusions section.

9.3 Forecast Visualization

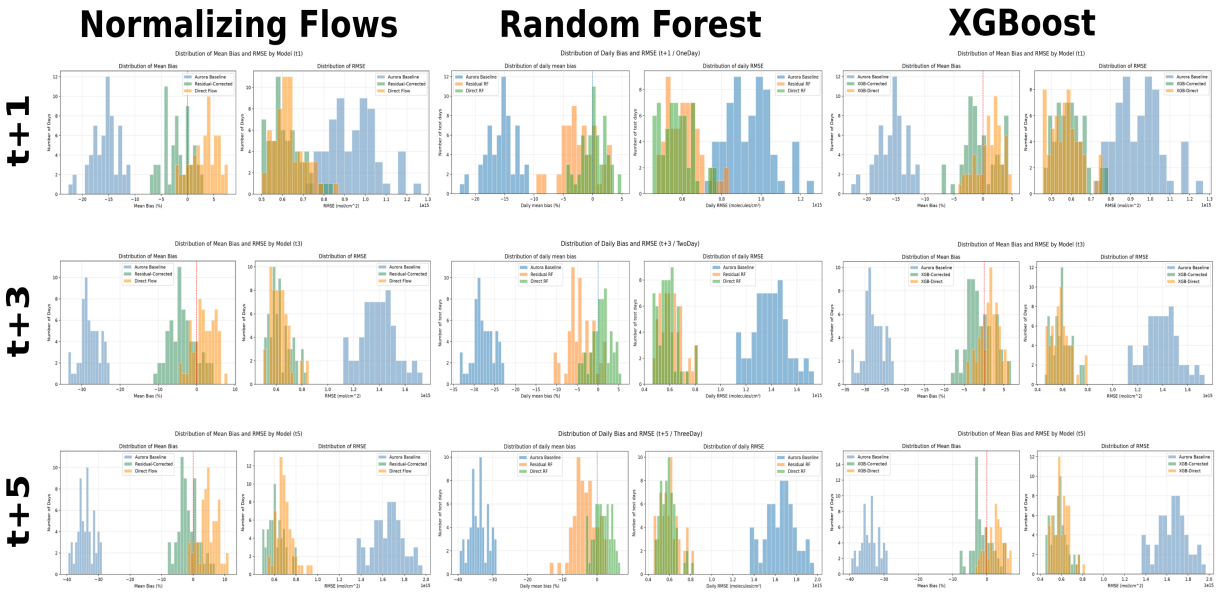
The maps in Figure 1b show a spatial comparison for a representative test day at each lead time, showing TEMPO ground truth, Aurora's baseline forecast, the residual-corrected prediction, and the direct prediction side by side on a shared color scale.

10 Conclusions and Future Work

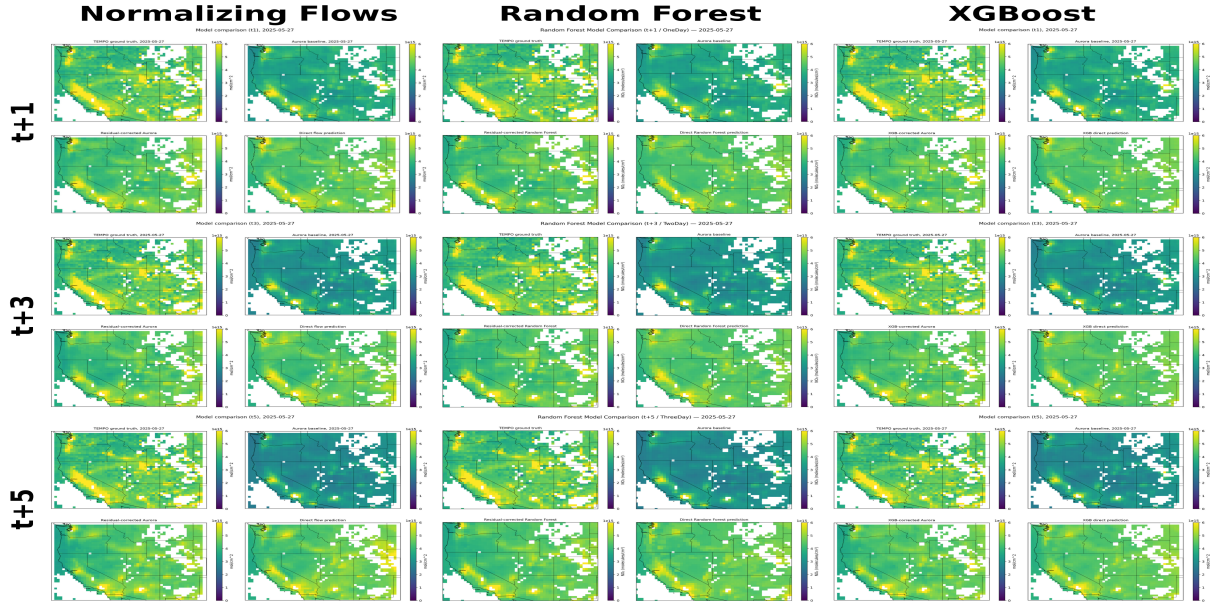
This study offers a direct comparison between correcting Microsoft's Aurora foundation model and predicting NO_2 independently of it. Three machine learning models were evaluated across two forecasting approaches and three lead times. From the current work, the following conclusions are drawn:

1. The relative performance of the residual correction and direct prediction approaches varies across models and forecast lead times. For longer lead times, residual correction performs slightly better, while for shorter lead times, direct prediction performs slightly better.

2. Among the three machine learning models evaluated, XGBoost and Random Forest generally perform better than Normalizing Flows. XGBoost and Random Forest achieve relatively



(a)



(b)

Figure 1: Visual summary of model performance: (a) distribution plots of daily mean bias (%) and daily RMSE across test days for T+1 (12 h), T+3 (36 h), and T+5 (60 h) leads; (b) spatial forecast visualizations for a representative test day at each lead time.

similar results overall, with XGBoost performing slightly better under residual correction and Random Forest performing slightly better under direct prediction.

11 Conclusions and Future Work

This study offers a direct comparison between correcting Microsoft's Aurora foundation model and predicting NO_2 independently of it. Three machine learning models were evaluated across two forecasting approaches and three lead times. From the current work, the following conclusions are drawn:

1. The relative performance of the residual correction and direct prediction approaches varies across models and forecast lead times. For longer lead times, residual correction performs slightly better, while for shorter lead times, direct prediction performs slightly better.

2. Among the three machine learning models evaluated, XGBoost and Random Forest generally perform better than Normalizing Flows. XGBoost and Random Forest achieve relatively similar results overall, with XGBoost performing slightly better under residual correction and Random Forest performing slightly better under direct prediction.

Future work includes evaluating model performance specifically during wildfire events, since this study assessed general atmospheric conditions rather than sudden NO_2 spikes that motivated this work. Further improvements to NO_2 prediction could also be explored using other types of models and additional training data, such as foundation model fine-tuning, patch-based deep learning models such as Vision Transformers and Swin Transformers, or physics-informed models. This research can be accessed through the UMBC Big Data REU GitHub Repository [**https://github.com/big-data-lab-umbc/big-data-reu**]

Acknowledgments

This work is supported by the grant "REU Site: Online Interdisciplinary Big Data Analytics in Science and Engineering" from the National Science Foundation (grant no. OAC–2348755). We acknowledge the computational resources in the UMBC High Performance Computing Facility (hpcf.umbc.edu) and the financial contributions from NIH, NSF, CIRC, and UMBC for this work.

References

[1] Cristian Bodnar, Wessel P. Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A. Weyn, Haiyu Dong, Jayesh K. Gupta, Kit Thambiratnam, Alexander T. Archibald, Chun-Chieh Wu, Elizabeth Heider, Max Welling, Richard E. Turner, and Paris Perdikaris. A foundation model for the earth system. *Nature*, May 2025.

[2] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori

Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir P. Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *ArXiv*, 2021.

[3] Elton L Cao. National ground-level no2 predictions via satellite imagery driven convolutional neural networks. *Frontiers in Environmental Science*, 11:1285471, 2023.

[4] Shengchao Chen, Guodong Long, Jing Jiang, Dikai Liu, and Chengqi Zhang. Foundation models for weather and climate data understanding: A comprehensive survey. *arXiv preprint arXiv:2312.03014*, 2023.

[5] Gonzalo González Abad, Caroline Nowlan, Huiqun Wang, Heesung Chong, John Houck, Xiong Liu, and Kelly Chance. Trace gas and cloud level 2 and 3 data products: User guide. User Guide Version 2.1, National Aeronautics and Space Administration, January 2026.

[6] A. Inness, M. Ades, A. Agustí-Panareda, J. Barré, A. Benedictow, A.-M. Blechschmidt, J. J. Dominguez, R. Engelen, H. Eskes, J. Flemming, V. Huijnen, L. Jones, Z. Kipling, S. Massart, M. Parrington, V.-H. Peuch, M. Razinger, S. Remy, M. Schulz, and M. Suttie. The cams reanalysis of atmospheric composition. *Atmospheric Chemistry and Physics*, 19(6):3515–3556, 2019.

[7] Daejin Kim, Youngin Cho, Dongmin Kim, Cheonbok Park, and Jaegul Choo. Residual correction in real-time traffic forecasting. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 962–971, 2022.

[8] Wu Liu, Xiaodong Li, Zuo Chen, Guangming Zeng, Tomás León, Jie Liang, Guohe Huang, Zhihua Gao, Sheng Jiao, Xiaoxiao He, et al. Land use regression models coupled with meteorology to model spatial and temporal variability of no2 and pm10 in changsha, china. *Atmospheric Environment*, 116:272–280, 2015.

[9] Jinghui Ma, Zhongqi Yu, Yuanhao Qu, Jianming Xu, and Yu Cao. Application of the xgboost machine learning method in pm2. 5 prediction: A case study of shanghai. *Aerosol and Air Quality Research*, 20(1):128–138, 2020.

[10] Microsoft. Microsoft predictions for air pollution, 2024.

[11] Ankita Mishra and Yogesh Gupta. Comparative analysis of air quality index prediction using deep learning algorithms. *Spatial Information Research*, 32(1):63–72, 2024.

[12] Takao Yoshikane and Kei Yoshimura. A bias correction method for precipitation through recognizing mesoscale precipitation systems corresponding to weather conditions. *PLoS Water*, 1(5):e0000016, 2022.

[13] Ruiyun Yu, Yu Yang, Leyou Yang, Guangjie Han, and Oguti Ann Move. Raq–a random forest approach for predicting air quality in urban sensing systems. *Sensors*, 16(1):86, 2016.