Neural network emulation of the formation of organic aerosols based on the explicit GECKO-A chemistry model

John S. Schreck[1], Charles Becker[1], David John Gagne[1], Keely Lawrence[1], Siyuan Wang[2,3], Camille Mouchel-Vallon[4], Jinkyul Choi[5], Alma Hodzic[1]

[1]National Center for Atmospheric Research (NCAR), Boulder, CO, USA
[2]Cooperative Institute for Research in Environmental Sciences (CIRES), University of Colorado, Boulder, CO, USA
[3]National Oceanic and Atmospheric Administration (NOAA), Chemical Sciences Laboratory (CSL), Boulder, CO, USA
[4]Laboratoire d'Arologie, Universit de Toulouse, CNRS, UPS, Toulouse, France
[5]Environmental Engineering Program, University of Colorado, Boulder, CO, USA

Key Points:

- Incorporation of explicit organic chemistry into 3D chemistry-simulations requires emulation.
- We developed two types of neural network emulators for the GECKO-A chemistry model.
- The emulators produced accurate and stable simulations for three precursor species.

Corresponding author: Alma Hodzic, `alma@ucar.edu`

Abstract

Secondary organic aerosols (SOA) are formed from oxidation of hundreds of volatile organic compounds (VOCs) emitted from anthropogenic and natural sources. Accurate predictions of this chemistry are key for air quality and climate studies due to the large contribution of organic aerosols to submicron aerosol mass. Currently, only explicit models, such as the Generator for Explicit Chemistry and Kinetics of Organics in the Atmosphere (GECKO-A), can fully represent the chemical processing of thousands of organic species. However, their extreme computational cost prohibits their use in current chemistry-climate models, which rely on simplified empirical parameterizations to predict SOA concentrations. Recent applications of atmospheric chemistry emulation with machine learning (ML) applied to the simpler chemical mechanisms of tropospheric ozone have shown its ability to produce realistic predictions and significantly reduce the computational cost. This study proves that ML can accurately emulate SOA formation from an explicit chemistry model for several precursors with 100 to 100,000 times speedup over GECKO-A, making it computationally usable in a chemistry-climate model. To train the ML emulator, we generated thousands of GECKO-A box simulations sampled from a broad range of initial environmental conditions, and focused on the chemistry of three representative SOA precursors: the oxidation by OH of two anthropogenic (toluene, dodecane), and one biogenic VOC (alpha-pinene). We compare fully-connected and recurrent neural network methods and use an ensemble approach to quantify their underlying uncertainty and robustness. The SOA predictions generally remain stable over a simulation period of 5 days with an approximate error of 2-8%.

Plain Language Summary

Detailed and accurate representation of organic aerosol chemistry is needed to predict the effect of atmospheric aerosols formed from natural and anthropogenic sources on both human health and climate. Ideally, these complex representations of chemistry would be directly included within state-of-the-art weather and climate models to get a fully coupled system with meteorological and climatological feedback all over the globe. However, we are many years away from having the computational power needed to run such fully coupled large-scale simulations due to the complexity of organic chemistry, which involves hundreds of thousands of organic gaseous and particle species and chemical reactions. As a potential solution, we test an approach that uses a neural network to mimic

the solution of an explicit representation of organic chemistry which would be computationally feasible to link with current air quality and climate models.

1 Introduction

Secondary organic aerosols (SOA) have been an active area of research in the past decade with the goal of improving their representation in air quality and climate models (Tsigaridis et al., 2014; Hodzic et al., 2016), which is essential for predicting their effect on human health (Mauderly & Chow, 2008) and their contribution to radiative forcing in the climate system (Boucher et al., 2013). The misrepresentation of SOA formation pathways in 3D models has led to a long-standing discrepancy between observed and modeled organic aerosol concentrations that has been reported from urban to remote regions (de Gouw, 2005; Hodzic et al., 2020). Unlike sulfate and other inorganic aerosols, which are made from a few dominant chemical pathways, SOAs result from the condensation of a very large number of partly oxidized gases. These gases are generated from the multi-generational oxidation of volatile organic compounds (VOCs) emitted from anthropogenic and natural sources. This complexity is not included in current 3D models that rely on simplified SOA parameterizations that have been developed and optimized based on laboratory measurements or ambient aircraft data (Ng et al., 2007; Hodzic & Jimenez, 2011). This empirical approach does not include the mechanistic understanding of processes leading to SOA formation, and the adequate sensitivity to environmental variables modulating SOA concentration.

Detailed chemistry models such as the widely used near-explicit Master Chemical Mechanism (MCM) (Jenkin et al., 2003) or the "fully-explicit" Generator of Explicit Chemistry and Kinetics of Organics in the Atmosphere (GECKO-A) (Aumont et al., 2005; Camredon et al., 2007) provide a mechanistic representation of the organic aerosol chemistry and relevant process, and lead to an improved agreement with ambient SOA measurements (Lee-Taylor et al., 2011; Mouchel-Vallon et al., 2020). Chemical mechanisms generated by GECKO-A typically include millions to tens of millions of reactions, and hundreds of thousands intermediate species (Aumont et al., 2005). MCM mechanisms are handwritten and much smaller than GECKO-A ones as they represent only 2-3 first generations of chemistry. Due to the remarkable computational cost, no air quality models or chemistry-climate models can afford to run with GECKO-A included in the foreseeable future. To our knowledge, only the study by (Li et al., 2015) attempted to in-

clude the MCM organic chemistry mechanism into a regional 3D model and faced computational challenges.

Recent years have seen quite a few inspiring applications in developing machine learning emulators using explicit/process-level models and implementing the trained emulators into large-scale models (Brenowitz & Bretherton, 2018; Beucler et al., 2020; Gettelman et al., 2021). Replacing complex processes with ML emulators have potential advantages by learning non-linear relationships that can represent underlying physical or chemical processes not captured in simple empirical characterizations, as well as multiple orders of magnitude speedups in computation when compared to fully coupled process-based models. However, maintaining both an acceptable level of accuracy and a system that remains numerically stable through an adequate amount of time with emulators remains challenging.

Current efforts in atmospheric chemistry emulation with machine learning (ML) have focused on inorganic gas-phase chemistry, such as ozone within GEOS-Chem (Kelp et al., 2018; Keller & Evans, 2019). Using random forest regression and neural network models they were able to reproduce the hourly concentration of 77 gaseous species predicted by the GEOS-Chem chemical mechanism, with a significantly reduced computational expense (250 times). However, the emulator for gas-phase chemistry was subject to runaway errors and numerical instability, as well as performance degradation on out-of-domain inputs. In a follow-up study, (Kelp et al., 2020) used a neural network with a recurrent training approach, where a multi-time step loss function was used in conjunction with dimensionality reduction of the chemical system, that resulted in observed reduced error accumulation and provided greater stability. The use of recurrent neural networks was not reported in any of these studies.

Our primary aim in this work is to extend atmospheric chemistry emulation to organic aerosols for which current climate models do not currently account for. Additionally, this proof of concept study evaluates two different types of neural network architectures: (1) a feed-forward, fully-connected network and (2) a recurrent neural network (RNN). Both models were designed to be feasibly integrated into current 3D transport models, that can provide fast and accurate predictions for organic aerosol concentrations. The RNN was chosen to determine if it could help to overcome numerical stability problems, as observed by others using fully-connected model architectures (Brenowitz & Brether-

ton, 2018; Kelp et al., 2020), as they come equipped with feedback connections that can store information about previous events in the form of a latent vector, e.g. RNNs possess memory (Hochreiter, 1991). As these architectures were developed to learn representations of sequential data to solve temporal problems, they offer the ability to use multi-length inputs. However, incorporating multi-length inputs into 3D models would require us to dissociate chemistry production from other processes (e.g. transport, removal). We address this with the development of a novel "1-step" RNN that only requires a single time step of input, but relies on a separate simple neural network to initialize the hidden state vector for the very first time-step.

The paper is organized as follows: Section 2 outlines the data generation, training, hyperparameter tuning, and evaluation procedures for the reference model and both neural network types. In Section 3, we characterize the two models performance relative to the GECKO-A data sets for different precursor species, and compared to each other to assess the overall strengths and weaknesses of each model architecture. Both model types are also tested on data sets that help assess the ability of the models to generalize into new domains. Sections 4 and 5 provide a brief discussion about the pros and cons of these different model architectures and the ongoing challenges regarding numerical stability, computational cost, and interpretability for emulating SOA production with ML.

2 Methods

2.1 Description of the reference model

To provide reference chemical mechanisms, we used the GECKO-A chemical generator (Aumont et al., 2005; Camredon et al., 2007), which describes in great details the chemical oxidation of organic compounds in the atmosphere. The resulting chemical mechanisms for each SOA precursor species are complete (down to the ultimate products CO_2 and H_2O), and explicit by preserving knowledge of the molecular structures of all the intermediate compounds. Compared to other widely used semi-explicit chemical models such as MCM, GECKO-A can consider many generations of oxidative chemistry i.e., 20 generations are considered here (Lee-Taylor et al., 2011). This has important implications for the formation of organic aerosols as SOA formation arises from a multitude of partly oxidized compounds, rather than from a few dominant molecules. In addition,

the SOA formation timescale varies greatly for different precursors. For example, at ambient conditions, it takes only a few hours to form SOA from dodecane vs. several days to form SOA from toluene (Hodzic et al., 2014). For the gas-particle partitioning of organic molecules, dynamic partitioning is used. It is reasonable to consider GECKO-A simulations as a benchmark for building an emulator for SOA chemistry given its reasonable agreement with observations shown for both comparisons with chamber measurements e.g. for alkanes and alkenes compounds (La et al., 2016) and ambient measurements e.g. during MIRAGE, BEACHON and Go-AMAZON, (Lee-Taylor et al., 2011, 2015; Mouchel-Vallon et al., 2020).

2.2 GECKO-A generated training datasets

We ran the GECKO-A model with systematically varied input parameters to generate the dataset used to train the machine learning emulator. At this stage we focus on three representative SOA precursors: toluene, dodecane, and α-pinene. Toluene and dodecane are emitted from a wide range of anthropogenic sources, while α-pinene is one of the major SOA precursors emitted by vegetation. These compounds together contribute substantially to the global SOA burden. We generate chemical mechanisms and corresponding datasets for these precursors including the OH oxidation mechanisms of toluene, dodecane and α-pinene. For each oxidation mechanism, the reactions of the precursor with oxidants other than OH were not considered. Thus, the considered chemistry is mostly representative of daytime conditions.

Based on our current understanding of atmospheric chemistry and the common chemistry-climate modeling frameworks, we identified the following six variables that are key to predicting SOA formation from VOC oxidation under tropospheric conditions: (1) temperature, (2) solar zenith angle, (3) pre-existing aerosol mass, (4) ozone concentrations, (5) nitrogen oxides (NOx) concentrations, and (6) OH radical concentrations. The range of variability considered for these parameters and the associated sampling scheme is summarized in Table 1 and illustrated in Figure 1. Additionally, a diurnal variation in temperature of an average 5 degrees amplitude is used. In each training data set, we use the Latin Hypercube sampling approach to obtain two-thousand environmental input combinations. Temperature, solar zenith angle, ozone, and OH were all sampled uniformly, whereas pre-existing aerosol concentrations and NOx were sampled as a logarithmic distribution. The combination of these ranges is relevant for a wide range of tropospheric

Temperature	240 - 320 K	Uniform
Solar zenith angle (SZA)	0-90 degrees	Uniform
Pre-existing aerosols	0.01-10 µg/m^3	Logarithmic
Ozone	1 - 150 ppb	Uniform
NOx	0.01-10 ppb	Logarithmic
OH	10^1–10^6 molecules/cm^3	Uniform

Table 1. Environmental parameter ranges used in GECKO-A simulations.

conditions, from remote to moderately polluted environments. These environmental variables, with the exception of temperature, are held constant in a given 5-day GECKO-A box model simulation in an attempt to coerce the ML models to generalize better and be more robust to over fitting.

In each dataset, the initial concentrations of a given SOA precursor were set to an arbitrary low value of 10 ppt similar to previous studies (Lannuque et al., 2018; Hodzic et al., 2014, 2015). Although the tropospheric concentrations of the precursor can be higher in polluted regions, this low value is representative of the remote atmosphere, and was chosen so that the amount of aerosol produced from the given precursor is small compared to preexisting OA and will not impact the gas/particle partitioning, nor the overall photochemical reactivity. Precursor spans several orders of magnitude in our 5-day GECKO-A simulations as it decays exponentially and is effectively consumed before the end of each simulation leading to the production of thousands of intermediate organic gases. As shown on Figure 1, complexities of the organic chemistry are illustrated by the wide variation of produced SOA mass with respect to each environmental variable. For example, significantly higher SOA mass concentrations are produced at colder temperatures.

As the precursor mass is exponentially distributed, before using the precursor data as input to the ML model, we transform the precursor values by taking the base-10 logarithm to avoid any stiffness in the system. Next, each input variable X_j in the training data, including chemical concentrations and environmental variables, were standardized independently into z-scores according to the formula $X_j = (X_j - u)(X_j - s)^{-1}$, where u and s are the mean and standard deviation of X_j. Standardization was chosen

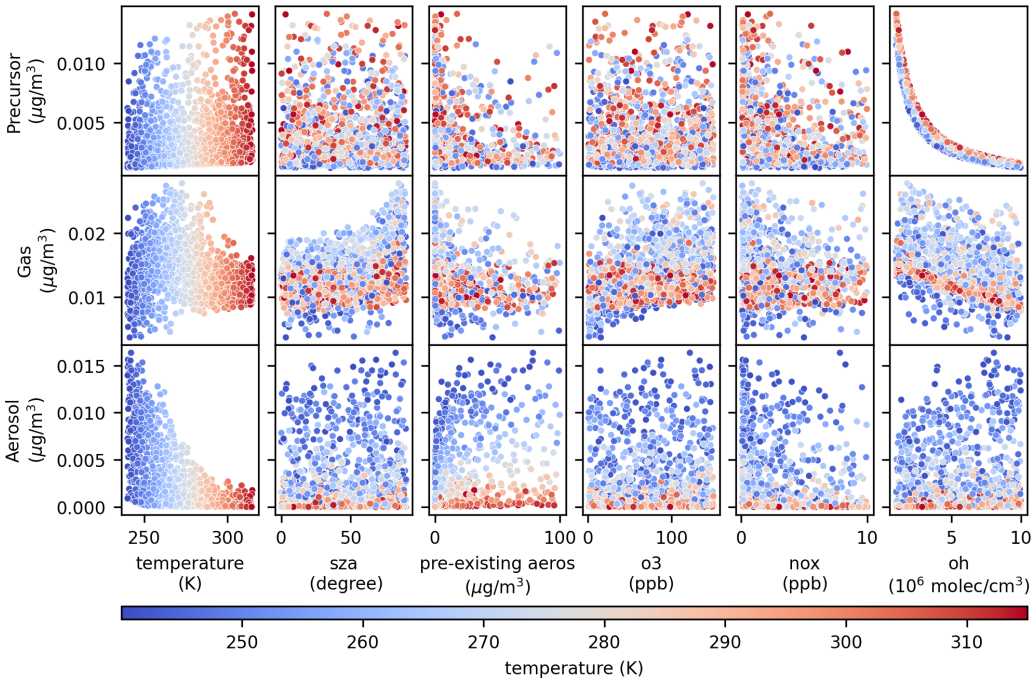


Figure 1. Training distributions (mean value through 5-days) of targets (rows) vs. environmental variables (columns).

over other common transformations because many features are not normally distributed in the data sets for the three species. Hence the transformation recasts the values of the input and output channels into a format where the values of each variable are centered and have similar spread. This is especially important when computing the error on the model predictions against the training data values when the weights in the model are being updated, for example, to prevent the model from over-fitting on the quantities having the largest spread.

A total of 2,000 experiments were run in GECKO-A for each precursor species, and output every five minutes over the course of five days. Thus, a total of 2,880,000 samples were generated per species. However, as the target variables are a subset of the input feature variables at the previous time step, we removed the first (final) time step of the output (input) variables from each experiment leaving a total of 2,878,000 samples. These simulations were then split into three subsets: training (80%), validation (10%), and testing (10%). The training set is used to optimize the weights of an ML model, while the validation set is used to select the hyperparameters, or meta-settings describing the

ML model architecture, such as the number of neurons in the hidden layer, that result in the best-performing model across the space of possible models (see below). The final testing set is not used to train or adjust the machine learning models and is only used in the final evaluations described herein.

2.3 Neural network models

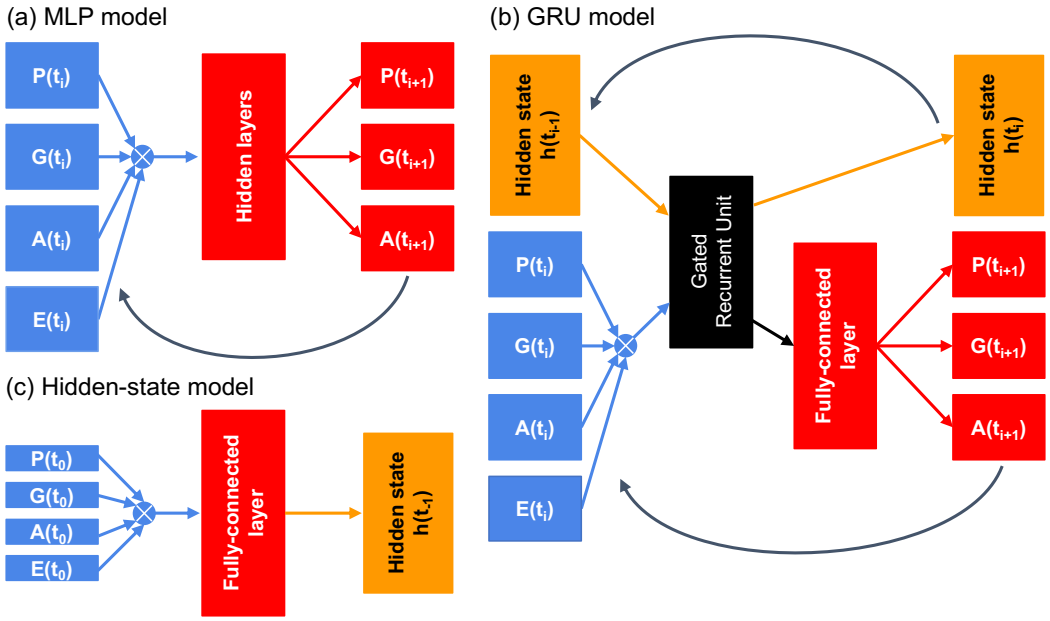


Figure 2. Model architectures showing the (a) MLP (multi-layer perceptron) and (b) GRU (Gated Recurrent Unit) models for predicting concentrations, and (c) the model for the initialization of the recurrent hidden state given an initial condition as input. In all three models, inputs at time t_i are colored blue (precursor, gas, and aerosol, and the environmental variables). The inputs are concatenated into a vector as illustrated by the cross-marked blue dot. The model outputs at time t_{i+1} are colored red in (a) and (b) for precursor, gas, and aerosol, and orange in (b) and (c) for the hidden state. In (a) and (b) when the models are used in box simulations, the black arrow represent using the output prediction from the model along with the environmental variables as input to the model for the next time step prediction.

Figure 2 shows several neural network models that we consider here as emulators for GECKO-A. In Figure 2(a), a multi-layer perceptron (MLP) architecture is shown (referred to as the "MLP model"). The MLP model accepts as input the scaled values of the precursor, gas, aerosol, and environmental variables at time t_i (blue boxes in the fig-

ure). These quantities are concatenated into a vector, and denoted $X(t_i) \equiv (P(t_i), G(t_i), A(t_i), E(t_i))$. The model outputs the precursor, gas, and aerosol values at time $t_{i+1} = t_i + \delta t$ (red boxes in the figure), denoted $Y(t_{i+1}) \equiv (P(t_i + \delta t), G(t_i + \delta t), A(t_i + \delta t))$, where δt here is 300 seconds. The MLP is an artificial neural network that contains, in addition to input and output layers, at least one hidden layer (horizontal red block in Figure 2(a)), and is the simplest neural network architecture available. Each hidden layer contains a set of perceptrons, which mathematically are linear regressions on the outputs of the previous layer followed by a non-linear transformation called the activation function. For our MLP, we use the Rectified Linear Unit (ReLU) activation function, which sets negative values to 0 and allows positive values to pass through unchanged. The final hidden layer is connected to the output layer, which is a linear regression on the hidden layer outputs. We tested using multiple hidden layers but found that a single hidden layer produced the lowest validation set error. The MLP model for each of the precursor species make future predictions for the values of the chemical quantities of interest, using only the current chemical state of the atmosphere. As such, any MLP model satisfies the Markovian condition, and possesses no memory about atmospheric chemical states visited in the past beyond the previous timestep.

As GECKO-A generates sequential trajectories describing the evolution of species' concentrations over time, we have also applied a recurrent neural network (RNN) model to investigate whether it may have an advantage over the MLP model due to its ability to utilize its memory about the past at $\{t_{i-n}, ..., t_{i-1}\}$ to make the next prediction at t_i. As such, a temporal model may be better equipped compared to the state-less MLP model to describe the changes occurring to the quantities over time, in addition to limiting and/or preventing runaway error propagation. The input to an RNN is a sequence and a hidden state. The sequence elements could be scalars, vectors, or other higher-dimensional tensors. The first (or last) element in the sequence is ingested by the RNN along with a starting hidden state, producing an encoding of the element and a hidden state for the current encoding. The next element in the sequence is then used as input along with the current hidden state, which produces an encoding of the second element and another hidden state. This encoding represents not just the second element, but the elements that came before it, due to the fact that the model leverages its feedback connections to produce the encoding. This process continues until all of the elements in the sequence have

been seen by the model, which produces a final encoding of the entire sequence, as well as a hidden state.

Figure 2(b) illustrates a model architecture that combines an RNN layer type called a Gated Recurrent Unit (GRU) (Chung et al., 2014b) with a fully-connected layer to make chemical concentration predictions. We refer to this model as the GRU model. The GRU layer performs three key operations: filtering the contents of the hidden vector from the previous time, calculating a new hidden state from a combination of the filtered hidden state and new inputs, and finally calculating a new output. The GRU is similar to the well-known long-short term memory (LSTM) model (Hochreiter & Schmidhuber, 1997) but has fewer parameters to learn. Even so, the GRU often performs comparably to the LSTM in language modelling tasks (Chung et al., 2014a).

Similar to the MLP, the GRU model shown in Figure 2(b) illustrates the model at the t_i time step such that t_{i-1} came before it, accepting as input $(P(t_i), G(t_i), A(t_i), E(t_i))$ at time t_i and hidden state $h(t_{i-1})$. The GRU layer produces an encoded representation of the input $X(t_i)$ denoted $Y^*(t_i)$, and a hidden state $h(t_i)$ for the current time, which has the same dimension as $h(t_{i-1})$. The GRU layer avoids the vanishing gradient problem by computing these quantities according to

$$
\begin{aligned}
z(t_i) &= \sigma\left(W_z h(t_{i-1}) + U_z X(t_i) + b_z\right) \\
r(t_i) &= \sigma\left(W_r h(t_{i-1}) + U_r X(t_i) + b_r\right) \\
c(t_i) &= \tanh\left(W_c \left(r_c \odot h(t_{i-1})\right) + U_c X(t_i) + b_c\right) \\
h(t_i) &= z(t_i) \odot h(t_{i-1}) + \left(1 - z(t_i)\right) \odot c(t_i) \\
Y^*(t_i) &= \operatorname{softmax}\left(W_y h(t_i) + b_y\right)
\end{aligned}
$$

where the sigmoid function $\sigma(x) = (1 + \exp^{-x})^{-1}$ projects input values to be within [0, 1]. The softmax function for a K-component z is $\exp(z_k) \,/\, \sum_{j=1}^{K} \exp(z_j)$. The tensors W_* and U_*, and bias terms b_*, contain the fit parameters that are updated through back-propagation during training. The $\odot$ symbol refers to element-wise multiplication.

The quantity $z(t_i)$ is the update gate, and $r(t_i)$ is the reset gate, which determines how much information over past time steps to forget. The quantity $c(t_i)$ represents the current memory content in the layer and utilizes the reset gate to store information from the past. The equation for the current hidden state $h(t_i)$ uses the update gate to determine how much information from the current time step to collect and how much to col-

lect from past time steps. The encoded information at the current time step, $Y^*(t_i)$, is computed using the current hidden state. Then, as Figure 2(b) illustrates, it is passed through a ReLU activation function (black arrow), and then through a fully-connected layer (tall red box), which outputs the scalar precursor, gas, and aerosol values at the time t_{i+1}.

It is common that the input sequence to an RNN is longer than length one, in which case a hidden state can be immediately informed by the sequence (trajectory) to make the next prediction. Applied to GECKO-A trajectories, we are free to choose the length of the input sequence, which does not have to be fixed. Indeed, in our GECKO-A box model simulations concentrations of organic species are only undergoing chemistry processing, whereas in 3D models other processes (e.g., transport, dry and wet removal) are included. Thus, we must also consider the added complexity of incorporating RNNs into 3D transport models. For a RNN that uses a sequence for input, each chemical variable needs to be transported and stored for the number of time steps needed as input, which could be memory intensive and programmatically challenging.

Here we describe a "1-step" approach, which means that when incorporated into a 3D climate simulation, the RNN is similar to the MLP in that only a single time step of input is required. The only difference being that a single hidden state is also input to the RNN, which can be understood mainly as a larger input compared to the MLP. Unlike the MLP, the RNN hidden state vector needs to be stored at every model grid cell to inform the calculation of the next time step. Depending on the size of the hidden state, this could create a large memory burden on the simulation but would be less disruptive to simulation codes than creating and managing multiple copies in time of every model field.

When there is no initial hidden state available, the initial condition $X(t_0)$ is passed through a separate MLP, referred to as the hidden-state model, to obtain $h(t_{-1})$. Figure 2(c) illustrates that this model's architecture accepts as input values at some t_0 for precursor, gas, aerosol, and the environmental variables $(P(t_0), G(t_0), A(t_0), E(t_0))$, and outputs a hidden state $h(t_{-1})$. The hidden-state model contains one fully-connected layer with a linear activation. The input size is equal to the length of $X(t_0)$ while the output size is equal to the length of the hidden state used by the GRU.

2.4 Training procedure

Each MLP model is trained by first initializing an architecture and the trainable weights. The training data split is randomly shuffled removing the time sequence in the data. A fixed number of training data points is selected from the training data, called a batch, and then passed through the model to obtain a prediction for each point in the batch. The mean-absolute error (MAE), the training loss, is computed for the batch from the prediction. Using the loss, the weights of the model are updated accordingly using gradient descent with back-propagation, (Rumelhart et al., 1986) and a pre-specified learning rate to reduce the error. This process is repeated until all of the training samples are passed through the model once, and is referred to as one epoch of model training. At the end of every epoch, the training data is randomly shuffled. This procedure is repeated for a prescribed number of epochs.

In order to train the GRU model and the hidden-state model, the input and output data for each experiment needs to be ordered by time, thus it is not shuffled along this coordinate, as is done with the MLP. The training procedure is then similar to how the model would be used in evaluation, and starts by setting the initial condition for an experiment along with the environmental variables as the initial input, X_0 to the model. As there is no hidden state available at the beginning of a box simulation, $X(t_0)$ is passed through the hidden state model to produce $h(t_0)$, then ($X(t_0)$, $h(t_{-1})$) is passed through the GRU model to obtain the prediction $Y(t_1)$ and $h(t_0)$. For the GRU model, we use the Huber formula as the loss function for the predicted chemical concentrations, which computes the mean-squared difference between the predicted output $Y(t_1)$ and the known values produced by GECKO-A, when the difference is greater than a fixed cutoff value, and computes the MAE otherwise.

Next, the predicted output $Y(t_1)$ is concatenated with the environmental variables one time step into the future to create the input to the model $X(t_1)$. This quantity is passed through the hidden state model to obtain $h^*(t_0)$, where the $*$ notation is used to distinguish this hidden state prediction from the one that the GRU model predicted. The mean absolute difference between $h(t_0)$ and $h^*(t_0)$ is computed. The total loss for the time step adds this quantity, multiplied by a loss weight, to the loss contributions computed for the chemical quantities. The loss weight is left as a parameter to be optimized (see below). The total loss for the time step is then used with a given learning

rate value to update the adjustable parameters of both the GRU and the hidden state models in tandem. This procedure is repeated until the second-to-last time step in an experiment trajectory is used as input to the model.

During training of the GRU model, we select random experiments to create batches of data when computing the total loss at each time step. One epoch is defined as all training experiment trajectories passing through the model once, so the same data as with training the MLP model, except that the data is ordered by time (and randomized by experiment). At the end of every epoch, the model is put into evaluation mode and used to predict the trajectories for the validation experiments. The MAE is then computed between the model predictions and the validation experiments. After each epoch the validation MAE is used to measure improvement of the model predictions in two ways: (1) to anneal the learning rate if the models performance does not improve after some number of epochs, e.g. it "over-fits" on the validation experiments, and (2) to stop the training entirely once the model does not improve on the validation experiments after some number of epochs. We chose 3 and 7 epochs in (1) and (2) respectively.

2.5 Evaluation Procedure

The ability of MLP and GRU-based models to predict the time evolution of precursor, gas, and aerosol mass concentrations is evaluated by comparing the box model predictions against the benchmark values as produced by the GECKO-A model. Here each model is placed into evaluation mode, which disables any stochastic components such as the recurrent dropout used when training the GRU, and is used to make predictions on the hold-out validation set of data that was not used during the training to influence the weight updates in each model. For each validation experiment, a starting amount of precursor, gas, aerosol, and environmental variables at time t_0 is passed through the MLP network to obtain the predicted quantities at the first time step t_1, where $t_1 = t_0 + \delta t$. These predictions are then used along with the environmental variables at the next time step as the next input to the model, and so forth for the length of the experiment (see Fig. 2(b)).

2.5.1 Ensembles

Machine learning models must be stochastically initialized with random weights, as gradient descent requires variation amongst the weights to perform an initial adjustment that is not uniform across the weights. As a result, two models trained with the exact same basic architecture (without setting a random seed for initialization) will yield a different set of weights and biases, and thus can have a different result during evaluation. Generally, if a model has sufficient data and ample time for training, identical models will converge to similar values, and differences in performance may be small and/or negligible resulting in a robust model. However, for transport or propagation problems that require the input from a prior model prediction in order to make a future prediction, these small differences may accumulate through time and quickly become non-negligible. To further evaluate the robustness or sensitivity to the initialization process, we trained and evaluated 30 ensemble members for each precursor model.

2.6 Hyperparameter optimization

At different stages in training an emulator model, from data post-processing to selecting an architecture, there are hyper parameters that need to be set that can affect the performance outcome of a trained model. They may include, for example, the learning rate used to update the model weights during training, or the size and number of the hidden layers in an MLP or GRU model. As the main objective is to minimize the difference between the model predictions and the test experiments in box simulations, we want to understand how the models performance depends on the hyper parameter choices. From such an understanding, an informed choice can be made in selecting potentially optimal parameter values.

To estimate such a dependency, we use the package Earth Computing Hyperparameter Optimization (ECHO) developed by the authors at NCAR (Schreck & Gagne, 2021), and perform hyper parameter optimization given an objective metric for the three species for both MLP and GRU models. The objective metric for both the MLP and GRU models is the box MAE on the validation holdout set. With the MLP model, a box simulation begins with the initial precursor concentration at t_0, while for the GRU model the MAE for box simulations is computed for a set of starting times $\{t_0, t_i, \cdots, t_j\}$ and added together, to also test the hidden-state model on different initial precursor amounts.

MLP and GRU models were optimized with ECHO for the three species, with the outcomes described in appendix Appendix B. Tables C1 and C2 list the best hyperparameters found in each optimization study for the two models. Using the best hyperparameter set, an ensemble of 30 models were trained, where each model had a different random weight initialization. See appendix Appendix C for more details.

Although it is rather trivial to train a model to output realistic predictions one time step ahead from the truth (primarily due to high auto-correlation), limiting cumbersome error accumulation when propagated through time is highly sensitive to model parameters in complex problems such as this. We found that efficient hyperparameter searches were crucial for finding models that could successfully stabilize and limit error accumulation through the length of the simulation. See appendix Appendix B for further details.

3 Results

3.1 Performance of trained MLP and GRU models

Table 2 lists the bulk validation performance metrics for the MLP and GRU models for toluene, dodecane and α-pinene. The metrics are the Pearson coefficient and the Hellinger distance computed for each prediction task, and the number of unstable or runaway experiments observed. Here, an experiment is considered as unstable when predicted values exceed $1\,\mu g/m^3$ which corresponds to an unrealistic formation yield from 10 ppt of initial precursor. Unstable experiments were not used in the computed metrics reported below.

Figures 3 and 4 illustrate the MLP and GRU models' performance on reproducing the experimental data for three experiments selected from the test set of toluene experiments. Overall, they show that both the MLP and GRU models can predict experiment trajectories that resemble the GECKO-A ones within a factor of two. The predicted ensemble mean matched closely with the GECKO-A trajectories for the three prediction tasks, for both models. For toluene, Table 2 shows that all of the model prediction tasks led to Pearson coefficients greater than 0.97. Additionally, the Hellinger distances for each task are all low for both MLP and GRU models, indicating that the temporal distributions predicted for different initial conditions matched closely with those generated by GECKO-A.

	Toluene			Dodecane			α-pinene		
Task / Metric	Pearson	Hellinger	Unstable	Pearson	Hellinger	Unstable	Pearson	Hellinger	Unstable
MLP precursor	0.989	0.0008	0	0.950	0.0002	0.01	0.556	0.0015	0.0005
MLP gas	0.977	0.0045	0	0.952	0.0033	0.01	0.971	0.0028	0.0005
MLP aerosol	0.927	0.0150	0	0.894	0.0161	0.01	0.939	0.0153	0.0005
GRU precursor	0.993	0.0018	0	0.950	0.0023	0	0.867	0.0025	0
GRU gas	0.990	0.0012	0	0.984	0.0020	0	0.991	0.0007	0
GRU aerosol	0.975	0.0105	0	0.961	0.0083	0	0.976	0.0091	0

Table 2. Table of computed metrics for MLP and GRU models, for each of toluene, dodecane, and α-pinene. The average Pearson coefficient and Hellinger distance are listed for the precursor, gas, and aerosol prediction tasks. The fraction of experiments that went unstable is listed for each model and task. All reported metrics for both models were computed using the testing hold-out set of experiments.

For the other precursor species, each model degraded in performance in different ways. The GRU model mainly did not perform as well at predicting precursor. For example, the Pearson coefficient was 0.886 for α-pinene compared with 0.992 for toluene. The Hellinger distance for the other two species also modestly increased compared to that for toluene. On gas and aerosol predictions, Table 2 indicates that the GRU performed about the same for all three species according to these two metrics, and that none of the predicted numerical values became unstable during the box simulations.

Table 2 shows the MLP model performance on precursor prediction was the best for toluene and then α-pinene, which had Pearson values of 0.989 and 0.950, respectively. However, the MLP struggled by comparison for dodecane, where the Pearson value was 0.556. On the gas and aerosol predictions, the MLP model was mostly consistent for the three species, with gas prediction performing better by comparison to aerosol prediction. Furthermore, out of 200 experiments that went unstable during a box simulation there were 13 for dodecane and 2 for α-pinene, despite the fact that the Pearson score for dodecane remained high across the prediction tasks.

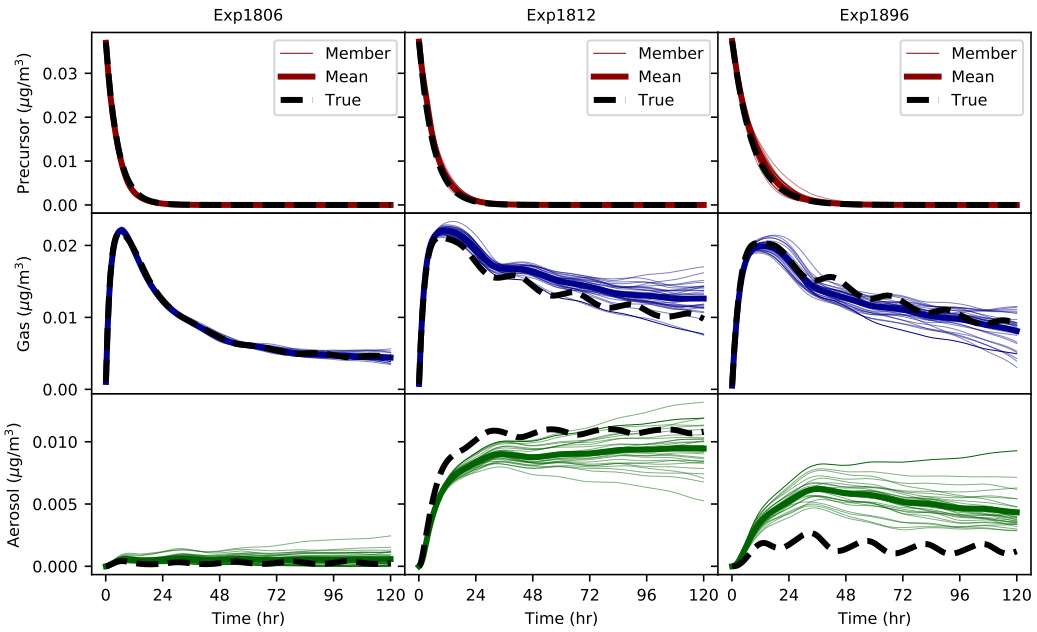


Figure 3. Three toluene sample experiment trajectories for the MLP model. Solid (thick) lines show the mean GECKO-A trajectories from 30 ensemble members, dashed lines show the reference GECKO-A trajectory, and the thin lines show each of the 30 ensemble member predictions.

3.2 Quantification of model variances

Figure 3 shows that the predicted aerosol quantities for the ensemble suite begins to diverge as the simulation time progresses, whereas the GRU variation (see Figure 4) appears to be roughly constant or improving as time progresses. In order to capture how well the models are predicting time-dependent quantities, we also computed the bootstrapped continuously-ranked probability score (CRPS) in Figure 5 and the mean standard deviation (Figure 6) between the different species across all 30 ensemble members. Figure 5 shows the CRPS (lines) and the 95% bootstrap confidence interval (shaded areas) changing in time for the two model types. For the three species, the MLP model has a lower CRPS on all predictions at early times, then the GRU at later times. Figure 5 shows the two models' CRPS values for precursor crossing typically within one simulation day, with the CRPS for the GRU starting out relatively high compared to the MLP, but then quickly declining. Except at the earliest simulation times, the GRU had a lower CRPS as well as a smaller confidence interval on the gas and aerosol prediction tasks and stayed flat or declined.

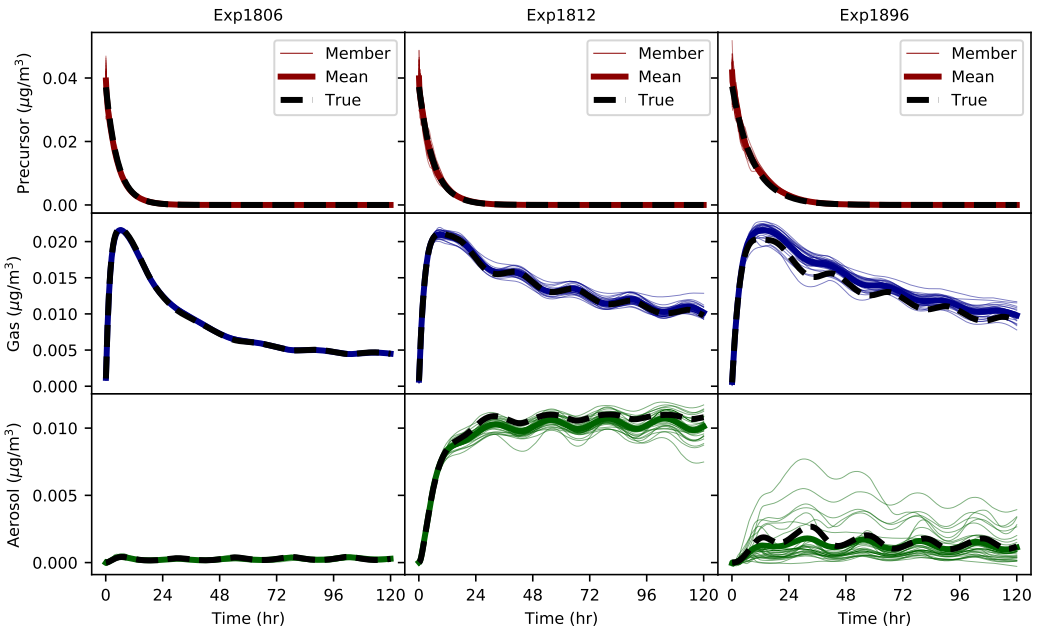


Figure 4. Three toluene sample experiment trajectories for the GRU model. Solid (thick) lines show the mean GECKO-A trajectories from 30 ensemble members, dashed lines show the reference GECKO-A trajectory, and the thin lines show each of the 30 ensemble member predictions.

Figure 6 shows a notable difference in the variation among ensemble members between the MLP and GRU, specifically the slope of the gas and aerosol trajectories. The steep positive slope of the MLP demonstrates the inherent growing uncertainty in the model itself as it progresses further from the starting condition, which is also seen in the ensemble spread on figure 3. The GRU has a much flatter trajectory, especially in the later time steps due to the short-term memory of the trajectory being encoded into the hidden state, which could potentially make it much more suitable for maintaining stability in much longer running simulations. Additionally, if it is computationally feasible, one could choose to run the ensemble suite and take the mean to increase accuracy. With this approach, the mean absolute percentage errors for the GRU are less than 2% for the gas and aerosol partitions, and approximately 3-8% for the MLP.

3.3 Performance dependency on initial conditions

Next the models' performance dependency on different initial conditions was probed by selecting initial values $X(t_N)$ for some t_N in a GECKO-A experiment trajectory as

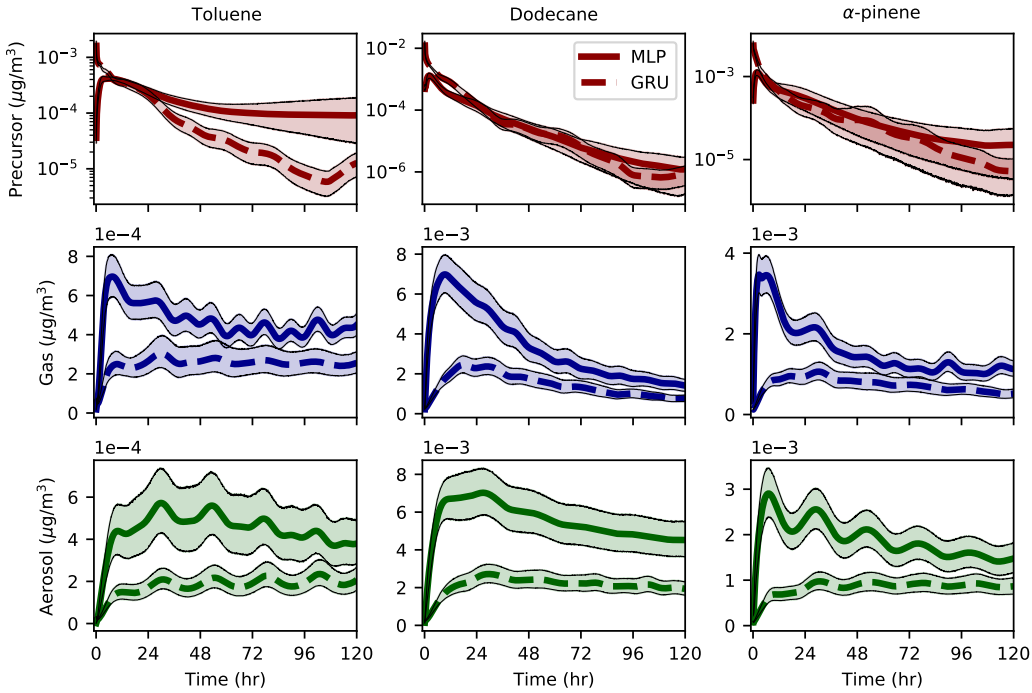


Figure 5. Ensemble bootstrapped CRPS through time. MLP (solid) and GRU (dashed) with the shaded regions representing the 95% confidence interval.

the initial starting point in a box simulation. For the GRU model, $X(t_N)$ is initially passed through the hidden-state model to obtained a starting hidden state. As the experiments contain 1440 total time steps, box simulations were left to run for 1440 - N time steps once the initial time was selected. As the instabilities observed in the MLP model discussed above occurred at a range of different time steps after the box simulation was first started, we wanted to check each models' stability over as many possible time steps as there was data available. This means that box simulations started at earlier times in experiments will run for more time steps compared to those which started at later times in the experiments. Figure 7 shows the average ensemble Pearson coefficient for MLP and GRU models versus the initial box simulation start time. A similar plot for the Hellinger distance is shown in Figure D1.

Figure 7 illustrates the broad variation in GRU and MLP model performance at predicting precursor versus initial simulation start time. The variation is more significant in dodecane and α-pinene, compared to toluene. For example, the lowest Pearson values are observed when the MLP box simulation is started about 2-3 days into the GECKO-

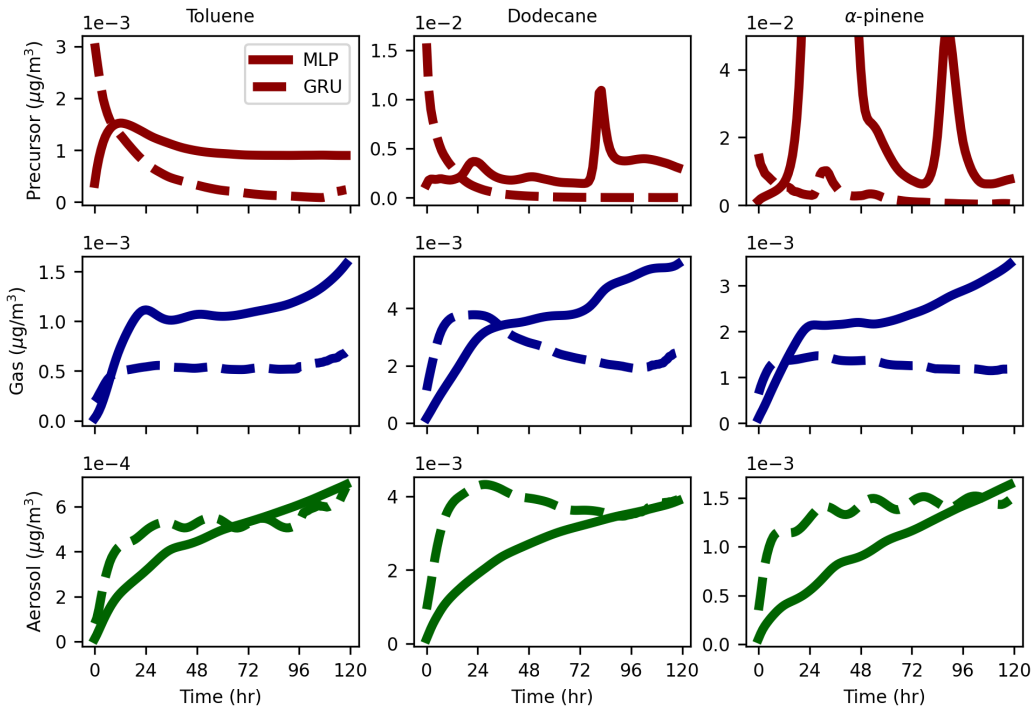


Figure 6. Mean ensemble standard deviation across all validation experiments as a function of simulation hour.

A experiment and runs for a similar amount of time. Then, some recovery is observed for the MLP model as observed by increasing Pearson coefficients at later start times for the prediction tasks, in particular on days 4 and 5, for the shorter box simulations. For α-pinene in particular, the GRU was also observed to go unstable at earlier start times. However, by comparison to the MLP model for all initial starting times, the total number of experiments having gone unstable was significantly less.

On the gas prediction task, the GRU model typically performed better than the MLP at earlier start times (longer box simulations), while at the later start times (shorter box simulations) the MLP had higher Pearson scores for precursor and gas prediction. We observed in some experiments the GRU struggling at later start times to reproduce the GECKO-A predicted gas values as accurately as the MLP, in particular, when those concentration values were very small compared to the initial experiment precursor amount, but the model predictions remained stable at these times.

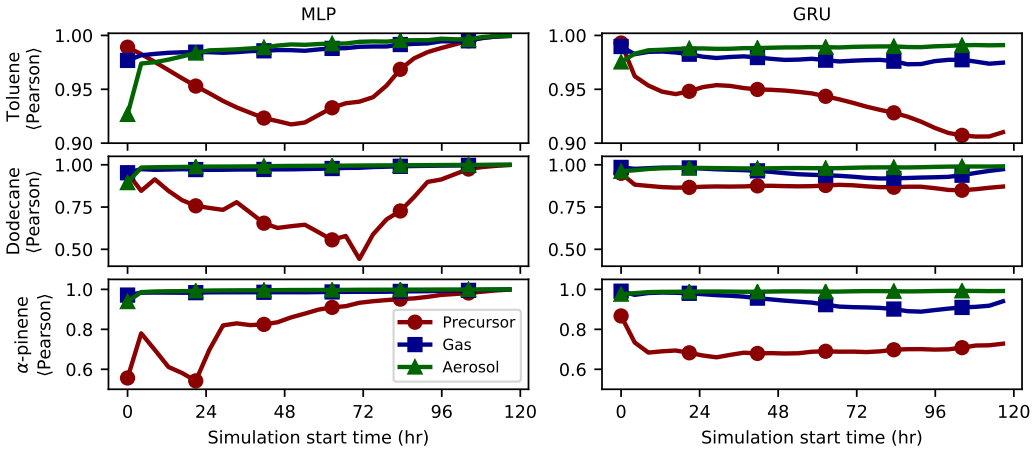


Figure 7. The mean Pearson coefficient versus the initial box simulation start time for all validation experiments.

3.4 Stabilization through fewer prediction targets

The results above indicate that predicting the evolution of the precursor's concentrations is the most difficult task of the three that we considered, especially for the MLP models. As both MLP and GRU models always have to predict finite quantities of precursor at early times before mass moves into the other two phases at later times, small precursor prediction inaccuracies can lead to numerically inaccurate predictions for gas and aerosol quantities, as well as lead to the observed experiments having gone unstable, as reported above.

In the reference model, the precursor decays exponentially from its initial concentrations, at different rates depending on the environmental conditions and the species, and could be estimated using other heuristic models (such as a linear regression model), or directly calculated within the chemical model. Thus, we consider MLP and GRU models that only perform gas and aerosol prediction, and not precursor, to probe whether reducing the total number of prediction targets will improve model performance on gas and aerosol predictions. The inputs to the model and all other architecture choices remains the same, just that the output layer size is size 2 rather than 3. Both model types were optimized using ECHO, and 30 ensemble members were trained using the parameters from the best study. Table 3 shows the same metrics as in Table 2 for the MLP and GRU models tasked with gas and aerosol predictions only. The Pearson coefficients and

	Toluene			Dodecane			α-pinene		
Task / Metric	Pearson	Hellinger	Unstable	Pearson	Hellinger	Unstable	Pearson	Hellinger	Unstable
MLP gas	0.980	0.0034	0	0.911	0.0093	0	0.957	0.0050	0
MLP aerosol	0.957	0.0095	0	0.908	0.0248	0	0.918	0.0321	0
GRU gas	0.989	0.0012	0	0.991	0.0014	0	0.990	0.0010	0
GRU aerosol	0.985	0.0148	0	0.990	0.0097	0	0.986	0.0105	0

Table 3. Table of computed metrics for MLP and GRU models which are tasked with prediction of gas and aerosol for each of toluene, dodecane, and α-pinene. The average Pearson coefficient and average Hellinger distance are listed for the two prediction tasks. The fraction of experiments that went unstable is listed for each model and task. All reported metrics for both models were computed using the testing set of experiments.

Hellinger distances are comparable for both model types and architectures. An overall improvement in scores is seen when predicting the precursor concentrations was not a target, but the most notable difference is that all model runs remained stable.

3.5 GECKO-A emulator evaluation with external datasets

The performance abilities of both MLP and GRU models were tested by expanding the data sets to include additional simulations, (1) for 10 times (X10, = 100 ppt) and 100 times (X100, = 1 ppb) higher initial concentrations of the precursor, which are more representative of somewhat polluted atmospheric conditions, and (2) for simulating the diurnal variation in the precursor levels, that was not present in the original data sets which did not include the daily variability on the emissions, and removal of the precursor.

3.5.1 Model performance on increased precursor concentrations

The simulations performed to create X10 and X100 data sets were carried out identically compared with the reference simulations starting at 10 ppt precursor concentrations, except that the initial precursor concentrations were increases by a factor of 10

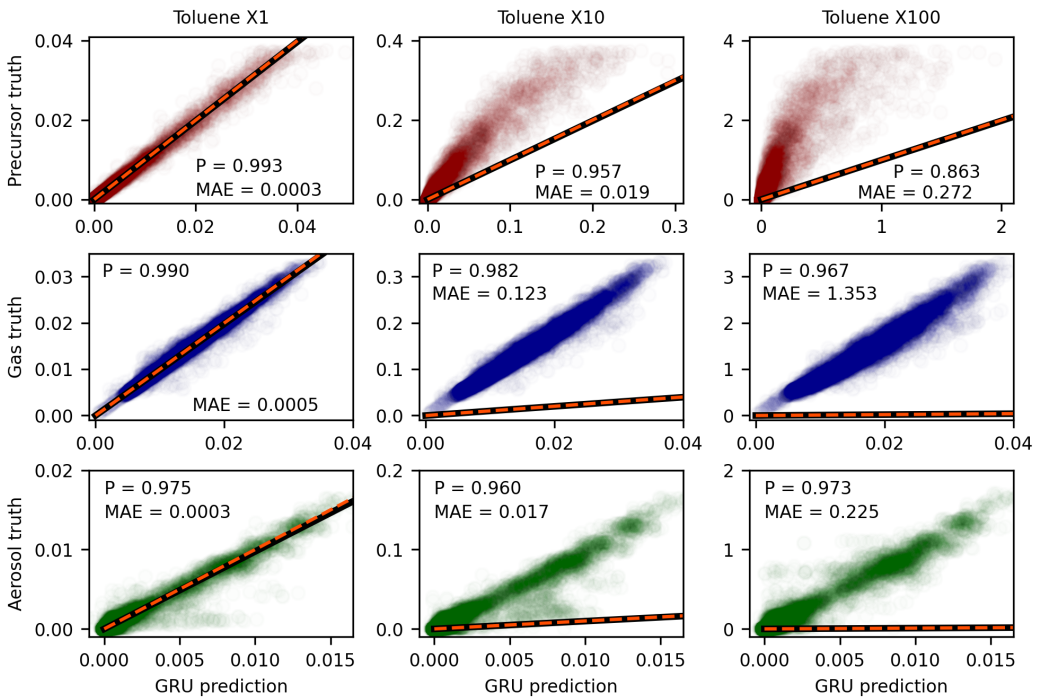


Figure 8. Scatter plots comparing the GECKO-A value (y-axis) against the GRU predicted value (x-axis) for models trained on X1 data and applied to test sets with 10X and 100X higher initial concentrations. The dashed orange line shows y = x, while the solid black line shows the linear relationship for models trained on the respective X* data set.

(X10) and 100 (X100). The new data sets were then split into train, validation, and test data sets just as before, then transformed using the fitted scaling transformations on the original data sets. Then, the X10 and X100 test data sets were passed through MLP and GRU models that were trained on the original data set containing the smaller initial value of the precursor.

Figure 8 shows the predictions of the GRU model on the reference test set of experiments (left column), and the expanded X10 and X100 test data sets (middle and right columns, respectively). The Pearson coefficient and MAE for each prediction task are listed in the sub-panels. The figure shows that the GRU trained on the smaller initial precursor concentrations made predictions on the X10 and X100 data sets that correlated strongly with the true values for precursor, gas, and aerosol, as is seen by high values of the Pearson coefficients for the different prediction tasks, but the MAE for each task increased by orders of magnitude with larger starting precursor concentrations.

549 The figure also clearly indicates that the GRU model under-predicted the true val-
550 ues for gas and aerosol by approximately 1 and 2 orders of magnitude for the X10 and
551 X100 data sets, respectively. For the precursor prediction task, the predicted decay times
552 were significantly shorter compared to that observed in the GECKO-A experiments. Over-
553 all, similar performance declines were observed for the MLP model (results not shown).
554 Models which did not have the precursor prediction task did better by comparison but
555 overall performance still declined. These results indicate that the neural models cannot
556 be extrapolated outside of the training data sets. This poses a real challenge for 3D model
557 applications given the wide range of precursor's concentrations in the atmosphere go-
558 ing from very clean conditions in the remote regions, and upper troposphere to polluted
559 conditions found i.e., in urban or fire plumes.

560 ### *3.5.2 Evaluation with varying environmental conditions*

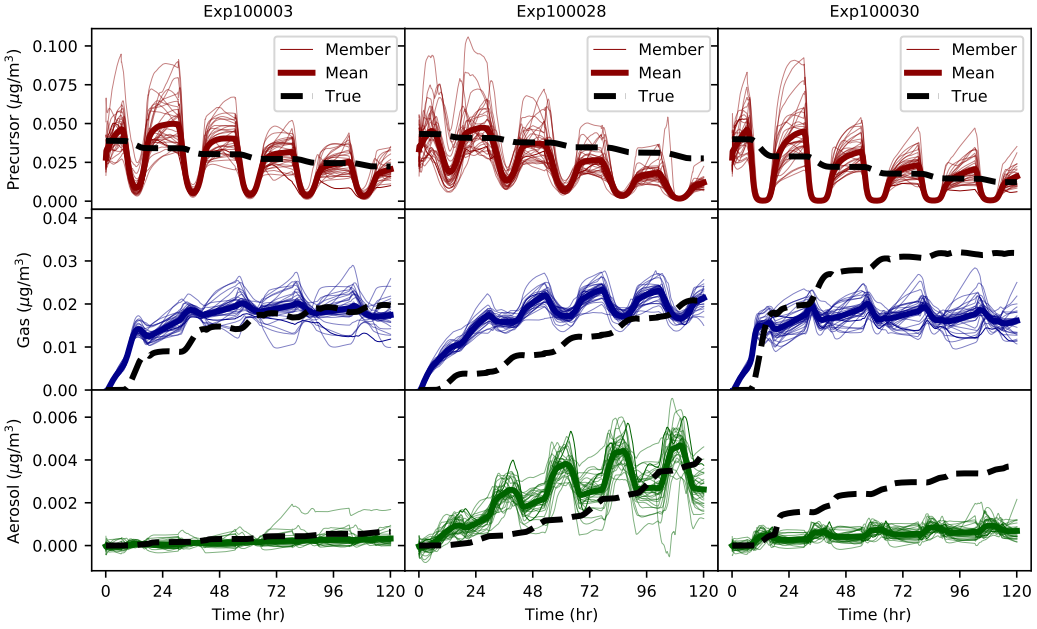


Figure 9. Examples of GRU box simulations where select environmental variables were allowed to vary with time.

561 Lastly, the models' performance was tested on 36 experiments run for toluene that
562 simulated daily varying conditions for five days. Like for the training data set, the pre-
563 cursor's initial concentration was set to 10 ppt. Initial temperature, pre-existing aerosol

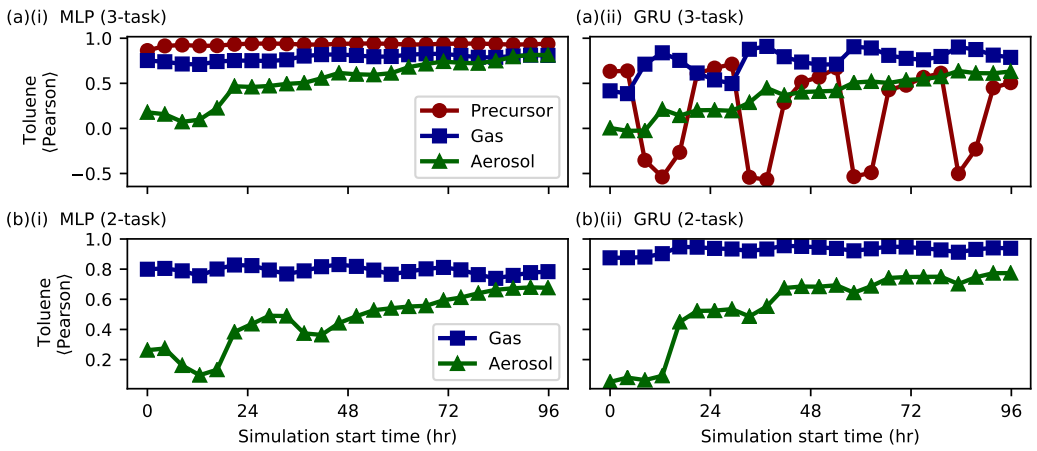


Figure 10. The average Pearson coefficient versus the initial box simulation start time computed from 36 experiments for toluene. (a) The results for the three-task MLP and GRU models are shown in (i) and (ii), respectively. (b) The same quantities as in (a) except for the two-task MLP and GRU models. All box simulations ran for 24 hours.

seed, ozone and NOx were randomly selected in the same ranges as the training data set (Tab. 1). CO mixing ratio was initialized to 100 ppb. Relative humidity was held constant to a random value picked in the 50-80 % range. The latitude was also randomly selected in the 80S-80N range. Contrary to the training data set, after initialization, all chemical concentrations were free to evolve with the diurnal cycle to simulate a realistic atmospheric degradation of toluene and the subsequent organic aerosol formation.

Because the experiments simulated a diurnal cycle and started at midnight, box simulations were performed with MLP and GRU models at different starting times in the experiments to assess the impact of training the models on daytime oxidation only. Three example experiment trajectories are shown in Figure 9 for the 3-task GRU model, for the full 5-day box simulations which all began at midnight (the same examples for the MLP model are shown in Figure D4). The simulations performed at other starting times covered a shorter 1-day window. Figure 10 shows the average Pearson coefficient for these shorter simulations for both the 3- and 2-task MLP and GRU models.

Figure 9 shows the predicted curves for simulations starting at midnight are notably different compared with those in Figure 4. In particular, the GRU model seems to have captured some of the diurnal changes, where oxidation appears to proceed during the day, but not at night. Drastic changes in the predicted precursor amounts are

observed during day-night transition periods. However, the predicted concentration values are clearly not in agreement with the true values. Although it is less obvious, close inspection of the MLP predicted precursor values in Figure D4 shows that it too responded to the diurnal variation in the extended experiments, but with poor numerical accuracy. Both MLP and GRU models predicted that all experiments remained stable at all simulation times.

Figure 10(a) shows that over a 1-day simulation window, the performance still dropped relative to the experiments where the environmental variables were held constant for toluene. The MLP model had comparably high Pearson score for precursor prediction across the start times, but gas and aerosol performance was lower by comparison. The periodic response of the GRU to the diurnal signal in Figure 10(a)(ii) is indicated by the sign-change in the average Pearson value for predicted precursor, which goes negative when box simulations were started during day-time hours, while those started overnight stayed positive. The GRUs performance on gas and aerosol prediction also peaked for simulations that started during the middle of the day-time, and was poorest by comparison for those started late at night. Figure 10(b) shows that MLP and GRU models, which were only tasked with predicting gas and aerosol, performed mostly similar to the 3-task models, with notable gas performance improvement for the 2-task GRU.

3.6 Computational performance of emulator models and GECKO-A

In addition to being able to reproduce reasonably well the evolution of concentrations of organic compounds on the test data sets for the three species, the MLP and GRU emulators also led to significant computational gains. Table A1 lists estimates for the time required by GECKO-A, MLP, and GRU models to advance one time step, e.g. five minutes of simulation time, for the three precursor species. For toluene, GECKO-A requires 0.9 seconds and is about 78 and 244 times faster than dodecane and α-pinene, respectively. By comparison, both MLP and GRU models require about the same time for the three precursor species on the CPU, typically a few microseconds, with the MLP faster than the GRU by up to a factor of five. Thus, for toluene both neural network models could be expected to perform hundreds of times faster, while for α-pinene the expected speed-up could be up to 4-6 orders of magnitude faster than the explicit model.

4 Discussion

In general, the comparative differences seen between the MLP and novel 1-step GRU applications show the advantages of a recurrent neural network emulator in a few key areas. First, its overall accuracy, especially at integrated time steps further away from its starting conditions, is notably higher than the MLP model. Furthermore, the encoding of a hidden state which can represent the trajectory of each input, the key feature of a recurrent network architecture, appears to help constrain model uncertainty and ultimately, numerical stability. Most neural network applications for atmospheric chemistry have not yet begun to examine such model uncertainties. Our use of training a suite of ensemble members using the exact same architecture for each model, and only initializing the weights differently prior to training, provides some evidence that model uncertainty can be sensitive to, and better constrained by, certain model types. Related, we also note that our recurrent model remains numerically stable for all species and for most starting initial conditions, which is not true for a small percentage of MLP member / experiment combinations. This insight may not have been detected without the inspection of an ensemble suite, as most of the MLP models remained stable.

Maintaining numerical stability with the use of emulators for atmospheric chemistry and other atmospheric parameterizations is a known issue and initial steps have been taken to address it (Brenowitz & Bretherton, 2018; Kelp et al., 2020, 2021). These recent studies found some performance improvements by using a "recurrent training" scheme, where a model was rolled out in time for n time steps during training, and a loss was calculated on the sum of n time steps, instead of a single time step. However, the models used in these studies were not recurrent neural networks, as the network architectures were that of an MLP (Brenowitz & Bretherton, 2018) and an encoder/decoder framework (Kelp et al., 2020, 2021), which only utilized feed-forward connections. Rather, training these models relied on the multi-time step loss function as a means to update the model weights using a sequence instead of a single length input. Our GRU model provides an alternative approach, by rolling out the model to the end of the training experiment and calculating the loss at successive single time steps, the feedback connections' memory of the trajectory through $t-1$ is simply used as input at t along side the current values of the precursor, gas and aerosol.

Recurrent neural networks have not yet been thoroughly explored in 3D atmospheric modeling, although there have been applications in other earth systems areas including hydrology (Kratzert et al., 2018; Ardabili et al., 2019), earthquake magnitude prediction (Mousavi & Beroza, 2020), rain-runoff (Boulmaiz et al., 2020) and wind velocity forecasting (Irrgang et al., 2020), as well as vegetation growth estimation (Reddy & Prasad, 2018). One reason for this might be the lower dimensional nature of many of these models, which would be computationally less burdensome to put into production as opposed to integrating a model that requires multiple time steps of input into a full 3D climate or weather model. For this reason, we have developed a method that still only requires one time step of input but maintains the advantage of having an encoded memory of past time steps. A small disadvantage of our framework is that it does require an additional model to predict the initial hidden state prior to running the GRU. However, if the community ultimately finds that it is computationally and programmatically feasible to couple large recurrent networks into full 3D transport models, investigation of training recurrent models with multiple time steps of input would be a recommended pathway.

Although there are clear benefits demonstrated from use of a recurrent network, there are computational limitations. The hidden state increases the input needed for each prediction from 9 for the MLP model to 1000 for the GRU. This is not problematic for 1D validation efforts, but would become too memory intensive if this model were integrated into 3D simulations. A smaller GRU hidden state is possible but may result in drops in performance. If directly shrinking the vector is not feasible, lossy compression of the vector with principal component analysis or an autoencoder may balance a smaller performance loss with slightly more computation. For this reason, despite the lower performance metrics, we still find value in simplified neural networks such as the MLP if they can still approximate a solution within the given tolerance. Additionally, some performance could be sacrificed for a smaller GRU model (see Figure B3).

Our results demonstrate some ML generalization challenges involving the selection and training of neural networks on experiments with both small initial precursor concentrations and select static environmental variables. For example, low precursor concentrations of 10 ppt were chosen primarily to limit the influence of a single precursor on the photochemical reactivity, and gas/particle partitioning in GECKO-A. However, this had a large impact on the generalizability outside the training range (Fig 8). If we were to implement our models into a 3D climate model, they would need to be trained

on a larger range of precursor values that are also representative of more polluted atmospheric conditions. Additionally, environmental variables outside of temperature were held constant in an effort to help the models generalize better, but this was not successful. While there did not appear to be any direct evidence of over-fitting to the training data, an open question remains of how to properly configure the reference box models to provide data to best capture the physical relationships in a complex chemical system. One speculation is that the GRU could generalize more effectively than observed here by having varying environmental fields within an experiment, to better parameterize the models feedback connections. Many other generalization questions also remain, such as the inclusion of night chemistry (oxidation with O3 and NO3), as well as reactions between species originated from various precursors.

To our knowledge, this is the first neural network emulation of organic atmospheric chemistry. As a result, there are many areas that warrant further exploration: (1) coupling both the MLP and GRU models to a 3D chemistry-climate model, such as WRF- or GEOS-Chem, to better understand their successes and shortcomings, (2) further quantification of the underlying uncertainties in model predictions to determine whether the error sources originate from the data or the model architecture choices, or both, (3) testing of different data sets, training regimes, and model architectures to better generalize across different chemical regimes (such as daytime vs. nighttime chemistry), (4) application of transfer learning for domain adaption (Kouw & Loog, 2018), and for potentially managing the cumbersome production of data sets, (5) incorporating physical constraints into the model architecture or training procedure as a means for constraining model outputs, for example the total mass or the number of C atoms needs to be conserved, and (6) utilizing explainable and interpretable methodologies to better understand what the model has learned, and what it is using to drive its predictions.

5 Conclusions

In summary, the neural network emulators proposed here, especially the GRU model, appear to provide fast and accurate representations of complex chemical processes. As such they may be incorporated into 3D models to potentially provide insight into important chemical processes currently absent climate models. The recurrent neural network considered contained feedback connections, and was generally more stable over longer box simulations and maintained higher numerical accuracy with the GECKO-A data sets,

as compared to the MLP architectures, which did not possess any memory capabilities by way of feedback connections. Additionally, models with only two output tasks for predicting gas and aerosol quantities and not precursor led to further performance and stability improvements in both models. Furthermore, extensive hyper-parameter search was a crucial step in finding the best models in each case. The novelty of our recurrent neural network that only requires one time step of input data allows for a similar ease of transfer compared to those already explored such as random forests and MLPs. This approach does not depend on the specific data set used for training and validation, and was designed so that a recurrent model can be integrated into current 3D models without adding additional transport complexity. Thus, this "1-step" approach could be applied in other areas where emulators are being used for the prediction of time-ordered quantities.

Appendix A Average time step comparison

Model	Toluene		Dodecane		α-pinene	
GECKO-A	0.9 s	1	71 s	1	220 s	1
MLP CPU	2.1 µs	430	0.8 µs	8.88×10^4	1.6 µs	1.38×10^5
MLP GPU	0.08 µs	11250	0.07 µs	1.01×10^6	0.08 µs	2.75×10^6
GRU CPU	3.1 µs	290	3.2 µs	2.22×10^4	3.3 µs	6.67×10^4
GRU GPU	0.38 µs	2368	0.38 µs	1.87×10^5	0.38 µs	5.79×10^5

Table A1. The average time each model required to advance 300 seconds of simulation time. For each species, the first column shows the time step in seconds while the second column shows the ratio of the GECKO-A time step to each model time step.

Table A1 compares the average speed in which GECKO-A and MLP and GRU models take to advance 300 seconds of simulation time. The neural network models were evaluated on NCAR's casper supercomputer, on a node that contained an 18-core 2.3-GHz Intel Xeon Gold 6140 processors (CPU) and a NVIDIA Tesla V100 32GB graphics cards (GPU). The GECKO-A simulations were performed on NCARs cheyenne supercomputer, on a node that contained a 2.3-GHz Intel Xeon E5-2697V4 (Broadwell) processor (CPU).

Appendix B Hyperparameter optimization

The ECHO package is based on Optuna (Akiba et al., 2019), and facilitates optimization using the high-performance computing clusters available at NCAR. The optimization procedure begins by initiating a "study" and performing the first "trial" where values are selected for a set of hyper parameters within specified ranges, the model is trained, and its performance measured in box simulations. The outcome of the trial is saved to the study along with other metadata. For the current objective, any trial is independent from any other trial. Trials are ran until the optimization converges or the number of trials saved to a study reaches a predetermined number. Upon the completion of a study the relative importance of each hyper parameter on model performance may be estimated. To sample hyper-parameters, we selected the Tree-structured Parzen Estimator (TPE) (Bergstra et al., 2011) for this task. For each hyper parameter, the TPE method fits a Gaussian Mixture model (GMM) $l(x)$ to the set of parameter values associated with the best MAE, for all of the trials that have been carried out to completion. TPE additionally fits a second GMM to the leftover parameter values, and then chooses the next parameter value that maximizes $l(x)$ / $g(x)$. We initially delay using the TPE sampler and use random sampling instead. We have observed for the present models that this initial step helps to inform the TPE sampler by initially supplying observations that the GMM models may leverage to make better informed parameter selections. For additional details about hyper parameter optimization, see the supporting information.

Once a study is complete, the parameters sampled in each trial along with the box-MAEs are used to compute the relative parameter importance. There are a variety of approaches for such estimation, including mean decrease impurity evaluation (MDI) (Louppe et al., 2013) and functional analysis of variance (functional ANOVA, or fANOVA henceforth) (Hutter et al., 2014). Both approaches utilize tree-based ensemble methods to estimate the relationship between the values of a set of hyperparameters used to train a model, and the optimization objective value that resulted. The MDI estimation of a hyperparameter is zero when it depends only on the relevant variables, hence it is irrelevant to making a prediction. The most relevant hyperparameter has the largest estimation value. Similarly, in fANOVA when the estimated variance between input x_i and output y_j is low or zero, it is not an important input feature, and vice versa.

We also compute the partial dependence plot for each parameter, which estimates the marginal effect one (or more) features have on the predicted outcome of a machine learning model (Friedman, 2001). A partial dependence plot can show whether the relationship between the box-MAE and an input feature is linear, monotonic or more complex. For example, when applied to a linear regression model, partial dependence plots always show a linear relationship.

MLP and GRU model optimization for the three species was performed and the best model parameterization was selected from each optimization study. Figure B1(a) plots the optimization objective for GRU model to the toluene data set versus number of optimization trials. Figure B1(b) illustrates partial dependence curves for the GRU layer size for the three species. Figures B2 and B3 illustrate partial dependence curves for each hyper parameter varied by ECHO for MLP and GRU models respectively. Tables B1 and B2 list the hyperparameter importance as measured by MDI and fANOVA.

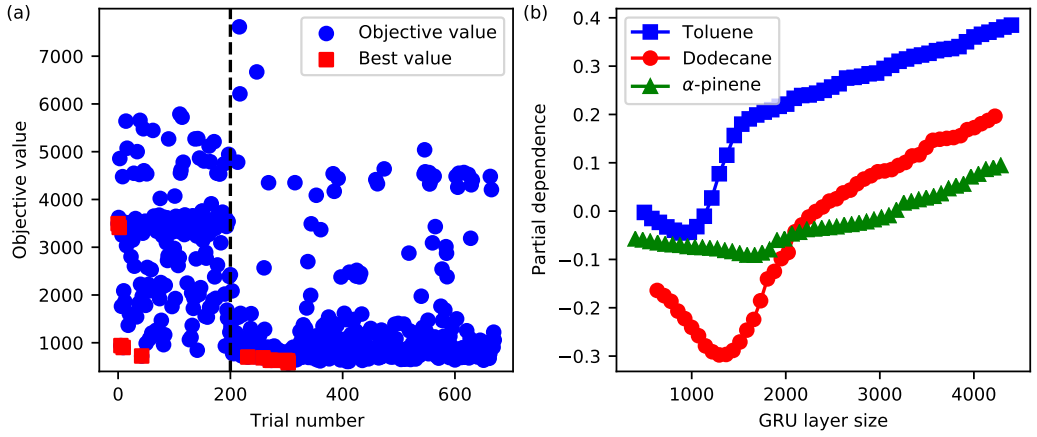


Figure B1. (a) The optimization history for the GRU model trained. on the α-pinene data set. (b) Partial dependence versus GRU layer size.

The blue dots in Figure B1(a) show outcomes of trails. A red dot indicates when a set of hyperparameters is the best performing one in a study. The horizontal line indicates the two stages of the algorithm. To the left, random sampling was used to select a set of hyperparmeters. To the right, TPE sampling was used. As the figure shows, the optimization procedure mostly converges to better performing models in the second stage, but no improvement in the study was observed after about 300 trial attempts in this example.

Figure B1(b) shows how the optimization metric depends on changes to values of hyperparmeters, referred to as the partial dependence. In the figure, the partial dependence shows how the GRU model MAE summed over 1439 time steps depends on the GRU model layer size, and a random seed in python 3.7 of 1000. The best layer size for each species is the one that leads to the most negative value of the partial dependence, and is at a curve's global minimum. In this example, toluene and dodecane are more sensitive to the value of the GRU layer size relative to α-pinene, especially for layer sizes that are near the best value. Table B2 additionally shows that the computed MDI and fAVONA values for the hidden size of the GRU are larger than that for α-pinene. The curve for α-pinene is also more flat in appearance and encompasses a smaller range of values of the partial dependence compared to toluene and dodecane. Overall, the most important hyperparameters in the optimization studies for the three species were the loss weight for hidden state model, the initial learning rate, and the GRU layer size. For dodecane, the aerosol loss weight and the batch size were also estimated to be important training parameters. For the MLP models, Table B1 shows that both MDI and fANOVA score the learning rate as the most important training parameter for all three species, with the batch size the second most important.

Figures B2 and B3 illustrate partial dependence curves for hyper parameters used in each optimization study, for MLP and GRU models, respectively. Tables B1 and B2 list the hyperparameter importance value estimations using the MDI and fANOVA methods, for MLP and GRU models, respectively.

Appendix C Model and training parameters

Tables C1 and C2 list the best hyperparameter values in optimization studies for the MLP and GRU models respectively. The parameters listed in these tables were used to train ensembles of models that were then used to produce the results shown in the figures in the main text. Figure C1 shows the CRPS for MLP and GRU models which are tasked with gas and aerosol prediction only.

Appendix D Additional results

Figure D1 shows the average Hellinger distance for MLP and GRU models tasked with predicting precursor, gas, and aerosol, versus the start time of the box simulation. Figures D2 and D3 show the average Pearson coefficient and the average Hellinger dis-

MLP	Toluene		Dodecane		α-pinene	
Parameter	fANOVA	MDI	fANOVA	MDI	fANOVA	MDI
Learning rate	0.969	0.945	0.759	0.540	0.759	0.618
Batch size	-	-	0.107	0.179	0.169	0.134
Hidden layer size	0.022	0.034	0.063	0.076	0.017	0.068
Epochs	0.006	0.013	0.055	0.124	0.037	0.120
L2 penalty	0.002	0.004	0.003	0.041	0.018	0.044
L1 penalty	0.001	0.001	0.013	0.039	0.001	0.016

Table B1. Hyperparameter importance values for optimization studies of a MLP model trained on toluene, dodecane, and α-pinene GECKO-A experiment trajectories. The maximum number of trees used and the maximum depth was set to 1000 in all estimations. The batch size was fixed at 8192 for toluene.

GRU	Toluene		Dodecane		α-pinene	
Parameter	fANOVA	MDI	fANOVA	MDI	fANOVA	MDI
Hidden loss weight	0.310	0.386	0.040	0.020	0.393	0.518
GRU hidden size	0.200	0.140	0.198	0.112	0.186	0.046
Learning rate	0.118	0.205	0.152	0.297	0.315	0.194
Precursor loss weight	0.101	0.052	0.054	0.024	0.026	0.081
Batch size	0.079	0.052	0.165	0.053	0.025	0.032
Aerosol loss weight	0.072	0.042	0.184	0.363	0.017	0.022
Gas loss weight	0.060	0.043	0.057	0.019	0.017	0.039
GRU dropout	0.045	0.042	0.112	0.023	0.013	0.032
L2 penalty	0.013	0.038	0.039	0.088	0.008	0.034

Table B2. Hyperparameter importance values for optimization studies of a GRU model trained on toluene, dodecane, and α-pinene GECKO-A experiment trajectories. The maximum number of trees used and the maximum depth was set to 1000 in all estimations.

Parameter	Toluene	Dodecane	α-pinene
Learning rate	1.39×10^{-5}	4.41×10^{-6}	6.39×10^{-6}
Batch size	8192	2538	6907
Hidden layer size	4902	2655	4049
Epochs	841	907	1450
L2 penalty	3.49×10^{-4}	2.60×10^{-3}	6.61×10^{-5}
L1 penalty	1.39×10^{-5}	1.22×10^{-11}	1.76×10^{-5}

Table C1. The values of the best hyperparameters in the optimization studies for the MLP models for the three species. The batch size for toluene was fixed at 8192. The leaky ReLU activation function was used after the hidden layer.

Parameter	Toluene	Dodecane	α-pinene
Hidden loss weight	0.161	0.980	1.896
GRU hidden size	1215	1253	1850
Learning rate	6.926×10^{-5}	5.275×10^{-5}	2.474×10^{-5}
Precursor loss weight	0.812	0.537	0.805
Batch size	1426	980	767
Aerosol loss weight	0.421	0.911	0.894
Gas loss weight	0.151	0.962	0.621
GRU dropout	0.122	0.137	0.415
L2 penalty	2.269×10^{-8}	4.138×10^{-8}	1.171×10^{-8}

Table C2. The values of the best hyperparameters in the optimization studies for the GRU models for the three species. Other fixed parameters used were an early stopping patience of 6 and the learning rate annealing patience of the 2.

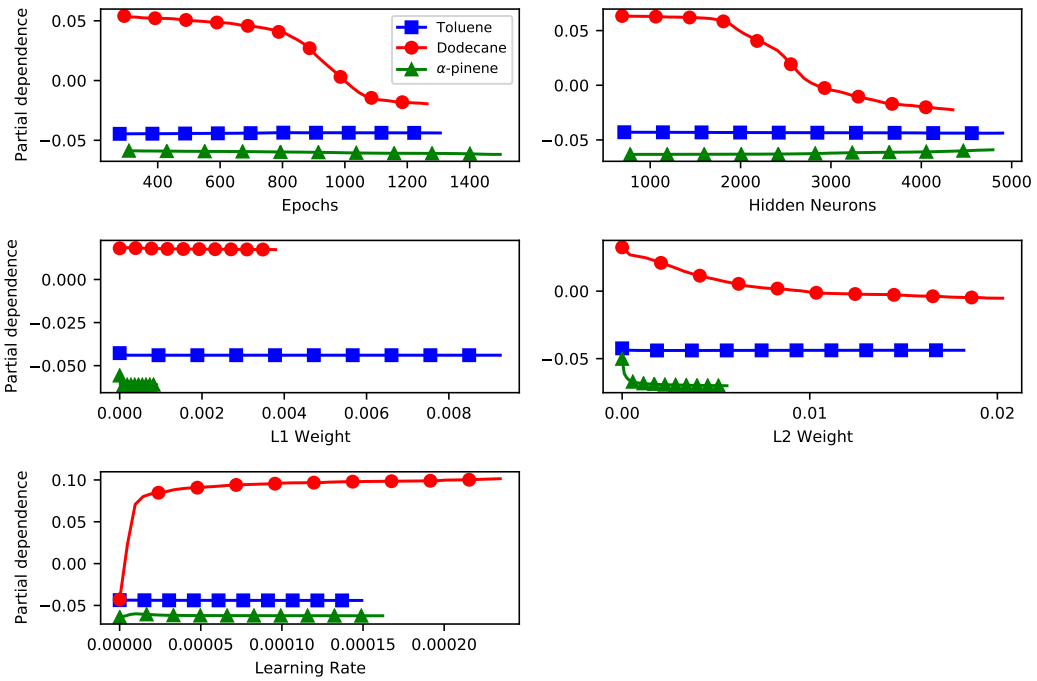


Figure B2. A partial dependence plot for each parameter varied by ECHO for the MLP model for toluene.

tance for MLP and GRU models tasked with predicting gas and aerosol, versus the start time of the box simulation. In these three figures, a box simulation began at the start time and continued until no more time steps were available to compare with the GECKO-A trajectories.

Three example experiment trajectories are shown in Figure D4 for the 3-task MLP model, for the full 5-day box simulations which all began at midnight (the same examples for the GRU model are shown in Figure 9). The simulations performed at other starting times covered a longer 4-day window compared to the 1-day simulations shown in the main text in Figure 10. Figure D5 shows the average Pearson coefficient for these 4-day simulations for both the 3- and 2-task MLP and GRU models.

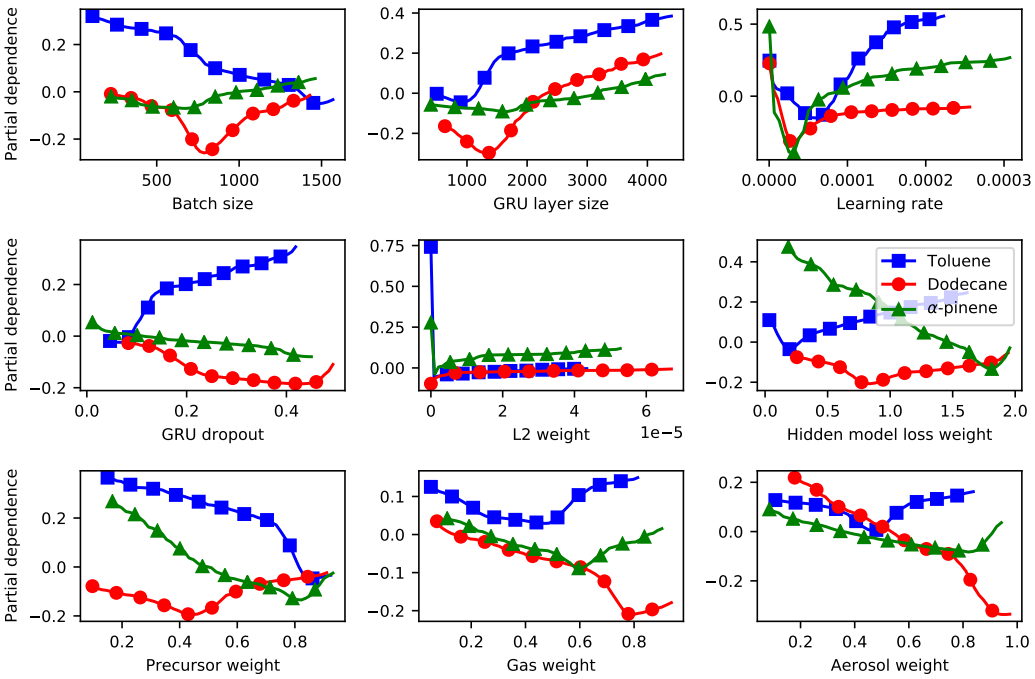


Figure B3. A partial dependence plot for each parameter varied by ECHO for the GRU model for toluene.

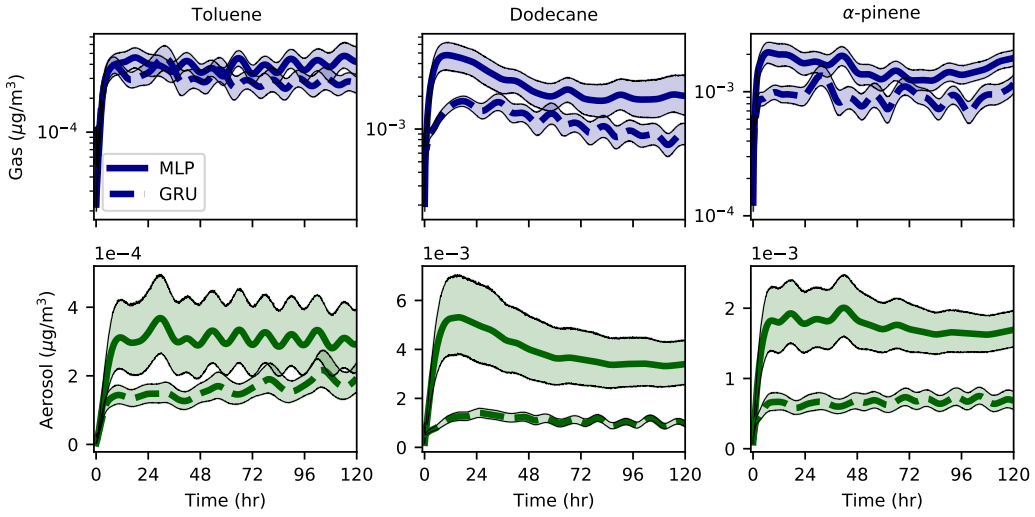


Figure C1. Computed CRPS versus simulation time for toluene, dodecane and α-pinene, for MLP and GRU models (solid and dashed lines, respectively) predicting gas and aerosol but not precursor. The precursor value at some time is still used as input to both models. The shaded regions show the 95% confidence interval.

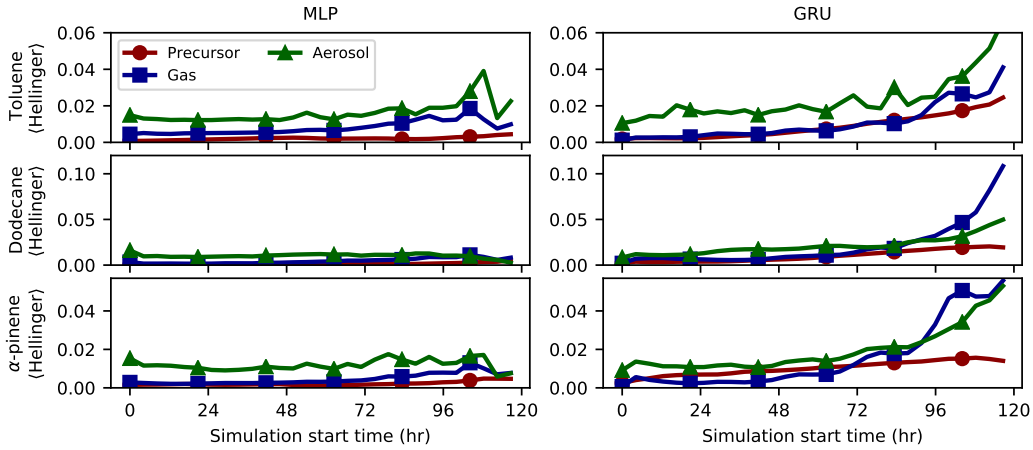


Figure D1. The average Hellinger distance versus the initial box simulation start time, computed from the 200 test experiments for the three species considered. The MLP and GRU results are shown in panels on the left and right, respectively.

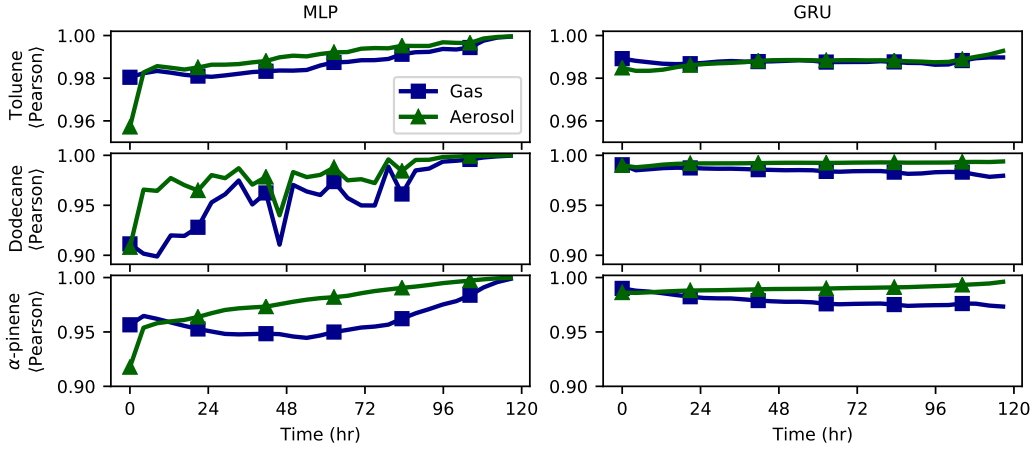


Figure D2. The average Pearson coefficient versus the initial box simulation start time, computed from the 200 test experiments for the three species considered. Both model types were tasked with gas and aerosol prediction only, but otherwise were the same as the versions which predicted precursor. The MLP and GRU results are shown in panels on the left and right, respectively.

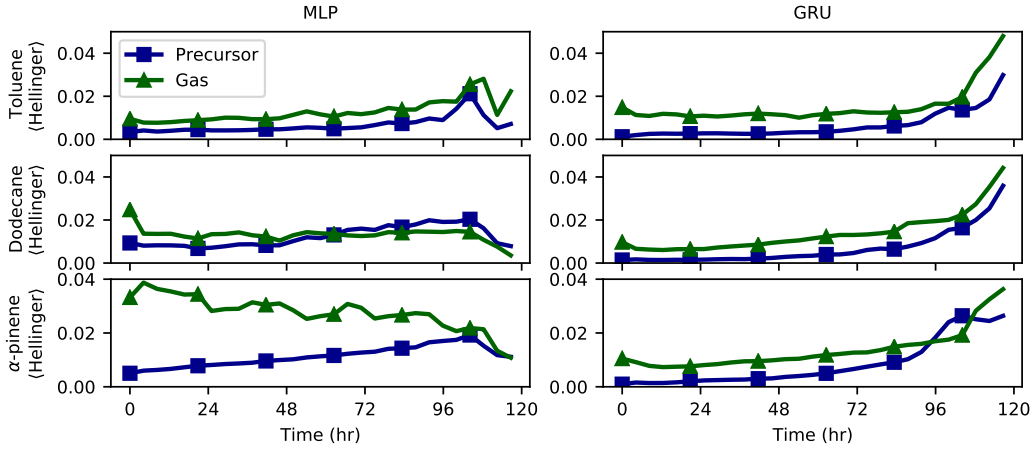


Figure D3. The average Hellinger distance versus the initial box simulation start time, computed from the 200 test experiments for the three species considered. Both model types were tasked with gas and aerosol prediction only, but otherwise were the same as the versions which predicted precursor. The MLP and GRU results are shown in panels on the left and right, respectively.

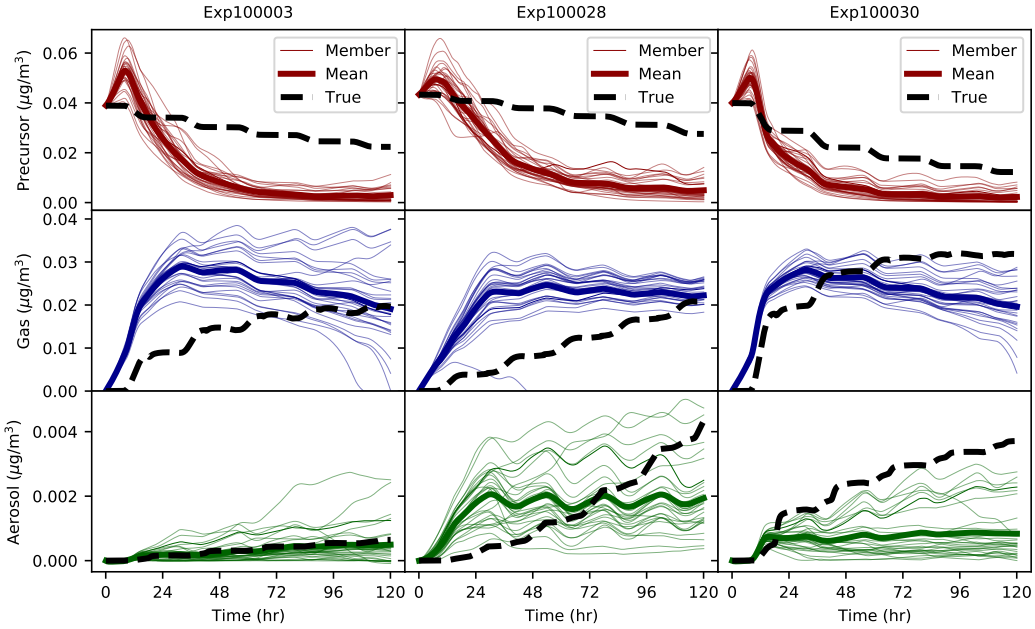


Figure D4. Examples of MLP box simulations where the environmental variables not including temperature and solar zenith angle were allowed to vary with time, compared with the GECKO-A simulations.

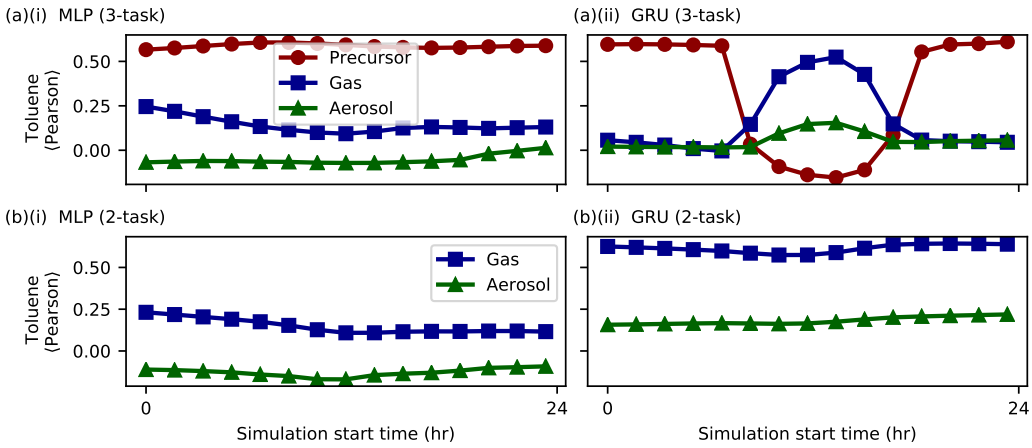


Figure D5. The Pearson coefficient versus the initial box simulation start time computed from 36 experiments for toluene. In (a), the results for the three-task MLP and GRU models are shown in (i) and (ii), respectively, while (b) shows the two-task MLP and GRU models. All box simulations ran for 96 hours.

Acknowledgments

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977. We would like to acknowledge high-performance computing support from Cheyenne and Casper (Computational and Information Systems Laboratory, CISL, 2020) provided by NCAR's Computational and Information Systems Laboratory, sponsored by the National Science Foundation. The emulators described here and simulation code used to train and test the models are archived at `https://github.com/NCAR/gecko-ml`. All GECKO-A data sets created for this study are available at `https://doi.org/10.5281/zenodo.5790042`

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery and data mining.*

Ardabili, S., Mosavi, A., Dehghani, M., & Várkonyi-Kóczy, A. R. (2019). Deep learning and machine learning in hydrological processes climate change and earth systems a systematic review. In *Int. res. conf. high educ.* (pp. 52–62).

Aumont, B., Szopa, S., & Madronich, S. (2005). Modelling the evolution of organic carbon during its gas-phase tropospheric oxidation: development of an explicit model based on a self generating approach. *Atmospheric Chemistry and Physics*, *5*(9), 2497–2517. Retrieved from `https://acp.copernicus.org/articles/5/2497/2005/` doi: 10.5194/acp-5-2497-2005

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, *24*.

Beucler, T., Pritchard, M., Gentine, P., & Rasp, S. (2020). Towards Physically-Consistent, Data-Driven Models of Convection. *International Geoscience and Remote Sensing Symposium (IGARSS)*(1), 3987–3990. doi: 10.1109/IGARSS39084.2020.9324569

Boucher, O., Randall, D., Artaxo, P., Bretherton, C., Feingold, G., Forster, P., ... Zhang, X.-Y. (2013). Clouds and aerosols. In T. [Stocker et al. (Eds.), *Climate change 2013: The physical science basis. contribution of working group i to*

the fifth assessment report of the intergovernmental panel on climate change. Cambridge, United Kingdom: Cambridge University Press.

Boulmaiz, T., Guermoui, M., & Boutaghane, H. (2020). Impact of training data size on the lstm performances for rainfall–runoff modeling. *Modeling Earth Systems and Environment*, *6*, 2153–2164.

Brenowitz, N. D., & Bretherton, C. S. (2018). Prognostic Validation of a Neural Network Unified Physics Parameterization. *Geophysical Research Letters*, *45*(12), 6289–6298. doi: 10.1029/2018GL078510

Camredon, M., Aumont, B., Lee-Taylor, J., & Madronich, S. (2007). The soa/voc/no$_x$ system: an explicit model of secondary organic aerosol formation. *Atmospheric Chemistry and Physics*, *7*(21), 5599–5610. Retrieved from `https://acp.copernicus.org/articles/7/5599/2007/` doi: 10.5194/acp-7-5599-2007

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014a). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014b). Empirical evaluation of gated recurrent neural networks on sequence modeling. arxiv 2014. *arXiv preprint arXiv:1412.3555*.

Computational and Information Systems Laboratory, CISL. (2020). *Cheyenne: HPE/SGI ICE XA System (NCAR Community Computing)* (Tech. Rep.). National Center for Atmospheric Research. Retrieved from `https://doi.org/10.5065/D6RX99HX`

de Gouw, J. A. (2005). Budget of organic carbon in a polluted atmosphere: Results from the new england air quality study in 2002. *J. Geophys. Res.*, *110*(D16). doi: 10.1029/2004JD005623

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Ann. Stat.*, 1189–1232.

Gettelman, A., Gagne, D. J., Chen, C. C., Christensen, M. W., Lebo, Z. J., Morrison, H., & Gantos, G. (2021). Machine Learning the Warm Rain Process. *Journal of Advances in Modeling Earth Systems*, *13*(2). doi: 10.1029/2020MS002268

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen. *Diploma,*

Technische Universität München, *91*(1).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hodzic, A., Aumont, B., Knote, C., Madronich, S., & Tyndall, G. (2014). Volatility dependence of Henry's law constants of condensable organics: Application to estimate depositional loss of secondary organic aerosols. *Geophysical Research Letters*, 4795–4804. doi: 10.1002/2014GL060649.Received

Hodzic, A., Campuzano-Jost, P., Bian, H., Chin, M., Colarco, P. R., Day, D. A., ... Jimenez, J. L. (2020). Characterization of organic aerosol across the global remote troposphere: a comparison of ATom measurements and global chemistry models. *Atmos. Chem. Phys.*, *20*(8), 4607–4635. doi: 10.5194/acp-20-4607-2020

Hodzic, A., & Jimenez, J. L. (2011). Modeling anthropogenically controlled secondary organic aerosols in a megacity: a simplified framework for global and climate models. *Geoscientific Model Development*, *4*(4), 901–917. Retrieved from `https://gmd.copernicus.org/articles/4/901/2011/` doi: 10.5194/gmd-4-901-2011

Hodzic, A., Kasibhatla, P. S., Jo, D. S., Cappa, C. D., Jimenez, J. L., Madronich, S., & Park, R. J. (2016). Rethinking the global secondary organic aerosol (soa) budget: stronger production, faster removal, shorter lifetime. *Atmospheric Chemistry and Physics*(16), 7917–7941. doi: 10.5194/acp-16-7917-2016

Hodzic, A., Madronich, S., Kasibhatla, P. S., Tyndall, G., Aumont, B., Jimenez, J. L., ... Orlando, J. (2015). Organic photolysis reactions in tropospheric aerosols: effect on secondary organic aerosol formation and lifetime. *Atmospheric Chemistry and Physics*, *15*(16), 9253–9269. Retrieved from `https://acp.copernicus.org/articles/15/9253/2015/` doi: 10.5194/acp-15-9253-2015

Hutter, F., Hoos, H., & Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. In *International conference on machine learning* (pp. 754–762).

Irrgang, C., Saynisch-Wagner, J., & Thomas, M. (2020). Machine learning-based prediction of spatiotemporal uncertainties in global wind velocity reanalyses. *Journal of Advances in Modeling Earth Systems*, *12*(5), e2019MS001876.

Jenkin, M. E., Saunders, S. M., Wagner, V., & Pilling, M. J. (2003, February). Protocol for the development of the Master Chemical Mechanism, MCM v3 (Part B): Tropospheric degradation of aromatic volatile organic compounds. *Atmospheric Chem. Phys.*, *3*(1), 181–193. doi: 10.5194/acp-3-181-2003

Keller, C. A., & Evans, M. J. (2019). Application of random forest regression to the calculation of gas-phase chemistry within the geos-chem chemistry model v10. *Geoscientific Model Development*, *12*(3), 1209–1225. Retrieved from `https://gmd.copernicus.org/articles/12/1209/2019/` doi: 10.5194/gmd-12-1209-2019

Kelp, M. M., Jacob, D. J., Kutz, J. N., Marshall, J. D., & Tessum, C. W. (2020). Toward stable, general machine-learned models of the atmospheric chemical system. *Journal of Geophysical Research: Atmospheres*, *125*(23), e2020JD032759. Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020JD032759` (e2020JD032759 2020JD032759) doi: https://doi.org/10.1029/2020JD032759

Kelp, M. M., Jacob, D. J., Lin, H., & Sulprizio, M. P. (2021). An online-learned neural network chemical solver for stable long-term global simulations of atmospheric chemistry. *Journal of Advances in Modeling Earth Systems*, Under Review. Retrieved from `https://eartharxiv.org/repository/view/2886/` doi: https://doi.org/10.31223/X52K7J

Kelp, M. M., Tessum, C. W., & Marshall, J. D. (2018). Orders-of-magnitude speedup in atmospheric chemistry modeling through neural network-based emulation. *arXiv*(206), 1–23.

Kouw, W. M., & Loog, M. (2018). An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*.

Kratzert, F., Klotz, D., Brenner, C., Schulz, K., & Herrnegger, M. (2018, November). Rainfall–runoff modelling using long Short-Term memory (LSTM) networks. *Hydrol. Earth Syst. Sci.*, *22*(11), 6005–6022.

La, Y. S., Camredon, M., Ziemann, P. J., Valorso, R., Matsunaga, A., Lannuque, V., ... Aumont, B. (2016). Impact of chamber wall loss of gaseous organic compounds on secondary organic aerosol formation: explicit modeling of soa formation from alkane and alkene oxidation. *Atmospheric Chemistry and Physics*, *16*(3), 1417–1431. Retrieved from `https://acp.copernicus.org/`

articles/16/1417/2016/ doi: 10.5194/acp-16-1417-2016

Lannuque, V., Camredon, M., Couvidat, F., Hodzic, A., Valorso, R., Madronich, S., ... Aumont, B. (2018). Exploration of the influence of environmental conditions on secondary organic aerosol formation and organic species properties using explicit simulations: development of the vbs-gecko parameterization. *Atmospheric Chemistry and Physics*, *18*(18), 13411–13428. Retrieved from https://acp.copernicus.org/articles/18/13411/2018/ doi: 10.5194/acp-18-13411-2018

Lee-Taylor, J., Hodzic, A., Madronich, S., Aumont, B., Camredon, M., & Valorso, R. (2015). Multiday production of condensing organic aerosol mass in urban and forest outflow. *Atmospheric Chemistry and Physics*, *15*(2), 595–615. Retrieved from https://acp.copernicus.org/articles/15/595/2015/ doi: 10.5194/acp-15-595-2015

Lee-Taylor, J., Madronich, S., Aumont, B., Baker, A., Camredon, M., Hodzic, A., ... Zaveri, R. A. (2011). Explicit modeling of organic chemistry and secondary organic aerosol partitioning for mexico city and its outflow plume. *Atmospheric Chemistry and Physics*, *11*(24), 13219–13241. Retrieved from https://acp.copernicus.org/articles/11/13219/2011/ doi: 10.5194/acp-11-13219-2011

Li, J., Cleveland, M., Ziemba, L. D., Griffin, R. J., Barsanti, K. C., Pankow, J. F., & Ying, Q. (2015). Modeling regional secondary organic aerosol using the Master Chemical Mechanism. *Atmos. Environ.*, *102*(February), 52–61. doi: 10.1016/j.atmosenv.2014.11.054

Louppe, G., Wehenkel, L., Sutera, A., & Geurts, P. (2013). Understanding variable importances in forests of randomized trees. *Advances in neural information processing systems*, *26*, 431–439.

Mauderly, J. L., & Chow, J. C. (2008, February). Health effects of organic aerosols. *Inhal. Toxicol.*, *20*(3), 257–288. doi: 10.1080/08958370701866008

Mouchel-Vallon, C., Lee-Taylor, J., Hodzic, A., Artaxo, P., Aumont, B., Camredon, M., ... Madronich, S. (2020). Exploration of oxidative chemistry and secondary organic aerosol formation in the amazon during the wet season: explicit modeling of the manaus urban plume with gecko-a. *Atmospheric Chemistry and Physics*, *20*(10), 5995–6014. Retrieved

from https://acp.copernicus.org/articles/20/5995/2020/ doi: 10.5194/acp-20-5995-2020

Mousavi, S. M., & Beroza, G. C. (2020). A machine-learning approach for earthquake magnitude estimation. *Geophysical Research Letters*, *47*(1), e2019GL085976.

Ng, N. L., Kroll, J. H., Chan, A. W. H., Chhabra, P. S., Flagan, R. C., & Seinfeld, J. H. (2007). Secondary organic aerosol formation from m-xylene, toluene, and benzene. *Atmospheric Chemistry and Physics*, *7*(14), 3909–3922. Retrieved from https://acp.copernicus.org/articles/7/3909/2007/ doi: 10.5194/acp-7-3909-2007

Reddy, D. S., & Prasad, P. R. C. (2018). Prediction of vegetation dynamics using ndvi time series data and lstm. *Modeling Earth Systems and Environment*, *4*(1), 409–419.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Schreck, J. S., & Gagne, D. J. (2021). *Earth computing hyperparameter optimization.* GitHub. Retrieved from https://github.com/NCAR/echo-opt

Tsigaridis, K., Daskalakis, N., Kanakidou, M., Adams, P. J., Artaxo, P., Bahadur, R., ... Zhang, X. (2014). The aerocom evaluation and intercomparison of organic aerosol in global models. *Atmospheric Chemistry and Physics*, *14*(19), 10845–10895. Retrieved from https://acp.copernicus.org/articles/14/10845/2014/ doi: 10.5194/acp-14-10845-2014

Figure 1.

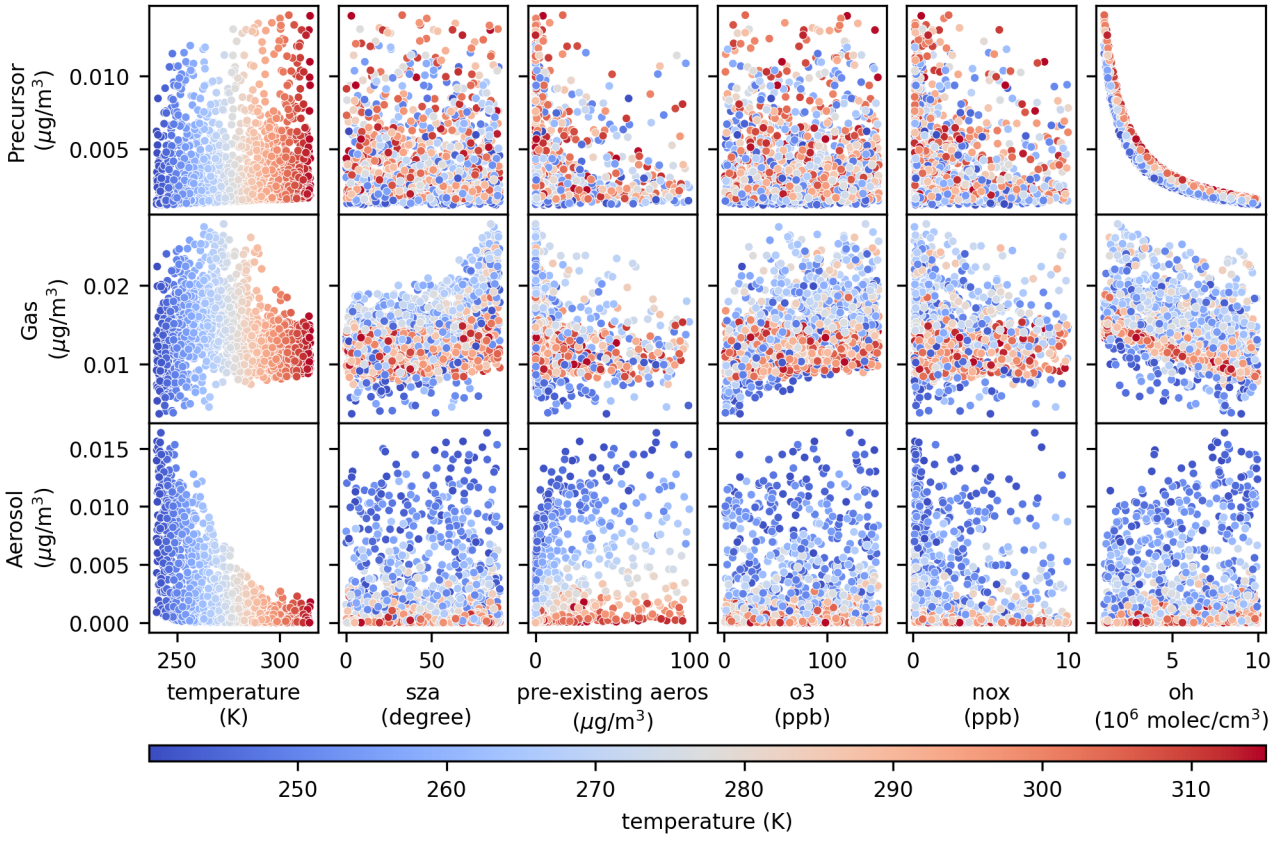

Precursor (μg/m^3)
Gas (μg/m^3)
Aerosol (μg/m^3)
0.010
0.005
0.02
0.01
0.015
0.010
0.005
0.000
250
300
temperature (K)
0
50
sza (degree)
0
100
pre-existing aeros (μg/m^3)
0
100
o3 (ppb)
0
10
nox (ppb)
5
10
oh (10^6 molec/cm^3)
250
260
270
280
290
300
310
temperature (K)

Figure 2.

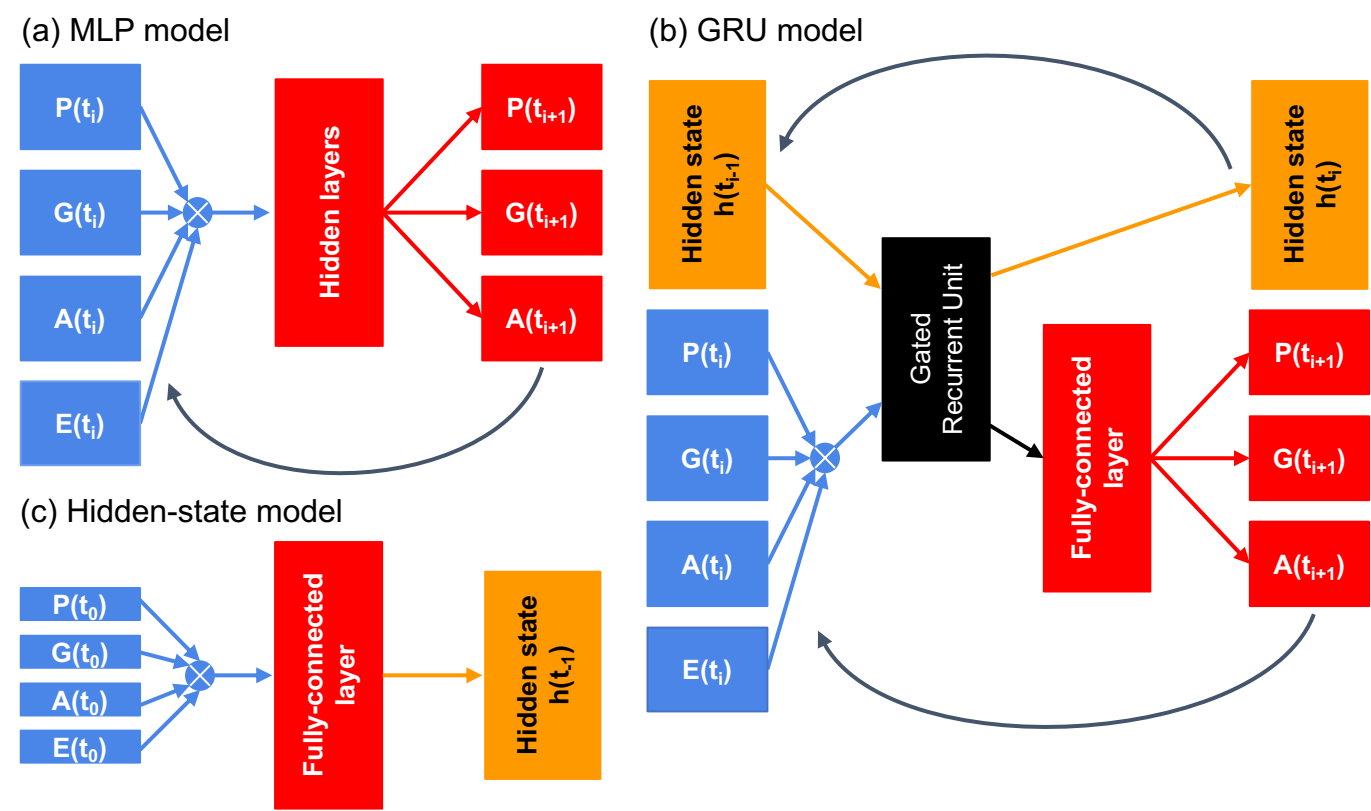

(a) MLP model
P(t_i)
G(t_i)
A(t_i)
E(t_i)
Hidden layers
P(t_i+1)
G(t_i+1)
A(t_i+1)
(b) GRU model
Hidden state h(t_i-1)
Hidden state h(t_i)
Gated Recurrent Unit
Fully-connected layer
P(t_i)
G(t_i)
A(t_i)
E(t_i)
P(t_i+1)
G(t_i+1)
A(t_i+1)
(c) Hidden-state model
P(t_0)
G(t_0)
A(t_0)
E(t_0)
Fully-connected layer
Hidden state h(t_-1)

Figure 3.

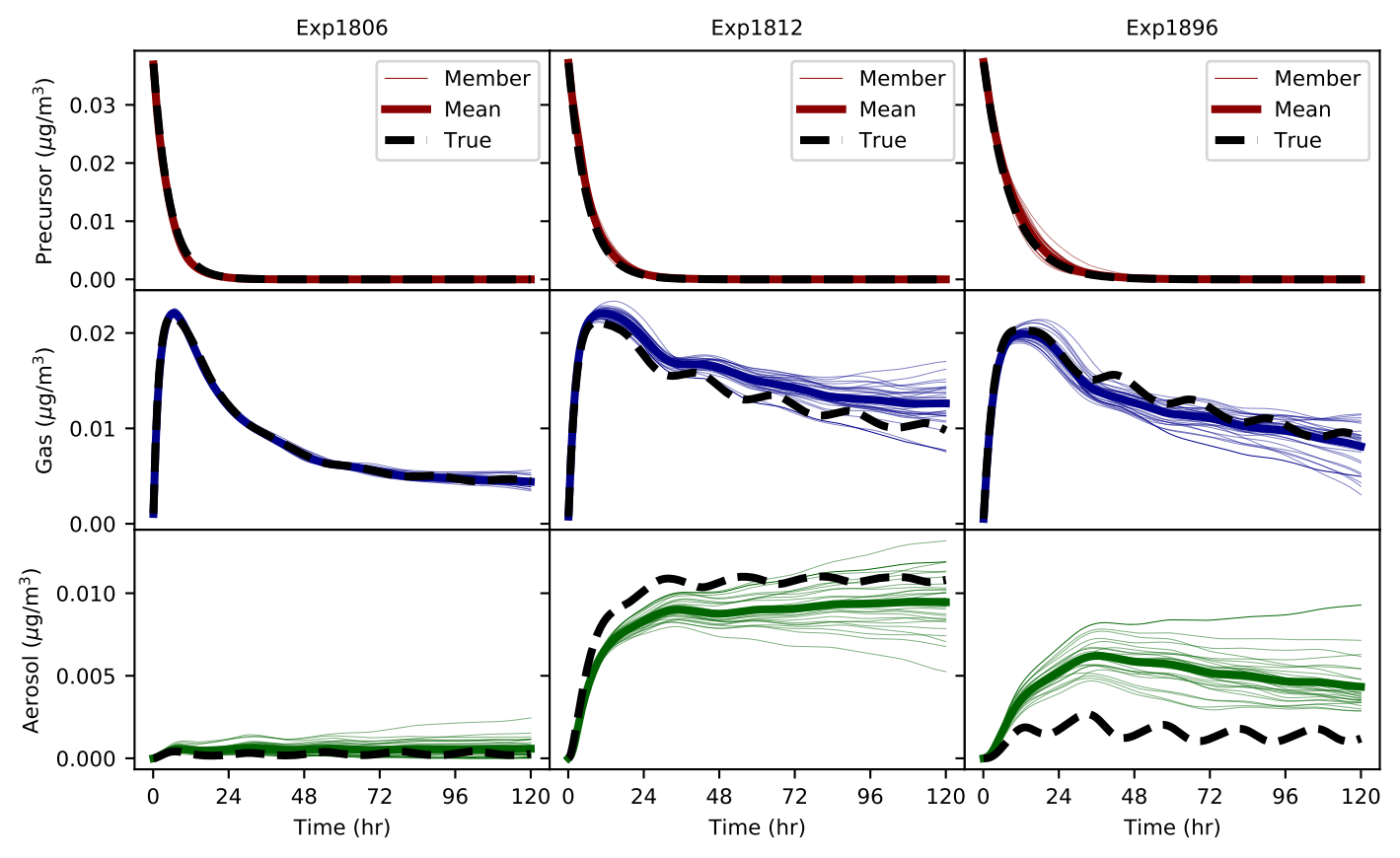

Exp1806
Exp1812
Exp1896
Member
Mean
True
Precursor (μg/m³)
Gas (μg/m³)
Aerosol (μg/m³)
0.00
0.01
0.02
0.03
0.000
0.005
0.010
0
24
48
72
96
120
Time (hr)

Figure 4.

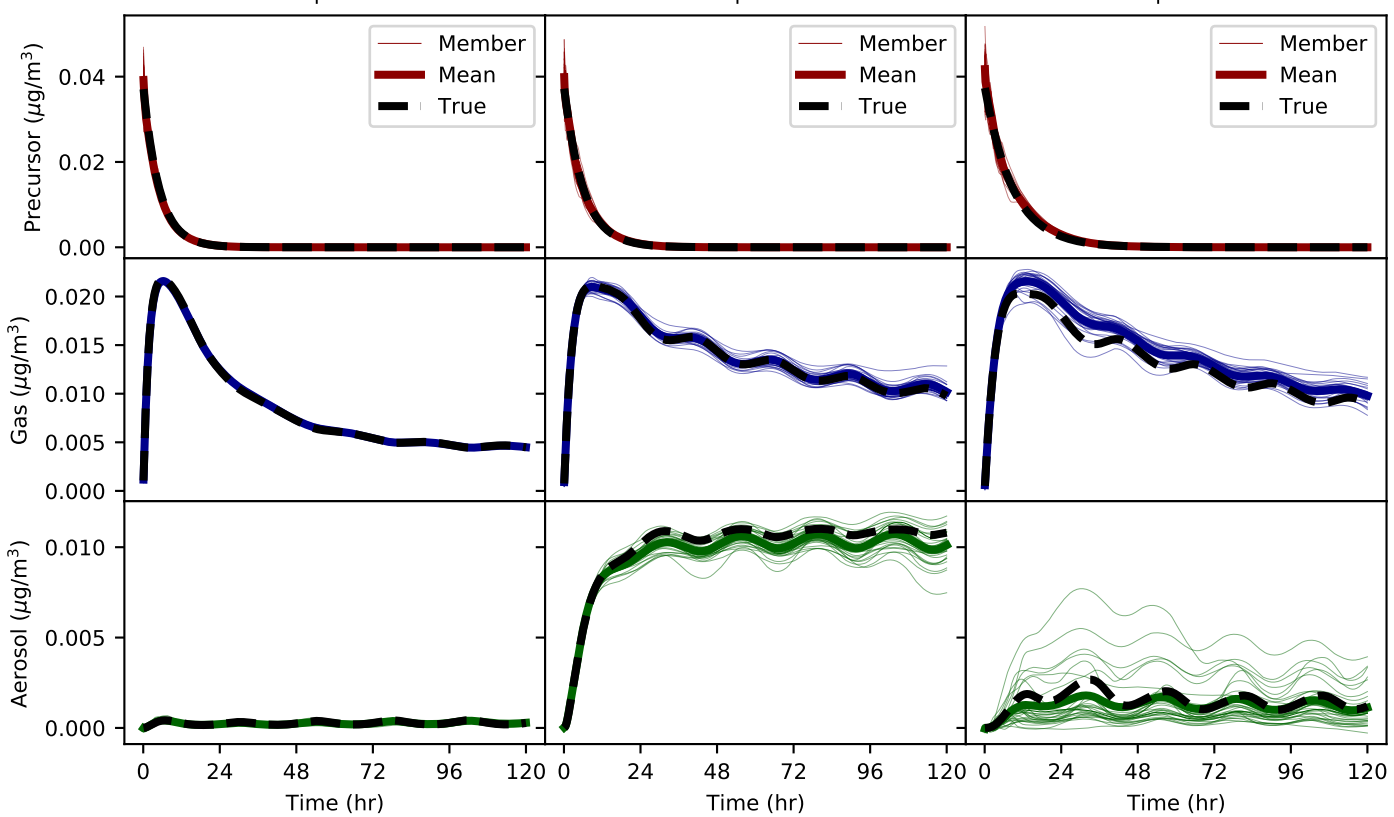

Exp1806
Exp1812
Exp1896
Member
Mean
True
Precursor (μg/m³)
Gas (μg/m³)
Aerosol (μg/m³)
0.04
0.02
0.00
0.020
0.015
0.010
0.005
0.000
0.010
0.005
0.000
0
24
48
72
96
120
Time (hr)

Figure 5.

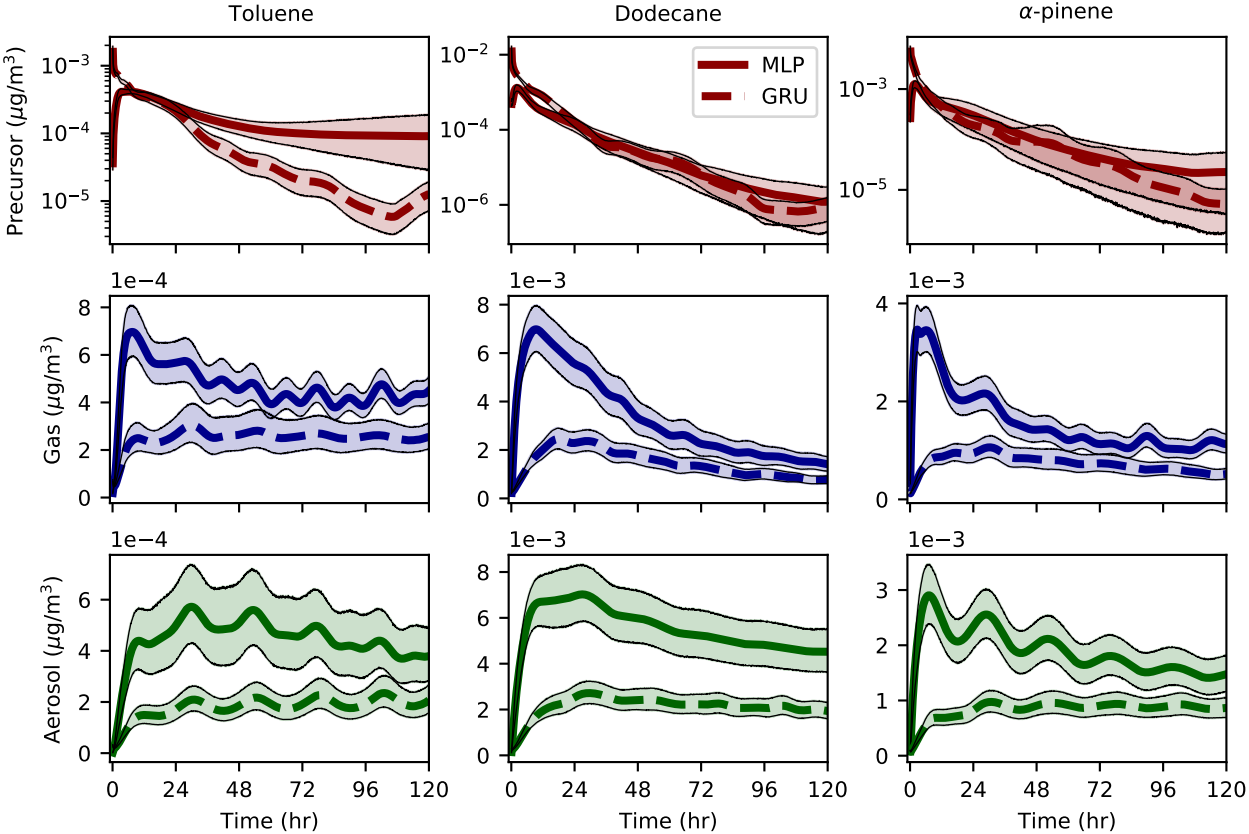

Toluene
Dodecane
α-pinene
MLP
GRU
Precursor (μg/m³)
Gas (μg/m³)
Aerosol (μg/m³)
1e−4
1e−3
Time (hr)
0
24
48
72
96
120

Figure 6.

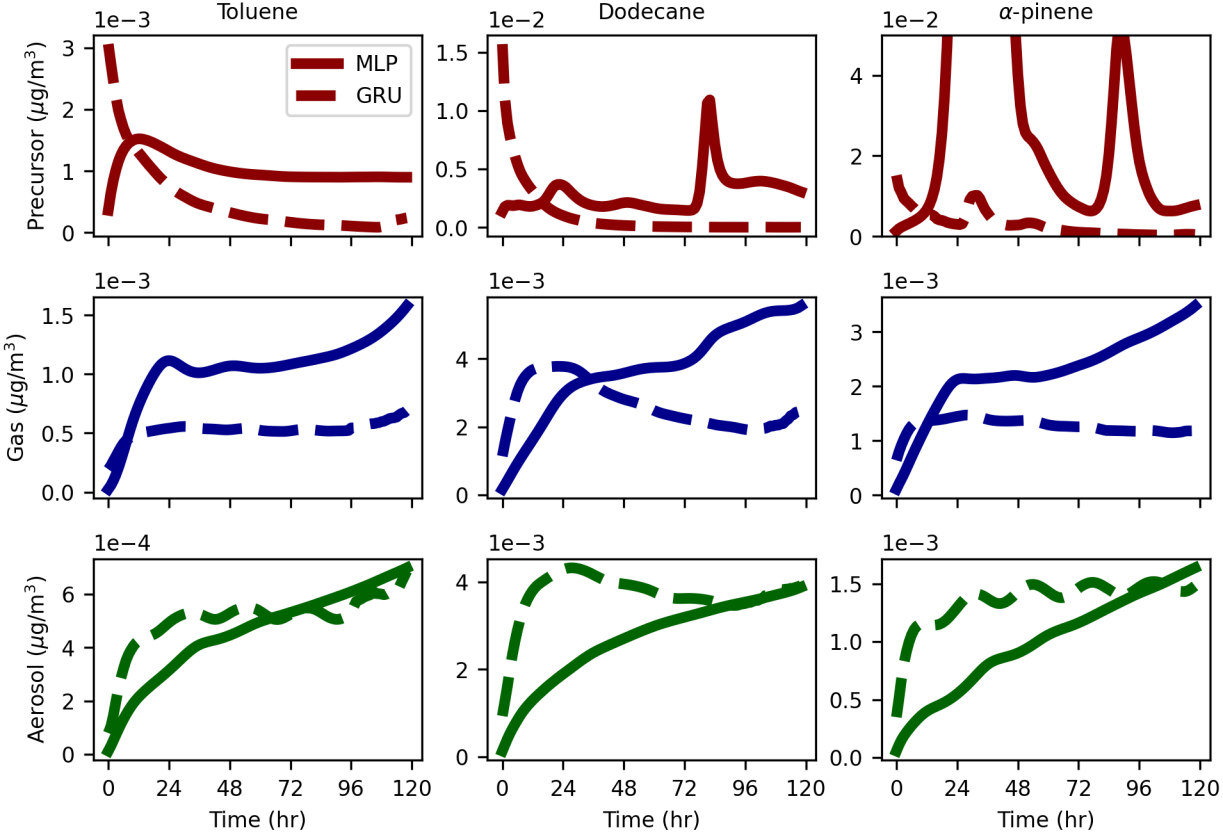

Toluene
Dodecane
α-pinene
MLP
GRU
Precursor (μg/m³)
Gas (μg/m³)
Aerosol (μg/m³)
1e−3
1e−2
1e−4
Time (hr)
0
24
48
72
96
120

Figure 7.

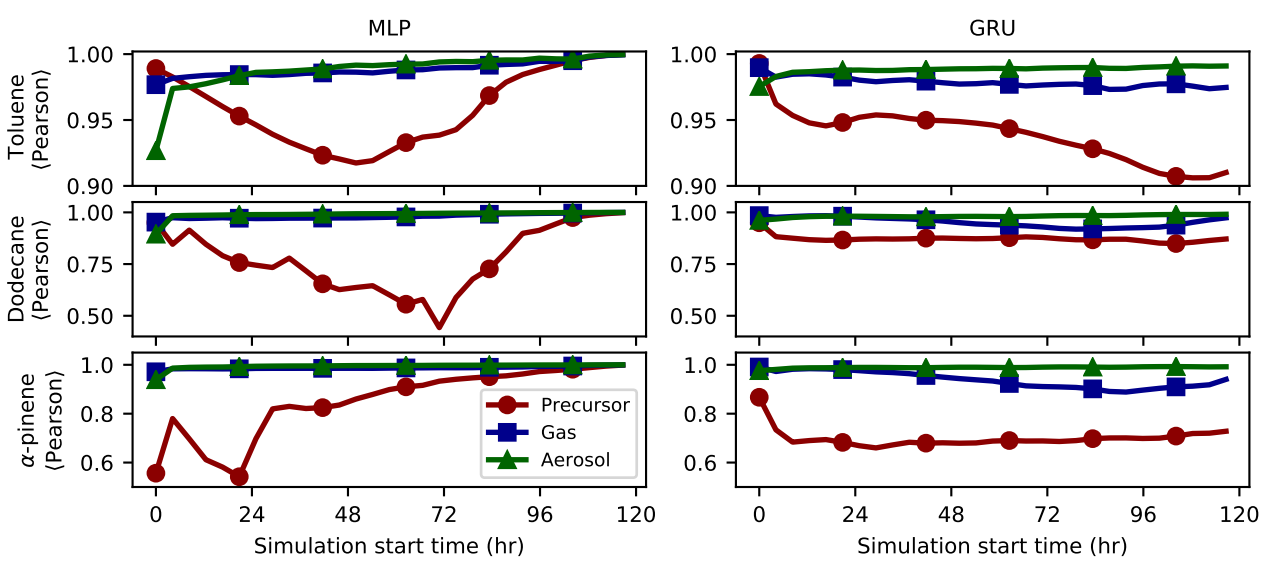

MLP
GRU
Toluene (Pearson)
Dodecane (Pearson)
α-pinene (Pearson)
1.00
0.95
0.90
0.75
0.50
0.8
0.6
Precursor
Gas
Aerosol
0
24
48
72
96
120
Simulation start time (hr)

Figure 8.

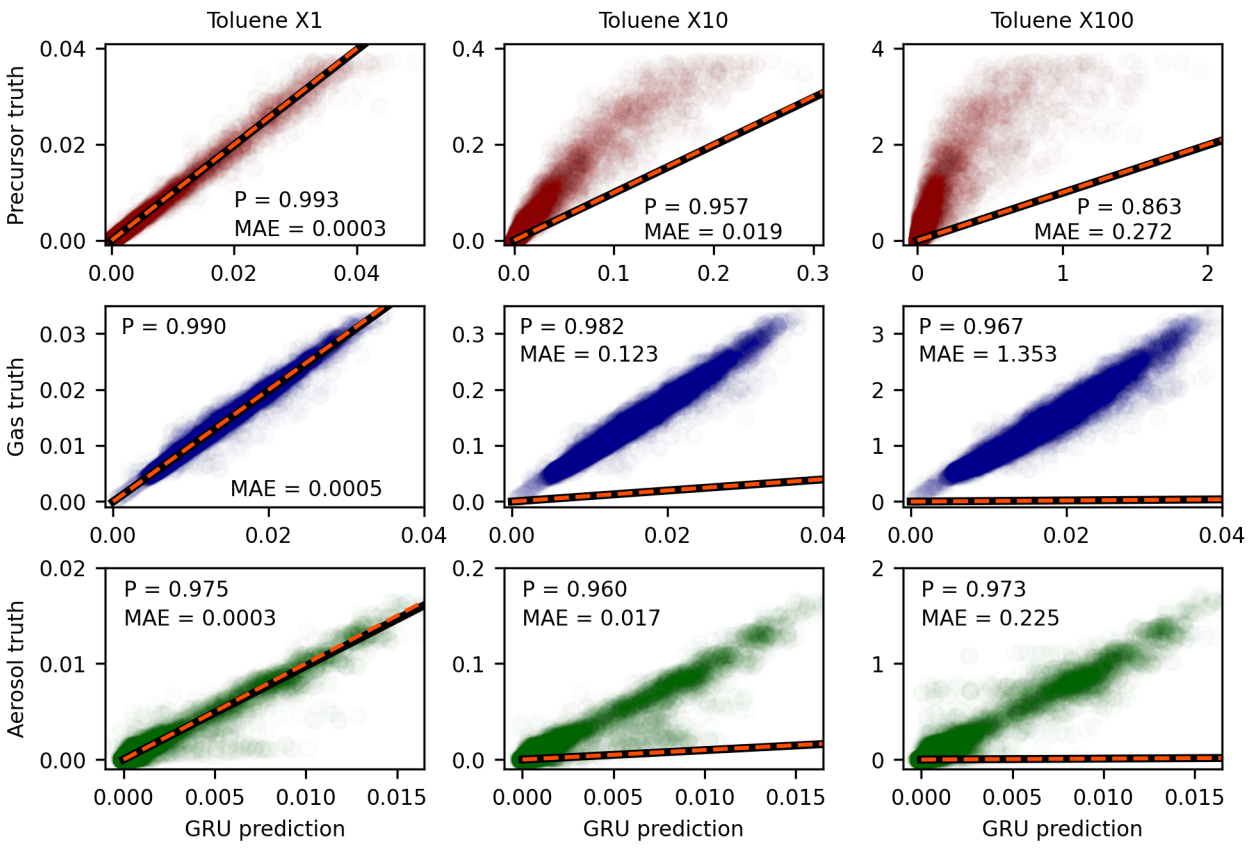

Toluene X1
Toluene X10
Toluene X100
Precursor truth
Gas truth
Aerosol truth
GRU prediction
P = 0.993
MAE = 0.0003
P = 0.957
MAE = 0.019
P = 0.863
MAE = 0.272
P = 0.990
MAE = 0.0005
P = 0.982
MAE = 0.123
P = 0.967
MAE = 1.353
P = 0.975
MAE = 0.0003
P = 0.960
MAE = 0.017
P = 0.973
MAE = 0.225

Figure 9.

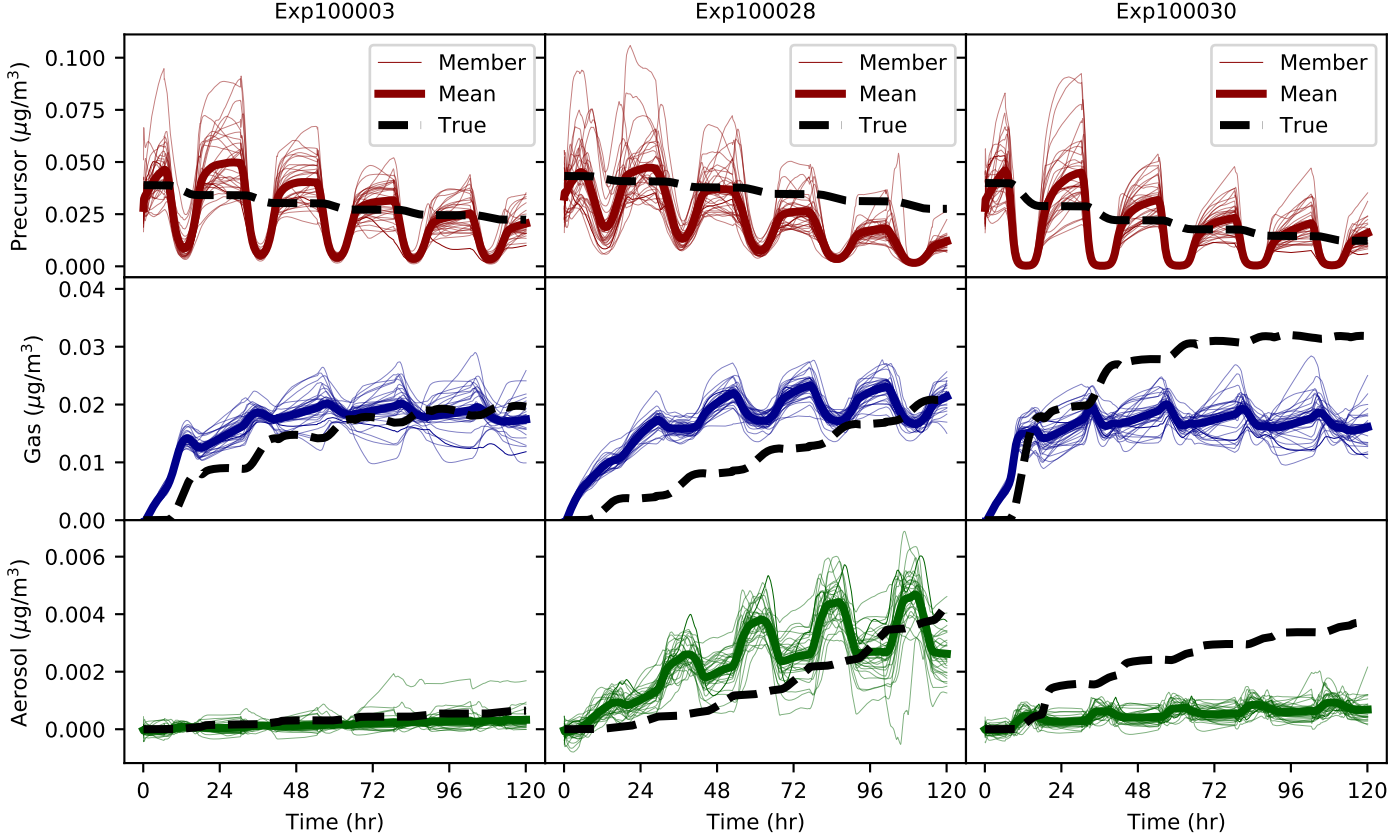

Exp100003
Exp100028
Exp100030
Member
Mean
True
Precursor (μg/m³)
Gas (μg/m³)
Aerosol (μg/m³)
0.100
0.075
0.050
0.025
0.000
0.04
0.03
0.02
0.01
0.00
0.006
0.004
0.002
0.000
0
24
48
72
96
120
Time (hr)

Figure 10.

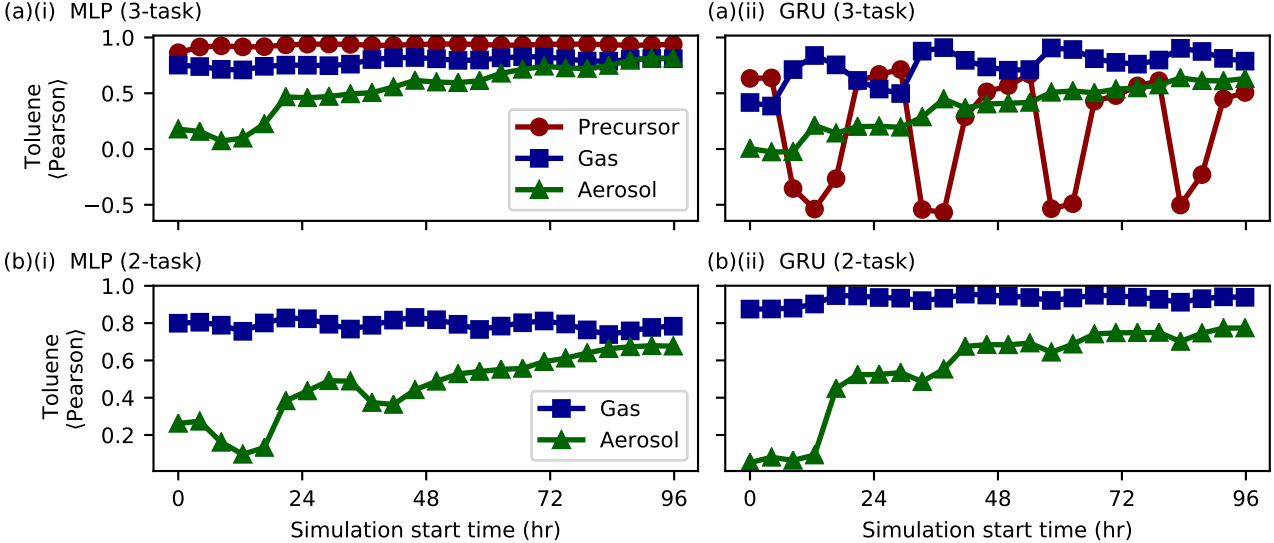

(a)(i) MLP (3-task)
(a)(ii) GRU (3-task)
Toluene ⟨Pearson⟩
1.0
0.5
0.0
−0.5
Precursor
Gas
Aerosol
(b)(i) MLP (2-task)
(b)(ii) GRU (2-task)
Toluene ⟨Pearson⟩
1.0
0.8
0.6
0.4
0.2
Gas
Aerosol
0
24
48
72
96
Simulation start time (hr)
Simulation start time (hr)

Figure 11.

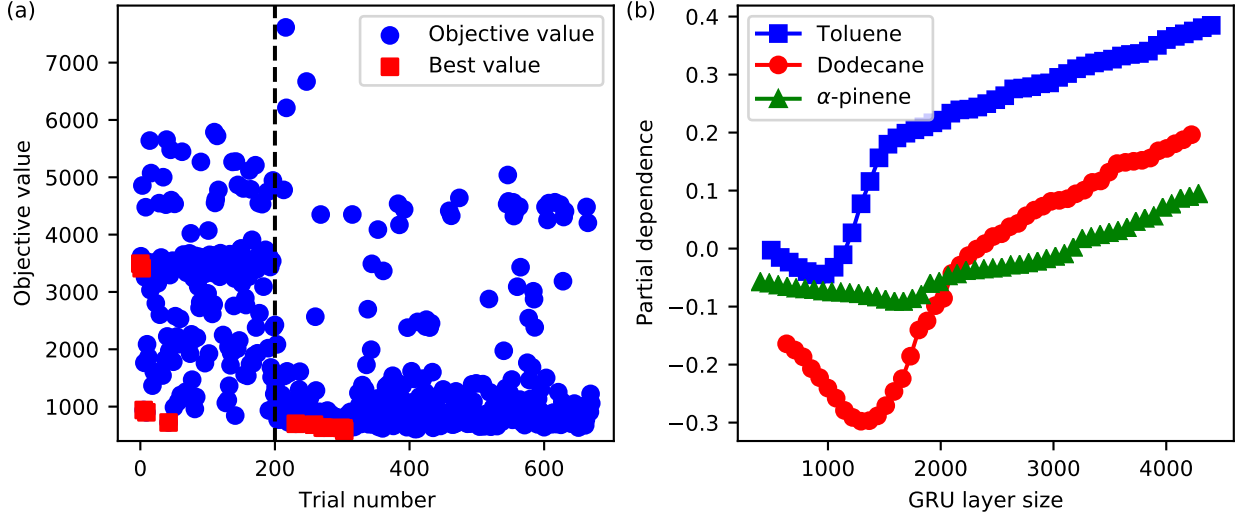

(a)
Objective value
Best value
Objective value
Trial number
(b)
Toluene
Dodecane
α-pinene
Partial dependence
GRU layer size

Figure12.

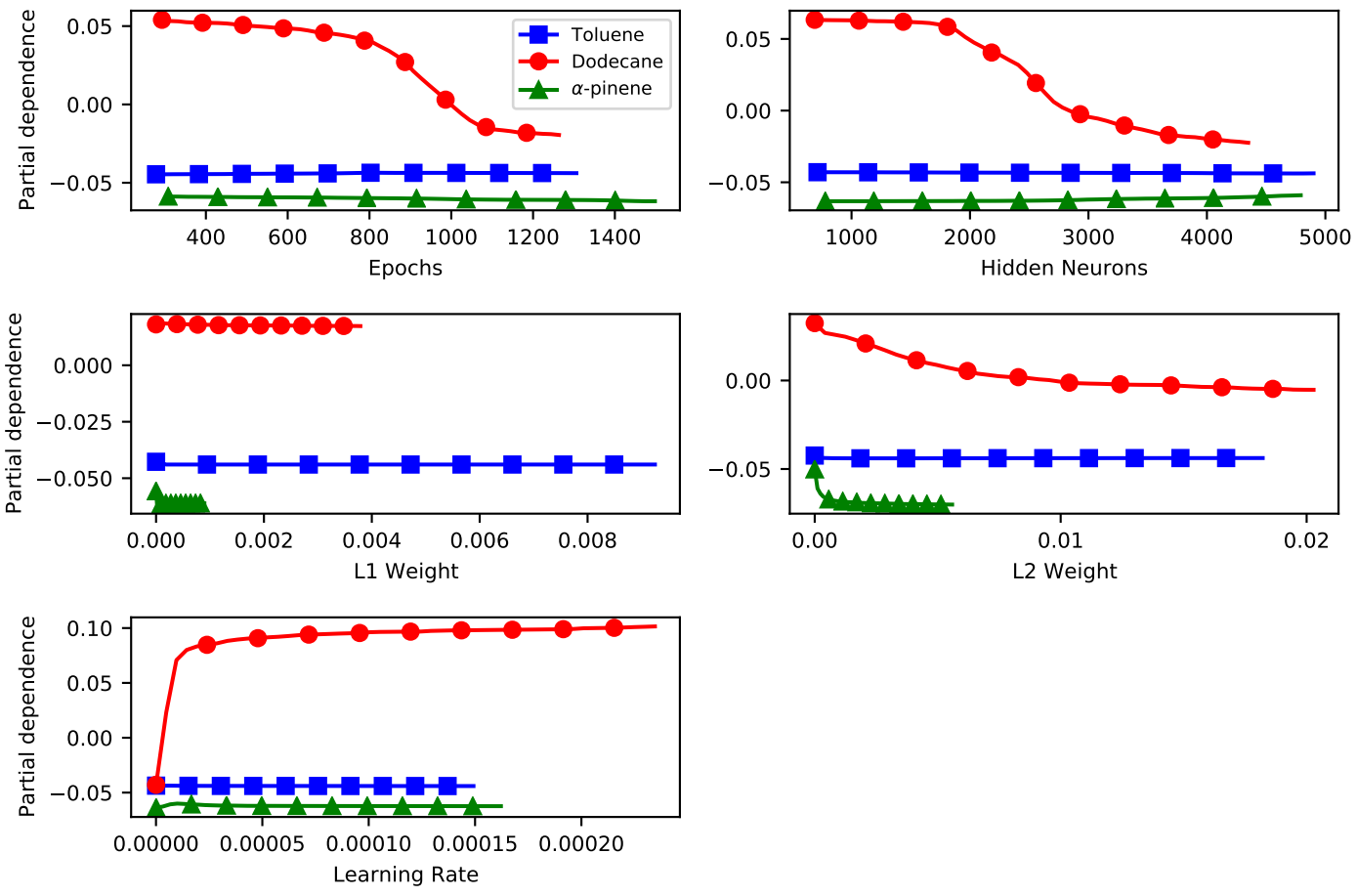

Partial dependence
0.05
0.00
−0.05
400
600
800
1000
1200
1400
Epochs
Toluene
Dodecane
α-pinene
1000
2000
3000
4000
5000
Hidden Neurons
0.000
−0.025
−0.050
0.000
0.002
0.004
0.006
0.008
L1 Weight
0.00
−0.05
0.00
0.01
0.02
L2 Weight
0.10
0.05
0.00
−0.05
0.00000
0.00005
0.00010
0.00015
0.00020
Learning Rate

Figure 13.

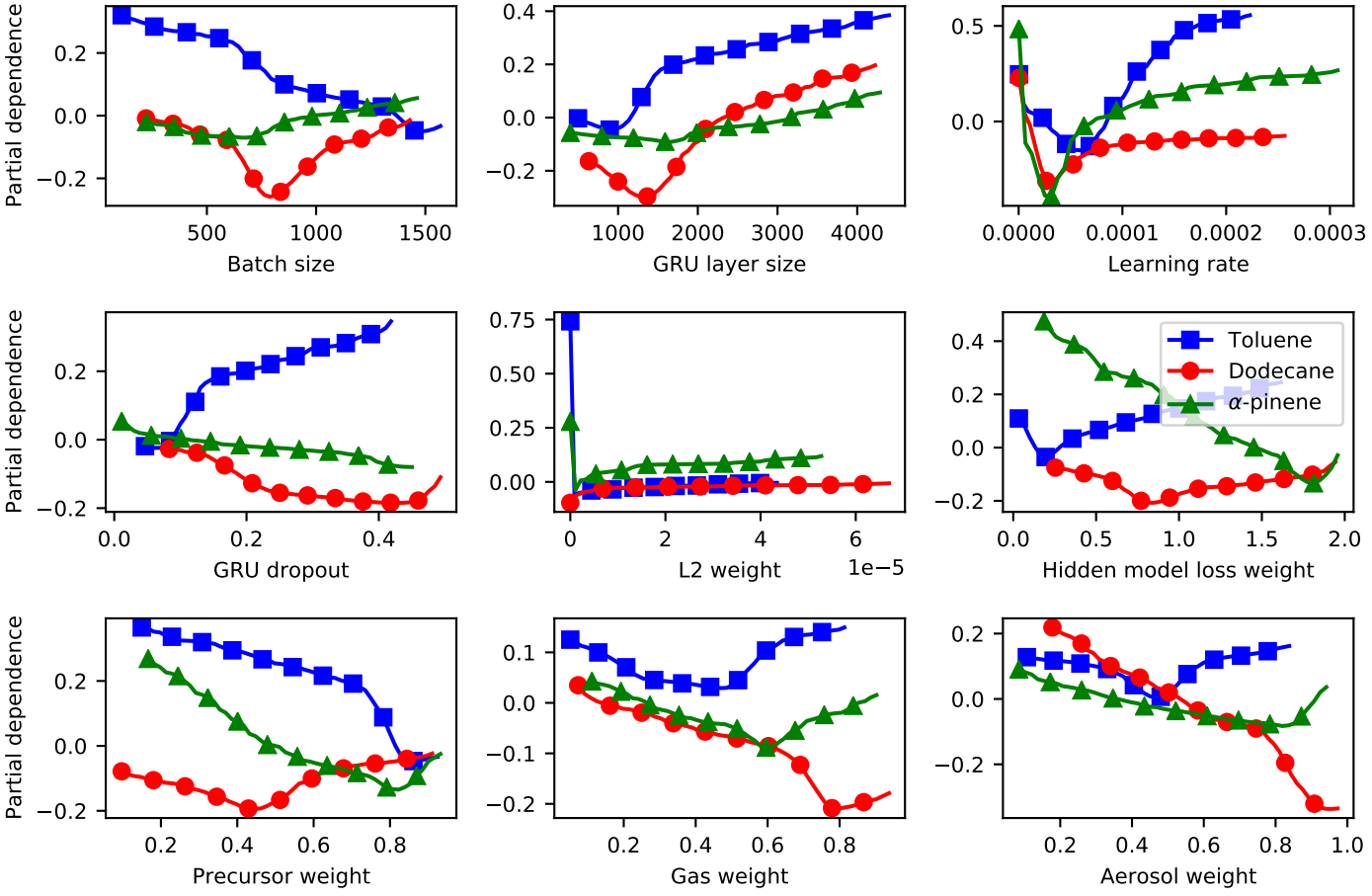

Partial dependence
Batch size
GRU layer size
Learning rate
GRU dropout
L2 weight
1e−5
Hidden model loss weight
Toluene
Dodecane
α-pinene
Precursor weight
Gas weight
Aerosol weight

Figure 14.

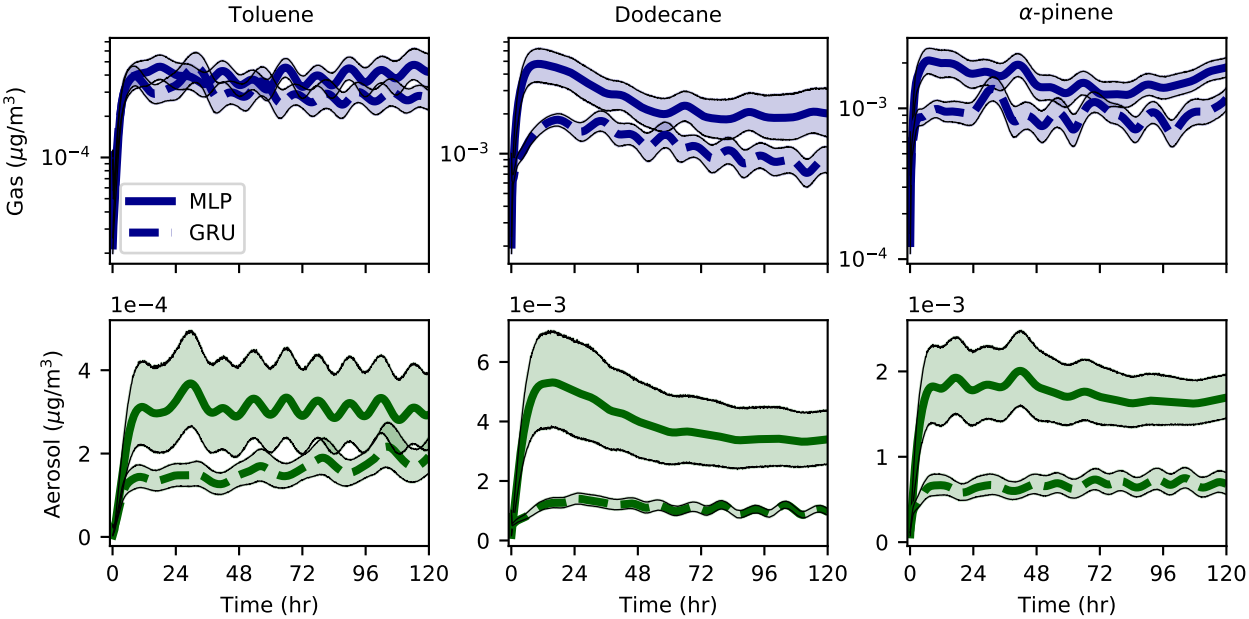

Toluene
Dodecane
α-pinene
Gas (μg/m³)
Aerosol (μg/m³)
MLP
GRU
10⁻⁴
10⁻³
1e−4
1e−3
Time (hr)
0
24
48
72
96
120

Figure 15.

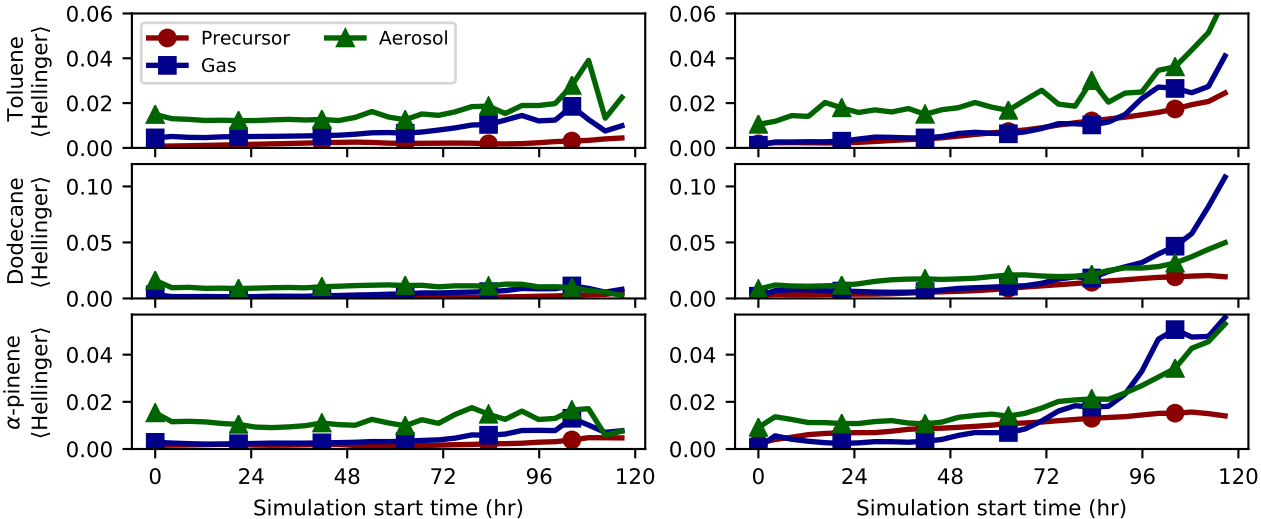

MLP
GRU
Precursor
Gas
Aerosol
Toluene (Hellinger)
Dodecane (Hellinger)
α-pinene (Hellinger)
0.06
0.04
0.02
0.00
0.10
0.05
0.00
0.04
0.02
0.00
0
24
48
72
96
120
Simulation start time (hr)
Simulation start time (hr)

Figure 16.

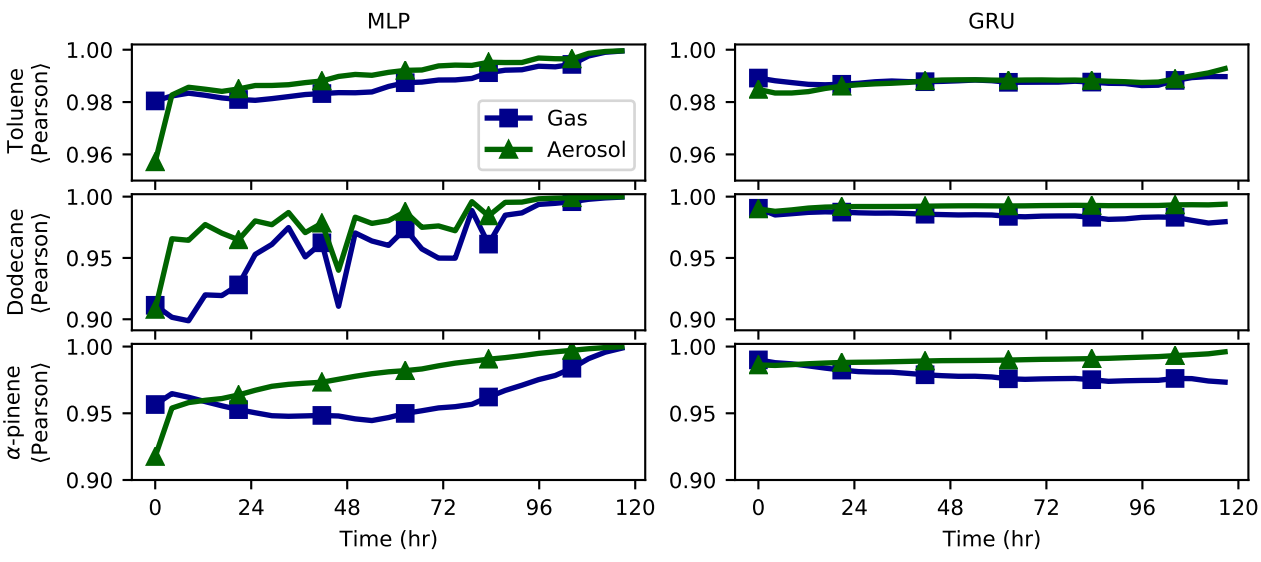

MLP
GRU
Toluene (Pearson)
Dodecane (Pearson)
α-pinene (Pearson)
Gas
Aerosol
1.00
0.98
0.96
0.95
0.90
0
24
48
72
96
120
Time (hr)

Figure 17.

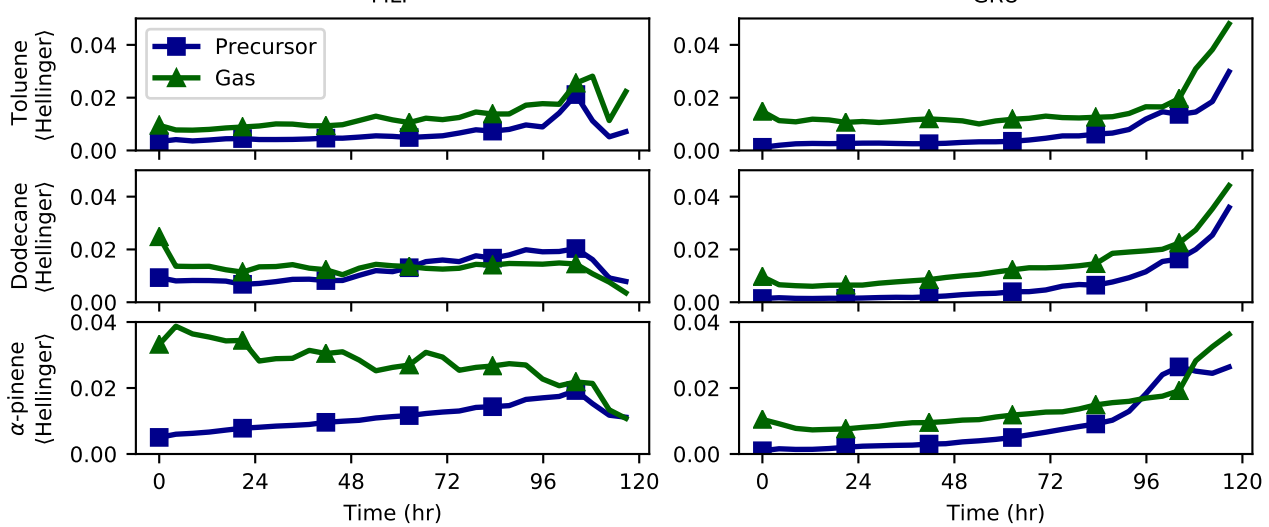

MLP
GRU
Precursor
Gas
Toluene (Hellinger)
Dodecane (Hellinger)
α-pinene (Hellinger)
0.00
0.02
0.04
0
24
48
72
96
120
Time (hr)

Figure 18.

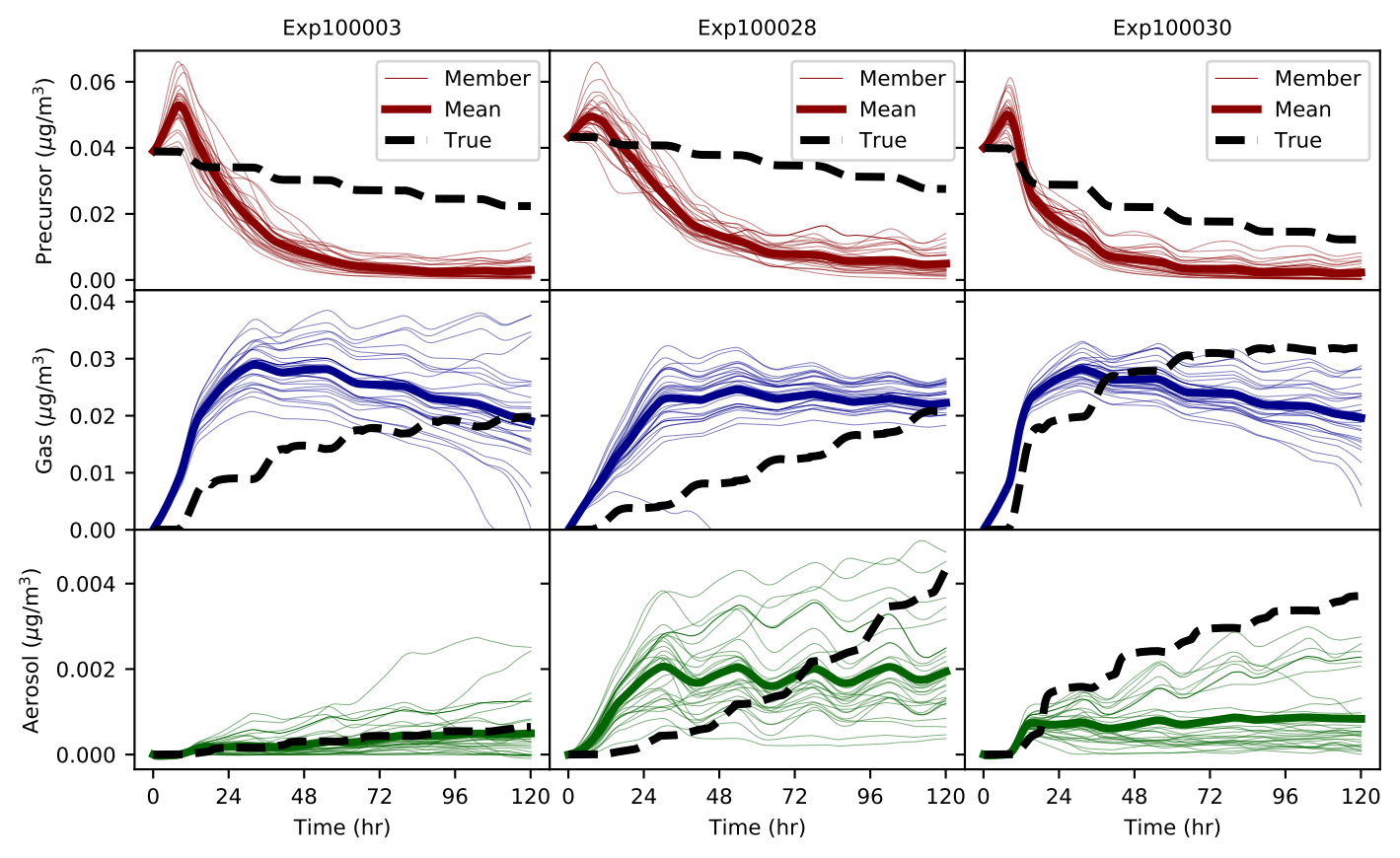
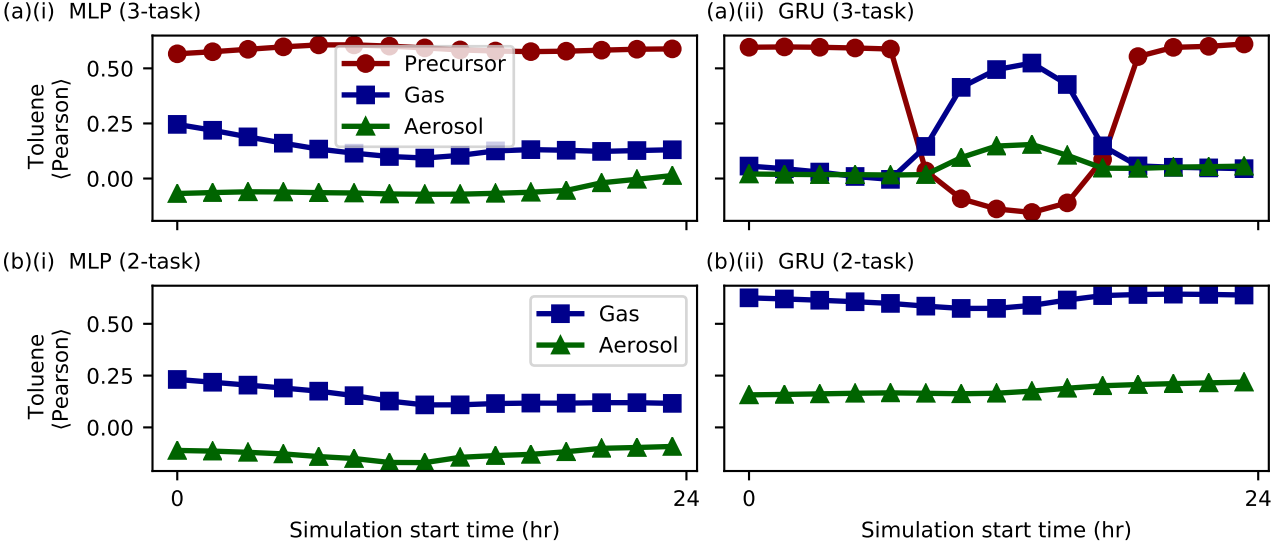

Exp100003
Exp100028
Exp100030
Member
Mean
True
Precursor (μg/m³)
Gas (μg/m³)
Aerosol (μg/m³)
0.06
0.04
0.02
0.00
0.04
0.03
0.02
0.01
0.00
0.004
0.002
0.000
0
24
48
72
96
120
Time (hr)

Figure 19.


(a)(i) MLP (3-task)
(a)(ii) GRU (3-task)
Toluene ⟨Pearson⟩
0.50
0.25
0.00
Precursor
Gas
Aerosol
(b)(i) MLP (2-task)
(b)(ii) GRU (2-task)
Toluene ⟨Pearson⟩
0.50
0.25
0.00
Gas
Aerosol
0
24
Simulation start time (hr)
0
24
Simulation start time (hr)