Environmental Data Science (2022), 4: 1–6
doi:10.1017/eds.2020.xx

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Supplementary Material: Physics-Informed Learning of Aerosol Microphysics

Paula Harder[1,2,3]*, Duncan Watson-Parris[1], Philip Stier[1], Dominik Strassel[2], Nicolas R. Gauger[4] and Janis Keuper[2,3,5]

[1]Atmospheric, Oceanic and Planetary Physics, Department of Physics, University of Oxford, Oxford, UK
[2]Fraunhofer Center High-Performance Computing, Fraunhofer ITWM, Kaiserslautern,Germany
[3]Fraunhofer Center Machine Learning, Fraunhofer Society, Germany
[4]Chair for Scientific Computing, TU Kaiserslautern, Kaiserslauter, Germany
[5]Institute for Machine Learning and Analytics, Offenburg University, Germany
*Corresponding author. Email: paula.harder@itwm.fraunhofer.de

Received: 28 January 2022

1. Modelled Variables

Table 1 shows all the variables used for our emulator. Overall we consider 36 quantities, the first eight shown in the Table are only input variables and are not changed by M7/our emulator. Masses and concentrations of different aerosol species are both inputs and outputs for the model. Additionally, we output the water content of different aerosol size bins.

2. Logarithmic Transformation

Training and evaluating on a logarithmic scale we can achieve much better scores than in the original units. On the test set, we achieve an R^2-score of 97.4%. The predictive quality on the logarithmic scale though does not transfer to the original scale, as we can see in the scores in Table 2.

3. Other ML Approaches

Apart from a neural network approach, we investigated other ML models to emulate the M7 module. We looked at linear regression (LR) and different ensemble methods such as a random forest regressor t (RF) and a gradient boosting model (GB). In Table 3 we report the test scores for linear regression, a random forest, and a gradient boosting approach. Being linear combinations of training points, linear regression and random forest both automatically conserve mass. Overall the random forest shows the best accuracy among these models. We note that the random forest performs worse than the neural network in terms of predictive accuracy, but could still be considered for future work because of the mass conserving property.

4. Network Architectures and Losses

The network architectures used all build on the same base neural network containing two hidden layers and ReLU activations. For enforcing hard constraints we add an additional layer at the end, either the

correction layer for the inequality constraint of positive masses or the completion layer of the equality constraint for mass conservation. Schematics and the combination of architectures and loss functions are shown in Figure 1.

5. Activation Functions

We explored different activation functions for our neural architecture, Sigmoid, Tanh, ReLU, and Leaky ReLU. We evaluated their performance (see Table 4) on the validation set and find that the ReLU-based activations result in better MSE and R^2-scores.

Table 1. *Modelled variables.*

VARIABLE	UNIT	INPUT	OUTPUT
PRESSURE	PA	√	
TEMPERATURE	K	√	
REL. HUMIDITY	-	√	
IONIZATION RATE	-	√	
CLOUD COVER	-	√	
BOUNDARY LAYER	-	√	
FOREST FRACTION	-	√	
H2SO4 PROD. RATE	$cm^{-3}s^{-1}$	√	
H2SO4 MASS	$\mu g\ m^{-3}$	√	√
SO4 NS MASS	$molec.\ m^{-3}$	√	√
SO4 KS MASS	$molec.\ m^{-3}$	√	√
SO4 AS MASS	$molec.\ m^{-3}$	√	√
SO4 CS MASS	$molec.\ m^{-3}$	√	√
BC KS MASS	$\mu g\ m^{-3}$	√	√
BC AS MASS	$\mu g\ m^{-3}$	√	√
BC CS MASS	$\mu g\ m^{-3}$	√	√
BC KI MASS	$\mu g\ m^{-3}$	√	√
OC KS MASS	$\mu g\ m^{-3}$	√	√
OC AS MASS	$\mu g\ m^{-3}$	√	√
OC CS MASS	$\mu g\ m^{-3}$	√	√
OC KI MASS	$\mu g\ m^{-3}$	√	√
DU AS MASS	$\mu g\ m^{-3}$	√	√
DU CS MASS	$\mu g\ m^{-3}$	√	√
DU AI MASS	$\mu g\ m^{-3}$	√	√
DU CI MASS	$\mu g\ m^{-3}$	√	√
NS CONCENTRATION	cm^{-3}	√	√
KS CONCENTRATION	cm^{-3}	√	√
AS CONCENTRATION	cm^{-3}	√	√
CS CONCENTRATION	cm^{-3}	√	√
KI CONCENTRATION	cm^{-3}	√	√
AI CONCENTRATION	cm^{-3}	√	√
CI CONCENTRATION	cm^{-3}	√	√
NS WATER	$kg\ m^{-3}$		√
KS WATER	$kg\ m^{-3}$		√
AS WATER	$kg\ m^{-3}$		√
CS WATER	$kg\ m^{-3}$		√

Table 2. *Test metrics for different architectures and transformations. Standard refers to learning with the standard transformation, log-scaled to the learning on logarithmitically transformed values. Base means the usage of only the base NN, correct adds the correction procedure and complete the completion procedure. Mass reg. and positivity reg. include the regularization terms. Best scores are bold and second best scores are highlighted in blue .*

	Standard					Log-Scaled			
Architecture	Base	+Correct	+Complete	+Mass Loss	+Positivity Loss	Base	+Correct	+Complete	+Mass Loss
R2	0.763	**0.771**	0.738	0.730	0.709	–	–	–	–
R2 log	–	–	–	–	–	0.974	0.974	0.959	**0.984**
RMSE	0.402	**0.401**	0.403	0.433	0.459	115431	115431	260706	21057757
Mass Bias SO4	$1.1 \cdot 10^{-5}$	$8.5 \cdot 10^{-5}$	**0.00**	$8.6 \cdot 10^{-6}$	$1.0 \cdot 10^{-3}$	$1.2 \cdot 10^{0}$	$1.2 \cdot 10^{0}$	**0.00**	$1.1 \cdot 10^{-3}$
Mass Bias BC	$3.8 \cdot 10^{-5}$	$1.4 \cdot 10^{-4}$	**0.00**	$3.4 \cdot 10^{-5}$	$3.6 \cdot 10^{-4}$	$2.6 \cdot 10^{-4}$	$2.6 \cdot 10^{-4}$	**0.00**	$3.2 \cdot 10^{-4}$
Mass Bias OC	$3.3 \cdot 10^{-5}$	$6.0 \cdot 10^{-5}$	**0.00**	$1.1 \cdot 10^{-5}$	$6.4 \cdot 10^{-4}$	$3.8 \cdot 10^{-4}$	$1.4 \cdot 10^{-4}$	**0.00**	$8.0 \cdot 10^{-5}$
Mass Bias DU	$1.0 \cdot 10^{-6}$	$3.9 \cdot 10^{-5}$	**0.00**	$2.8 \cdot 10^{-7}$	$1.5 \cdot 10^{-5}$	$1.0 \cdot 10^{-3}$	$2.2 \cdot 10^{-4}$	**0.00**	$8.5 \cdot 10^{-5}$
Mass Violation	$3.7 \cdot 10^{-4}$	$1.1 \cdot 10^{-3}$	**0.00**	$1.4 \cdot 10^{-4}$	$2.4 \cdot 10^{-3}$	$4.0 \cdot 10^{2}$	$4.0 \cdot 10^{2}$	**0.00**	$3.2 \cdot 10^{-1}$
Neg. Fraction	0.134	**0.00**	0.146	0.144	0.0894	0.0807	**0.00**	0.0812	0.061
Neg. Mean	0.00151	**0.00**	0.00170	0.00169	0.000081	0.00921	**0.00**	0.108	0.00115

Table 3. *Test metrics for different other ML models.*

	Accuracy			Mass Conservation					Positivity	
Architecture	R2	R2 log	RMSE	Bias SO4	Bias BC	Bias OC	Bias DU	RMSE	Neg. Fraction	Neg. Mean
LR	+2.10e-1	-4.19e+0	+6.95e-1	-4.21e-18	-1.03e-16	-1.08e-16	+1.03e-17	+4.16e-16	+1.92e-1	-1.32e-2
RF	+7.07e-1	-7.30e-1	+4.66e-1	+9.90e-17	-4.78e-16	-6.19e-17	-9.27e-19	+3.85e-16	+1.31e-1	-7.28e-4
GB	+6.66e-1	-2.95e-1	+4.76e-1	+1.90e-5	+1.75e-5	+1.45e-5	-8.56e-5	+1.14e-2	+1.27e-1	-1.25e-3

Table 4. *Validation scores for different activation functions.*

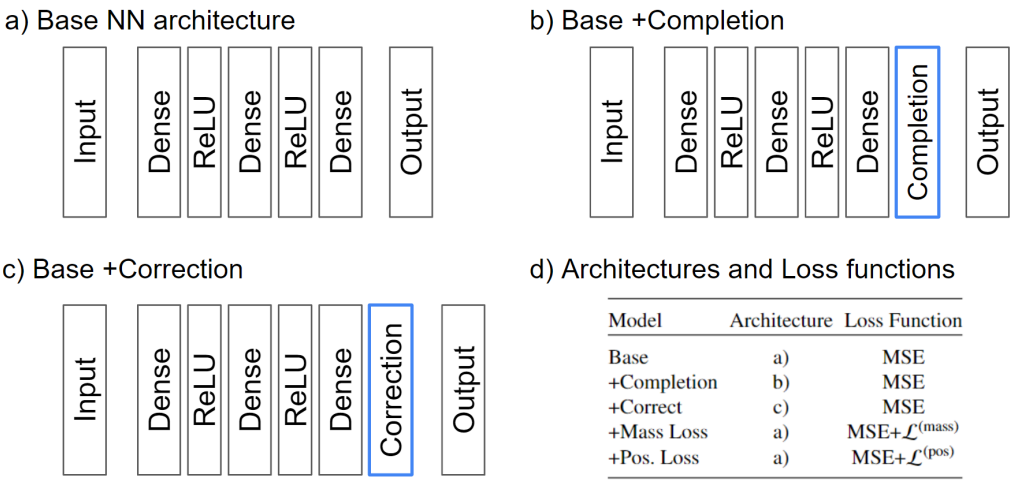
Activation Function	MSE	R^2
sigmoid	0.442	74.2
tanh	0.446	73.4
relu	0.395	78.9
leaky relu	0.395	78.7

6. Parameters for Regularization

The neural network's success is very sensitive to changes in the parameters of both mass and positivity regularizers. The parameters we found to work the best in this scenario are $\alpha = [10^{-7}, 2 \cdot 10^{4}, 2 \cdot 10^{3}, 10^{-1}]$ and $\beta = [10^{-11}, 10^{7}, 10^{7}, 10^{3}, 10^{-8}, 10^{1}]$ for the training with linear transformed values and $\alpha = [10^{-8}, 10^{3}, 10^{4}, 10^{5}]$ for log-transformation. β was tuned for the different variable groups (SO4, BC, OC, DU, NUM, WAT) and not each variable independently, which could lead to further improvement.

7. Transformation for Constraining

For clarity and simplicity, we omit the fact that we need to back-transform the variables before calculating masses and negative fraction in the main paper. The more technically precise formulation is introduced here. Let $\mu_x, \sigma_x \in \mathbb{R}^{32}$ be the mean and standard deviation vectors over all training input data and $\mu_y, \sigma_y \in \mathbb{R}^{28}$ the mean over all training output data. Then we set the function $g : \mathbb{R}^{28} \mapsto \mathbb{R}^{28}$



Model	Architecture	Loss Function
Base	a)	MSE
+Completion	b)	MSE
+Correct	c)	MSE
+Mass Loss	a)	MSE+$\mathcal{L}^{(\text{mass})}$
+Pos. Loss	a)	MSE+$\mathcal{L}^{(\text{pos})}$

Figure 1. *Different architectures and loss functions used for comparison..*

as the back transformation for the output data, $g(y) = y * \sigma_y + \mu_y$ and $h : \mathbb{R}^{32} \mapsto \mathbb{R}^{32}$ as the back transformation for the input data, $h(x) = x * \sigma_x + \mu_x$. With that, the loss terms are defined as:

$$\mathcal{L}^{(\text{mass})}(y_\theta) := \sum_{s \in S} \alpha_s | \sum_{i \in I_s} g(y_\theta^{(i)})|, \tag{1}$$

for the mass conservation and

$$\mathcal{L}^{(\text{pos})}(y_\theta) := \sum_{i=0}^{n} \beta_i ReLU(-(g(y_\theta^{(i)}) + h(x_i)))^2 \tag{2}$$

for the positivity term.

The same applies for hard-constraining. The equations for that are:

$$y_\theta^{(i)} = \text{ReLU}((g(\tilde{y}_\theta^{(i)}) + h(x_i))) - h(x_i)$$

and

$$y_\theta^{(j)} = (- \sum_{i \in I_s \setminus \{j\}} g(y_\theta^{(i)}) - \mu_{y_j})/\sigma_{y_j}.$$

8. Metrics Development

In Figure 2 we show the development of MSE, R^2-score, average mass conservation violation, and overall negative fraction for different losses throughout training. We can observe that the accuracies expressed in MSE and R^2-score develop very similar for all loss types. Adding the mass violation term we can see an improvement in mass conservation, especially for later layers, adding the positivity term we can see a decrease in negative fraction for all the epochs. It is also noticeable, that the negative fraction shows high variability between different epochs.

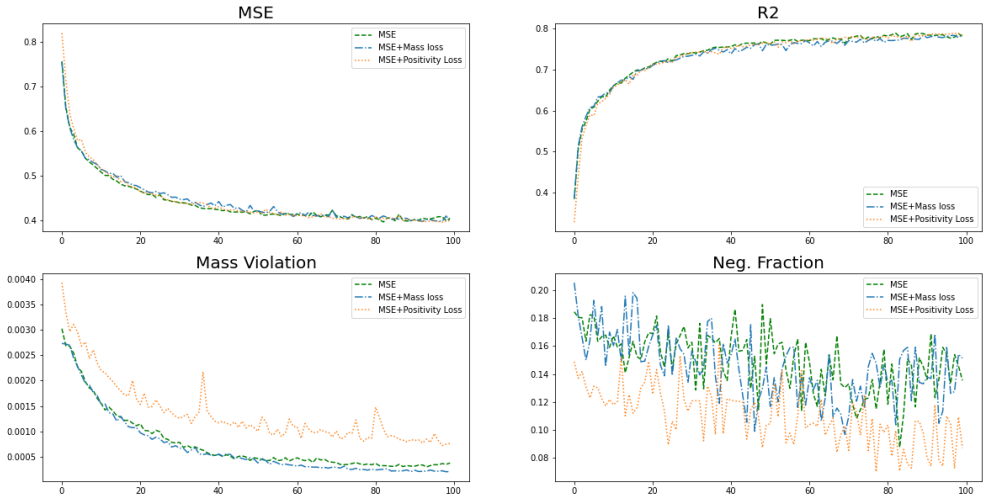


Figure 2. *Development of MSE, R^2-score, mass conservation violation and negative fraction over the course of training epochs, measured on the validation set..*

Table 5. *Scores for classification neural network, calculated on test set.*

Variable	SO4 NS	SO4 KS	SO4 AS	BC KS	BC AS	OC KS	OC AS	DU AS	NUM NS	NUM KS	NUM AS
Accuracy	0.986	0.962	0.999	0.960	0.999	0.962	0.999	0.999	0.989	0.984	0.959
Precision	0.954	0.931	0.999	0.914	0.999	0.908	0.999	1.000	0.972	0.755	0.919
Recall	0.990	0.988	1.000	0.972	1.000	0.968	1.000	1.000	0.989	0.844	0.768

Table 6. *R^2 scores on the test set for each variable. For our neural network that learns the log-transformed tendencies we consider the R^2 score on the log-transformed values, for the neural network trained on standardized values we consider the R^2 in original units.*

	SO4					Black Carbon				Organic Carbon			
Variable	H2SO4	NS	KS	AS	CS	KS	AS	CS	CI	KS	AS	CS	CI
Log. (R^2)	0.999	0.993	0.981	0.993	0.992	0.979	0.995	0.990	0.997	0.984	0.997	0.994	0.998
Stand. (R^2)	0.983	0.877	0.652	0.875	0.292	0.837	0.844	0.000	0.867	0.616	0.669	0.200	0.710

9. Results for Classification

To enable log-transformed learning we need to combine our regression neural network with a classification network. The classification network predicts whether a tendency is positive or negative. The scores for this binary classification task can be found in Table 5. Note that this classification problem is highly imbalanced and could be improved by using techniques for this case.

Table 7. *Same as table above for additional variables.*

	Dust				Number particles							Water content			
Variable	AS	CS	AI	CI	NS	KS	AS	CS	KI	AI	CI	NS	KS	AS	CS
Log. (R^2)	0.995	0.996	0.995	0.996	0.993	0.997	0.998	0.997	0.997	0.977	0.983	0.979	0.982	0.979	0.982
Stand. (R^2)	0.872	0.966	0.945	0.966	0.900	0.663	0.874	0.932	0.949	0.936	0.940	0.979	0.978	0.928	0.954

10. Individual Scores and Plots

In Table 6 and 7 we present the R^2 scores for all predicted variables individually, using the base neural network for logarithmic transformed or standardized values. Figure 3 and 4 show plots for all predicted variables.

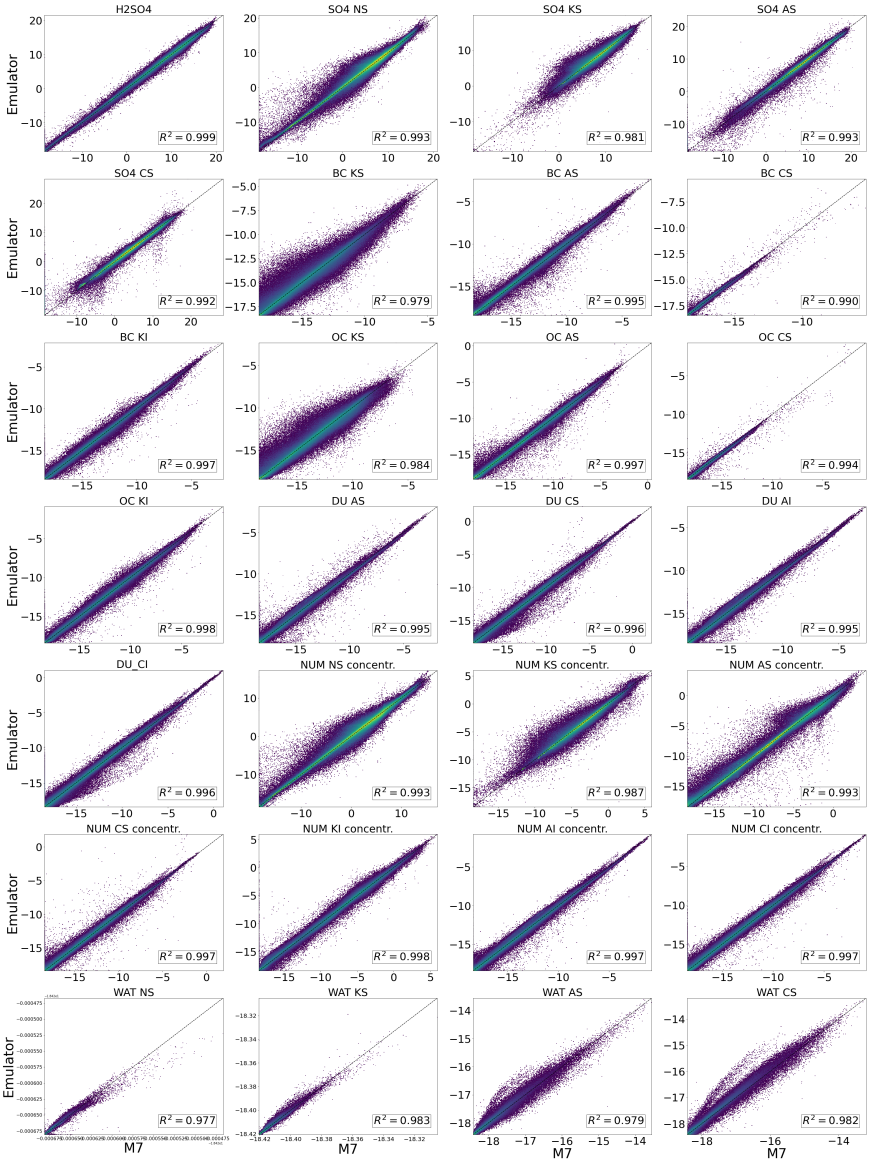


Figure 3. *Predicted test values by emulator versus true values from M7 model. Trained on logarithmically transformed quantities and plotted in the logarithmic scale.*

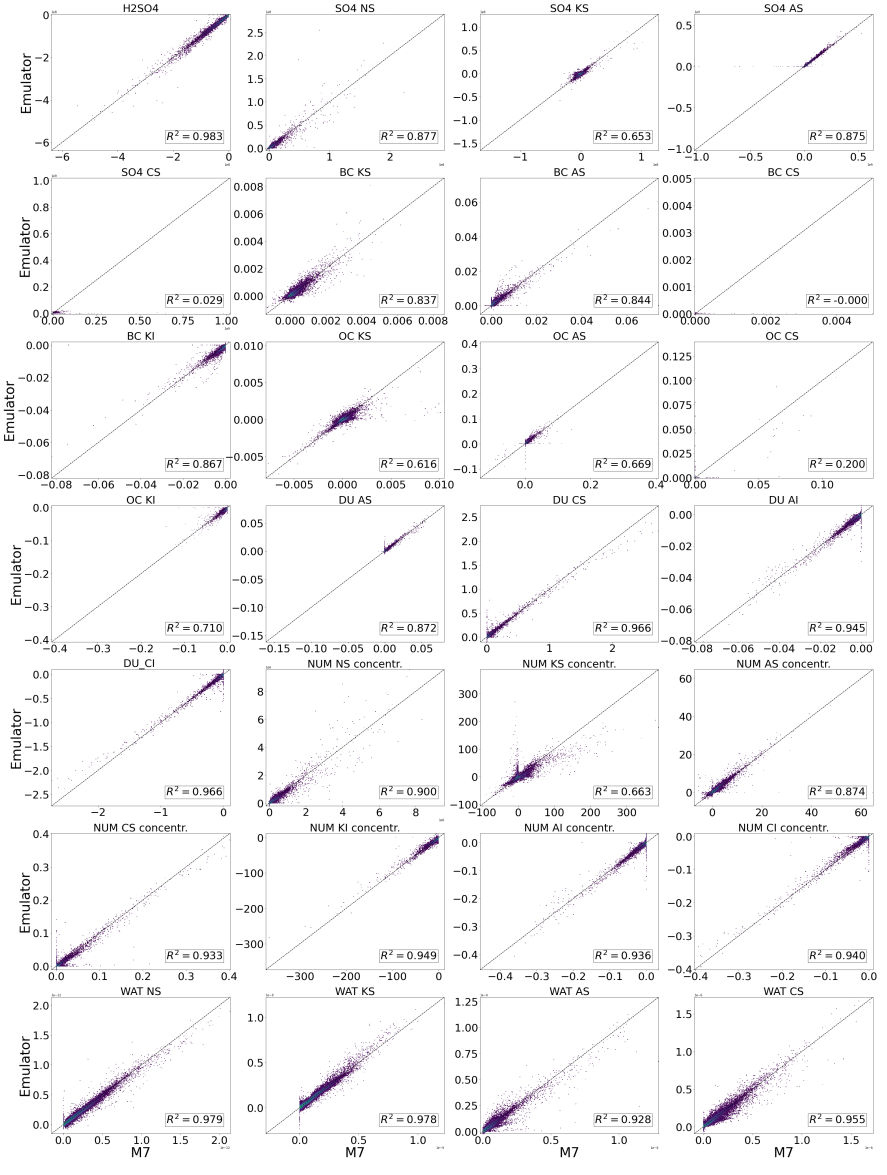


Figure 4. *Predicted test values by emulator versus true values from M7 model. Trained on standardized quantities and plotted standardized.*