IAMA Conference 2023

Physics-Constrained Learning of Aerosol Microphysics

Paula Harder

Work done with: Duncan Watson-Parris, Phillip Weiß, Philip Stier, Janis Keuper, Dominik Strassel and Nico Gauger

Content

Motivation

Motivation

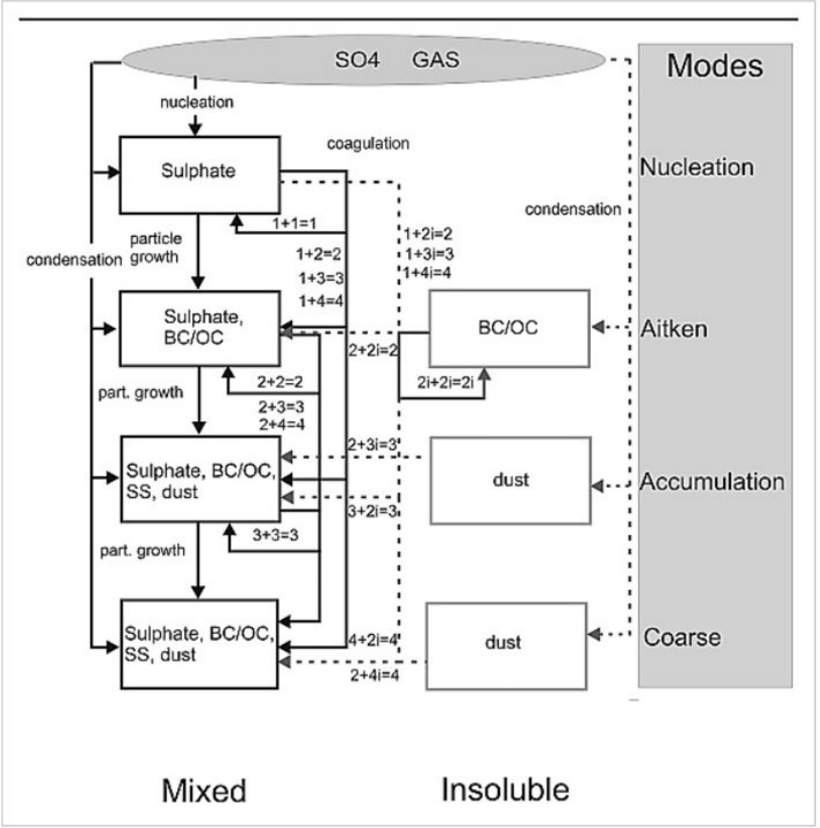
Most climate models do not include aerosols in detail and if they are included the computational costs do make long-term high-resolution runs impossible

Idea: Replace original computationally expensive model with cheap ML model = Emulation

The machine learning model is trained once offline and we benefit from fast inference time online

Data & Model

Aerosol Microphysics Model M7



Aerosol microphysics model by Vignati et al. 2004

Different aerosol types modeled with size bins

M7: An efficient size-resolved aerosol microphyiscs module for large-scale aerosol transport models, Vignatti et al. 2004

Data generation

Input/output data generated from runs of global aerosol-climate model ICON-HAM

Data covers 5 years, subsampled from locations, days, levels

Train-val-test split within time dimension, 4 years for training

~5M data points for training, ~0.5M for validation, ~0.5M for testing, each from different days spread out through the year

Data

Idea: Same input and output as original model, e.g. no spatial information

Predict one time step for fixed time step length, here 450s

35 input and 28 output variables

Input/Output: 26 values for masses and numbers of different aerosol types

Input only: Atmospheric variables, such as temperature, pressure, RH

Output only: Water content of aerosols

Data challenges

Changes in variables are small to full variables → we model changes and evaluate on changes

Changes themselves are exponentially distributed

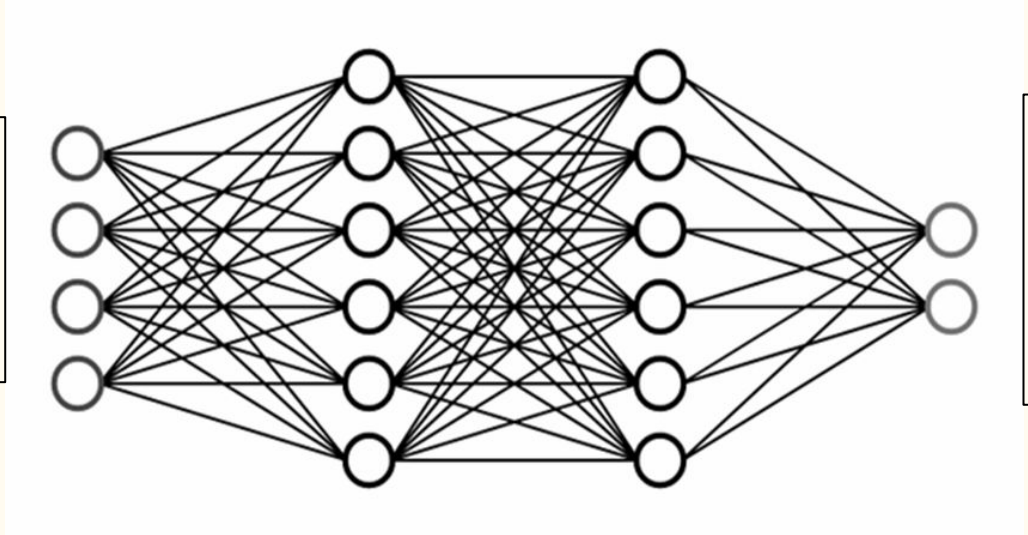
We tried both using logarithmically transformed and linearly transformed variables

Log-transformation requires an additional network to classify whether a change is positive or negative

Network architecture

Neural network with 2 hidden layers, 256 nodes per hidden layer, ReLU activation

Inputs:
Temperature, RH, pressure,...
Aerosol masses
Aerosol number



Outputs:
Change in aerosol masses
Change in aerosol numbers
Water content

Physics-Constraining

Equality and inequality constraints

Let x be the NN's input and y the output, h, g (linear) functions

Equality constraints:

$$h(x, y) = 0$$

Inequality constraints:

$$g(x, y) \geqslant 0$$

See also: Enforcing Analytic Constraints in Neural Networks Emulating Physical Systems, Beucler et al. & DC3: A learning method with hard constraints, Donti et al.

Soft vs. Hard Constraining

Let $\tilde{y}$ be the target

Soft constraining: The NN's loss L is extended by additional terms

$L(y,\tilde{y}) := ||y-\tilde{y}||^2 + \lambda$ Penalizer term

Hard constraining: Enforce constraints in additional network layer

Soft Constraints

$$L(x,y,\tilde{y}) := ||y-\tilde{y}||^2 + \lambda \underbrace{||h(x,y)||^2}_{\text{equality constraints}} + \mu \underbrace{||\text{ReLU}(-g(x,y))||^2}_{\text{inequality constraints}}$$

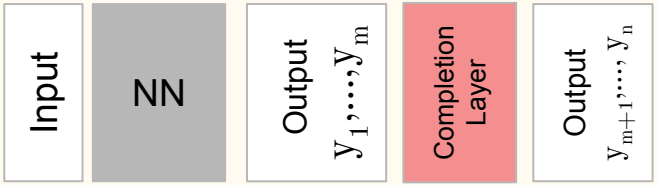
Equality constraints: Minimizing ||h(x,y)|| encourages the NN to output y, such that h(x,y) is close to 0

Inequality constraints: ReLU(z) = max(0, z), it sets the negative part to 0. Minimizing ||ReLU(-g(x,y))|| encourages the NN to output y, such that g(x,y) is positive

Hard Constraints

Equality constraints: Output partial set (y_1, ..., y_m) and then calculate (y_{m+1}, ..., y_n), such that $h(x,y)=0$

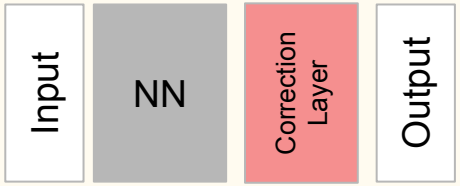
This process is called **completion**



Inequality constraints: If $g(x,y)<0$ then set y such that $g(x, y) = 0$

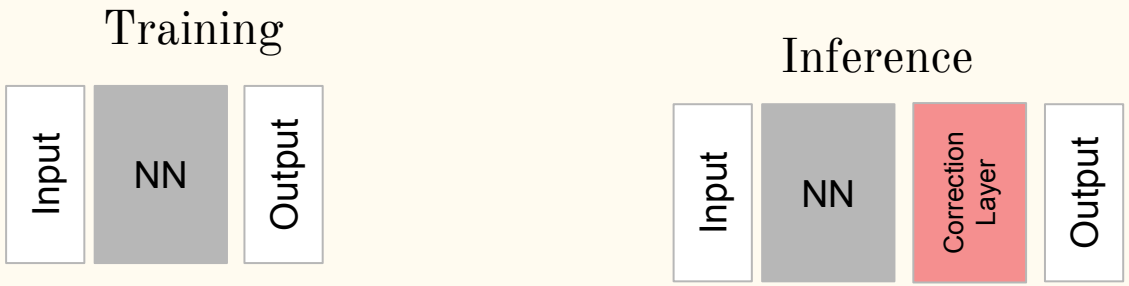
Example: Constraint is that $y \geq 0$. If $y_i < 0$ then set $y_i = 0$

This process is called **correction**



Hard Constraints - Offline vs Online

Offline:



Online:

Our Constraints

Mass conservation - Equality constraints

Let $S = \{SO4, DU, OC, BC\}$ be the set of aerosol species. For every $s \in S$ let I_s be the indices of our output y corresponding to value of that species. Mass conservation is given by

$$\sum_{i \in Is} y_i = 0$$

Soft constraining: Add loss term $||\sum_{i \in Is} y_i||^2$

Hard constraining: Choose $j \in I_s$ and set $y_j := -\sum_{i \in Is \setminus \{j\}} y_i$

Our Constraints

Mass positivity - Inequality constraints

All predicted masses have to be positive. For our input x (masses at time 0) and our output y (change in mass), the constraint is

$$y_i + x_i \geq 0$$

Soft constraining: Add loss term $||\text{ReLU}(-(y_i + x_i)||^2$

Hard constraining: Add correction layer $y_i = \text{ReLU}(y_i + x_i) - x_i$

Offline results

Results - Mass conservation

Model	Base NN	NN + mass loss (soft-constr.)	NN + offl completion (hard-constr.)	NN + onl completion (hard-constr.)
R^2	**0.87**	0.86	0.85	0.84
Overall Mass Violation	0.0015	0.00097	**0**	**0**

Hard-constrained NN satisfies mass conservation constraints completely, soft constraints slightly decrease mass violation

Constraints only very slightly decrease accuracy

Difference between online vs offline constraints small

Results - Positivity

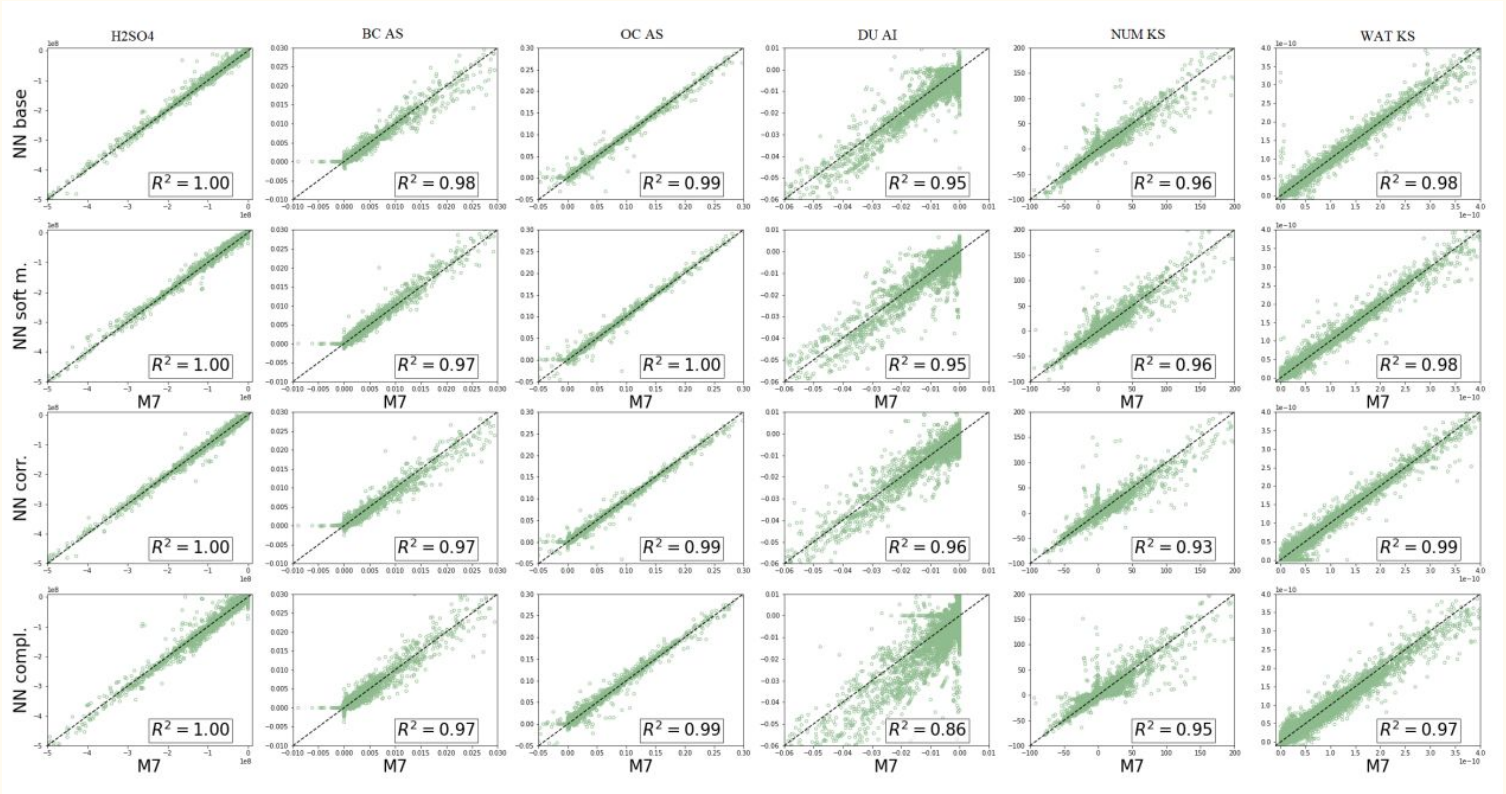
Model	NN Base	NN + Positivity Loss (soft-constr.)	NN + offl. Correction (hard-constr.)	NN + onl. Correction (hard-constr.)
R^2	**0.87**	0.68	**0.87**	0.86
Negative Fraction	0.20	0.15	**0**	**0**
Negative Mean	0.0016	0.0016	**0**	**0**

Hard-constrained NN satisfies positivity constraints completely, soft constraints decrease negative fraction, but not negative magnitude

Hard constraints do not decrease accuracy, soft do decrease

Difference between online vs offline constraints small

Results - Plots



Results - Speed-ups

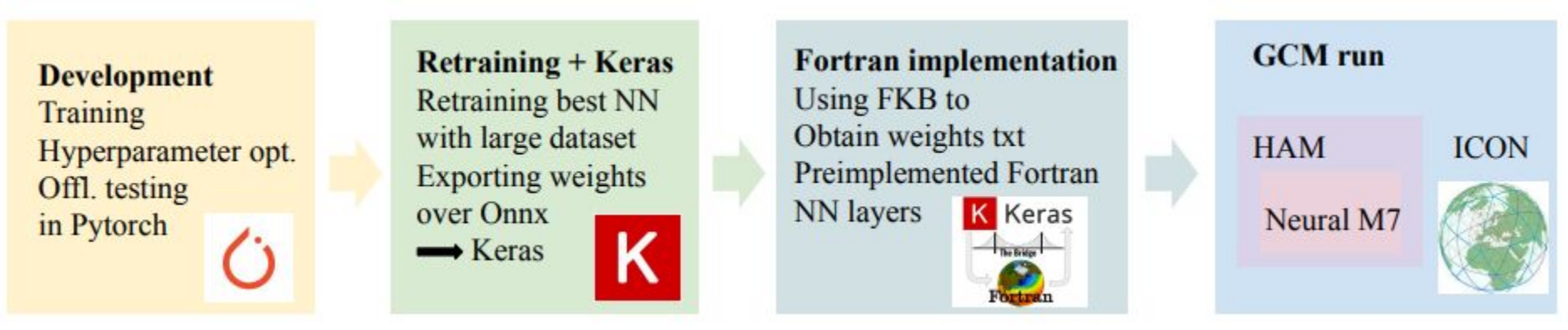
Runtimes for the prediction of one global time step:

Model	M7	NN pure GPU	NN CPU-GPU-CPU	NN CPU
Time (s)	5.781	0.000517	0.0897	2.042
Speed-up	1	11181.8	64.4	2.80

Integration in ICON

Fortran-Keras-Bridge

Use Fortran-Keras-Bridge (FKB) to integrate NN into ICON



[1] https://github.com/scientific-computing/FKB

Stability

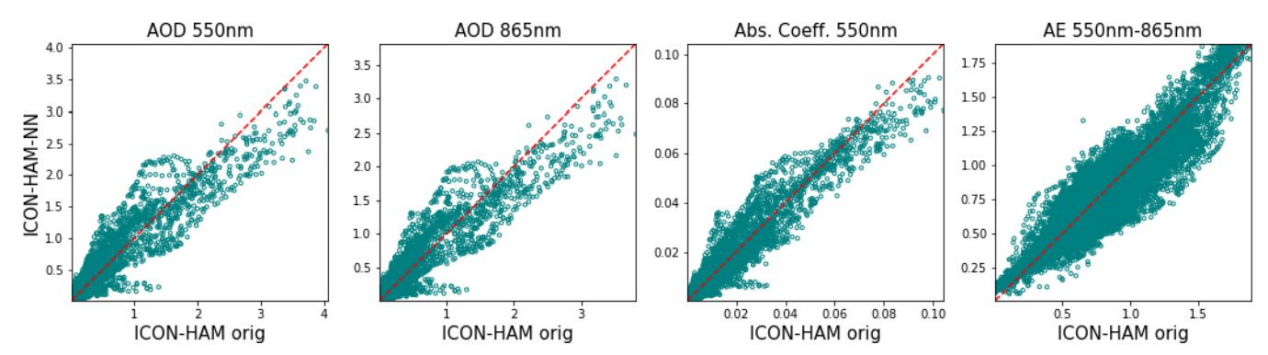
Simulation easily unstable, if out-of-distribution samples appear - > important to include samples from all year, all times of day, all areas, all vertical levels in training data

Things that helped to achieve stable runs

- Retrain with all data (including training,validation and testing) to achieve stable runs
- Bigger NN
- Constraints such as soft mass and correction layers

Accuracy

Aerosol properties after a year of ICON-NN run vs reference run



Summary

Neural networks can accurately emulate aerosol microphysics model

We can incorporate physical constraints with hard constraining

A significant speed-up, especially on a GPU can be achieved

Achieve stable and accurate online runs with the help of constraints

Online speed-up very small, due to batch size of 1

Future work

Speed-up of the emulator when coupled

- Library that supports bigger batch size
- Pruning methods to make NN smaller
- Setup where a GPU can be utilized (GPU version of ICON)

Thanks for your attention!

Contact: paula.harder@mila.quebec