Machine learning to inform bulk microphysical parameterizations

Axel Seifert
Deutscher Wetterdienst

with contributions from
Stephan Rasp and Christoph Siewert

EWGLAM Meeting
27 September 2021

Why machine learning for parameterizations?

Why machine learning for parameterizations?

➔ **Parameterization problem in atmospheric models:**
Find a nonlinear relation between resolved model variables x = (x1, x2, x3,…) and unresolved processes P = (p1, p2, p3, …)

$$P = f(x)$$

using physical reasoning, measurements or benchmark simulations.

Why machine learning for parameterizations?

➔ **Parameterization problem in atmospheric models:**
Find a nonlinear relation between resolved model variables x = (x1, x2, x3,…) and unresolved processes P = (p1, p2, p3, …)

$$P = f(x)$$

using physical reasoning, measurements or benchmark simulations.

Why machine learning for parameterizations?

➔ **Parameterization problem in atmospheric models:**
Find a nonlinear relation between resolved model variables x = (x1, x2, x3,…) and unresolved processes P = (p1, p2, p3, …)

$$P = f(x)$$

using physical reasoning, measurements or benchmark simulations.

➔ **Machine learning (supervised learning):**
Find a nonlinear relation between features x = (x1, x2, x3,…) and labels y = (y1, y2, y3, …)

$$y = f(x)$$

using data (measurements or benchmark simulations).

Bulk microphysics parameterizations

- Bulk microphysics parameterizations need to approximate integrals of the type

$$AU = \int_{x'=0}^{x_*} \int_{x''=x_*-x'}^{x_*} f(x') f(x'') K(x', x'') x' \mathrm{d}x' \mathrm{d}x'',$$

  This is the warm-rain autoconversion and *x* is particle mass, *f(x)* is the particle size distribution (PSD) and *K(x,y)* is the collision kernel. Similar integrals apply for accretion, riming and aggregation.

- Traditionally there are 3 different approaches to parameterize these integrals
  1. Analytic parameterizations with an assumed PDF for *f(x)*
  2. Tabulated schemes with pre-calculated values using an assumed PDF for *f(x)*
  3. Regression models based on simulations with bin or particle-based models as data. In this case f(x) is provided by the benchmark simulation.

- In some sense, ML-based bulk microphysics is simply a new workflow for a regression model.

ML-based microphysics development workflow

ML-based microphysics development workflow

- Do reference simulations using a resolved microphysics. Here we use the super-particle method of Shima et al. (2009)

ML-based microphysics development workflow

- Do reference simulations using a resolved microphysics. Here we use the super-particle method of Shima et al. (2009)
- The simulations should cover the whole range of physically plausible values, because the ML-based model is good for interpolation but quite bad for extrapolation.

ML-based microphysics development workflow

- Do reference simulations using a resolved microphysics. Here we use the super-particle method of Shima et al. (2009)
- The simulations should cover the whole range of physically plausible values, because the ML-based model is good for interpolation but quite bad for extrapolation.
- Calculate the bulk microphysical process rates like autoconversion or accretion based on the reference simulations.

ML-based microphysics development workflow

- Do reference simulations using a resolved microphysics. Here we use the super-particle method of Shima et al. (2009)
- The simulations should cover the whole range of physically plausible values, because the ML-based model is good for interpolation but quite bad for extrapolation.
- Calculate the bulk microphysical process rates like autoconversion or accretion based on the reference simulations.
- Train a neural network for each process rate. Here we can play with different sets of predictors.

ML-based microphysics development workflow

- Do reference simulations using a resolved microphysics. Here we use the super-particle method of Shima et al. (2009)
- The simulations should cover the whole range of physically plausible values, because the ML-based model is good for interpolation but quite bad for extrapolation.
- Calculate the bulk microphysical process rates like autoconversion or accretion based on the reference simulations.
- Train a neural network for each process rate. Here we can play with different sets of predictors.
- To validate the parameterization we solve the ODE system using the neural networks for the process rates and compare the solutions with the reference simulations.

Training the ML model

- We use the Keras/Tensorflow library to train an artificial neural network (multilayer perceptron).
- Separate data in training, validation and testing data
- Log transform of process rates (labels) and predictors (features)
- Standardization of transformed variables. After this the predictors have a mean of zero and a standard deviation of one.
- For warm-rain autoconversion possible predictors are:
  - cloud water content L_c
  - cloud droplet number N_c or drop mass x_c
  - shape parameter nu
  - rain water content L_r
  - dimensionless time scale tau=$L_r/(L_c+L_r)$

Overview of training and testing data

	Initial conditions	
	training (+validation)	testing
L_0 [g m^{-3}]	0.2, 0.4, 0.6, 0.8, 1.0, 2.0	0.3, 0.5, 0.7, 0.9, 1.5
$\bar{r}_0$ [µm]	9, 10, 11, 12, 13, 14, 15	9, 10, 11, 12, 13, 14, 15
ν	0,1,2,3,4	0.5, 1.5, 2.5, 3.5
n	5 (+2)	5

potential predictors P (features)

AU	AC	SC_c	SC_r
$L_c, \bar{x}_c, \nu, L_r, \tau$	$L_c, \bar{x}_c, \nu, L_r, \bar{x}_r, \tau$	$L_c, N_c, \bar{x}_c, \nu, \tau$	$L_r, N_r, \bar{x}_r, \tau$

data reduction for label-feature vectors

AU	AC	SC_c	SC_r
$AU > \epsilon_1$	$AC > \epsilon_0$	$SC_c > \epsilon_0$	$SC_r > \epsilon_0$
$P_{au} > \epsilon_0$	$P_{ac} > \epsilon_0$	$P_{sc,c} > \epsilon_0$	$P_{sc,r} > \epsilon_0$
$\tau < 0.85$	$\tau < 0.99$	$L_c > 10^{-4}$ g m^{-3}	$L_r > 10^{-4}$ g m^{-3}

total number of samples after data reduction (without validation data)

AU		AC		SC_c		SC_r	
train	test	train	test	train	test	train	test
179,133	114,312	316,141	185,956	365,705	220,119	309,151	181,247

Note. n is the number of ensembles using different random number seeds for the same initial condition.

Simple fully connected neural net with 3 layers

```
def build_model(ncol):
  model = keras.Sequential([
    layers.Dense(16, activation='tanh', input_shape=[ncol]),
    layers.Dense(16, activation='tanh'),
    layers.Dense(16, activation='tanh'),
    layers.Dense(1)
  ])

  optimizer = tf.keras.optimizers.RMSprop(0.001)

  model.compile(loss='mse',
                optimizer=optimizer,
                metrics=['mae', 'mse'])
```

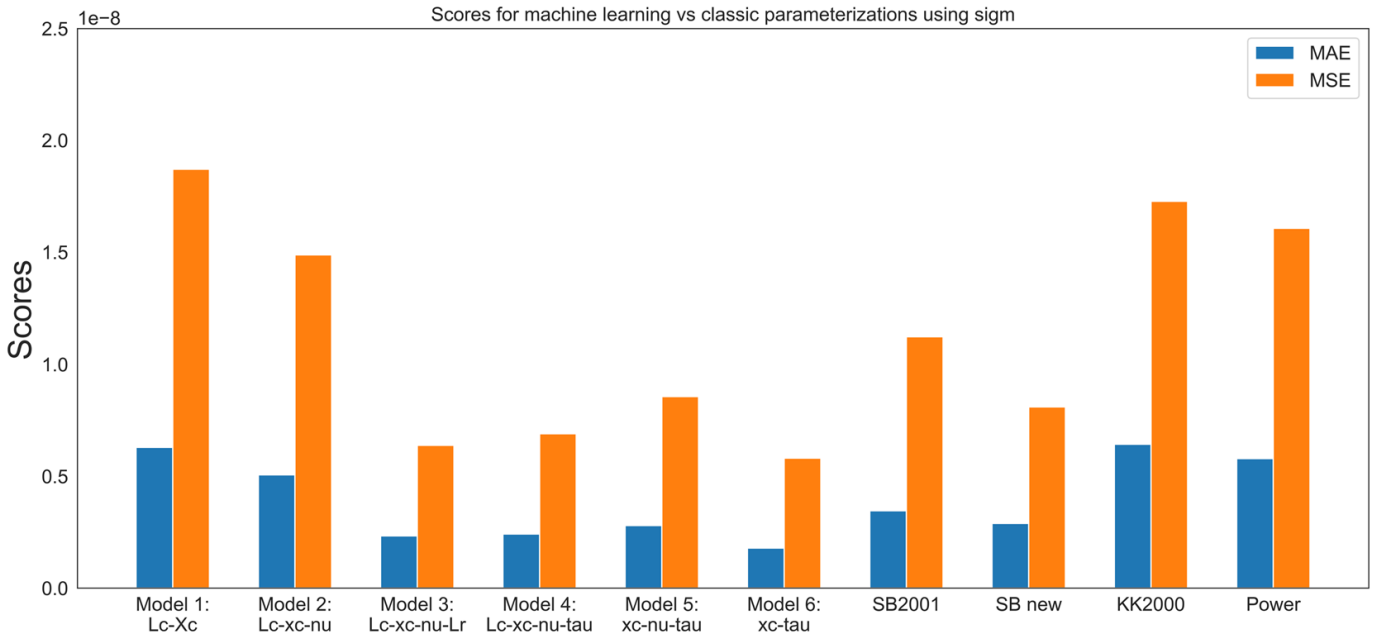
We can play with the size of the NN and test different activation functions like tanh, sigmoid or ReLU.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 16)                64
_________________________________________________________________
dense_1 (Dense)              (None, 16)                272
_________________________________________________________________
dense_2 (Dense)              (None, 16)                272
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
=================================================================
Total params: 625
Trainable params: 625
Non-trainable params: 0
_________________________________________________________________
```

The model has ~600 trainable parameters. Hence, overfitting should not be an issue.

Result of the machine learning step:

➔ Error measures against the testing data



➔ Machine learning seems be able to improve over bulk schemes like SB2001

➔ Including rain as predictor improves the autoconversion rate.

Partial dependency analysis for all 4 process rates

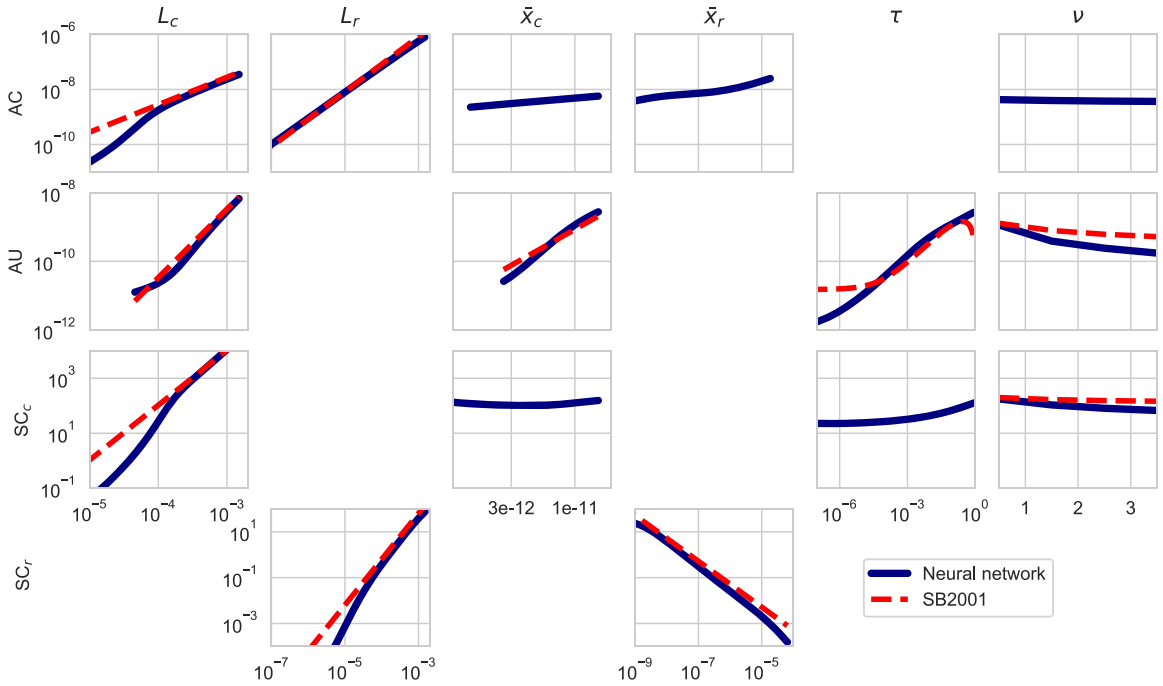


Figure 6. Partial dependence plots for the neural networks as described and selected in section 5. Autoconversion is ML Model 4 of Table 2. Red dashed lines denote predictions for SB2001. For *AC*, we used their Equation 21, for *AU* Equation 16, for SC_c Equation 14, and for SC_r Equation 19. For the variables that were not varied, the mean over the test set was used to create the SB2001 predictions.

- Hence, ML can indeed recover the dependencies of the SB2001 scheme.
- ML is not necessarily a black box, we can check what the scheme is doing.

The warm-rain parameterization needs to solve the ODE

$$\frac{dL_c}{dt} = -AU - AC,$$

$$\frac{dL_r}{dt} = +AU + AC,$$

$$\frac{dN_c}{dt} = -2AU_N - AC_N - SC_c = -\frac{2}{x_*}AU - \frac{1}{\bar{x}_c}AC - SC_c,$$

$$\frac{dN_r}{dt} = +AU_N + AC_N - SC_r = +\frac{1}{x_*}AU - SC_r,$$

➔ Will the ML models perform as well as SB2001 for the ODE solutions?

ODE solutions and super-droplet benchmark:

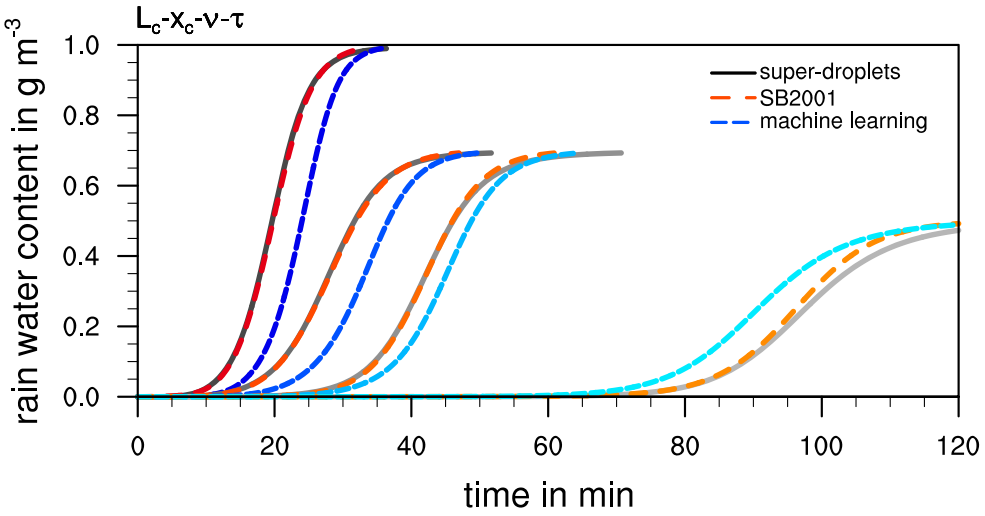


Figure 7. Time series of the rain water content for the solution of the KCE and the ODE solutions using SB2001 and ML Model 4 with autoconversion predictors $L_c, \bar{x}_c$, ν, and τ. Black and gray colors are the KCE solutions, red to orange colors for SB2001, and bluish colors the ML model. Shown are four difference initial conditions with (from left to right, different hue of colors) (1) $L_0 = 1$ g m^{-3}, $\bar{r}_0 = 14$ µm, $\nu = 0$; (2) $L_0 = 0.7$ g m^{-3}, $\bar{r}_0 = 14$ µm, $\nu = 0$; (3) $L_0 = 0.7$ g m^{-3}, $\bar{r}_0 = 11$ µm, $\nu = 0$; (4) $L_0 = 0.5$ g m^{-3}, $\bar{r}_0 = 11$ µm, $\nu = 2$.

➔ The ML-based model does okay, but the dependencies are not quite right.

The warm-rain ML-based microphysics

The ML approach provides a viable warm-rain parameterization, but does not perform as good as the Seifert and Beheng (2001) scheme.

The reasons for the deficiencies of the ML-based warm-rain scheme are discussed in the paper:

Seifert, A., & Rasp, S. (2020). Potential and limitations of machine learning for modeling warm-rain cloud microphysical processes. *Journal of Advances in Modeling Earth Systems*, 12, https://doi.org/10.1029/2020MS002301

Python scripts and the training data for the warm-rain scheme are publicly available from

https://gitlab.com/axelseifert/warmrain

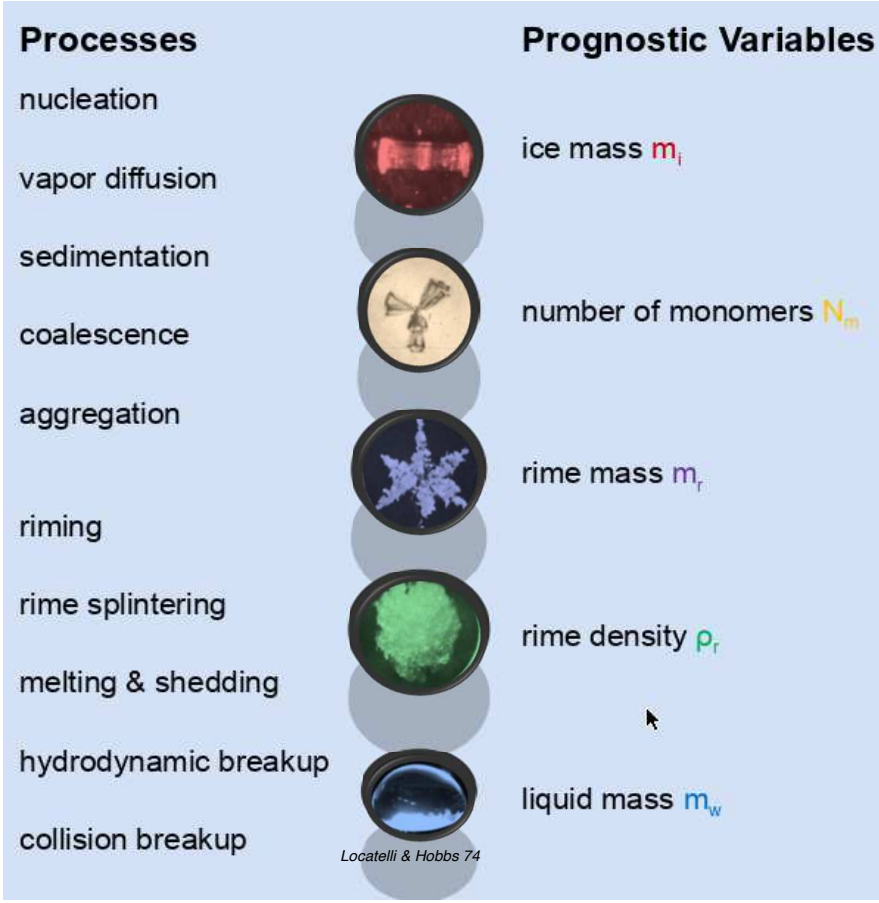
An ML-based P3-like multimodal extension of the two-moment bulk scheme in ICON

- **ML-based**: The new scheme is based on machine learning (ML) using neural nets or perceptrons and supervised-learning to approximate microphysical processes
- **P3-like**: Following the P3 scheme of Morrison and Milbrandt (2015) the scheme predicts particle properties like rime mass (or rime fraction) and rime density in addition to traditional bulk moments like mass and number density.
- **Multimodal**: In contrast to the original P3 scheme the ML-based scheme still uses several categories or modes.
- **Extension**: The ML-based parameterizations replace only the ice microphysical processes in the ICON two-moment scheme. The warm-rain parameterizations and other processes like ice nucleation remain unchanged.

ML for ice microphysics

- To generate the training data we use the Lagrangian particle model McSnow that explicitly resolves ice processes (Brdar and Seifert 2018)
- Each McSnow particle has severals variables that describe its current microphysical state.
- Needs at least 1000 particles per grid point, better 10000 to reduce Monte-Carlo noise.
- These are expensive simulation that are even today hardly feasible in 3d.

McSnow processes and variables



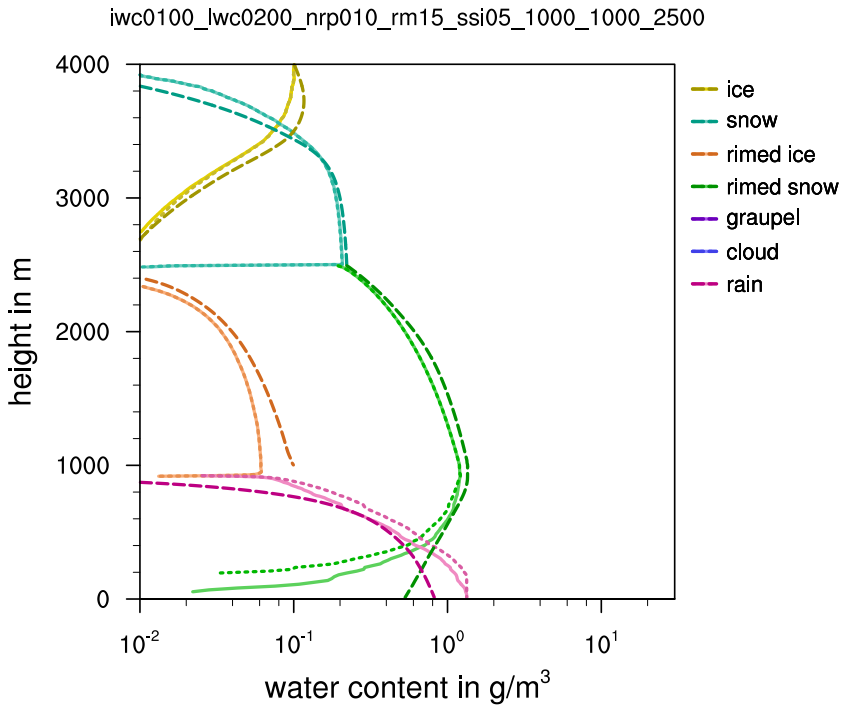
ML for ice microphysics

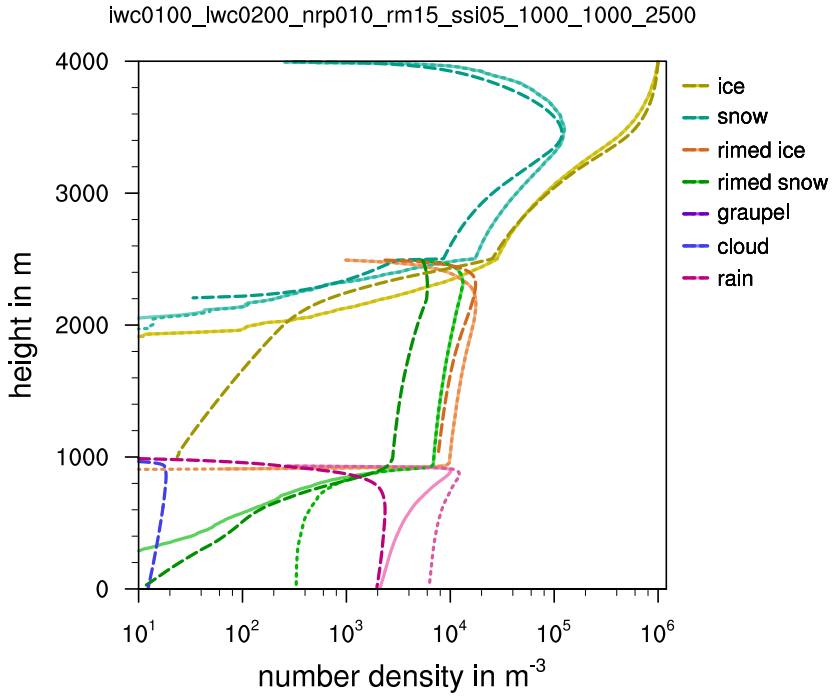
- We try to built an ODE system with 6 particle classes:
  - ice monomers, snow aggregates, rimed ice, rimed snow graupel and rain (and cloud droplets).
- All classes have mass and number densities, rimed classes (including graupel) have in addition rime mass, rime volume and liquid mass.
- Hence, for rimed particle classes with have rime fraction, rime density and melted fraction as bulk properties (P3-like scheme).
- This makes a total of 23 variables and more than 100 process rates.
- Can we „learn“ all those process rates from McSnow output and come up with an ODE system that works reasonably well?

Training data for bulk ice microphysics:

- We use an idealized box model falling through a prescribed atmosphere.
- The idealized box model allows many simulations. This is preferred here over a few 3d simulations.
- Training data is generated by Latin hypercube sampling of initial condition and atmospheric profile resulting more than 10.000 simulation.
- This can cover a large range of parameters.
- It proved to be necessary to include updraft parcels in the training data to better represent processes within the convective core.
- Maybe another advantage of the idealized box model approach is that it does not contain an imprint of the current climate, in contrast to more realistic simulations.

Parcel falling through an atmospheric profile





- height = f(time), i.e. parcel falling through a prescribed atmosphere.
- Mass and number densities for McSnow (solid), diagnosed ODE (dotted) and the ML-based ODE (dashed)

Some more details on ML approach

- Here we have used Tensorflow/Keras to train rather simple fully connected neural nets (perceptron).
- Features (input variables) and labels (output variables, process rates) are log-transformed, when appropriate, and standardized, i.e., normalized by mean and standard deviation.
- One small neural net with 16 nodes and 2 hidden layers for the regression model for each microphysical process rate.
- ReLU activation is used and different optimizers (SGD, Adam) are applied to find the best network parameters.
- For some processes a classifier network is used to decide whether the process is non-zero, before the regression neural net is applied (Gettelmann et al. 2020, JAMES).
- Parameters of neural nets are stored in NetCDF files.

Implementation in the ICON model

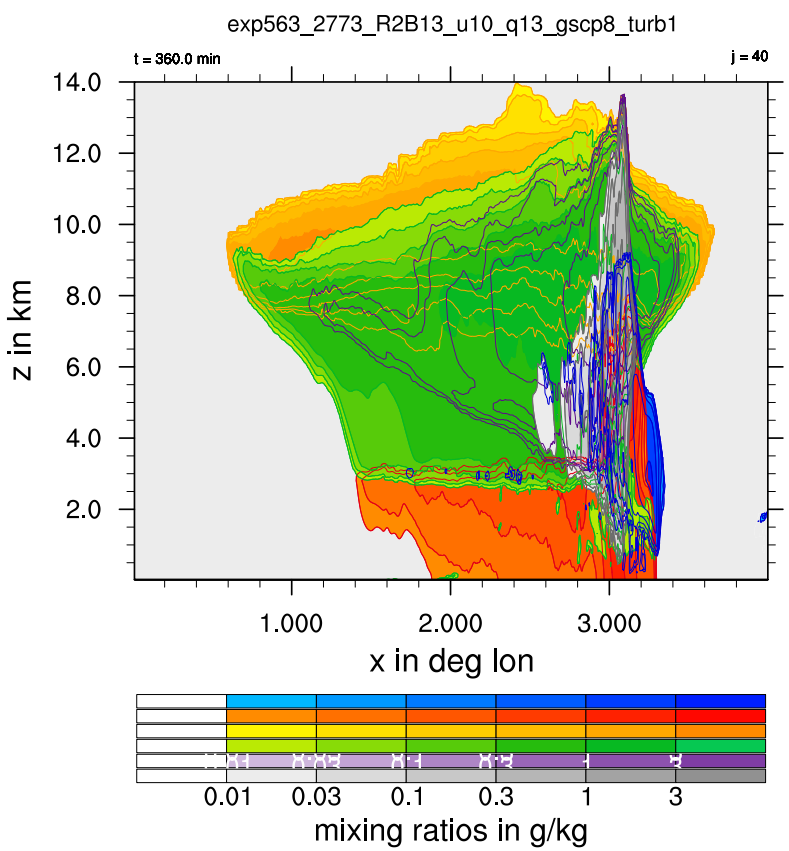
- Read neural network parameters from NetCDF and broadcast to all processors (only once during model initialization).
- Use a Fortran implementation of the evaluation (inference) of the neural net (based on Fornado of Leonhard Scheck and Fabian Jakub, LMU)

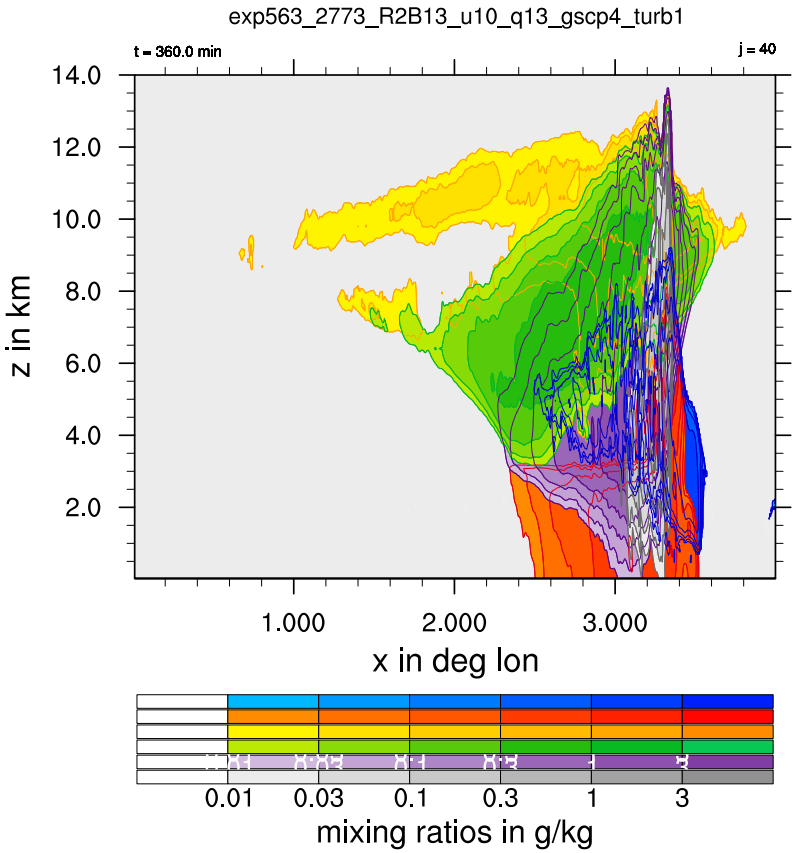
  https://gitlab.com/LeonhardScheck/fornado

- Vectorized on NEC Aurora using index lists.
- About 50 % of the time spent on the microphysics scheme is then the evaluation of the neural nets.
- Remaining time includes the preparation of the index lists, but also sedimentation, ice nucleation, warm-rain processes etc.
- The ML-based scheme with 23 variables is about twice as expensive as the standard two-moment scheme with 13 variables.

Simulation of an idealized squall line with ICON

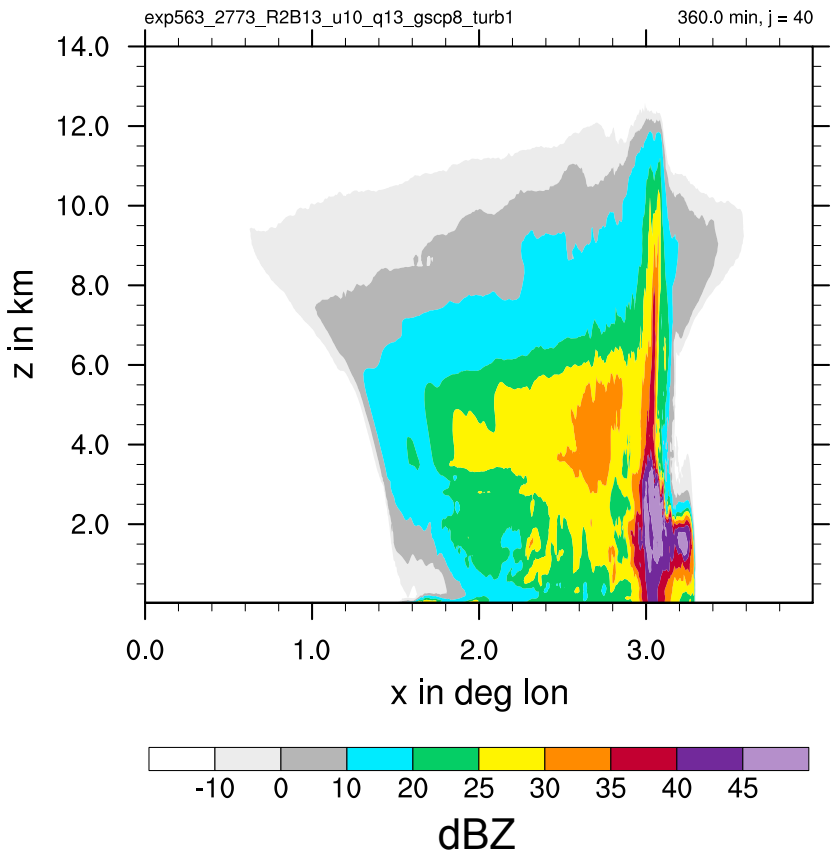
➔ Vertical cross section of hydrometeors: ML-based vs classic two-moment

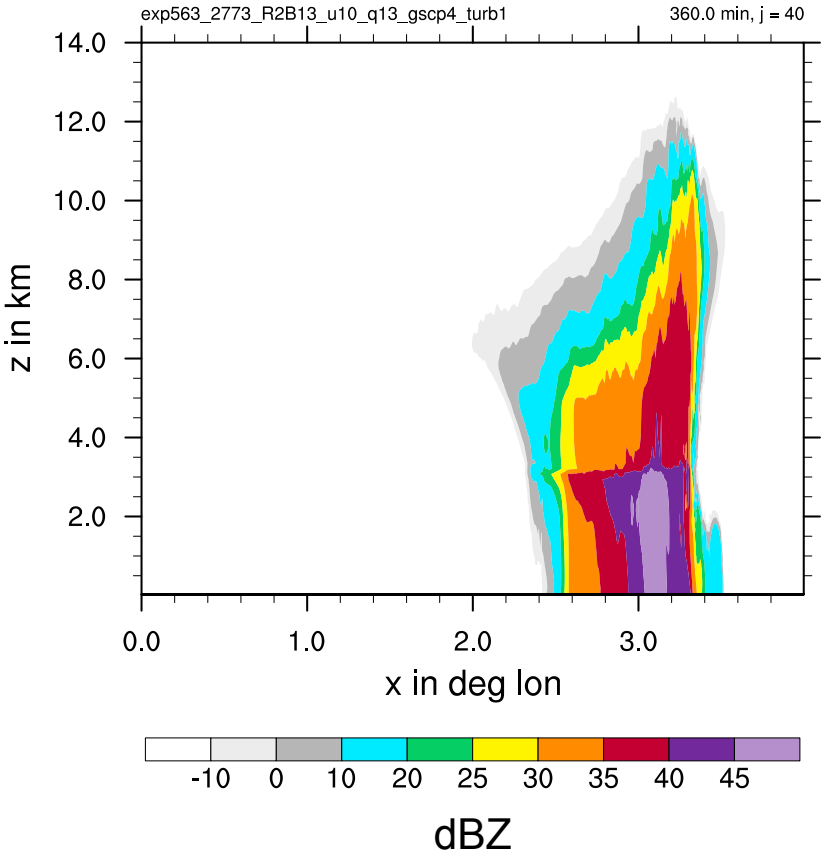




Simulation of an idealized squall line with ICON

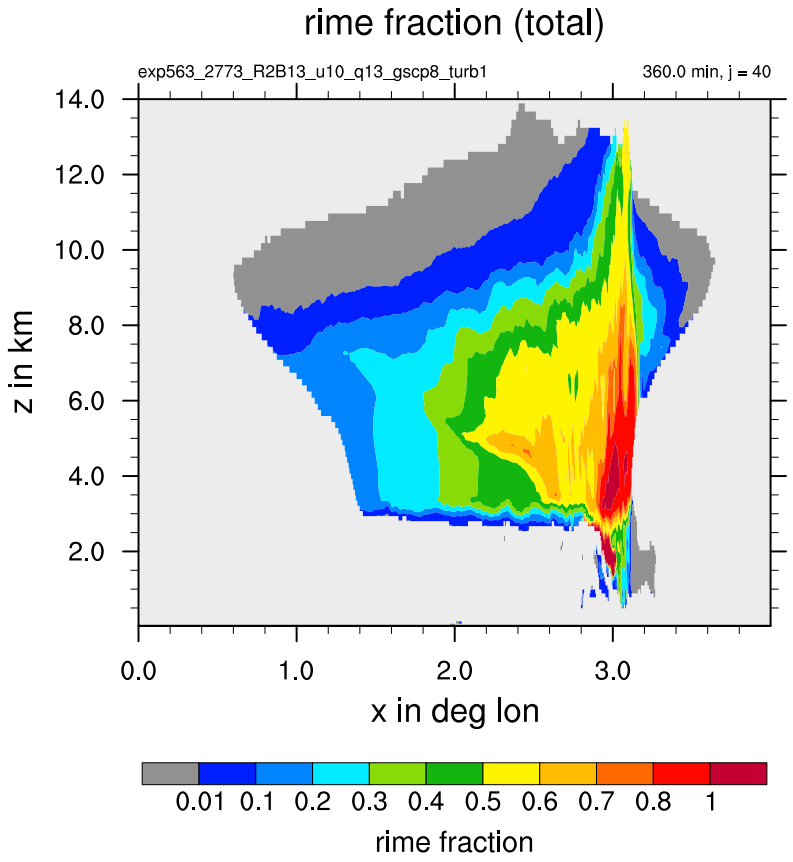
➔ Radar reflectivity (Rayleigh approximation)

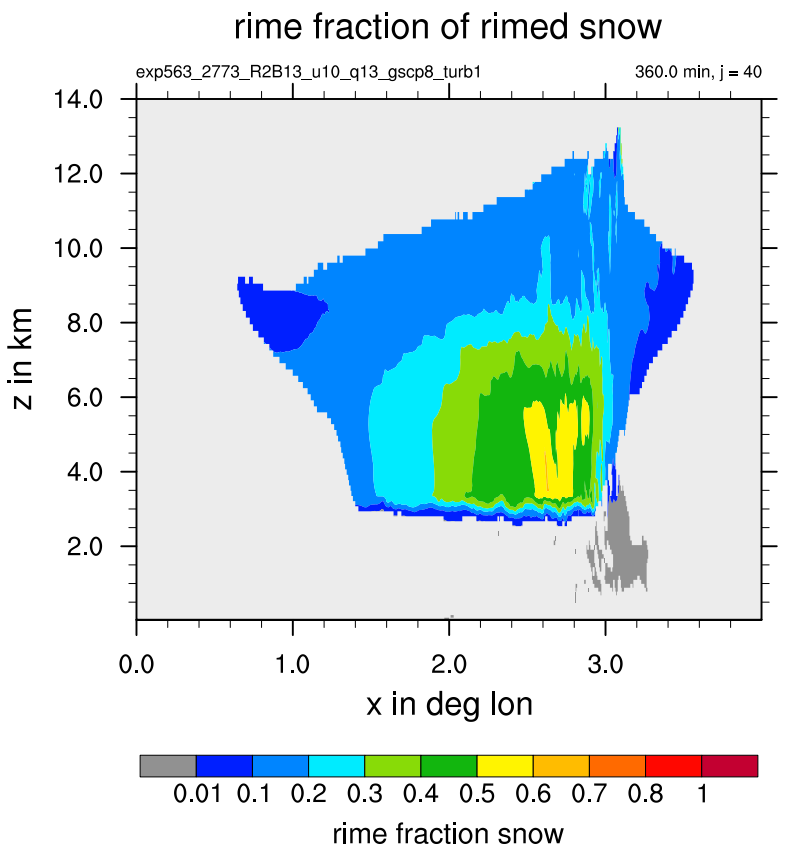




Simulation of an idealized squall line with ICON

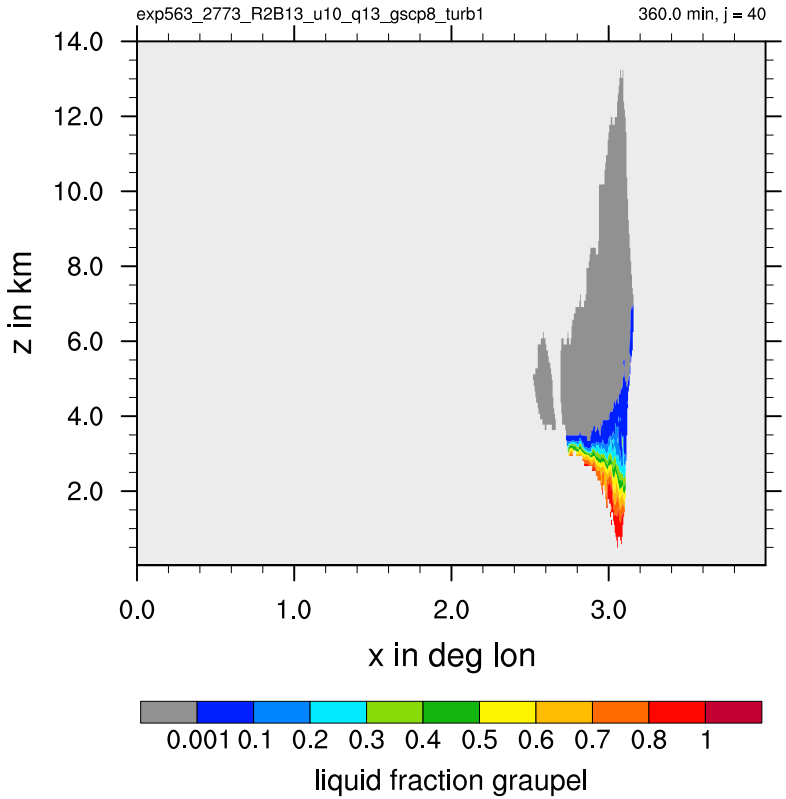
➔ Total rime fraction and rime fraction of „rimed snow“





Simulation of an idealized squall line with ICON

➔ liquid fraction of graupel



The scheme includes prognostic melting, but the wet growth regime is not yet well represented (too low liquid fraction in updraft).
This is probably a deficiency of the training data.

Lessons learned

- Machine learning provides an easy-to-use workflow to built regression models from training data. This can be used to develop bulk microphysical schemes.
- In my opinion, this is interesting also for „physics people“, because
  1. The most important step is to develop the benchmark model that is used to create the training data.
  2. The choice of the variables for the bulk model will determine it’s behavior.
  3. The setup of the simulations with the benchmark model is a crucial step and requires a good understanding of the relevant physics.
  4. Afterwards we should take the time and investigate what the ML model is doing, e.g., does it have the correct asymptotic behavior? How can we guarantee that?
- Overall, ML methods provide an alternative approach to develop physical parameterization with the promise to speed-up the development process.

Conclusions and Outlook

- Machine learning can indeed be used to built regression models of microphysical processes based on benchmark particle simulations, e.g., using super-droplets or the Lagrangian particle model McSnow.
- For warm-rain autoconversion a straightforward ML-based scheme is inferior to established parameterizations like Seifert and Beheng (2001), see Seifert and Rasp (2020) for details.
- The extension of the ICON two-moment microphysics scheme with a ML-based P3-like ice microphysics works well and produces a more pronounced stratiform region for the idealized squall line.
- Hence, the straightforward approach to train process rates with standard ML methods works reasonably well, but more advanced approaches should be explored in the future.