JAMES | Journal of Advances in Modeling Earth Systems

RESEARCH ARTICLE
10.1029/2020MS002301

Key Points:
- Machine learning is successfully applied to the warm-rain parameterization problem
- Training and testing data for the warm-rain kinetic collection equation are provided using the superdroplet method
- Standard training methods show some limitations for the resulting ODE system

Supporting Information:
- Supporting Information S1

Correspondence to:
A. Seifert,
axel.seifert@dwd.de

Citation:
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*, e2020MS002301. https://doi.org/10.1029/2020MS002301

Received 18 AUG 2020
Accepted 10 NOV 2020
Accepted article online 17 NOV 2020

Potential and Limitations of Machine Learning for Modeling Warm-Rain Cloud Microphysical Processes

Axel Seifert[1] and **Stephan Rasp**[2]

[1]Deutscher Wetterdienst, Offenbach, Germany, [2]TU München, Munich, Germany

Abstract The use of machine learning based on neural networks for cloud microphysical parameterizations is investigated. As an example, we use the warm-rain formation by collision-coalescence, that is, the parameterization of autoconversion, accretion, and self-collection of droplets in a two-moment framework. Benchmark solutions of the kinetic collection equations are performed using a Monte Carlo superdroplet algorithm. The superdroplet method provides reliable but noisy estimates of the warm-rain process rates. For each process rate, a neural network is trained using standard machine learning techniques. The resulting models make skillful predictions for the process rates when compared to the testing data. However, when solving the ordinary differential equations, the solutions are not as good as those of an established warm-rain parameterization. This deficiency can be seen as a limitation of the machine learning methods that are applied, but at the same time, it points toward a fundamental ill-posedness of the commonly used two-moment warm-rain schemes. More advanced machine learning methods that include a notion of time derivatives, therefore, have the potential to overcome these problems.

Plain Language Summary In our work, we are trying to teach a computer how rain forms in clouds. We show that computer hundreds of cases in the form of data. To be honest, the data are not real data but only results of simulations with a more complicated computer model. This complicated model can track the collisions of 10,000 of droplets, and we save all that data about the growth of the droplets into larger raindrops. This is what we then give to the simpler computer model to teach it something about clouds and rain. Afterward, it can make pretty good predictions about which clouds will rain and how long it will take them to produce the first rain. Unfortunately, the current machine learning methods are still a bit stupid because they only learn from the data but do not understand the mathematics and the physics behind the data. Therefore, the new computer model is still not as good at predicting rain as some clever mathematical formulas that were developed 20 years ago. Maybe we first have to teach the computer model more about calculus before it can learn to predict rain.

1. Introduction

During the last decade, machine learning (ML) has become an essential tool in data science, engineering, medical research, and many other applied sciences. The success of ML techniques can be explained by its general ability to extract (learn) complex nonlinear relationships from data. Hence, ML methods like random forests or deep neural networks (NNs) provide very powerful techniques for many kinds of classification and nonlinear regression problems. Besides, these methods have been greatly popularized by the availability of easy-to-use Python libraries like Tensorflow and Keras (Abadi et al., 2015; Chollet et al., 2015; Géron, 2019).

In atmospheric science ML methods are very appealing for nowcasting applications (Sønderby et al., 2020; Xingjian et al., 2015), remote-sensing retrievals (Lary et al., 2016; Maxwell et al., 2018), and many other applications that require a postprocessing of numerical weather prediction (NWP) or climate model output. This area of research is growing rapidly, and for a general overview we refer to recent reviews of ML techniques and their application in climate science (Huntingford et al., 2019; Reichstein et al., 2019) and fluid mechanics (Brunton et al., 2020).

The usefulness of these tools for the development of atmospheric models is much less clear. Subgrid parameterizations, which are crucial to describe unresolved processes in NWP and climate models, are functional relationships between resolved model variables and those unresolved quantities. This problem might be accessible to ML methods given that appropriate training data are available either from observations or from highly resolved benchmark simulations. Some efforts have been made to use ML to replace convection parameterizations in general circulations models (O'Gorman & Dwyer, 2018; Rasp et al., 2018) or even to describe all subgrid processes in the atmosphere (Brenowitz & Bretherton, 2018, 2019; Yuval & O'Gorman, 2020). The results are promising, but not without challenges, which are especially related to the generalization of the ML predictions outside of the chosen training data. Further improvements can probably be expected if conservation principles and/or invariance and similarity laws can be built into ML methods (Beucler et al., 2020; Ling, Jones, et al., 2016; Ling, Kurzawski, et al., 2016). Advanced regularization methods might be another important ingredient to stabilize the behavior of complex ML-based parameterizations (Brenowitz et al., 2020).

When we discuss ML methods for the parameterization problem, it can be helpful to distinguish two different approaches. First, the use of ML to emulate an existing parameterization to achieve an increase in computational performance, that is, a speedup (Chevallier et al., 2000), and second, to derive new parameterizations from original data when the parameterizability of the problem is not necessarily clear. There is no sharp line between the two approaches, though, because the emulation of expensive highly resolved models shares many problems with the general parameterization problem.

In the following, we apply regression using NNs to the parameterization of warm-rain cloud microphysics. The formulation of the corresponding autoconversion and accretion rates is a classic problem in cloud modeling going back to first parameterizations of Kessler (1969) and Berry and Reinhardt (1974). This problem seems especially well suited for ML methods because most warm-rain parameterizations used today are based on data from numerical solutions of the kinetic collection equation (KCE). Hence, this looks like a well-defined regression problem.

The paper is organized as follows. In section 2 we introduce the KCE and define the bulk microphysical conversion rates. In section 3 we introduce the superdroplet method as a benchmark solver and discuss the training data in section 4. In section 5 ML is applied to the parameterization problem. In section 6 we attempt to interpret the predictions from the ML method. In section 7 we solve the corresponding ordinary differential equations (ODEs) and compare the ML results with an established warm-rain parameterization. Sensitivities to hyperparameters and other assumptions are discussed in section 8, followed by our conclusions in section 9.

2. The Kinetic Collection Equation

The collision-coalescence of droplets in a cloud is described by the KCE:

$$\frac{df(x)}{dt} = \frac{1}{2}\int_0^x f(x-y)f(y)K(x-y,y)\mathrm{d}y - \int_0^\infty f(x)f(y)K(x,y)\mathrm{d}y, \tag{1}$$

also known as Smoluchowski equation or quasi-stochastic collection equation (Drake, 1972; von Smoluchowski, 1916, 1917). Here $f(x)$ is the number of droplets per unit volume in the mass range $[x, x+\mathrm{d}x]$, and x and y are the drop mass. The collision-coalescence kernel $K(x,y)$ describes the physics of the collision process. For clouds it is specified as

$$K(x,y) = \pi\,[r(x)+r(y)]^2\,|v(x)-v(y)|\,E_{coll}(x,y), \tag{2}$$

where $r(x)$ is the drop radius, $v(x)$ is the terminal fall velocity, and $E_{coll}(x,y)$ is the collision efficiency. The first two terms constitute the geometric swept volume. The collision efficiency quantifies the details of the fluid dynamics interaction and is tabulated from detailed trajectory calculations. In the following, we use the collision efficiency of Hall (1980) with modifications by Beheng (1994).

For small droplets the hydrodynamic collision kernel is nonlinear with $K \sim x^2$, and analytical approaches to solve the KCE are of limited use in this case. The solutions describe the colloidal instability, which is the formation of larger and larger drops due to binary collisions. The kinetic equation is invariant under the transformation

$$f \rightarrow cf, \qquad t \rightarrow t/c \tag{3}$$

with a constant c and therefore similarity solutions of the form $\tilde{f}(x,t) = cf(x,ct)$ exist for each solution $f(x,t)$.

Atmospheric models do not necessarily predict $f(x,t)$; instead, only partial moments of this distribution in two size ranges are used, which are defined as

$$M_c^{(k)} = \int_0^{x_*} x^k f(x)\mathrm{d}x, \tag{4}$$

$$M_r^{(k)} = \int_{x_*}^{\infty} x^k f(x)\mathrm{d}x, \tag{5}$$

and drops smaller than the mass $x_* = 2.6 \times 10^{-10}$ kg are cloud droplets, and larger drops are raindrops. $M_i^{(0)} = N_i$ is the number density, and $M_i^{(1)} = L_i$ the mass density or cloud resp. rain water content. Depending on the number of moments that are used for each particle category, parameterizations are classified as single-, double-, or triple-moment schemes. In the following, we focus on double-moment schemes with the variables N_c, L_c, N_r, and L_r. It is straightforward to prove that the liquid water content $L = L_c + L_r$ is conserved by Equation 1 and the scaling constant c of similarity solutions can be identified as $c = \tilde{L}/L$ (Drake, 1972).

The time evolution described by Equation 1 establishes a system of ODEs for the partial moments N_c, L_c, N_r, and L_r given by

$$\frac{dL_c}{dt} = -AU - AC, \tag{6}$$

$$\frac{dL_r}{dt} = +AU + AC, \tag{7}$$

$$\frac{dN_c}{dt} = -2AU_N - AC_N - SC_c = -\frac{2}{x_*}AU - \frac{1}{\bar{x}_c}AC - SC_c, \tag{8}$$

$$\frac{dN_r}{dt} = +AU_N + AC_N - SC_r = +\frac{1}{x_*}AU - SC_r, \tag{9}$$

with the mean cloud droplet mass $\bar{x}_c = L_c/N_c$. The autoconversion rate AU, the accretion rate AC, and the two self-collection rates SC_c and SC_r are unknown, and to specify these process rates in terms of the partial moments is known as the warm-rain parameterization problem. Note that we have already made the approximation to couple the number rates of autoconversion and accretion, AU_N and AC_N, to the mass rates (Beheng, 1994, 2010) by the assumption that autoconversion creates droplets of the mass x^* and accretion collects on average the cloud droplets with mass $\bar{x}_c$. All moments and the process rates are positive semidefinite quantities. The time evolution of L_c is monotonically decreasing, because it has only sink terms. Correspondingly, L_r increases monotonically.

The autoconversion rate AU and the accretion rate AC can be calculated directly from known solutions of the kinetic equation by

$$AU = \int_{x'=0}^{x_*} \int_{x''=x_*-x'}^{x_*} f(x')f(x'')K(x',x'')x'\mathrm{d}x'\mathrm{d}x'', \tag{10}$$

$$AC = \int_{x'=0}^{x_*} \int_{x''=x_*}^{\infty} f(x')f(x'')K(x',x'')x'\mathrm{d}x'\mathrm{d}x'', \tag{11}$$

and similar integral forms exist for SC_c and SC_r (Beheng, 2010; Doms & Beheng, 1986).

Having such data reduces the parameterization problem to a regression task and depending on the input data and regression assumptions different parameterizations have been derived in the last decades (Beheng, 1994; Berry & Reinhardt, 1974; Khairoutdinov & Kogan, 2000). In the following, we will compare with the double-moment parameterization of Seifert and Beheng (2001; SB2001 hereafter) given by

$$AU_{sb} = \frac{k_c}{20x_*}\frac{(\nu+2)(\nu+4)}{(\nu+1)^2}L_c^2\bar{x}_c^2\left[1 + \frac{\Phi_{au}(\tau)}{(1-\tau)^2}\right], \tag{12}$$

$$AC_{sb} = k_r L_c L_r \Phi_{ac}(\tau), \tag{13}$$

where $k_c = 9.44 \times 10^9$ m^3 kg^{-2} s^{-1} and $k_r = 5.78$ m^3 kg^{-1} s^{-1} are coefficients of the Long-kernel (Long, 1974) and ν is the shape parameter of an assumed Gamma distribution. The functions $\Phi_{au}(\tau)$ and $\Phi_{ac}(\tau)$ are given by

$$\Phi_{au} = 600\tau^{0.68}(1 - \tau^{0.68})^3, \tag{14}$$

$$\Phi_{ac} = \left(\frac{\tau}{\tau + 5 \times 10^{-4}}\right)^4 \tag{15}$$

and depend only on the liquid water time scale $\tau = L_r/(L_c + L_r) = L_r/L$, which is motivated by the above-mentioned invariance of the kinetic equation.

3. The Benchmark Simulations

For the benchmark solutions of the KCE, we apply the superdroplet method of Shima et al. (2009), which is a Monte Carlo algorithm that simulates the collision-coalescence processes explicitly. The Monte Carlo algorithm actually implements a more general equation which takes into account stochastic fluctuations (Dziekan & Pawlowska, 2017). The KCE (1) is in fact only the mean-field approximation of this more general stochastic collection equation (Alfonso et al., 2008).

As initial condition, we use a Gamma distribution for cloud droplets with

$$f(x) = Ax^\nu e^{-Bx}, \tag{16}$$

where A and B are calculated from the initial mean radius $\bar{r}_0$ and the initial liquid water content $L_0 = L_{c,0}$ and ν is the shape parameter. In the following, we will make use of simulation from the three-dimensional phase space $L_0 \in [0.3, 2]$ g m^{-3}, $\bar{r}_0 \in [9,15]$ μm, and $\nu \in [0, 4]$. The Monte Carlo algorithm draws samples from the initial conditions that represent a certain number of real droplets given by the (average) multiplicity $\xi_0 =$ 25,600 in a control volume of 25 m^3. The initialization follows the single-SIP approach of Unterstrasser et al. (2017), which improves the convergence properties of the Monte Carlo method. Typically more than 10,000 superdroplets are used for a single simulation, which ensures a converged solution. The actual number of superdroplets scales with the number density of the initial conditions. If the number of superdroplets would exceed 50,000, it is limited to that value. Due to the stochastic nature of the Monte Carlo algorithm, which also causes additional variance, it can be useful to do simulations with the same initial condition multiple times. This will be denoted as a simulation ensemble.

The autoconversion and accretion rates could be calculated from the integral representation Equations 10 and 11, but the Monte Carlo algorithm allows a direct calculation of those conversion rates. For each superdroplet collision, it is known to which drop category the drops belong to, and hence, the autoconversion rate can be exactly calculated from

$$AU = \sum_1^N x'_k \xi'_k = \sum_1^N (\tilde{\gamma}_\alpha x_j + x_k)\xi'_k, \qquad j,k \in c \wedge k' \in r, \tag{17}$$

where k is the superdroplet with the smaller multiplicity and j is the superdroplet with the larger multiplicity. The prime denotes quantities after the collision event. Hence, $k \in c \wedge k' \in r$ means that the superdroplet with the smaller multiplicity became a raindrop after the collision event. The mass x'_k is the mass of the superdroplet k after the collision event with $x'_k > x_*$. The portion of the superdroplet j that is added to k is given by the probability $\tilde{\gamma}_\alpha$ that is defined and calculated as specified in section 5.1.3 of Shima et al. (2009). Hence, autoconversion is simply the sum over x'_k for all superdroplets that became raindrops weighted by their multiplicity ξ'_k.

For accretion we have to distinguish two cases and $AC = AC_1 + AC_2$. In the first case the superdroplet k is a raindrop, and j is the cloud droplet. The raindrop k grows by collecting the fraction $\tilde{\gamma}_\alpha$ of j:

$$AC_1 = \sum_1^N x_j \xi_k \tilde{\gamma}_\alpha, \qquad j \in c \wedge k \in r. \tag{18}$$

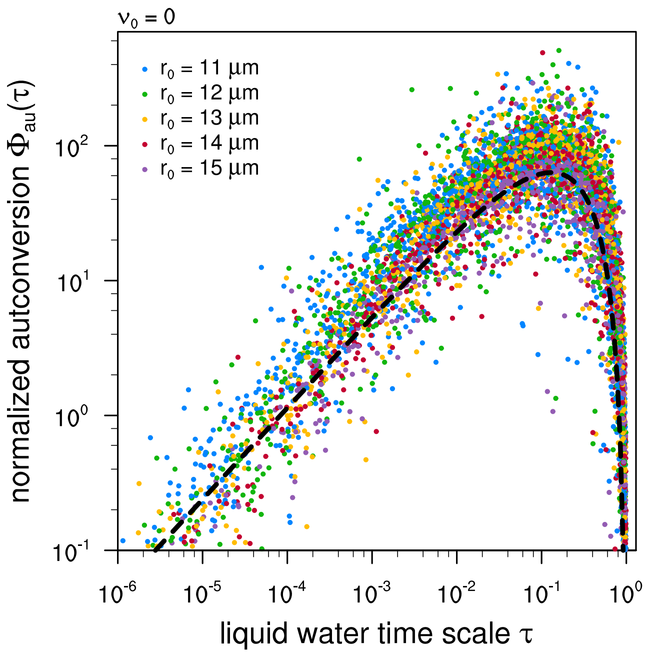


Figure 1. Normalized autoconversion rate $\Phi(\tau)$ as a function of the liquid-water time scale $\tau = L_r/(L_c + L_r)$ for a subset of the training data with $\nu_0 = 0$.

In the second case the superdroplet j is the raindrop, and k is the cloud droplet. Hence, the mass transfer to rain due to accretion is only the mass of k before the collision event:

$$AC_2 = \sum_1^N x_k \xi_k', \qquad j \in r \wedge k \in c. \tag{19}$$

The self-collection rates are simply

$$SC_c = \sum_1^N \xi_k \tilde{\gamma}_\alpha, \qquad j, k \in c, \tag{20}$$

$$SC_c = \sum_1^N \xi_k \tilde{\gamma}_\alpha, \qquad j, k \in r, \tag{21}$$

and the sum is taken over the list of all superdroplet pairs for this time step. In passing we note that the number rates AU_N and AC_N can be calculated simply by dropping the mass weights in the equations above.

The simulations are performed with a time step of 1 s, and output is written every 20 s. Each individual simulations of the collision-coalescence process is long enough such that all cloud water is eventually converted to rain. The length of the individual simulations depends on L_0 and $\bar{r}_0$. For example, for $L_0 = 1$ g m^{-3} and $\bar{r}_0 = 13$ μm, it is 60 min. For different L_0 the simulation time scales with $1/L_0$.

Figure 1 shows the normalized autoconversion rate in terms of Φ_{au}; that is, it reproduces Figure 1 from SB2001 for a subset of our simulation output. This confirms that the superdroplet simulations and the derived autoconversion rate are comparable to SB2001. Although the superdroplet method should have a higher accuracy given that we are using a very large number of particles, it is also obvious that the Monte Carlo method exhibits significant noise. This becomes even more severe for larger ν, that is, narrower initial cloud droplet distributions. Some of these fluctuations might be correct physical behavior, though, and especially for larger ν, the droplet growth can be dominated by rare events ("lucky droplets"). Nevertheless, it is encouraging that the old fit of SB2001 (black dashed line) is a good approximation to the new data, although some bias is visible.

4. Preparation of Training and Testing Data

As predictors, we use the moments N_c, L_c, N_r, and L_r, which would be the prognostic variables in a double-moment warm-rain scheme. In addition, we assume that the shape parameter ν is known, for example, from the properties of a given aerosol distribution. Hence, these five variables are the basic features or predictors in the training data. The process rates AU, AC, SC_c, and SC_r are the labels or target variables.

The solutions of the KCE exhibit exponential growth, and therefore, the moments and the process rates span several orders in magnitude. Hence, it is advisable to use log-transformed features and labels for training. This can also be motivated from the fact that established regression-based parameterization like Beheng (1994) or Khairoutdinov and Kogan (2000) uses a multiplicative power law ansatz. The log-transform requires that we specify a minimum value ϵ and feature-label vectors with values smaller than ϵ are removed from the training and testing data. We use $\epsilon_0 = 10^{-15}$ for all data except the autoconversion rate for which a larger value of $\epsilon_1 = 10^{-12}$ is applied (Table 1). The reasoning for this choice is discussed in section 7.

Number and mass densities are linearly dependent in the sense that they increase by a factor c when the size distribution $f(x)$ is multiplied by the same constant factor. This can be avoided when $\bar{x}_i = L_i/N_i$ is used instead of N_i. The alternative choice N_i and x_i would be truly orthogonal. Here we prefer the combination L_i and $\bar{x}_i$ for the mass rates because L_c and L_r are the most important predictors for AU and AC. Finally, we may replace L_r with τ for predicting autoconversion to make the set of predictors identical to SB2001.

In the following, we train a neural net for each label separately. This allows us to test different sets of predictors for each process rate. Another consideration is that autoconversion is most important when L_r is still small; later on the system is dominated by accretion. Hence, we do not want to contaminate the training

Table 1
Overview of the 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.

data with rather irrelevant autoconversion labels, for example, when $L_r \gg L_c$. Even accretion becomes irrelevant when L_c is small. Hence, we can remove labels based on τ, L_c, and L_r to focus the training on the relevant part of the phase space.

As testing data, we use a separate set of simulations with different values of L_0 and ν. This makes sure that the testing data are clearly separated from the training data. The selection of the training and testing data is summarized in Table 1. The testing data span a marginally smaller range of initial conditions as the training data. This means the ML models will, in some sense, only need to interpolate to achieve a good skill on the testing data. This is sufficient here because cases that have lower L_0 and/or smaller $\bar{r}_0$ are nonprecipitating and will not produce rain by the warm-rain collision-coalescence process within reasonable time. Cases with higher L_0 and larger $\bar{r}_0$ rain so quickly that the error in the timing becomes irrelevant. The shape parameter ν is not a prognostic variable in a two-moment scheme and is set a priori and is usually within the range shown here.

Finally, the training, validation, and testing data are standardized using the mean and standard deviation to ensure that all features zero mean and a standard deviation of one. Hence, we apply the transformation $\check{\chi} = (\chi - \bar{\chi})/\sigma_\chi$, where χ is a feature vector with mean $\bar{\chi}$ and standard deviation σ_χ and $\check{\chi}$ is the standardized value of the feature.

A subset of the training data before standardization is visualized by a pairs plot in Figure 2 showing AU and the predictors L_c, $\bar{x}_c$, τ, and ν. This shows clearly that we have defined $\nu = \nu_0$ as the shape parameter of the initial conditions. Note that discrete values of, for example, ν and $\bar{x}_c$, are artificially smeared out in this plot by the kernel density estimator, which assumes a Gaussian distribution. It can also be seen that $\bar{x}_c$ does not change much, especially not for large ν. This is explained by the fact that initially, only the tail of the distribution is affected by collisional growth, but not the mean, and later accretion does collect all cloud droplets. Only for broad distributions $\bar{x}_c$ decreases during the final collection stage, because very small cloud droplets are not collected efficiently by the raindrops (see, e.g., Figure S40).

5. The ML Models

To parameterize the process rates with ML, we train several small NNs. Each has three fully connected layers with 16 nodes per layer. Each neural net has only a single output node corresponding to our choice to train a

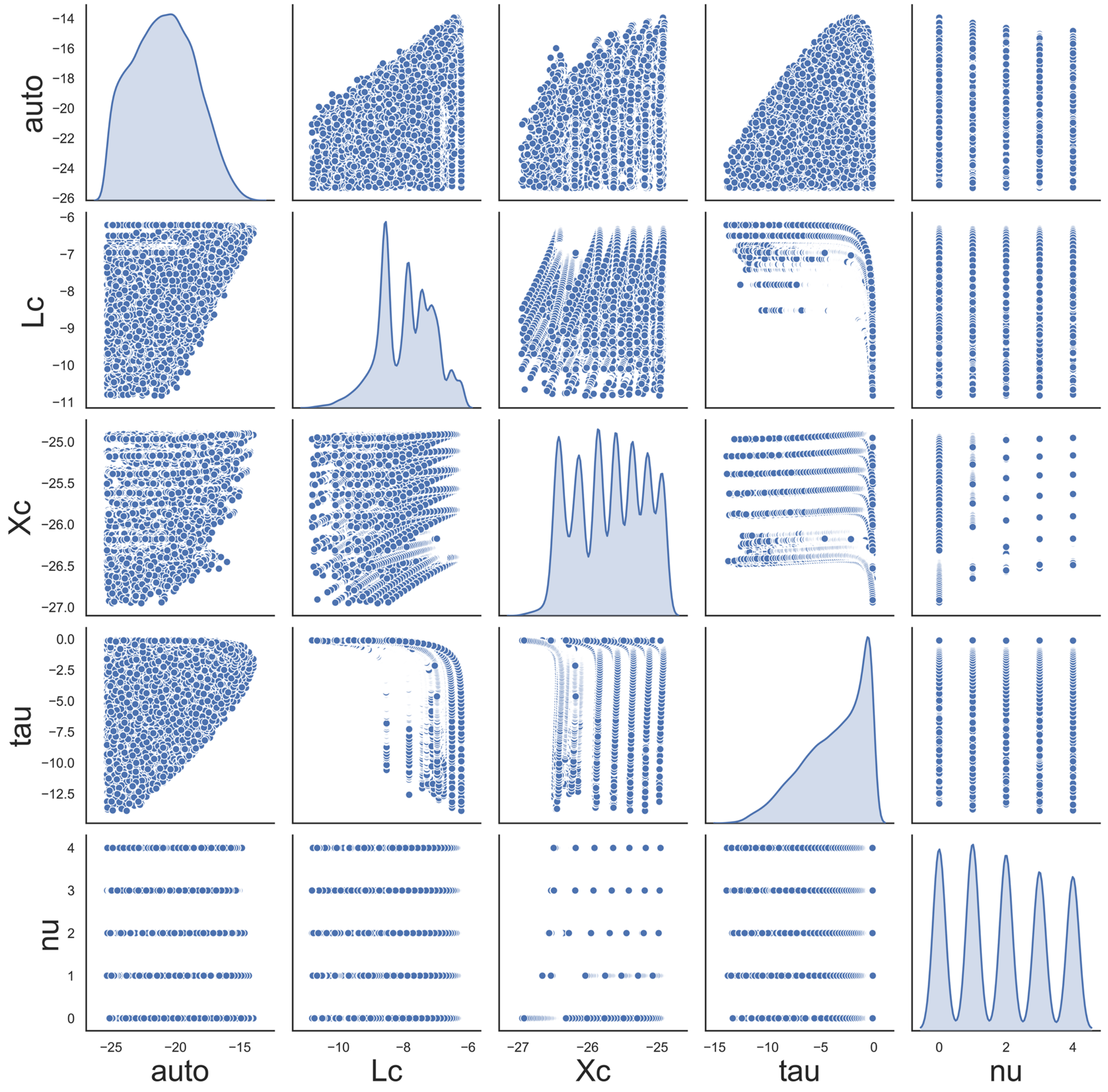


Figure 2. Pairs plot for log-transformed training data showing the autoconversion rate, the cloud water content, the mean cloud droplet mass, the time scale τ, and the shape parameter ν. The plots on the diagonal show a kernel density estimate.

separate NN for each individual process rate. This gives a total of 609 trainable parameters for each NN. The sigmoid activation function is applied. Larger nets and other activation functions have been tested, and those results are documented in the supporting information. However, they did not lead to any improvements. We choose the mean squared error (MSE) as loss or cost function and RMSprop as the optimizer. RMSprop is a variant of gradient descent that automatically adjusts the gradient magnitude during the minimization. The initial learning rate is set to 10^{-3}. We use early stopping with a patience of 20 epochs and a maximum number of 200 training epochs.

In this first ML step, we primarily test different predictors for the process rates. For autoconversion, our first models follow classic formulations like Beheng (1994) or Khairoutdinov and Kogan (2000) and use only the cloud variables L_c and $\bar{x}_c$ (Model 1). As a second model, we can, in addition, assume that the (initial) shape parameter ν is known as well (Model 2). Our Models 3 and 4 make use of the rainwater content L_r as an

Table 2
Overview of Analyzed and Tested ML Models for Autoconversion (and SB2001 for Comparison)

No.	Predictors (features)	Trainable param.	Testing data MAE	Testing data MSE	ODE solutions, t_{10} errors MAE	MSE	ME	MRE
SB	$L_c, \bar{x}_c, \nu, \tau$	(6)	3.46	11.23	1.43	2.13	−0.91	−2.70
1	$L_c, \bar{x}_c$	609	6.29	18.71	—	—	—	—
2	$L_c, \bar{x}_c, \nu$	609	5.06	14.88	15.41	22.19	−15.26	−30.59
3	$L_c, \bar{x}_c, \nu, L_r$	609	2.33	6.38	5.76	8.18	0.92	10.31
4	$L_c, \bar{x}_c, \nu, \tau$	609	2.41	6.90	6.40	8.79	1.90	13.14
5	$\bar{x}_c, \nu, \tau$	609	2.79	8.56	6.09	8.47	0.81	8.59
6	$\bar{x}_c, \tau$	609	1.79	5.81	6.74	8.71	2.39	13.86

Note. MAE and MSE of AU for testing data are given in 10^{-9} m^3 kg^{-2} s^{-1}. MAE, MSE, and ME of t_{10} in min, and MRE of t_{10} in %. All ML models use sigmoid activation and $\epsilon_0 = 10^{-15}$ for accretion and both self-collection rates.

additional predictor of AU, either explicitly (Model 3) or by using τ (Model 4). The latter two models are ML counterparts of SB2001 in the sense that they have the same information provided as predictors.

Two additional ML models using only $\bar{x}_c - \nu - \tau$ and $\bar{x}_c - \tau$ as set of predictors are an attempt to improve the ML models by borrowing the dependencies from SB2001. Hence, for Model 5 with $\bar{x}_c - \nu - \tau$ we have eliminated the L_c dependency from AU by dividing the training data (labels) by L_c^2. For model 6 with predictors $\bar{x}_c - \tau$ we have, in addition, removed the ν dependency with the corresponding term of Equation 12. See also Table 2 where all six models are listed.

All six models show a good convergence. The MSE and mean absolute error (MAE) of the validation data follow the training loss perfectly as shown for the first four models in Figure 3. Hence, there is little sign of overfitting. Overfitting is also not to be expected, because we have much more training data than trainable parameters. Interestingly, the lines of Models 3 and 4 can hardly be distinguished, which already indicates that it is rather irrelevant whether we use L_r or τ. At the same time, it is obvious that having more predictors provides a better model and, especially, the jump from Models 2 to 3 and 4 is large. Models 5 and 6 use different training and validation data and cannot be directly compared with Models 1–4 in this kind of diagnostic.

More important than training or validation loss is the comparison with the testing data, which has not been used to optimize the model parameters. This is shown in Figure 4 and in Table 2. Also for the testing data, we see that Models 3 and 4 are clearly superior to Models 1 and 2. We compare with four classic, in the sense of non-ML, parameterizations. First, SB2001 with its original parameters. ML Models 3 and 4, which use the same set of predictors, are significantly better than SB2001, whereas SB2001 is better than ML Models 1 and 2. For "SB new" we have recalibrated Φ_{au} on the training data using a Levenberg-Marquardt

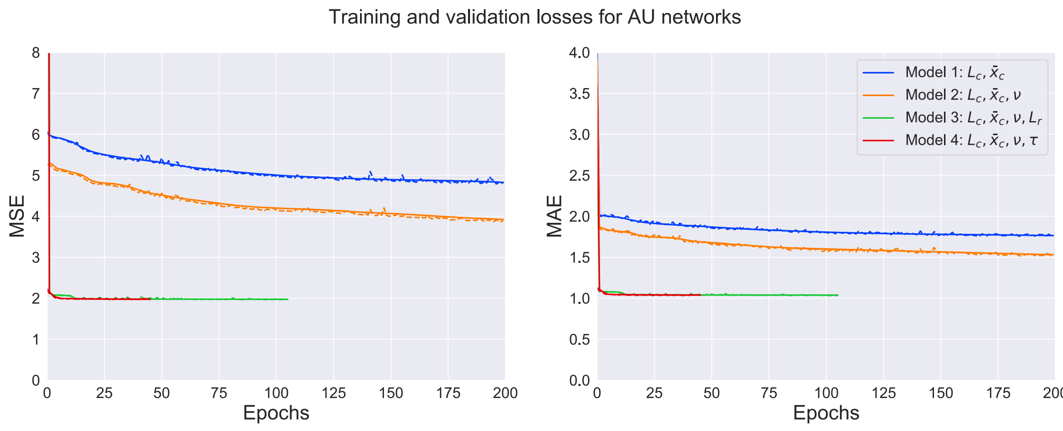


Figure 3. Mean squared error (MSE) and mean absolute error (MAE) for training (solid) and validation set (dashed) for Autoconversion Models 1–4.

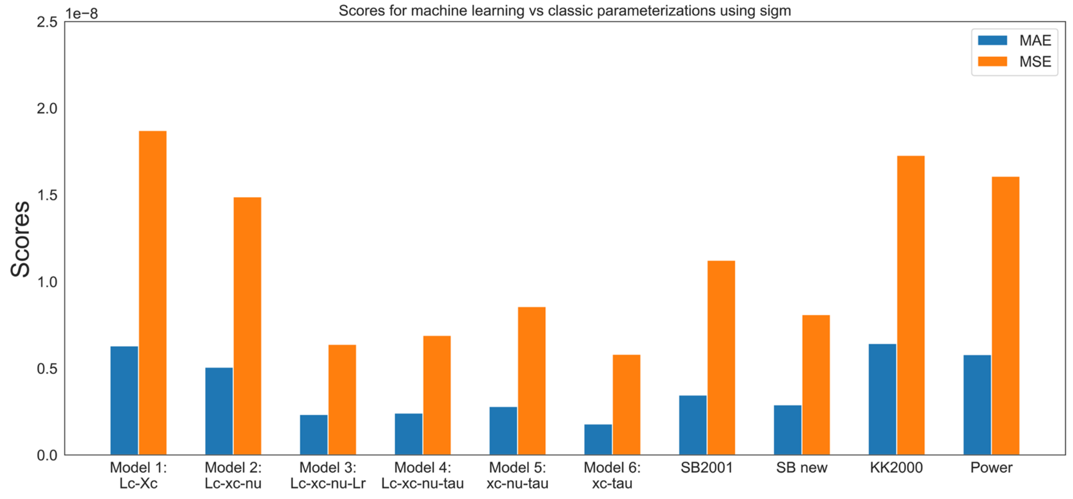


Figure 4. Mean squared error (MSE) and mean absolute error (MAE) for autoconversion calculated against the independent testing data. The ML models are compared with the parameterizations of SB2001, KK2000, and a power law regression model.

minimization. This brings the SB2001 ansatz closer to the MAE and MSE of ML Models 3 and 4. KK2000 is the parameterization of Khairoutdinov and Kogan (2000), which is of similar quality as ML Models 1 and 2 that also use only cloud variables. The model named "power" is a regression model using a power law ansatz for L_c, N_c, and ν, optimized using Levenberg-Marquardt minimization on the training data. This is only marginally better than KK2000.

This seems to make little difference for the scores against the testing data, Model 5 is slightly worse than Model 4, but Model 6 with $\bar{x}_c - \tau$ has the best MAE and MSE against the testing data. The result can be interpreted such that ML Models 3 and 4 are able to approximate those dependencies well enough from the original data and do not benefit from a simplification of the problem when we remove these dependencies. That Model 6 is better than Models 3 and 4 suggests that there are some issues with the generalization of those two models to the testing data.

Figure 5 shows a similar comparison of ML models with SB2001 for accretion and both self-collection rates. In general, we find that the use of additional predictors leads to an improvement of the models. The results are ML models that provide a significant improvement over SB2001 (and KK2000 for accretion). The additional improvement from including τ in the self-collection rates is small though. For accretion, it is not necessarily favorable to include ν, but this difference is not significant for our training and testing data. In a full atmospheric model, one could hope that simpler parameterizations or ML models might generalize better and drop those variables that provide only a marginal improvement.

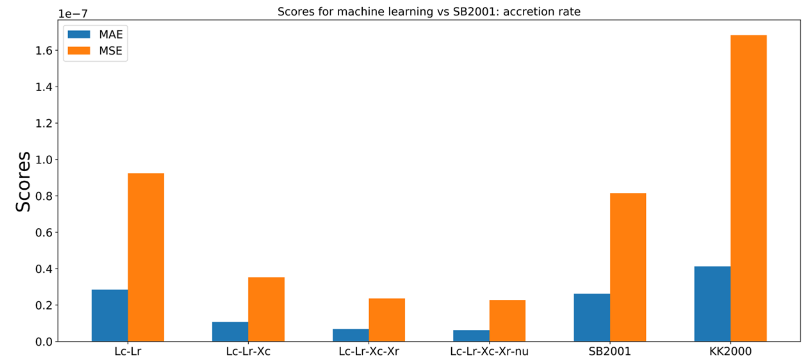
In the following (sections 6 and 7), we use $L_c - \bar{x}_c - \tau - \nu$ for autoconversion, $L_c - L_r - \bar{x}_c - \bar{x}_r - \nu$ for accretion, $L_c - \bar{x}_c - \nu - \tau$ for self-collection of cloud droplets, and $L_r - \bar{x}_r$ for the self-collection of rain.

As a pure ML application, we would already be done at this point. The ML models can make skillful predictions of the process rates that are of similar quality, or even better, than standard parameterizations used in atmospheric models. From the point of view of model development for NWP and climate models, on the other hand, we want to go at least one step further and test whether the ML-based process rates lead to skillful solutions of the ODE system that approximates the original KCEs. Before we do so, we first analyze the ML models in some more detail in the next section.

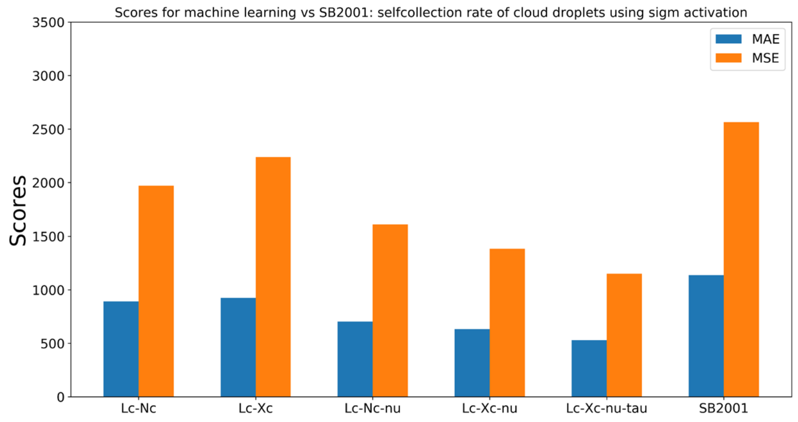
6. Interpreting the NNs With Partial Dependence Plots

While we have seen that the ML models can achieve similarly good predictions of the conversion rates as SB2001, they do not directly reveal how they do so. However, there are ways to peek inside the "black box" that allow to gain some insight from the trained models (McGovern et al., 2019). In particular, we use a method called partial dependence plots (PDP) (Friedman, 2001). To create a PDP, one predictor is set to a fixed value for all samples in the test set, which is then passed to the trained model to create a prediction.

(a) accretion



(b) self-collection of cloud droplets



(c) self-collection of raindrops

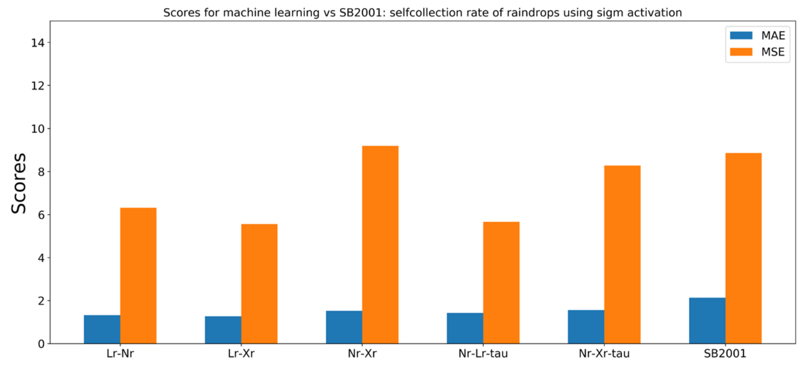


Figure 5. Mean square error (MSE) and mean absolute error (MAE) for accretion and self-collection calculated against the independent testing data and compared to SB2001.

By repeating this process for a range of predictor values and averaging the results, one obtains the effect of changing this one predictor on the prediction, all else being equal. This can then be repeated for every feature to obtain the plots in Figure 6. Note that in these univariate PDP plots, dependencies between the predictors are thus not taken into account. We also plotted the dependencies in SB2001 for comparison (see figure caption for details).

For accretion, the dependencies of the ML model as revealed by the PDP technique are very close to SB2001 for L_c and L_r, whereas the additional effects from taking into account $\bar{x}_c$, $\bar{x}_r$, and ν are small. This is also consistent with the results using different sets of predictors from the previous sections, which already suggested that L_c and L_r are the dominant predictors. For autoconversion ML Model 4 the PDP analysis shows

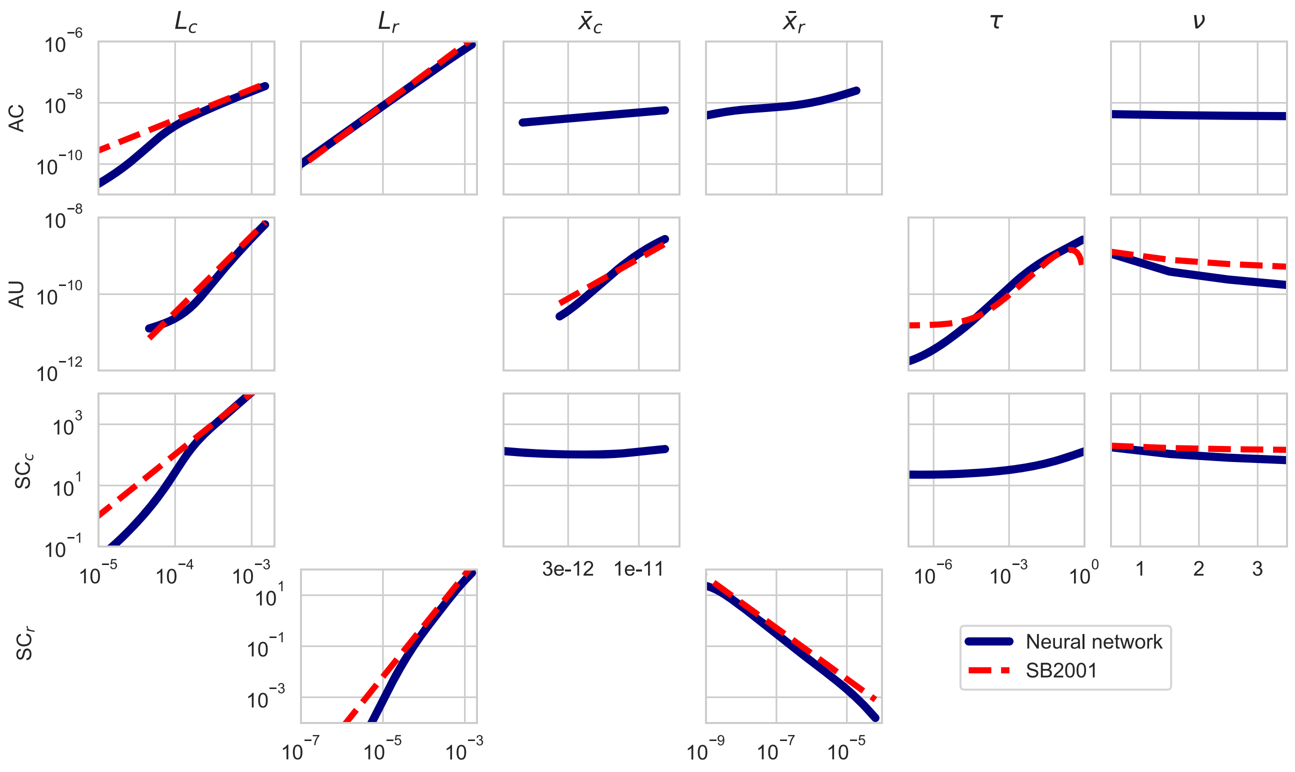


Figure 6. Partial dependence plots for the neural networks as described and selected in section 5. Autoconversion is ML Model 4 of Table 2. Red dashed lines denote predictions for SB2001. For AC, we used their Equation 21, for AU Equation 16, for SC_c Equation 14, and for SC_r Equation 19. For the variables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

that the ML approach finds dependencies on $L_c, \bar{x}_c, \tau$, and ν that are very similar to SB2001. The dependency on ν is somewhat, the one $\bar{x}_c$ only marginally, weaker than the analytic relations of SB2001. The latter originate directly from the Long (1974) kernel, but the Long-kernel itself is only an approximation, and, hence, it is not obvious that it provides the correct functional relationships. ML Model 4 predicts an increase of AU with τ that is qualitatively similar to $\Phi_{au}(\tau)$ of SB2001. Note that the τ dependency that results from the PDP analysis corresponds to

$$AU \sim \left[1 + \frac{\Phi_{au}(\tau)}{(1-\tau)^2}\right] \tag{22}$$

rather than just $\Phi_{au}(\tau)$ itself. This explains why AU of SB2001 levels off at small τ in Figure 6. For small τ the ML model shows smaller AU than SB2001.

Also, for the two self-collection rates the agreement between ML Model 4 and SB2001 is surprisingly good, although we have to emphasize that Figure 6 is plotted in log scales. For a small cloud liquid water content $L_c < 10^{-4}$ the ML model predicts a lower self-collection rate (SC_c resp.) than SB2001. The same is observed for rain with $L_r < 10^{-5}$ leading to a lower self-collection rate SC_r compared to SB2001.

The dependency of cloud droplet self-collection on ν shows a good agreement, whereas the additional dependency on $\bar{x}_c$ is weak. Cloud droplet self-collection shows an increase for large τ, which should be interpreted with caution due to the fact that this is only a box model without sedimentation or other effects. Note that the SB2001 cloud droplet self-collection rate as shown here includes the loss due to autoconversion.

Thus, the PDP technique confirms that the ML algorithm is able to extract physically reasonable nonlinear relationships for the warm-rain processes from the training data. The main dependencies are consistent with the well-established warm-rain parameterization of SB2001. The additional sensitivities are physically reasonable and promise to provide an improvement over SB2001. Whether they actually hold that promise will be to focus of the next section.

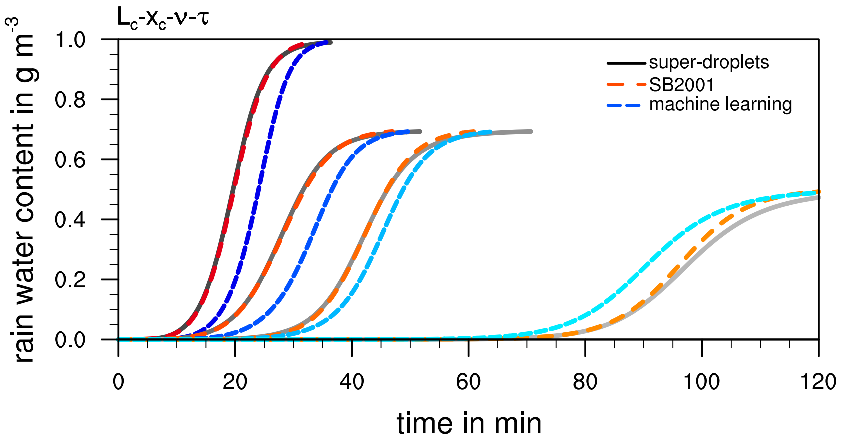


Figure 7. Time series of the rain water content for the solution of the KCE and the ODE solutions using SB2001 and ML Model 4 with autoconversion predictors $L_c, \bar{x}_c$, ν, and τ. Black and gray colors are the KCE solutions, red to orange colors for SB2001, and bluish colors the ML model. Shown 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, $\nu = 2$.

7. Results for the ODE System

In atmospheric models the warm-rain ODE system, Equations 6–9, is part of a much larger PDE system, and these source and sink terms are usually integrated with a simple Euler forward time stepping. Hence, we do the same here using a sufficiently small time step of 5 s. In NWP and climate models time steps of 20 s and up to several minutes are common, but this can deteriorate the solution of the warm-rain ODE system.

Note that we solve here only the ODE system as given by Equations 6–9 and no additional processes like drop sedimentation, drop breakup, or large-scale dynamics are taken into account. This ODE system should therefore parameterize the KCE as defined by Equation 1 and be interpreted as a box model or as spatially homogeneous cloud.

In the following, we focus on comparing the different ML models for autoconversion with benchmark solutions of the KCE and the parameterization of SB2001. In this section, all ML models use the same choices (and hyperparameters) for accretion and the two self-collection rates by using the ML models that gave the best results against the testing data.

Figure 7 shows the rainwater content L_r for four solution with different initial conditions with zero initial rainwater, but with different initial liquid water content $L_0 = L_c(t = 0)$, different initial mean volume radius $\bar{r}_0 = \bar{r}_c(t = 0)$ and different shape parameter ν. The time evolution is typical of all solutions of the KCE and shows a conversion from the pure cloud water initial condition with no rain to a pure rain water state where all cloud water has been depleted. Hence, the solutions change primarily in their time evolution, namely, the onset of the cloud-to-rain conversion, which is dominated by autoconversion, and the speed of the conversion to rain once a first significant amount of rain has formed. This slope is set by the accretion rate. Obviously, the amount of rain is limited by the total liquid water in the system L_0. Lower L_0, smaller $\bar{r}_0$, and larger ν lead to a slower formation of rain and a delay of the transition to a pure rain state. For these four cases, SB2001 provides almost perfect solutions that are very close to the benchmark solutions of the KCE. The ML model, here Model 4, shows a delay for the three cases with $\nu = 0$ and a too fast formation of rain for the slowest case with $\nu = 2$. Hence, the dependency on ν seems to be suboptimal in the ML model. To analyze those dependencies more systematically, we define a time scale t_p as the time when p percent of the cloud water has been converted to rainwater. The time scale t_{50} is a good measure for the overall performance of the ODE systems, whereas t_{10} focuses on the initial stage where autoconversion tends to dominate.

Figure 8 presents the results for t_{10} for ML Model 2 with predictors $L_c, \bar{x}_c$, and ν and Model 4 that, in addition, includes τ. ML Model 2 has clear deficiencies and fails to represent the timing of the benchmark solutions in a large part of the parameter space. In contrast, Model 4 shows a reasonable behavior, and also, the dependencies on $\bar{r}_0$ and ν are qualitatively correct. But ML Model 4 shows a significant bias with a delay in most parts of the phase space and a too weak sensitivity to $\bar{r}_0$. The dependency on ν is actually captured quite well

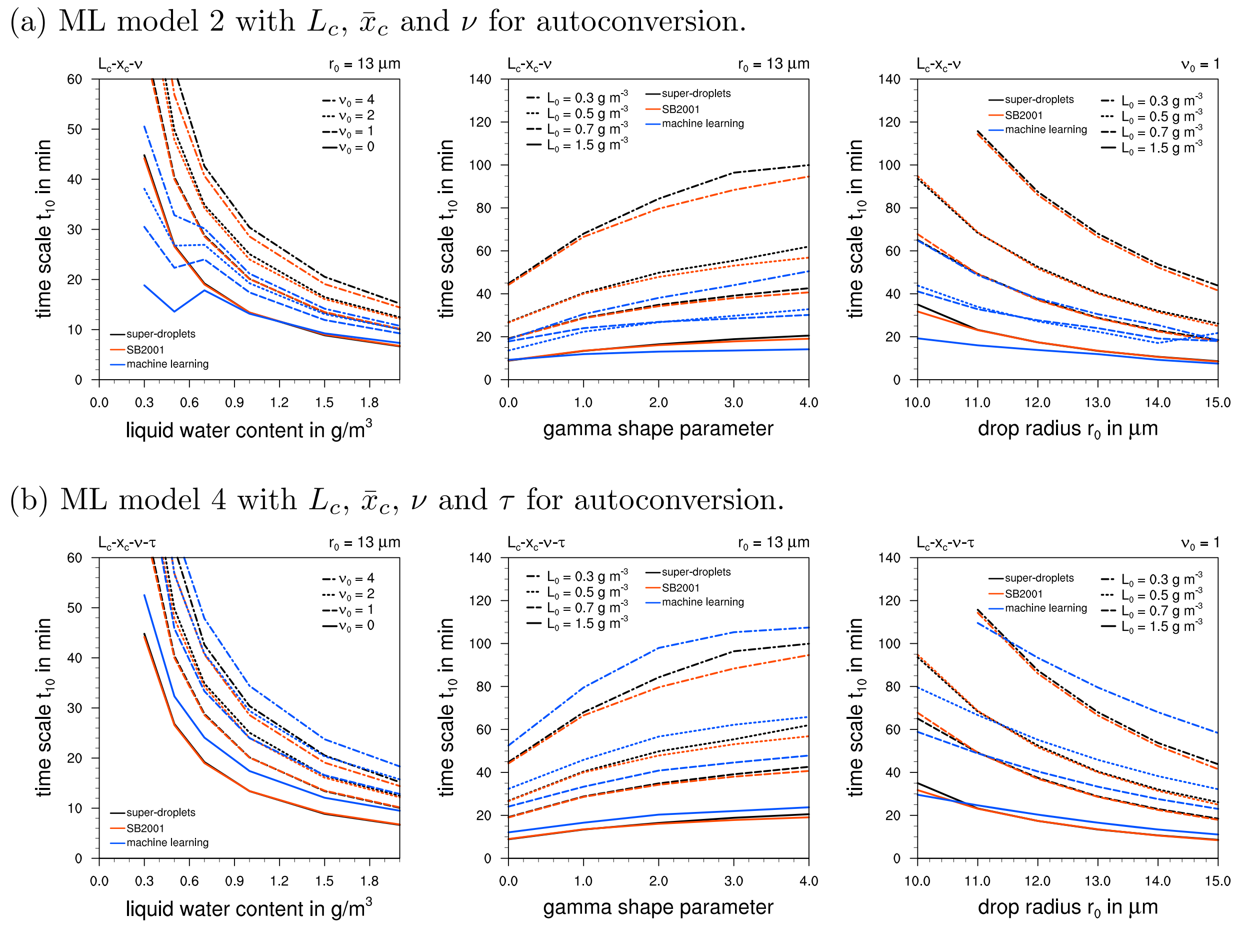


Figure 8. Time scale t_{10} in min as a function of L_0, ν, and $\bar{r}_0$ for two ML models without (top row) and with τ as predictor (bottom row) for autoconversion. The superdroplet benchmark solutions (black) are shown together with SB2001 (red) and the respective ML model (blue).

here and only slightly too strong. SB2001 is significantly better than ML Model 4 and matches the benchmark simulation surprisingly well given that it has not been retraining or retuned for this data set but is the original parameterization as published in 2001. On the other hand, we have used exactly the same collection kernel as in SB2001 and only applied a different numerical method. This shows that the good scores of the ML models against the testing data of the process rates do not directly transfer to the ODE solutions. In contrast, SB2001 is better for the ODE solutions, although it is worse against the testing data for the process rates. This will be discussed and explained in section 8. Here we just note that ML Model 4 would be an acceptable warm-rain parameterization, but in this metric, it is clearly outperformed by SB2001.

The deficiencies of ML Model 2 can also clearly be seen in the time series of *AU* and *AC* shown in Figure 9. ML Model 4 provides a very good approximation of both AU and AC, for this case, and is in some aspects even superior to SB2001. The latter shows a remarkable overestimation of *AU* in the first 10 min. Hence, for cases like this, for which ML Model 4 has only a small temporal bias, the solution reproduces the detailed evolution of the KCE reasonably well. In contrast, the autoconversion rate of Model 2 starts much too high, decreases with time and even shows some signs of instability later in the simulation. The inability to represent the time evolution of *AU*, and especially the increasing *AU* in the first stage of rain formation, can also be observed for many other parameterizations that are only based on cloud variables like Berry and Reinhardt (1974), Beheng (1994), and Khairoutdinov and Kogan (2000). Hence, this is caused by our choice of predictors, not by the ML approach itself.

To compare with the other ML models, we analyze the overall errors for t_{10} as shown in Figure 10 and in Table 2. The scores are evaluated over a wide range of initial conditions with $L_0 \in [0.3, 2]$ g m^{-3}, $\bar{r}_0 \in [10,15]$ µm, and $\nu \in [0, 4]$. This reveals that the other ML models with different predictors, including those

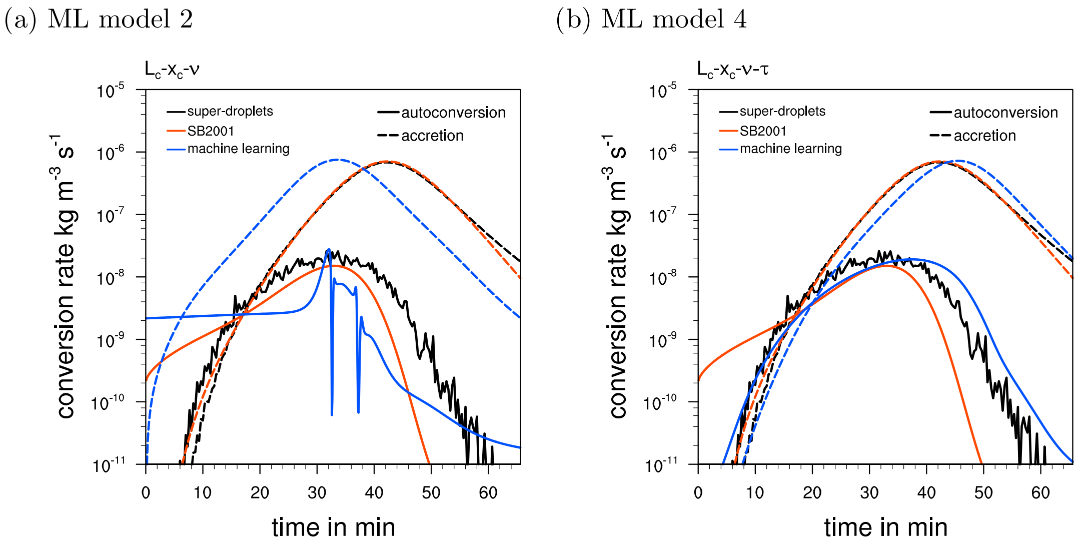


Figure 9. Time series of the autoconversion rate AU and the accretion rate AC of ML Models 2 and 4 (blue) for the initial condition $L_0 = 0.7$ g m^{-3}, $\bar{r}_0 = 11$ μm, $\nu = 0$. For comparison the superdroplet benchmark simulations are shown (black) as well as SB2001 (red).

borrowing some dependencies from SB2001, cannot improve significantly over Model 4. Only Model 3, which uses L_r instead of τ, is slightly better for the ODE solutions. In our opinion, this small difference is not significant. In terms of mean error (ME) and mean relative error (MRE) Model 5 is the best ML model. Overall, none of the ML models comes even close to the low MAE and MSE of SB2001. A detailed inspection shows that all ML-based models have difficulties with the slowly evolving cases of low L_0, small $\bar{r}_0$, and large ν.

In the next section, we will analyze this behavior in more detail and test some more ML models with the goal to remedy these deficiencies.

8. Sensitivities and Discussion

ML models can be fine-tuned by altering the design of the neural net and the optimization algorithm that determines the trainable parameters (Géron, 2019). In the ML community, such choices are called hyperparameters. Most prominently for neural nets is the size of the network. A wider and deeper neural net has more trainable parameters and can fit the training data better, but this will not necessarily generalize to the testing data or the application.

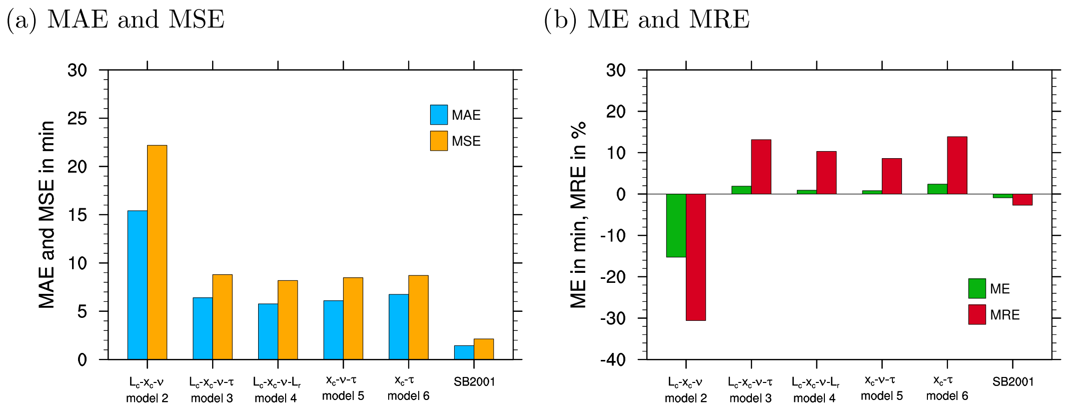


Figure 10. Mean absolute error (MAE) and mean squared error (MSE) for t_{10} for various ML models with different predictors and SB2001 (left), as well as mean error (ME) and mean relative error (MRE) for the same models (right).

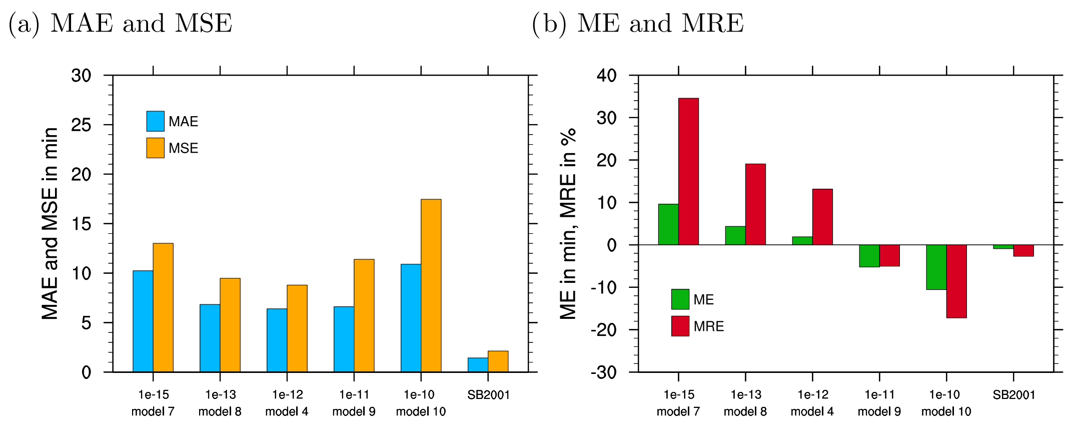


Figure 11. Mean absolute error (MAE) and mean squared error (MSE) for t_{10} for ML Models 13 to 18 and 4 using different thresholds ϵ_1 as well as SB2001 (left), and mean error (ME) and mean relative error (MRE) for the same models (right).

As part of this study, we have made many attempts to further improve the performance of the ML model to come closer to the ODE results of SB2001. We have, for example, tested deeper and wider NNs as well as different activation functions. None of this leads to a significant improvement in the behavior of the ODE solutions, and these further experiments are therefore documented in the supporting information. The main bottleneck for a further improvement seems to be that improved results for the training or test set do not necessarily transfer to the ODE solutions.

Some hint for the underlying cause for the difficulties to further improve the ML models can be gained from experiments that change not the ML model itself, but the threshold ϵ_1 in the data reduction of the training data. Changing ϵ_1 has a pronounced and systematic impact on the timing of the ODE solutions: A smaller ϵ_1 leads to a further delay of the rain formation, especially for the difficult initial conditions, for example, large ν, whereas a larger ϵ_1 results in a more rapid formation of rain (Figure 11 and Table 3). At this point, we may also reveal that the initial setting of $\epsilon_1 = 10^{-12}$ is an a posteriori choice that minimizes the MAE, MSE, and ME of the ODE solutions; only for MRE an even larger ϵ_1 would be better.

Further evidence for the cause of the problem can be gained from Figure 9. The autoconversion rate AU can be very small at initial times, and for large ν it becomes virtually zero for the benchmark solutions and, hence, in the training data. If AU is zero at initial time in the ODE system, though, no rain will ever develop because AC is also zero due to L_r being zero by definition. When we remove the small values of AU at initial time with help of an increased ϵ_1, then the ML model extrapolates ending up with a higher value for AU. This explains the change in the ME and MRE for different ϵ_1 switching for a delay at low ϵ_1 to a too fast development at high ϵ_1. Hence, we have to conclude that the benchmark simulations do not provide useful training data for AU at initial time. The underlying cause for this behavior is that the concept of autoconversion is, in fact, ill-posed for large ν and small $\bar{r}_0$. The two-moment parameterization is only based on the moments $M_c^{(0)} = N_c$ and $M^{(1)} = L_c$. For a very slowly evolving narrow cloud droplet distribution

Table 3
The Same as Table 2 but for ML Models With Predictors L_c, $\bar{x}_c$, ν, τ (as Model 4) but Using Different Thresholds ϵ_1 in the Data Reduction of AU

	Trainable		Testing data		ODE solutions, t_{10} errors			
No.	param.	ϵ_1	MAE	MSE	MAE	MSE	ME	MRE
SB	(6)	—	3.46	11.23	1.43	2.13	−0.91	−2.70
7	609	1e−15	2.32	6.41	10.24	13.01	9.60	34.57
8	609	1e−13	2.34	6.48	6.84	9.48	4.36	19.08
(4)	609	1e−12	2.41	6.90	6.40	8.79	1.90	13.14
9	609	1e−11	2.62	7.85	6.61	11.40	−5.22	−5.04
10	609	1e−10	2.36	6.06	10.90	17.46	−10.56	−17.23

these two moments contain no information about the evolution of the drop size distribution during this earliest stage of the rain formation. The collisional growth is initially only affecting the higher moments of the cloud droplet distribution, which are, unfortunately, not part of the two-moment model. Hence, when we follow this interpretation, the ML models have to fail because the training data contain no information that can lead to an ODE system that describes those solutions of the KCE. The puzzling piece here is that such an ODE system seems to exist in form of the SB2001 parameterization.

So how and why does SB2001 actually work? SB2001 does overestimate AU for $\tau \ll 0.1$, especially for the difficult cases with small L_0, small $\bar{r}_c$, and large ν. This can clearly be seen in Figure 9 and becomes even more severe for more extreme cases. The parameterization compensates this by a decrease of AC for small τ that is not based on data (see the supporting information). The parameters in $\Phi_{ac}(\tau)$, Equation 15, have been chosen such that the solutions of the ODE system match the timing of the reference solution of the KCE. Thus, the good agreement of SB2001 for t_{10} is a combination of the self-similarity of the solutions, which is the mathematical foundation that makes this possible, and a tuning of the ODE system for t_{10}. The self-similarity is instrumental in this because for each slowly evolving solution with small $\bar{r}_c$, small L_0, and large ν we can in principle find a corresponding fast solution with nonvanishing AU by taking the limit to large L_0. The self-similarity will then guarantee that the rescaled similarity solution matches the time evolution of the true slowly evolving solution. This neither requires nor guarantees that AU and AC are both correct for the slowly evolving solution, though. Hence, for those extreme cases SB2001 is not a rigorous approximation of the process rates based on the numerical data, but a parameterization that mimics the time evolution of solutions of the KCE.

In some sense ML, at least in the approach that we have chosen here, is an attempt to be more rigorous than SB2001, but such a direct solution based only on data may not exist for this problem. In the parameterization community, this is known as a closure problem. The chosen model, here the ODE system for the two moments $M_c^{(0)} = N_c$ and $M^{(1)} = L_c$, cannot be derived rigorously from the fundamental physical equations (here the KCE) without the help of additional and independent assumptions, for example, making use of conservation laws or invariance properties that may be part of the underlying equations but are not yet contained in the parameterized model. Due to the closure problem associated with vanishing autoconversion for small $\bar{r}_c$ and large ν, it remains unclear whether a two-moment scheme is actually able to represent autoconversion properly over the full range of parameters. The results presented here indicate that it is not. Hence, the problem might be fundamentally ill-posed. Nevertheless, the SB2001 model provides a good parameterization because it makes use of the invariance of the KCE to time transformations as an additional closure assumption.

9. Conclusions

In this study we have applied standard ML methods in form of fully connected neural nets to the parameterization problem of warm-rain cloud microphysics. We find that ML-based models provide a reasonable representation of the microphysical process rates. The ML approach confirms previous results that including rain water information is essential to parameterize warm-rain autoconversion, that is, in a standard two-moment model cloud water variables alone are insufficient to predict autoconversion. The dependencies of appropriate ML models of the cloud microphysical process rates are consistent with established warm-rain parameterizations. The resulting ML-based parameterizations can be applied in actual atmospheric models to describe the warm-rain processes.

A great advantage of the ML approach is that different model formulations can be implemented and evaluated quite quickly, for example, models using different sets of predictors. In the supporting information, we document an extension of the warm-rain two-moment model to predict the number change due to autoconversion and accretion explicitly. To our knowledge, this has never been done with standard bulk schemes, but it can, in our case, improve the prediction of cloud droplet size.

More generally, the ML approach seems especially well suited to parameterize the complications and details of ice microphysical processes, which often require a mix of analytical relations and look-up tables in state-of-the-art parameterizations. Hence, the ML approach has the potential to greatly simplify and speed up the development process of microphysical parameterizations for NWP and climate models.

The main problem of the ML approach presented here is that for the actual application in form of an ODE system, the quality of the ML-based models is inferior to the classic non-ML parameterizations. The ML models show significant biases especially for slowly evolving solutions of the KCE. This behavior cannot be improved by modifying the ML model using standard hyperparameters like size of the network, activation function, and so forth. We posit that these biases of the ML models are not so much a deficiency of the ML approach itself but are caused by inherent limitations of the chosen two-moment warm-rain model. During the first stage of rain formation, the two-moment model is insufficient to describe the evolution of the process rates, and, hence, it cannot be trained properly with the process rates of the benchmark simulations and standard ML approaches. In physically based parameterizations such limitations can be overcome by making additional closure assumptions, but often, this leads to compensating errors being introduced.

A logical step would be to include additional higher moments as predictors. This could maybe overcome the problems and the ML methods do offer a powerful technique to derive such more complicated parameterizations. Still, there might be new challenges when going to more and higher moments like the treatment of activation and turbulent mixing and the effect of these processes on the higher moments of the cloud droplet distribution. In addition, three-moment schemes are more costly in a computational sense, when applied in a full NWP or climate model.

Maybe more advanced ML techniques like, for example, neural ODEs (Chen et al., 2018; Rackauckas et al., 2020) can resolve the issues even on the level of a two-moment scheme by learning the equations from data in the sense that the ML models include the concept of an ODE or of a function and its derivative. Being able to include the time evolution of the model variables (the ODE solutions) and the process rates (the derivatives of the ODE solutions) in the training data would probably resolve the closure problem related to the ill-posedness of the two-moment warm-rain parameterization. Then we would only have to remove the misleading autoconversion rates during the initial stage from the training data and replace those with the corresponding benchmark solution. By being able to train directly on the time evolution of the model variables, such advanced ML methods might succeed to extract an ODE system that is as good as or even better than SB2001. Even more so if it would be possible to build the time invariance of the KCE into the neural net similar to the Galilean invariance that is included in ML-based turbulence models (Ling, Jones, et al., 2016; Ling, Kurzawski, et al., 2016). But to achieve this, the ML model first needs a concept of time and time derivatives.

In this study, we have not analyzed the computational performance or efficiency of the ML approach versus classic parameterizations. This has to be done in a full model framework and, probably, taking into account hardware accelerators like GPUs. For the simple warm-rain ODE and using CPUs, the ML models are in fact more expensive than the classic parameterizations. But this might change dramatically for full mixed-phase microphysics schemes and using an implementation and hardware that is tailored toward ML models. Hence, we do not want to draw any conclusions from the current study regarding computational performance or efficiency.

Data Availability Statement

The training and testing data, the Python notebooks, NCL scripts, and ODE solutions are provided at Zenodo (https://doi.org/10.5281/zenodo.3988974) and at Gitlab under the public project (https://gitlab.com/axelseifert/warmrain). The Lagrangian microphysics model McSnow is part of the ICON modeling framework, which is a joint effort of Deutscher Wetterdienst (DWD) and the Max Planck Institute for Meteorology (MPI-M). ICON licenses for scientific use are available at not cost online (at https://code.mpimet.mpg.de/projects/iconpublic/). Subsequently, the access to the McSnow GIT archive can be granted by A. S.

Acknowledgments
Stephan Rasp acknowledges funding from the German Research Foundation under Grant No. 426852073. We thank Christoph Siewert for discussions regarding the details of the Monte Carlo superdroplet algorithm. We thank Bjorn Stevens for numerous comments that helped to improve the manuscript. A. S. is deeply grateful to Detlev Majewski, who made this study possible by personally relieving the first author from many administrative and operational duties.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Retrieved from https://www.tensorflow.org/ (Software available from tensor flow.org).
Alfonso, L., Raga, G. B., & Baumgardner, D. (2008). The validity of the kinetic collection equation revisited. *Atmospheric Chemistry and Physics*, *8*(4), 969–982. https://doi.org/10.5194/acp-8-969-2008
Beheng, K. D. (1994). A parameterization of warm cloud microphysical conversion processes. *Atmospheric Research*, *33*, 193–206.
Beheng, K. D. (2010). The evolution of raindrop spectra: A review of microphysical essentials. *Rainfall: State of the science*, *191*, 29–48.
Berry, E. X., & Reinhardt, R. L. (1974). An analysis of cloud drop growth by collection: Part IV. A new parameterization. *Journal of the Atmospheric Sciences*, *31*, 2127–2135.

Beucler, T., Pritchard, M., Gentine, P., & Rasp, S. (2020). Towards physically-consistent, data-driven models of convection. *arXiv, physics.ao-ph*, 2002.08525.

Brenowitz, N. D., Beucler, T., Pritchard, M., & Bretherton, C. S. (2020). Interpreting and stabilizing machine-learning parametrizations of convection. *arXiv, physics.ao-ph*, 2003.06549.

Brenowitz, N. D., & Bretherton, C. S. (2018). Prognostic validation of a neural network unified physics parameterization. *Geophysical Research Letters*, *45*, 6289–6298. https://doi.org/10.1029/2018GL078510

Brenowitz, N. D., & Bretherton, C. S. (2019). Spatially extended tests of a neural network parametrization trained by coarse-graining. *Journal of Advances in Modeling Earth Systems*, *11*, 2728–2744. https://doi.org/10.1029/2019MS001711

Brunton, S. L., Noack, B. R., & Koumoutsakos, P. (2020). Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, *52*, 477–508.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31* (pp. 6571–6583). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/7892-neural-ordinary-differential-equations.pdf

Chevallier, F., Morcrette, J.-J., Chéruy, F., & Scott, N. A. (2000). Use of a neural-network-based long-wave radiative-transfer scheme in the ECMWF atmospheric model. *Quarterly Journal of the Royal Meteorological Society*, *126*(563), 761–776.

Chollet, F., et al. (2015). Keras. https://keras.io

Doms, G., & Beheng, K. D. (1986). Mathematical formulation of self-collection, autoconversion and accretion rates of cloud and raindrops. *Meteorologische Rundschau*, *39*, 98–102.

Drake, R. L. (1972). A general mathematical survey of the coagulation equation. In G. M. Hidy & J. R. Brock (Eds.), *Topics in current aerosol research* (pp. 201–376). Oxford: Pergamon Press.

Dziekan, P., & Pawlowska, H. (2017). Stochastic coalescence in Lagrangian cloud microphysics. *Atmospheric Chemistry and Physics*, *17*(22), 13,509–13,520.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of statistics*, 1189–1232.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media, Inc.

Hall, W. D. (1980). A detailed microphysical model within a two-dimensional dynamical framework: Model description and preliminary results. *Journal of the Atmospheric Sciences*, *37*, 2486–2507.

Huntingford, C., Jeffers, E. S., Bonsall, M. B., Christensen, H. M., Lees, T., & Yang, H. (2019). Machine learning and artificial intelligence to aid climate change research and preparedness. *Environmental Research Letters*, *14*(12), 124007.

Kessler, E. (1969). *On the distribution and continuity of water substance in atmospheric circulations*. Boston: Amer. Meteor. Soc.

Khairoutdinov, M., & Kogan, Y. (2000). A new cloud physics parameterization in a large-eddy simulation model of marine stratocumulus. *Monthly Weather Review*, *128*, 229–243.

Lary, D. J., Alavi, A. H., Gandomi, A. H., & Walker, A. L. (2016). Machine learning in geosciences and remote sensing. *Geoscience Frontiers*, *7*(1), 3–10. Special issue: Progress of machine learning in geosciences https://doi.org/10.1016/j.gsf.2015.07.003

Ling, J., Jones, R., & Templeton, J. (2016). Machine learning strategies for systems with invariance properties. *Journal of Computational Physics*, *318*, 22–35.

Ling, J., Kurzawski, A., & Templeton, J. (2016). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, *807*, 155–166.

Long, A. B. (1974). Solutions to the droplet collection equation for polynomial kernels. *Journal of the Atmospheric Sciences*, *31*, 1040–1052.

Maxwell, A. E., Warner, T. A., & Fang, F. (2018). Implementation of machine-learning classification in remote sensing: an applied review. *International Journal of Remote Sensing*, *39*(9), 2784–2817. https://doi.org/10.1080/01431161.2018.1433343

McGovern, A., Lagerquist, R., Gagne, D. J., Jergensen, G. E., Elmore, K. L., Homeyer, C. R., & Smith, T. (2019). Making the black box more transparent: Understanding the physical implications of machine learning. *Bulletin of the American Meteorological Society*, *100*, 2175–2199. https://doi.org/10.1175/bams-d-18-0195.1

O'Gorman, P. A., & Dwyer, J. G. (2018). Using machine learning to parameterize moist convection: Potential for modeling of climate, climate change, and extreme events. *Journal of Advances in Modeling Earth Systems*, *10*, 2548–2563. https://doi.org/10.1029/2018MS001351

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., et al. (2020). Universal differential equations for scientific machine learning. *arXiv, cs.LG*, 2001.04385.

Rasp, S., Pritchard, M. S., & Gentine, P. (2018). Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, *115*(39), 9684–9689. https://doi.org/10.1073/pnas.1810286115

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., & Deep learning and process understanding for data-driven Earth system science (2019). Deep learning and process understanding for data-driven Earth system science. *Nature*, *566*(7743), 195–204.

Seifert, A., & Beheng, K. D. (2001). A double-moment parameterization for simulating autoconversion, accretion and selfcollection. *Atmospheric Research*, *59–60*, 265–281.

Shima, S., Kusano, K., Kawano, A., Sugiyama, T., & Kawahara, S. (2009). The super-droplet method for the numerical simulation of clouds and precipitation: A particle-based and probabilistic microphysics model coupled with a non-hydrostatic model. *Quarterly Journal of the Royal Meteorological Society*, *135*(642), 1307–1320.

Sønderby, C. K., Espeholt, L., Heek, J., Dehghani, M., Oliver, A., Salimans, T., et al. (2020). MetNet: a neural weather model for precipitation forecasting. *arXiv, cs.LG*, 2003.12140.

Unterstrasser, S., Hoffmann, F., & Lerch, M. (2017). Collection/aggregation algorithms in Lagrangian cloud microphysical models: Rigorous evaluation in box model simulations. *Geoscientific Model Development*, *10*(4), 1521.

von Smoluchowski, M. (1916). Drei Vorträge über Diffusion, Brownsche Molekularbewegung und Koagulation von Kolloidteilchen. *Physikalische Zeitschrift*, *17*, 557–599.

von Smoluchowski, M. (1917). Versuch einer mathematischen Theorie der Koagulationskinetik kolloidaler Lösungen. *Zeitschrift für Physikalische Chemie*, *92*, 129–168.

Xingjian, S. H. I., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *29th Annual Conference on Neural Information Processing Systems, NIPS 2015; Montreal; Canada* (pp. 802–810). San Diego, CA: NIPS foundation.

Yuval, J., & O'Gorman, P. A. (2020). Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions. *Nature Communications*, *11*(1), 1–10.