Responses to the comments by Reviewer 1.

We greatly appreciate constructive comments and recommendations by Reviewer 1.

Changes recommended:

(I) The weight of the manuscript needs to be on the ML approach, this is currently not the case.

We have removed some details about the aerosols and have added the following section to the introduction:

"This paper investigates the potential of using advances in machine learning to invert aerosol properties (aerosol extinction coefficient profiles, single scattering albedo and scattering phase function) from a hyperspectral remote sensing technique called multi-axis differential optical absorption.

Machine learning (ML) is a branch of artificial intelligence that derives its roots from pattern recognition and statistics. The goal of ML is to build statistical (or mathematical) models of a real-world phenomenon by relying on training examples. For instance, in supervised ML, a model is first presented with a set of paired examples (termed as the training set), where every training example contains a pair of input variables and output variables, and the goal of ML algorithms is to find the statistical structure of mapping from the input variables to the output variables that match with the training examples and can be generalized to unseen examples (termed as test set). The learned mapping (or the model) can be applied to the inputs of test examples to make predictions on their outputs. There are several advantages of using ML. Firstly, it can sift through vast amounts of training data and discover patterns that are not apparent to humans. Secondly, ML algorithms can have continuous improvement in accuracy and efficiency with increasing amount of training data. Thirdly, ML algorithms are usually very fast to apply on test examples since the time-consuming training process of ML models is offline and one-time. With these advantages as well as the availability of faster hardware, ML has soon become the most popular data analytic technique since the 1990s. In recent years, it has also been applied to the field of remote sensing (Efremenko et al., 2017; Hedelt et al., 2019).

Artificial neural networks (ANN) is one of the many methods studied in the ML field that has found a lot of success in recent years over a number of commercial problems such as image detection, text translation, and speech recognition. It is inspired by the biological neural networks constituting animal brains. As an analogy to a biological brain, an ANN is based on artificial neurons. An artificial neuron is a mathematical function receiving and processing input signals and producing outputs signals or activations. Each neuron comprises of weighted inputs, an activation function, and an output. Weights of the neuron are parameters to be adjusted, while the activation function defines the relationship from the input signals to the output signals. When multiple neurons are composed together in a layered manner (where the output signals of neurons in a given layer are used as inputs for the neurons in the next layer), we call it an artificial neural network (ANN). A common algorithm for training ANNs is the backpropagation algorithm, that passes the gradients of errors on the training set from the output layer to inner

layers to refine the weights at all layers in an incremental way. The backpropagation algorithm converges when there is no change in ANN weights across all layers beyond a certain threshold. There are several optimization methods that are used for performing backpropagation and are behind standalone ANN packages commonly used by the ML community. ANNs have many different types depending on the specifics of the neuron arrangement or architecture. A simple type of ANN is a multilayer perceptron (MLP), where all neurons at a given layer are fully connected with all neurons of the next layer, also termed as dense layers. Other complex types of ANN include convolutional neural network (CNN) and recurrent neural network (RNN). Two important types of artificial neural networks used in this study are the convolutional neural networks (CNN) (Fukushima, 1980; LeCun et al., 1999) and the Long short-term memory (LSTM) neural networks (Hochreiter and Schmidhuber, 1997), which are variants of recurrent neural networks.

Convolutional neural network (CNN) is a class of deep neural networks that uses the convolution operation to define the type of connections from one layer to another. While they have shown impressive results in extracting complex features from images in computer vision applications (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015), they are relevant in many other applications involving structured input data, e.g., 1D-sequences. A CNN is composed of an input layer, multiple hidden layers and an output layer. The hidden layers usually consist of several convolutional layers, followed by pooling layers, fully connected layers (dense layers) and normalization layers. Figure 1 shows a simple example of CNN. The input vector (or sequence) is first passed through a convolutional layer where it is convolved with 3 filters (convolution kernels) of size 3 using the same padding to produce three 6x1 feature maps. Since the ReLU function ($f(x) = \max(0, x)$) is commonly chosen as the activation function in CNNs, the feature maps only contain positive values. Then the max pooling layer picks the maximum value every 3 elements for each feature map, generating three 2 x 1 vectors. After passing through a flatten layer, the max pooling output is reshaped into a 6 x 1 vector, which is followed by a dense (fully connected) layer with 2 nodes. The dense layer multiplies its input by a weight matrix and add a bias vector for generating the output of the model. The computer adjusts the model's convolutional kernel values or weights through a training process called backpropagation, a class of algorithms utilizing the gradient of loss function to update weights. For the case in Figure 1, there are 26 tunable parameters. ($(3 + 1)\times 3 = 12$ from convolution kernels and $(6 + 1)\times 2 = 14$ from the dense layer.)

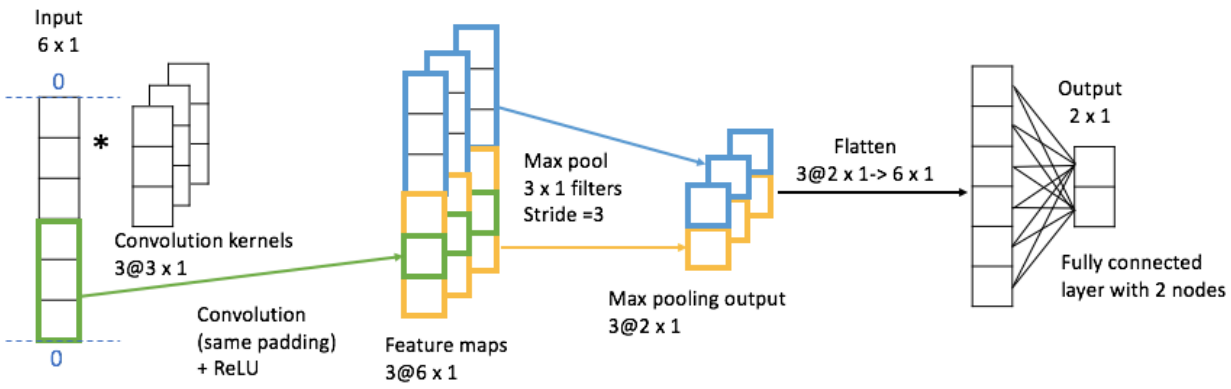

Figure 1: a simple example of CNN

Long short-term memory (LSTM) neural networks have many applications such as speech recognition (Li and Wu, 2015) and handwriting recognition (Graves et al., 2008; Graves and Schmidhuber, 2009). They are a special kind of ANNs termed as recurrent neural networks (RNNs). RNNs are designed for modeling sequence dependent behavior (e.g., in time). They are called "recurrent" because they perform the same operation for every element of a sequence, with the output at a given element dependent on previous computations at earlier elements (Britz, 2015). This is different from traditional neural networks wherein all the input-output examples are assumed to be independent of each other.

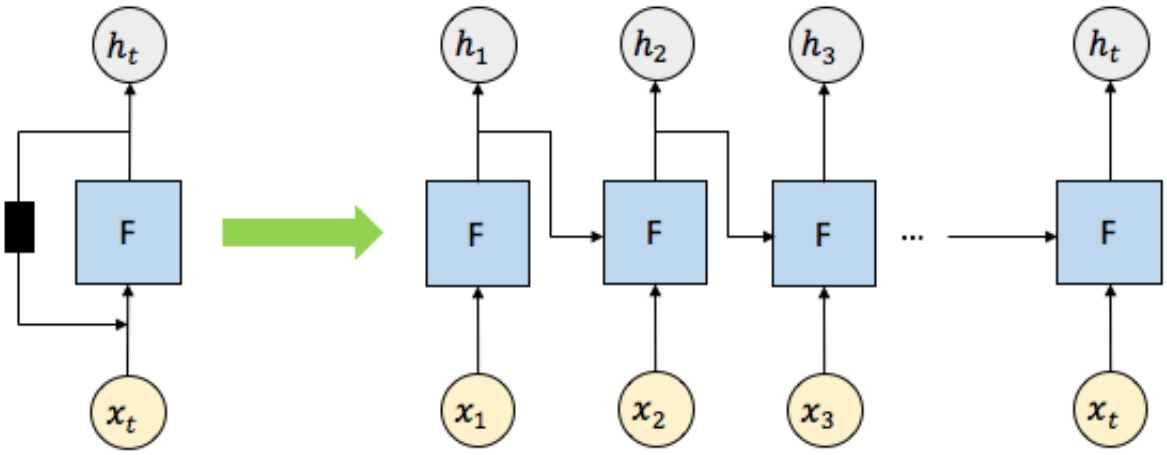

Figure 2: Unrolled recurrent neural network

Figure 2 is a diagram of an unrolled RNN with t input nodes, where "unrolled" means showing the network for the full sequence of inputs and outputs. The RNNs work as follows. At the first element of the sequence, the set of input signals x_1 (which can be multi-dimensioinal) is fed into the neural network F to produce an output h_1. At the next element of the sequence, the same neural network F takes both the next input x_2 and previous output h_1, generating the next output h_2. This recurrent computation continues for t times to produce the output at the t^{th} element of the sequence, h_t. While RNNs are powerful architectures for modeling sequence behavior, classical RNNs are inadequate to capture long-term memory effects where the inputs-outputs at a given element of the sequence can affect the outputs at another element of the sequence separated by a long interval. Long-short-term memory (LSTM) models are variants of RNNs that are able to overcome this challenge and are efficient at capturing long-term dependencies as well as short-term dependencies. It does so by introducing an internal memory state that is operated by neural network layers termed as gates, such as the "input gate," that adds new information from the input signals to the memory state, the "forgot gate," that erases content from the memory state depending on the input signals, and the "output gate," that transforms information contained in the input signals and the memory state to produce output signals.

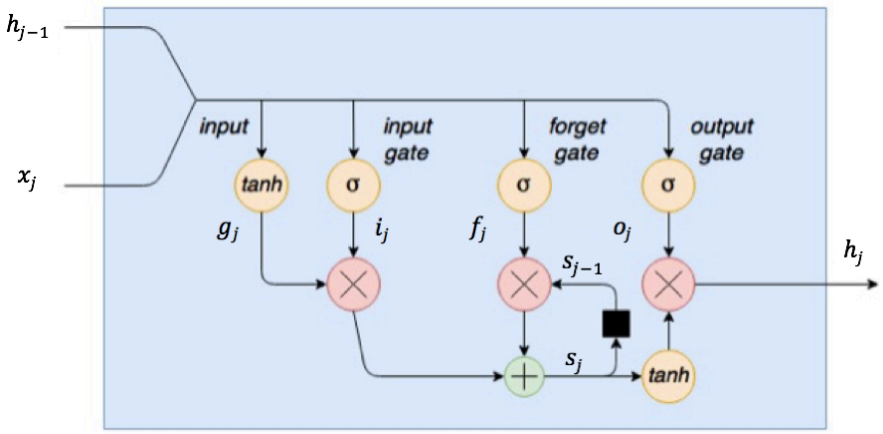

Figure 3: LSTM cell diagram (modified from Thomas, 2018).

The LSTM cell used in this study is illustrated in Figure 3, of which the update rules are:

$$g_j = \tanh\left(b^g + x_j U^g + h_{j-1} V^g\right)$$

$$i_j = \sigma(b^i + x_j U^i + h_{j-1} V^i)$$

$$f_j = \sigma(b^f + x_j U^f + h_{j-1} V^f)$$

$$s_j = s_{j-1} \circ f_j + g_j \circ i_j$$

$$o_j = \sigma(b^o + x_j U^o + h_{j-1} V^o)$$

$$h_j = \tanh(s_j) \circ o_j$$

where j is the element index, $\sigma(x)$ represents the sigmoid function, and $\tanh(x)$ represents the hyperbolic tangent function. $x \circ y$ denotes the element-wise product of x and y. U^g, U^i, U^f, U^o are the weights for the input x_j, while V^g, V^i, V^f, V^o are the weights for the other input h_{j-1}, and b^g, b^i, b^f, b^o are the scalar terms (termed as bias). The term g_j is the input modulation gate, which modulates the input $b^g + x_j U^g + h_{j-1} V^g$ by a hyperbolic tangent function, squashing the input between -1 to 1. The term i_j is the input gate, which applies a sigmoid function to its input, limiting the output values between 0 and 1. The input gate i_j determines which inputs are switched on or off when multiplying the modulated inputs ($g_j \circ i_j$). The term s_j is the internal cell state that provides an internal recurrence loop to learn the sequence dependence. The terms f_j and o_j are the forgot gate and output gate, respectively. They have similar function to the input gate i_j, regulating the information into and out of the LSTM cell. The term h_j is the output at step j. ”

(II) Normally, for ML, the data is split into three sets: (1) a training dataset (2) an evaluation dataset used during training to identify when the training results in overfitting (3) a completely new set of data for testing. ... The authors seem to have only used a validation data set (25% of the total data set) for the testing but no proper testing with parameters outside the training range (so not only "not this specific combination") was performed.

We thank the reviewer for bringing this point on the correctness of our evaluation setup and we would like to clarify that no part of the test data was used in any way during training, thus ensuring the validity of our test results in representing the performance of our ML model on samples outside the training set. Note that in any supervised ML experiment, it is very important to ensure that there is no overlap between the training and test sets, so that the performance on the test set is a true indicator of generalization performance, i.e., the performance on "unseen" instances never seen before during training (also known as out-of-sample instances). This is generally done by holding off a fraction of the overall data during training, thus partitioning the overall data into two sets: a "training set" used only for model building, and a "test set" used only for model evaluation (for further details on evaluating supervised ML models, see Tan et al., 2018 and Fiedman et al., 2001). A common approach for partitioning the overall data into training and test sets is to consider random sampling, also known as the random holdout method [ak1]. In our experiments, we randomly partitioned the overall data into a training set (comprising of 75% instances) and a test set (comprising of 25% instances). Further, an optional procedure that is sometimes followed is to hold a certain fraction of the training set as the "validation set" and monitor the performance on the validation set during training to either avoid overfitting or to tune the hyper-parameters of the ML model. Since the validation set is used during model building (although indirectly) it is no longer considered as a representation of "unseen" instances, and hence, the validation performance is not a true indicator of generalization performance. Note that in our work, we did not make use of any validation set, as the values of all hyper-parameters in our ML model were kept constant across all experiments. Instead, we only report our results on the test set that was not used during training, either directly or indirectly.

We have added the following text to the revised paper to address this comment:

Section 4 (lines 315 to 318): "ML algorithm was trained on 75% randomly selected measurement simulations (1094400 samples) and model performance was tested on the remaining 25%. Note, that no validation data was held off from the 75% training set for tuning hyper-parameters of our ML model, as all ML hyper-parameters were kept constant across all experimental settings in this paper."

Section 5 (lines 378 to 385): "We trained the model on 75% of the dataset for 124 epochs with a batch size of 640. The following choice of hyperparameters was used: choice of optimizer=RMSprop, lr=0.001, rho=0.9, epsilon=None, and decay=0.0. We did not perform any hyper-parameter tuning on a separately held validation set inside the training set, and the values of all hyper-parameters in our ML model were kept constant throughout all experiments in the paper on the test set. In order to ensure that there was no overlap between the training and testing steps, we did not make use of the test data either directly or indirectly during the training phase, either for learning parameter weights or selecting hyper-parameters."

General comments:

(1) There is a lengthy (and maybe not super accurate, see below) description on the aerosol phase function and the asymmetry parameter, both in the introduction and in Sect.4. However, there is no information on ML in the intro. This does not at all reflect the title. After all, this manuscript claims to be about the ML as inversion algorithm. Suggestion: Bundle the aerosol

information from here and from Sect. 4 in the section about training/validation/test data creation (which is currently Sect. 4) and include some paragraph or two on ML use in inverse modelling and general ML.

We've shorten the aerosol part and added general description of ML and detailed description of the ML model. See reply to general comments (1)

(2) I suggest a different ordering: (1) general introduction including advances in ML, aerosol importance in general, current retrieval techniques and why ML should be applied to aerosol retrieval (2) MAXODAS method description (3) Aerosol properties and modelling and forward modelling with VLIDORT (4) Overview of the methodology of the 3 necessary steps (instead of selling it as two steps as done in Sect. 3, where the first of the two has itself two steps) and a detailed description of the specific ML setup and choice of hyperparameters. (5...) as before

See reply to general comments (1)

(3) While what is written about OEM and parametrized methods is true, most of it is true for ML as well (i.e regarding e.g. the T/P profile). This section paints an overly dark image of OEM and parametrized codes. I think that the main problem with "traditional" methods is indeed the time they take, and this should be clearly (even more clearly) stated, since this is the one huge advantage of using ML. Also, especially around line 136, it gives the impression of full profile retrieval of asy and ssa, while in fact, it is "only" the aod profile and single scalar values asy and ssa valid for all layers.

We have reworded this part (*lines 203-206*):

"Aerosol extinction coefficient profiles are inverted while aerosol single scattering albedo and asymmetry factor are typically assumed based on the co-located AERONET measurements. They also require external information about the atmosphere (e.g. temperature and pressure profiles) that might not be readily available at the measurement time scales, and a priori information that does not typically exist."

(4)

(a) Which backend was used? Tensorflow, Theano? Some other? Why the mentioning of the yupiler notebook? Why was it used at all? Certainly no web-based interactivity is needed? Why wasn't it simply put in a plain python script?

TensorFlow backend was used. We mention Jupyter Notebook just because the code is implemented in Jupyter Notebook. Yes, web-based interactivity is not needed and of course we can use plain python script. We used Jupyter notebooks just for easy sharing of code, analysis of results, and reproducibility of experiments.

(b) CNN is normally used in ML for image recognition, why is it used here? Why is LSTM used? Maybe some intro on recurrent neural networks in general is needed. This seems to indicate … that scans are not considered separately, but as a function of time.... (so a scan from now and then from 10 minutes, not one from here and now and the next one from tomorrow and somewhere else). However, this seems not to fit your introduction and abstract where you very specifically write about a single scan. This is very confusing and needs explanation. Also maybe, you can start with explaining what a SimpleRNN layer is and why this was not chosen?

Different from image recognition in which 2D CNN is usually used, what we use in our model is 1D CNN which is good for capturing features from 1D-sequences. We've also added general introduction of LSTM. As mentioned above, we consider the profile as a sequence, that's the reason we use the LSTM. We do not use the LSTM in a typical way where the input or output sequence is a time series. In our case, it is nothing related to time but a series of partial AOD values at sequential heights. Simple RNN is inadequate to capture long-term memory effects where the inputs-outputs at a given element of the sequence can affect the outputs at another element of the sequence separated by a long interval. Actually we've tried simple RNN and it does not work as well as LSTMs.

We have added the following text to the paper in Section 5 (lines 330 to 340) to address this comment:

"Note, that in our supervised ML formulation, there are sequences in both the input signals and output signals, namely $\Delta AMF^{aerosol}$ sequence and partial AOD sequence, respectively. Further note that the input and output signals used in our problem setting are of very different types and thus have different dimensionalities (e.g., $\Delta AMF^{aerosol}$ takes 16 values at varying VZAs while partial AOD takes 23 values at varying atmospheric layer depths). We thus first apply a 1-dimensional CNN to extract features from the sequence part of the input signals. Note that our input signals are not image-based, which is one of the common types of input data for which CNNs are used. Instead, our input data is structured as a 1D sequence, and the convolution operations of CNN help in extracting sequence-based features from the input signals that are then fed into subsequent ANN components. We also use an LSTM to model the sequence part of the output signals. Note that our data contains no time dimension as we are only working with single scan data. However, it is the sequence-based nature of the output signals that motivated us to use LSTM models for sequence-based output prediction. Furthermore, the dataset we use for training is produced by a physical model (VLIDORT), where the relationship between the inputs and outputs are known."

(b) Why was it decided to split for profile and ssa/asy retrieval?

We split profile and SSA/ASY retrieval because we consider the profile as a sequence (the partial AODs at adjacent layers are related) that needs to be modeled using an LSTM, but the SSA/ASY are scalars that can be modeled using Dense layers. We've tried a lot of architectures and find that combining profile and SSA/ASY as a single output sequence results in inferior performance.

We have added the following text in the paper in Section 5 (lines 346 to 357) to address this comment:

"To extract sequence-based features from MAX-DOAS inputs, a 1-dimensional Convolutional Neural Network (CNN, Fukushima, 1980; LeCun et al., 1999) is first applied on the sequence of inputs (we concatenate $\Delta AMF^{aerosol}$ sequence with SZA and RAA to obtain an 18-length input sequence), which results in a sequence of preliminary hidden features. These preliminary hidden features are then sent to two different branches of 1D-CNN layers that perform further compositions of convolution operators to produce

non-linear hidden features for predicting two different types of outputs: (a) scalar outputs: SSA and ASY, and (b) sequence-based outputs: aerosol extinction profile. For the branch corresponding to scalar outputs, the features extracted from 1D-CNN layers are simply passed on to a fully-connected dense layer to produce a two-dimensional output of SSA and ASY. For the branch corresponding to sequence-based outputs, the features extracted from 1D-CNN layers are fed to a Long Short-Term Memory network (LSTM, Hochreiter and Schmidhuber, 1997) to produce a sequence of partial AOD values at varying atmospheric layers."

(c) What were the choices of the hyperparameters? Which batch size was used? Which lr was used for the RMSprop? Where there any drop out layers? Which activation function was used? There is no information on any of the parameters. How many nodes do the layers have?

We've added a plot of the detailed architecture of the ML model in the supplement with all the information.

We have also added the following text in the paper in Section 5 (lines 375 to 380) to provide more details about the hyper-parameters of our model:

"RMSprop was chosen as the optimizer and the mean squared error was used as the loss function (Hinton, 2012). We trained the model on 75% of the dataset for 124 epochs with a batch size of 640. The following choice of hyperparameters was used: choice of optimizer=RMSprop, lr=0.001, rho=0.9, epsilon=None, and decay=0.0."

(5) what happens if the network gets data that is by no means covered by the training data (i.e. completely outside the range in one or more parameters?) What is the effect of measurement noise (also including "noise" from situations that are not 1 dimensional)?

Though the outputs of the test set are not outside the range of the training data, however, the mappings contained in the test set are different. And different combination of SSA/ASY/profile produces different values of radiance. The model hasn't seen these input values and output combinations of SSA/ASY/profile before. As for the point you mentioned here, there are next steps of our work. ML itself is a technique learning from the statistics of the data. If applying on the dataset which is too different from the training set, of course with high probability it cannot provide reliable predictions. The more the ML model 'see', the better it works. Thus, we need to include more realistic aerosol inputs and radiative transfer simulations as mentioned in the 'Conclusion and Future Work' section. We will also consider noise in future work. For this work, our key point is the 'feasibility', which aiming at demonstrating that it is feasible to use ML technique into MAX-DOAS aerosol retrieval.

Specific comments:

(1) page 1, line 23 "... and have relative short lifetimes..." –> relative to what? Also, few minutes to few weeks spans about 5 order of magnitude in time, while one end of this span can be considered as short, the other cannot really. Please specify "relative".

- we replaced "relatively short" with "variable".

(2) page 1, line 26: apart from all the properties already listed, what else do you mean with "physical properties" as opposed to optical? This is very unclear.

- we removed this sentence since it did not add any new information: “The aerosol classification depends on the aerosol source, composition, size and number distribution, aging processes, and optical and physical properties.”

(3) page 1, line 28 "The spatial and temporal distribution of aerosols ... is greatly affected by ... the type of aerosols". I think this is incorrect, the correct verb here is "depends on".

- we replaced “is greatly affected by” for “greatly depends on”

(4) page 2, line 39–40: If you put this statement, then you need to explain more. I also cannot see any connection of this statement to the rest of the paper. The minimum that should be added is how it depends on the surface albedo.

- to shorten the aerosol discussion in introduction we removed line 37 – 63 on p.2.

(5) page 2, line 41–42: "escpecially of anthropogenic origin" "of"? or "for"? This sentence does not make too much sense like it is, reformulation needed.

- to shorten the aerosol discussion in introduction we removed line 37 – 63 on p.2.

(6) page 2, eq2 and eq3: I would think that the range of the asymmetry parameter as such depends on the normalization of the phase function, so you need to have integral(phase function) over 3D angle = 1. If so, then the first moment <cos theta> is the asymmetry btw. forward and backward scattering. So with this, would you not have a factor of 1/4pi missing in the HG phase function? Maybe you could check the normalization factors for consistence btw. g and P.

- to shorten the aerosol discussion in introduction we removed line 37 – 63 on p.2.

(7) page 2, eq. 4: You seem to use tensor notation to make a difference btw. covariant and contravariant tensors and apply Einstein summation convention. However, you still put the summation sign, but without indicating what you are actually summing over.

- to shorten the aerosol discussion in introduction we removed line 37 – 63 on p.2.

(8) page 4, line 101: "approximately known"? Please clarify.

- we added (e.g., temperature and pressure profiles from atmospheric sounding or models)

(9) page 5, around line 136: Since it was highlighted before that

- not sure how to interpret this comment

(10) page 5, line 153..154: both input and output states run to N, one of them should have a different limit, maybe... M? Otherwise it is confusing, especially because it is written that x has 67 layers, but y has "only" 16 angles.

- we replaced y number of elements with M

(11) page 7, line 196: Although VLIDORT has as direct input the viewing zenith angle, most people in the MAXDOAS community are more familiar with the elevation angle. Maybe it is an idea to change this to make it easier to connect to.

- we agree that “elevation angle” is a more familiar term but the MAX-DOAS community is well aware of the zenith angle definition.

(12) page 8, line 199, 201, and other listings in the text of parameters that are summarized in the Table 1: I do not think that they need to be repeated, I think it is enough if they are in the table.

- we replaced the exact listing with the following: “… and nineteen viewing zenith angles between 0 and 89^o (see Table 1). To ensure that the training dataset contains all observation geometries feasible for MAX-DOAS sky scans we have included: nineteen relative azimuth angles (0 to 180^o, 10^o step), and twelve solar zenith angles (0 to 85^o, 89^o see Table 1).”

(13) page 8, table1: Can you comment on how the direct sun cases for raa=0, sza=vza are handled?

- it is not handled in any special way. We do recognize that no meaningful profile information is available from such geometries and the forward scattering has large uncertainties.

(14) page 9, line 223: why do you need ozone absorption?

- strictly speaking we do not need ozone absorption, but since there is no harm in its presence we left it in.

(15) page 9, line 230: maybe a small sketch to explain the aerosol profile parameters (with the two components of the profile) would be helpful

- we added: “Figure 11 demonstrates the aerosol profile samples, where the near surface aerosol partial optical depth profiles are described by the exponential function and the layers aloft are described by the Gaussian function with various widths and heights added to the exponential function profile.”

(16) page 9, line 237: The 25% were fixed between the 20 realizations, or not? It would be really good to see some plots here of the evaluation loss as a function of epoch. Also, please comment on how over-fitting was mitigated.

- we did not perform the hyper-parameter optimization in a formal way, so no cross evaluation was done. However, we did monitor training loss and it converged. To eliminate the confusion, we have replaced “evaluated” with “performance tested”.

(17) page 9, line 236: this height is the middle height or the height of the upper boundary? This is not clear.

- we replaced layer “heights” with “depths”

(18) page 9, line 247ff: I would certainly describe the architecture of the network here, not only the Fig. 3. Also, dense layers are not explained. Also, how many nodes in the layers? Do you use maxpooling layers btw. your conv1d layers? What is the size of your convolution window? And again, how was the architecure chosen? Why does it make sense to separate the SSA and the ASY the way you do? Do you extract the SZA and RAA as well? They should certainly be == the input? Is there a test on this?

- We've added a plot of model architecture in the supplement.

(19) page 9, line 259: While you do use 25% for test (or do you actually use this for evaluation? Not really, because you use it to test the network. What was used for the evaluation then?)

- Hyper-parameter selection was done offline. We have not seen a significant difference between the hyper-parameter choices for the selected architecture and did not include the cross-evaluation at all during the training so 25% of the data that the model NEVER saw are used for testing the performance of the model.

Because you use the same type of parametrization, this is not a good test. A different, unseen set of data should be used.

- The model never saw the test data so the test is valid

How do features that were not included in the training dataset at all (by all means outside the parameter range) affect the result?

- Ideally, the final MAX-DOAS inversion algorithm based on ML would "see" most of the possible ranges, for example, profiles from the LIVAS database, but optical properties varied across all aerosol types for the same profile so the solution is more reliable.

What about thin cloud layers above 4 km, do they affect the result?

- Friess et al 2018 used NASA real-time aerosol retrieval algorithm that is the basis for the dAMF (-dAMF= AMF – $AMF_{Rayleigh}$) analysis in this study. It was shown that the method is not sensitive to the aerosol/cloud layers above 4 km. We assume the same applies to the ML-based algorithm.

The tests included here are not very useful.

- We disagree with this statement. Most studies evaluating the performance of MAX-DOAS algorithms (e.g. Fiess et al, 2018) have significantly simpler and smaller data sets from both profile variability, observation geometry and optical properties that were tested in this study.

(20) page 10, line 275 & 285: given the range of parameters, using eq. 270, the maximum error is about 20%, not 100%. This puts these low numbers in context.

- This is assuming that the ML-based algorithm always retrieves the ranges of the training data set. The fact that the ranges were within the realistic roam of aerosol profiles and properties is not a weakness, but a goal for a robust inversion ML algorithm.

(21) page 11, Fig. 5: When you wrote earlier that the mean error is "-0.14", you really just took the mean over all angles? What is the significance of this? If it were in have the parameter space +50% and in the other -50%, its mean is still 0... and the model really not good, so what is the significance of the mean error here?

- We agree, that as a stand-alone mean error over all observation geometries and all aerosol scenarios is somewhat meaningless unless the goal is to detect any systematic

biases. That is why we also show 2 standard deviation results and dependency on observation geometry.

(22) page 12, Fig. 7: please explain the box whisker plot. Is the line mean or median? The box is how much percentage? The whisker? There are different conventions...

- we added the following text to the Figure 5 caption: "The central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the '+' symbol."

(23) page16, line351: This paper does not present the ML-based algorithm. It presents some of the results and that's it. There is not enough information on the ML model. This sentence is not summarizing the paper.

- we have made modifications to expend the discussion of the ML-based algorithm

(24) page 16, line 363: maybe this is because of the choice of training parameters as a linear distributed AOD?

- while the AOD itself is linearly distributed the dAMF used in training is not. The more realistic reason for the lower accuracy for low AOD is probably a smaller signal in dAMF.

Technical corrections and suggestions:

(1) Many times, there are definite articles missing (e.g. page 3 line 65 "The MAXDOAS..", page 3 line 84: "The DOAS technique", page 4 line 91 "The offset term...")

- thank you for pointing this out

(2) Eq. 5 on page 4 is not referred to in the text.

- we added a reference to (Eq. 5) on line 90.

(3) page7, line204: I highly doubt that Clemer et al 2010 is the only code here. I would add a "e.g.".

- added

(4) page 9, line 229: I think you miss the AOD=0 case in this list

- we assumed that the algorithm will retrieve the properties perfectly in the absence of noise and AOD = 0 (dAMF = 0) and did not want to skew the data. However, we will include very small AOD in the next version of the model with the real data.

(5) page 10ff: I suggest to use an equi-distant grid for the raa-sza plots, as they are now, it gives a biased impression to the eye.

- we replaced Fig 4.

(6) page 12, line 311 f: I cannot quite understand the sentence "This error contribution..." maybe you can reformulate

- the phrase was replaced with “Layer partial AOD retrieval error relative to the total AOD”

References

P. Tan, M. Steinbach, A. Karpatne, and V. Kumar “Introduction to Data Mining (2nd Ed.),” Pearson Addison–Wesley, ISBN-13: 978-0133128901, 2018.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1, no. 10. New York: Springer series in statistics, 2001.