We sincerely thank both reviewers for their detailed comments and feedback. We have implemented several changes to the manuscript in response to this feedback and believe it is improved over the original submission. The primary change is the expansion of the discussion section with some consideration of computational costs. We have also fixed several small issues throughout the paper and added detail and clarification in places in response to Reviewer 2. Finally, we have added more comments and a detailed readme file to the NeuralMie and TAMie code explaining how to use it. The code is now published at: https://github.com/pnnl/NEURALMIE and has been archived on Zenodo.

Reviewer #1 (Peter Ukkonen):

We agree that the computational cost of the new parameterization is a major concern that should be addressed in the paper (and will impact the parameterization's adoption). We have benchmarked the Fortran versions of the neural networks and found that calls to the homogeneous sphere model take about 3µs on a single core and calls to the core-shell model take 6.3µs (on the same hardware the Mie code was tested on). We have also now run two short test simulations with E3SM and a control simulation using the neural network to perform the aerosol optics calculations and have added new discussion to the paper (lines 510-529) regarding computational requirements. In short, we found that the neural networks can be used in E3SM simulations (the simulations run stably and have plausible aerosol radiative effects), but the homogeneous sphere model increased runtime by about 2x while the core-shell model increased runtime by about 4x. This is fast enough to perform single, aerosol-focused experiments perhaps, but too slow to serve as the default optics parameterization. We are exploring several options for accelerating the neural network solution (with 1 and 2 below suggested by the reviewer) and will document the results in a separate manuscript documenting the implementation and evaluation of the emulator in E3SM:

1) Batched inference. The neural network needs to be called once for each wavelength and aerosol mode combination, which amounts to 120 times per grid-cell and level for MAM4, so there is potential to increase efficiency here.

2) Predicting all wavelengths in a single inference step. The neural network can be re-configured to output values for each wavelength band used in E3SM. If the size of the ANN does not increase substantially that would result in a large speedup. We would like to avoid this type of solution because it is less generalizable and would be specific to the wavelengths used in E3SM (though these are shared by several other ESMs at least). Another difficulty with this solution is that the refractive index is wavelength dependent.

3) Reducing model size: we chose model configurations based on Figure 2 in the paper, and reducing model size increases error, but the relationship is non-linear. Reducing the model size by a factor of 5 increases error by a factor of 2 and reducing model size by a factor of 10 increases error by a factor of 4. Because the MAE of the original neural networks is so low it may be acceptable to use a smaller model. We may also be able to reduce model size through pruning or using a different model architecture.

4) Using multiple smaller neural networks. Currently we use the same emulator for all wavelengths. In the longwave, only absorption needs to be calculated, however. Also, some of our experiments have indicated that the specific extinction can be calculated accurately with a smaller network if it is handled by a separate network from the single scattering albedo and asymmetry parameter.

5) Only calling the neural network in cases where there is a significant amount of aerosol: currently the model is run for every grid-cell, level, time-step, wavelength-band, and aerosol mode (when Rayleigh or geometric approximations cannot be used). This means it is called at locations and times with very low aerosol loading and, in these cases, we could fall back to the existing parameterization.

Also, Reviewer 1's comment noted that we indicate there are 32 wavelength bands used in EAM in the paper. This was a mistake and there are actually 30. The manuscript has been updated to reflect this.

Reviewer #2:

We have made several changes to the manuscript in response to Reviewer 2's feedback:

Reviewer 2 noted that a README and LICENSE file were not provided with the code. The code is now publicly available from https://github.com/pnnl/NEURALMIE under a BSD-2 license. We have added a readme that describes how to use both TAMie and NeuralMie. There are also scripts that provide examples of how to load and inference the trained neural networks (with appropriate input and output scaling) and how to use the Rayleigh approximation code for small particles. We have also released the Zenodo repositories originally provided privately to the reviewers to the public. This information has been added to the code/data availability section of the paper.

We have clarified statements about the accuracy of NeuralMie, to make it clear that it is more accurate than the parameterization currently used in E3SM but not more accurate than Mie code. Also, we have clarified the use of the term "negligible" in the abstract, and have either removed or clarified such modifiers elsewhere in the paper.

We have also made some slight changes to wording and ordering in the abstract and introduction to make it clear that the main novel scientific contribution is NeuralMie while TAMie was a part of its development and is based on existing algorithms. We have included the details about TAMie's development and testing in the paper and are releasing the code because we think this software fills a gap in the currently available Mie scattering libraries and others that wish to do fast Mie scattering calculations in Python will find the code useful.

Below are responses to some of Reviewer 2's specific comments:

[Line 22-23] "Recently, machine learning (ML), and neural networks in particular, have emerged as powerful tools for developing new, more accurate, faster, and more capable, physics parameterizations for atmosphere modeling." This is a strong statement and probably a large number of modelers would say that ML methods could provide a more accurate or more capable parametrization but not always. Also the statement says everything at the same time: accurate, faster and capable. Is this really so general?

We have edited this statement in the paper to be a bit more conservative. We certainly do not mean to imply that all parameterizations can, or should, be replaced by ML, just that ML has potential to replace some with solutions that are more accurate, faster, **and/or** (changed from "and") more capable. In this case, NeuralMie is more accurate and more capable, but slower than the existing parameterization for example.

[Line 24-26] "Conventional parameterizations typically take the form of simplified hand-derived physical models, basic statistical models like lookup tables and linear regressions, and sometimes simply rely on expert heuristics to make decisions about the behaviour of a system." Again I don't think that this is general, and for sure there is not a full agreement in the research community about this statement. A key set of physical parametrization are derived from fundamental physics and chemistry theories. Just two examples (but there are more): many relations of (cloud) microphysics are derived from Molecular Physics and Thermodynamics, also radiative transfer parametrization are not basic statistical models, lookup tables, linear regressions, expert heuristics or simplified hand-derived physical models (despite specific assumptions/datasets used in current radiative transfer schemes could be improved using ML methods)

We have expanded this statement to be broader and open-ended, and note that ML should be seen as a potentially powerful addition to the existing set of methods.

[Line 29-31] Do the authors think that everything is positive about the use of ML emulators here, or there are some trade-off to consider that could be worth to mention?

There are certainly tradeoffs, and we added a sentence to this paragraph to make this clear. We do not want to provide a full audit of the pros and cons of using neural networks for parameterizations and physics emulation here though, so as not to overburden the introduction. We have also added a more detailed discussion of the computational expense of NeuralMie to the Discussion section (lines 510-529) which is a tradeoff for this particular ML solution.

[Line 41-43] Just note that several climate models have aerosols that are not coated aerosols but an internal mixture of an aerosols with inclusions of other species. They usually redefine the refractive index using Maxwell Garnett mixing rule [2]

We added a note here (line 43)

[Line 70-75] Here, I would be cautious, although the authors' evaluations are reasonable. First, the comparison has been done with two specific Fortran Mie codes. Numba probably is correctly setting the best performance for the LVVM tool chain for the TAMie.py code, but this comparison would need to be sure that the optimization flags of the compiler are optimal for the comparison. The performance improvement of TAMie.py with respect to PyMieScatt code is remarkable even without the use of Numba. The general value of PyMieScatt more than speed is the flexibility of their API and very good documentation, two aspects that are not present in TAMie code.

Information about the Fortran compiler used for the comparison of TAMie to Fortran codes was added on line 180.

We have also added comments to Section 2.2 noting some of the drawbacks of TAMie and limitations of the comparison, specifically:

- Other, more modern, compiled Mie scattering codes exist.
- Faster run times could likely be achieved by the Fortran codes using appropriate compiler optimization flags.
- While TAMie is substantially faster than PyMieScatt, PyMieScatt has many more features.

Equation (20) and the discussion in lines [238-240] There are modal aerosols where include internally and externally mixture of aerosols. If this something considered when the NeuralMie was developed?

NeuralMie is flexible and can handle internal or external mixing assumptions (or a combination) by doing multiple inferences. For instance, to run NeuralMie with MAM4, which assumes internal mixing within four separate modes, the model will be inferenced once per mode. For fully external mixing it would be inferenced once per aerosol species. We added a note on MAM4's mixing assumptions on line 42, and have added a comment about how NeuralMie handles this on line 323.

Figure 2. These are interesting results to understand how increase the complexity of the model it not always improving the accuracy. Still in the case of the core-shell it seems to have an asymtopic decay. In this figure if the two branches have an structural difference they can be represented with different colors. If I understand well this figure is specific of ke because the Table 6 gives for it the higher errors. However, it the authors have results analogous to Figure 2 for the other parameters ω and $\bar{g}$ are they similar?

The main purpose of this figure is to illustrate the asymptotic behavior and how we chose a model architecture based on these results. We would prefer not to add complexity by coloring different layer counts (we tested 2, 3, and 4 layer networks in this work), but included this comment in the caption to explain why this separate group of models appears in the plot since we thought readers might find this curious. In any case, rather than adding more detail to this plot we added a reference to our previous work on this topic that involved a much more detailed analysis of model architectures and contains a similar plot with models separated out by number of layers, and that is based on error across all three output variables.