






Parameterizing pressure–temperature profiles of exoplanet atmospheres with neural networks[★]

Timothy D. Gebhard^{1,2,3} , Daniel Angerhausen³ , Björn S. Konrad³ , Eleonora Alei³ ,
Sascha P. Quanz³ , and Bernhard Schölkopf^{1,4}

¹ Max Planck Institute for Intelligent Systems, Max-Planck-Ring 4, 72076 Tübingen, Germany
e-mail: tgebhard@tue.mpg.de

² Max Planck ETH Center for Learning Systems, Max-Planck-Ring 4, 72076 Tübingen, Germany

³ ETH Zurich, Institute for Particle Physics & Astrophysics, Wolfgang-Pauli-Str. 27, 8092 Zurich, Switzerland

⁴ Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland

Received 12 March 2023 / Accepted 7 August 2023

ABSTRACT

Context. Atmospheric retrievals (AR) of exoplanets typically rely on a combination of a Bayesian inference technique and a forward simulator to estimate atmospheric properties from an observed spectrum. A key component in simulating spectra is the pressure–temperature (PT) profile, which describes the thermal structure of the atmosphere. Current AR pipelines commonly use ad hoc fitting functions here that limit the retrieved PT profiles to simple approximations, but still use a relatively large number of parameters.

Aims. In this work, we introduce a conceptually new, data-driven parameterization scheme for physically consistent PT profiles that does not require explicit assumptions about the functional form of the PT profiles and uses fewer parameters than existing methods.

Methods. Our approach consists of a latent variable model (based on a neural network) that learns a distribution over functions (PT profiles). Each profile is represented by a low-dimensional vector that can be used to condition a decoder network that maps P to T .

Results. When training and evaluating our method on two publicly available datasets of self-consistent PT profiles, we find that our method achieves, on average, better fit quality than existing baseline methods, despite using fewer parameters. In an AR based on existing literature, our model (using two parameters) produces a tighter, more accurate posterior for the PT profile than the five-parameter polynomial baseline, while also speeding up the retrieval by more than a factor of three.

Conclusions. By providing parametric access to physically consistent PT profiles, and by reducing the number of parameters required to describe a PT profile (thereby reducing computational cost or freeing resources for additional parameters of interest), our method can help improve AR and thus our understanding of exoplanet atmospheres and their habitability.

Key words. methods: data analysis – methods: statistical – planets and satellites: atmospheres

1. Introduction

With now over 5000 confirmed planet detections outside the solar system (see, e.g., [Christiansen 2022](#)), the exoplanet science community is increasingly expanding from detecting new planets to also characterizing them. One important tool for this are atmospheric retrievals (AR), that is, “the inference of atmospheric properties of an exoplanet given an observed spectrum” ([Madhusudhan 2018](#)). The properties of interest here include, for example, the chemical composition of the atmosphere (i.e., abundances of different chemical species) or the presence of clouds or haze. Deriving these properties from a spectrum is a classic example of an inverse problem. As with many such problems, the standard approach is to combine Bayesian inference methods, such as nested sampling ([Skilling 2006](#)), with a simulator for the “forward” direction – in our case, turning a set of atmospheric parameters θ into the corresponding spectrum. Given the spectrum y of a planet of interest, one can then estimate a posterior distribution $p(\theta|y)$ by iteratively drawing a sample of parameter

values from a prior $\pi(\theta)$, passing them to a simulator, and comparing the simulated spectrum \hat{y} to the observed one through a likelihood function L to guide the next parameter sample that is drawn. In practice, the simulator is often deterministic, in which case L is essentially equivalent to the noise model: For example, optimizing θ to minimize $\|y - \hat{y}(\theta)\|^2$ implies the assumption that the noise in y comes from an uncorrelated Gaussian.

Traditionally, due to computational limitations, much of the existing work on atmospheric retrievals has made the simplifying assumption of treating atmospheres as one-dimensional. More recent studies, however, have also begun to explore 2D and 3D approaches (see, e.g., [Chubb & Min 2022](#); [Nixon & Madhusudhan 2022](#); [Zingales et al. 2022](#)).

Pressure-temperature profiles. One of the key factors that determines the observable spectrum of a planet’s atmosphere is its thermal structure, that is, the temperature of each atmospheric layer. The thermal structure is described by the pressure–temperature profile, or PT profile, which is a function $f: \mathbb{R}^+ \rightarrow \mathbb{R}^+$ that maps an atmospheric pressure to its corresponding temperature. In reality, of course, this function also depends on the location (e.g., in the case of the Earth, the air over the equator is warmer than over the poles), and such variations can be accounted for when using general circulation models (GCMs).

[★] All code for our method and experiments is available at: <https://github.com/timothygebhard/ml4ptp>. Our datasets and the final trained models are available here: <https://doi.org/10.17617/3.K2CY3M>.

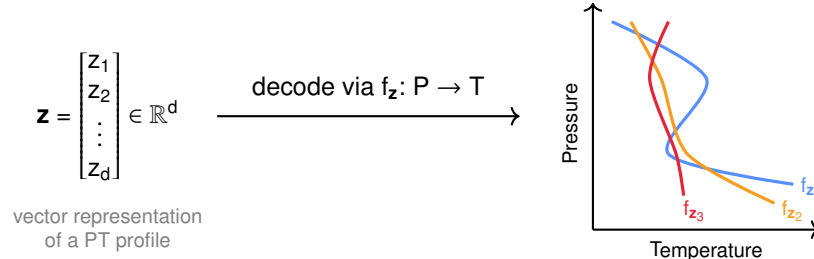


Fig. 1. Schematic illustration of the problem setting considered in this paper: “Parameterizing PT profiles” means finding a way to represent PT profiles as vectors $\mathbf{z} \in \mathbb{R}^d$, together with a decoding mechanism $f_{\mathbf{z}} : P \rightarrow T$ that converts these \mathbf{z} -vectors back into a function that maps pressure values onto temperature values. Our goal in this work is to learn both the representations \mathbf{z} and the mechanism $f_{\mathbf{z}}$ from data obtained with simulations using full climate models.

However, as mentioned above, it is common to limit oneself to one-dimensional approximations. The implications of such an approximation are discussed, for example, in Blečić et al. (2017).

There are, in principle, two different ways to determine the PT profile during an AR: One may either solve a set of (simplified) radiative transfer equations, or approximate the PT profile through an ad hoc fitting function (see, e.g., Seager 2010; Line et al. 2013; Heng 2017). In the latter case, which is the one that we consider in this work, one has to choose a parameterization for the PT profile, that is, one needs to assume a parametric family of functions $f_{\mathbf{z}} : P \rightarrow T$ which map pressure values onto corresponding temperature values and have tunable parameters $\mathbf{z} \in \mathbb{R}^d$ that control how exactly this mapping looks like (see Fig. 1 for an illustration). These parameters \mathbf{z} are a subset of the full set of parameters θ for which the retrieval procedure estimates a posterior distribution (see above). Throughout this work, we commonly refer to \mathbf{z} as a (latent) representation of a PT profile. The choice of $f_{\mathbf{z}}$ will, in general, be determined by a trade-off between different (and possibly conflicting) requirements and desiderata, including the following:

- For every profile that we can expect to find during the retrieval, there should exist a \mathbf{z} such that $f_{\mathbf{z}}$ is a good approximation of this profile (subjectivity).
- For every value of \mathbf{z} from some given domain (e.g., a subset of \mathbb{R}^d), $f_{\mathbf{z}}$ is a valid PT profile. Together with subjectivity, this implies that the image of $\mathbf{z} \mapsto f_{\mathbf{z}}$ is equal to the set of physically sensible (and relevant) PT profiles.
- The mapping $\mathbf{z} \mapsto f_{\mathbf{z}}$ should be smooth in the sense that small changes in \mathbf{z} should only result in small changes in $f_{\mathbf{z}}$ (i.e., continuity); otherwise, the retrieval loop may not converge.
- The dimensionality $d = \dim(\mathbf{z})$ should be as small as possible, since methods such as nested sampling often do not scale favorably in the number of retrieval parameters.
- We should be able to define a simple prior $p(\mathbf{z})$ for \mathbf{z} , and for $\mathbf{z} \sim p(\mathbf{z})$, the induced distribution over functions $f_{\mathbf{z}, \mathbf{z} \sim p(\mathbf{z})}$ should constitute an appropriate prior distribution over the PT profiles of the population of planets that we are considering.
- Ideally, the dimensions of \mathbf{z} are interpretable as physical quantities, or correspond to physical processes (e.g., “ z_1 is the mean temperature,” or “ z_2 controls the presence of an inversion”).

Of course, as suggested above, no choice of parameterization will perfectly satisfy all requirements simultaneously, and some sort of compromise is required.

Related work. Barstow & Heng (2020), who have identified the parameterization of PT profiles as one of the “outstanding challenges of exoplanet atmospheric retrievals,” find the approaches of Madhusudhan & Seager (2009) and Guillot (2010)

to be the most popular among the atmospheric retrieval community. These parameterizations, as well as the ones proposed by Hubeny et al. (2003), Hansen (2008), Heng et al. (2011), and Robinson & Catling (2012), all consist of (semi)-analytic expressions that seek to describe the relationship between pressure and temperature mainly in terms of physically interpretable quantities, such as the opacity or optical depth. Simpler (yet less physically motivated) approaches include the use of splines (e.g., Zhang et al. 2021) as well as low-order polynomials (e.g., Konrad et al. 2022). Finally, a data-driven approach was presented by Schreier et al. (2020), who compute a singular value decomposition of a representative set of PT profiles, and parameterize new profiles as a combination of the first k singular vectors. All these approaches typically use four to six parameters to describe a PT profile, which is often a substantial fraction of the total number of retrieved parameters. To the best of our knowledge, no studies to date have used machine learning (especially neural networks) to parameterize pressure–temperature profiles of exoplanet atmospheres.

Contributions. In this work, we explore a conceptually new approach to parameterizing PT profiles that is based on the idea of using neural networks to learn efficient (i.e., low-dimensional) representations of PT profiles, as well as the corresponding decoding mechanisms, from simulated data¹. Using two different datasets of PT profiles obtained with full climate models, we show that our approach can achieve better fit quality with fewer parameters than our two baseline methods, while still easily integrating into an existing atmospheric retrieval framework.

2. Method

Our goal is the following: We want to learn a model (in our case: a neural network) that takes two inputs: (1) a pressure value p , and (2) a vector $\mathbf{z} \in \mathbb{R}^d$, also called “representation”, which contains a highly compressed description of a PT profile. The output of the model is the temperature t at pressure p , for the profile described by \mathbf{z} . During an atmospheric retrieval, the Bayesian inference routine can then propose values of \mathbf{z} , which can be converted to a PT profile by conditioning the network on the proposed \mathbf{z} (that is, fixing the input \mathbf{z}) and evaluating it at different pressures p_i to get the corresponding temperatures t_i . Thus, we will infer a posterior distribution over \mathbf{z} , which corresponds directly to a posterior over PT profiles.

¹ We presented early versions of this work at AbSciCon 2022 and the “ML and the Physical Sciences” workshop at NeurIPS 2022.

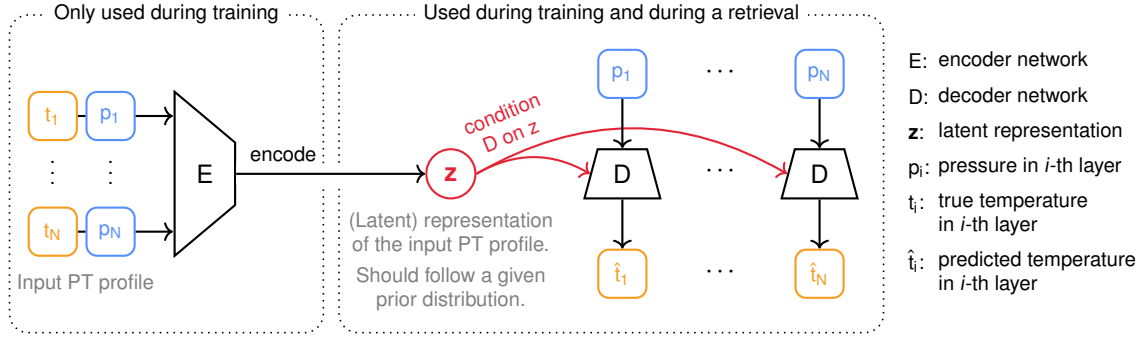


Fig. 2. Schematic illustration of our model: During training, the encoder network E maps a PT profile (consisting of a vector $\mathbf{p} = (p_1, \dots, p_N)$ of pressure values and a vector $\mathbf{t} = (t_1, \dots, t_N)$ of corresponding temperature values) onto a latent representation $\mathbf{z} \in \mathbb{R}^d$. This latent code is then used to condition a decoder network D , and $D(\cdot | \mathbf{z})$ is evaluated on each p_i to get the corresponding predicted temperatures \hat{t}_i . During an atmospheric retrieval, \mathbf{z} is proposed by the Bayesian inference method (e.g., nested sampling).

To learn such a network, we assume that we have a dataset \mathcal{D} of PT profiles which were generated by a full climate model or radiative transfer code to ensure they are physically consistent. These models typically simulate an atmosphere in a layer-by-layer fashion, meaning that each PT profile is given in the form of a set support points, $\{(p_i, t_i)\}_{i=1, \dots, N}$, where N is the number of layers, and p_i and t_i are the pressure and temperature in the i -th layer. Importantly, our method does not require that p_i be the same for all profiles in \mathcal{D} : Different profiles can use different pressure grids, and this is indeed the case for the datasets we work with. Moreover, in principle, even the number of layers N can vary between profiles. (We do not explicitly consider this case here, but it implies only minor technical complications at training time.) This flexibility also makes it easy, for example, to combine PT profiles from different models or pre-existing atmosphere grids into a single training dataset.

The approach that we propose in the following is inspired by the idea of a (conditional) neural process (NP; Garnelo et al. 2018a,b). A neural process is a type of machine learning model that learns a distribution over functions and combines the advantages of Gaussian processes with the flexibility and scalability of neural networks. In astrophysics, the usage of (conditional) neural processes is still relatively rare; examples include Park & Choi (2021), Čvorović-Hajdinjak et al. (2022), and Jankov et al. (2022), all of which use conditional NPs for interpolation.

Our method works as follows (see also Fig. 2). We employ two neural networks: an encoder E and a decoder D . The encoder is only required during training and will not be used during an AR. At training time, we take a PT profile – consisting of a vector $\mathbf{p} = (p_1, \dots, p_N)$ of pressure values and a vector $\mathbf{t} = (t_1, \dots, t_N)$ of corresponding temperatures – from our training data set and pass it to E , which outputs a representation $\mathbf{z} \in \mathbb{R}^d$ of the profile:

$$\mathbf{z} = E(\mathbf{p}, \mathbf{t}). \quad (1)$$

The dimensionality d of \mathbf{z} can be chosen freely (but changing d will require re-training E and D). Generally, we want to keep d as low as possible (e.g., $d = 2$), but of course, higher values also imply more expressive or informative representations.

In the next step, the representation \mathbf{z} is then used to condition the decoder network D . Once conditioned on \mathbf{z} , the decoder network is a PT profile, that is, a function that takes in a single value (a pressure) and outputs a single value (a temperature). We then pass all pressure values p_i of our input profile through D to get the respective predicted temperature values \hat{t}_i :

$$\hat{t}_i = D(p_i | \mathbf{z}), \quad (2)$$

where we use $D(\cdot | \mathbf{z}) : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ to denote the function that is given by conditioning D on \mathbf{z} (i.e., fixing \mathbf{z} as an input). We train the encoder and decoder jointly (i.e., we update the weights of the neural networks) to minimize the difference between the true and predicted temperatures. More specifically, we minimize the following mean squared error (MSE) reconstruction loss, which is a standard choice in machine learning for regression problems:

$$\mathcal{L}_{\text{rec}}(\mathbf{t}, \hat{\mathbf{t}}) = \frac{1}{N} \sum_{i=1}^N (t_i - \hat{t}_i)^2. \quad (3)$$

In practice, we do this not only for a single PT profile, but minimize the average of \mathcal{L}_{rec} over our entire training dataset in a batched fashion. We also note that this choice of reconstruction loss assigns the same weight to all N atmospheric layers, and refer to Sect. 5.4 for a brief discussion of possible alternatives.

To ensure that we can define a simple prior for \mathbf{z} during a retrieval, we furthermore want to constrain the distribution of the \mathbf{z} 's produced by the encoder. There are various potential approaches for this (see Appendix A). For this work, we adopt an idea from Zhao et al. (2017) and add a second term to the loss function that encourages \mathbf{z} to follow a standard Gaussian:

$$\mathcal{L}_{\text{MMD}} = \text{MMD}^2(\{\mathbf{z}_i\}_{i=1, \dots, b}, \{\mathbf{s}_i\}_{i=1, \dots, b}), \quad (4)$$

where:

$$\mathbf{s}_i \sim \mathcal{N}_d(0, 1), \quad i = 1, \dots, b.$$

Here, $\mathcal{N}_d(0, 1)$ denotes a d -dimensional standard Gaussian distribution (i.e., mean zero and identity matrix as covariance matrix). Furthermore, b is the batch size of our training, and MMD stands for Maximum Mean Discrepancy (Borgwardt et al. 2006; Gretton et al. 2012), which is a kernel-based metric that measures whether two samples come from the same distribution (see Appendix A.2 for more details). By minimizing this MMD term, we encourage the model to produce values of \mathbf{z} that follow the same distribution as the \mathbf{s}_i , that is, a d -dimensional standard Gaussian.

Experimentally, we found that despite this MMD term, there are sometimes a few profiles that are mapped to a \mathbf{z} very far from the origin, which is undesirable if we want to define a simple prior for \mathbf{z} during a retrieval. To prevent this problem, we introduce a third loss term with a softplus barrier function on the norm of \mathbf{z} :

$$\mathcal{L}_{\text{norm}}(\mathbf{z}) = 1/k \cdot \log(1 + \exp\{k \cdot (\|\mathbf{z}\|_2 - \tau)\}). \quad (5)$$

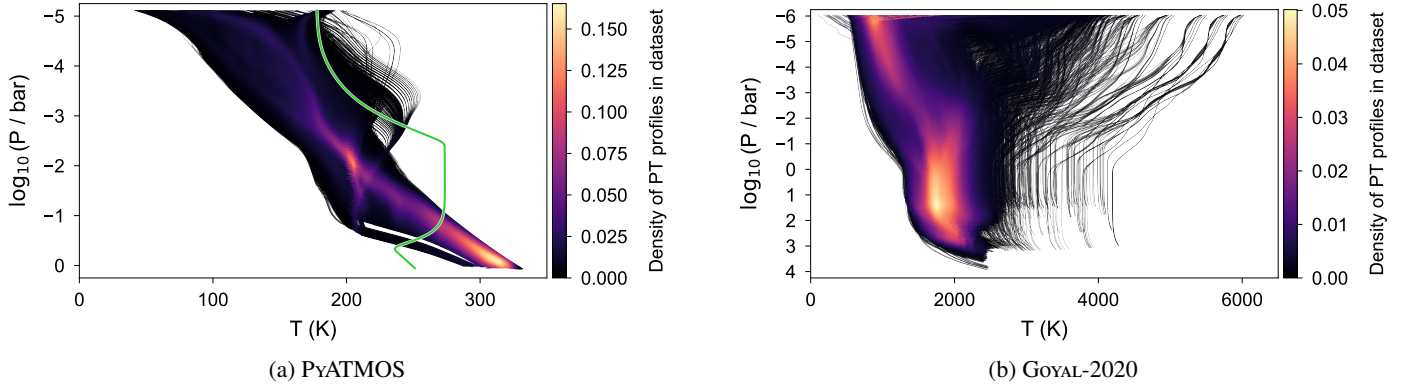


Fig. 3. All the PT profiles in our two datasets. Each segment of a line – that is, the connection between the points (p_i, t_i) and (p_{i+1}, t_{i+1}) – is color-coded by the density of the profiles at its respective (p, t) coordinate, which was obtained through a 2D KDE. The green line in the PYATMOS plot shows the one PT profile that we manually removed for being out-of-distribution (<https://github.com/timothygebhard/ml4ptp/blob/main/scripts/plotting/fig-3-datasets/plot-dataset.py>).

This is a smooth version of the $\text{ReLU}(x) := \max(0, x)$ function, where k controls the amount of smoothing: Larger values of k increase the similarity to ReLU. For this work, we have fixed k at 100 (ad hoc choice), after observing that using a standard ReLU function destabilized the training for us. The parameter τ defines the threshold on $\|z\|_2$ and was set to $\tau = 3.5$ (ad hoc choice).

The total loss that we minimize is then just a weighted sum (with $\beta, \gamma \in \mathbb{R}^+$) of our three loss terms:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \beta \cdot \mathcal{L}_{\text{MMD}} + \gamma \cdot \mathcal{L}_{\text{norm}}. \quad (6)$$

The hyperparameters β and γ control the tradeoff between the different loss terms: (1) Increasing β corresponds to placing a stronger prior on the distribution of z in the latent space, usually at the cost of a higher reconstruction error. We have tried different values of β and found that the performance is relatively similar over a range of about 0.1 to 1. Too high values of β can lead to model collapse, where the prior on z becomes too strong and prevents the encoder from learning meaningful representations. For this work, we set $\beta = 0.1$. This was an ad hoc choice and not the result of systematic optimization, which would likely yield a different value for each dataset and value of $\dim(z)$. (2) The value of γ should be chosen such that $\gamma \cdot \mathcal{L}_{\text{norm}}$ dominates the other loss terms when $\|z\|_2 > t$. We set $\gamma = 10$ for this work (again without optimization).

To use our trained models during atmospheric retrieval, we provide a convenient Python wrapper that allows the model to be loaded as a normal function, taking as input a value for z and a numpy array of pressure values, and returning an array of corresponding temperatures. We note again that neither the lengths of the pressure grids nor the exact values need to match the pressure grid(s) used during training. (However, the range of pressure values should be approximately the same; otherwise, evaluating $D(\cdot|z)$ at pressure values it has never encountered during training may result in nonphysical outputs).

3. Datasets

We validate our proposed approach using two different, publicly available datasets of PT profiles: one for terrestrial planets, and one for hot Jupiters. A visual comparison of the two datasets and the respective P - T space that they cover is shown in Fig. 3.

3.1. PYATMOS

The PYATMOS dataset was generated by Chopra et al. (2023) and consists of 124 314 stable atmospheres of Earth-like planets around solar-type stars generated with Atmos, a 1D coupled photochemistry-climate model (Arney et al. 2016; Meadows et al. 2018). The atmospheres in this dataset were simulated using $N = 101$ layers covering an altitude range of 0–80 km. Consequently, each PT profile is given by two 101-dimensional vectors for the pressures and temperatures. In addition, about 80 other atmospheric quantities are available, such as fluxes and abundance profiles of chemical species. We manually removed one PT profile from the dataset because it was completely out-of-distribution and caused stability problems during training.

Furthermore, it has been brought to our attention that the simulations of the PYATMOS dataset suffered from a couple of issues that affect the physical correctness of the simulated PT profiles. In particular, the `climaMain.f2` specified a H_2O k -coefficient file that assumed pressure broadening by H_2O itself, which is not appropriate for a N_2 -dominated atmosphere. Furthermore, the model version also did not allow for convection above 40 km. Consequently, the simulated PT profiles tend to underpredict stratospheric and mesospheric temperatures compared to the expectation from a real Earth twin. Newer versions of the Atmos code are not affected by these issues and use updated H_2O and CO_2 k -coefficients (Teal et al. 2022; Vidaurri et al. 2022). While this may limit the scientific usefulness of the PT profiles, it should not affect our ability to use them to demonstrate the applicability of our method as such, that is, to show how we can learn efficient, low-dimensional parameterizations of PT profiles from simulated data.

3.2. GOYAL-2020

This dataset was published by Goyal et al. (2020) and consists of 11 293 PT profiles, corresponding to 89 hot Jupiters, each with four recirculation factors, six metallicities and six C/O ratios³. Due to missing data, we manually removed three profiles. The data were simulated with Atmo (Amundsen et al. 2014; Tremblin et al. 2015, 2016; Drummond et al. 2016)⁴, a radiative-convective

² https://github.com/PyAtmos/atmos_fdl/blob/b2294838aeadd04d306f747830e46c0ec577db9a/ClimaMain.f

³ The authors report that around 12 % of the models did not converge; hence the dataset is smaller than the expected 12 816 combinations.

⁴ We point out that Atmos and Atmo are not the same code.

equilibrium model for atmospheres, using $N = 51$ atmospheric layers. In addition to the PT profiles, the dataset also includes transmission and emission spectra for each atmosphere, as well as over 250 abundance profiles of different chemical species.

4. Experiments and results

In this section, we describe the experiments that we have performed to test our proposed approach and show our results.

4.1. Training and evaluation procedure

We use stochastic gradient descent to train our models (i.e., one pair of neural networks E and D for each combination of a dataset and a value of $\dim(\mathbf{z})$) on a random subset of the datasets described in Sect. 3, and evaluate their performance on another random subset that is disjoint from the training set. For more detailed information about our network architectures, their implementation, and our training procedure, see Appendix B, or take a look at our code, which is available online.

4.2. Reconstruction quality

As a first experiment, we study how well our trained model can approximate previously unseen PT profiles from the test set (cf. Appendix B.3), and compare the performance with two baseline methods: (1) low-order polynomials, and (2) a PCA-based approach similar to the method of Schreier et al. (2020).

Setup. For each $\dim(\mathbf{z}) \in \{1, 2, 3, 4\}$, we train three instances of our model (using three different random seeds to initialize the neural networks and control the split between training and validation) on the training set. We do not consider higher values of $\dim(\mathbf{z})$ here mainly because one of the goals of this work was to demonstrate that our method requires fewer fitting parameters than the baseline methods. Then, we take the learned decoder and, for each profile in the test set, use nested sampling to find the optimal \mathbf{z} (denoted \mathbf{z}^*) to reproduce that profile. In this case, “optimal” means that \mathbf{z}^* minimizes the mean squared error (MSE):

$$\mathbf{z}^* = \arg \min_{\mathbf{z}} \frac{1}{N} \cdot \sum_{i=1}^N (t_i - D(p_i | \mathbf{z}))^2. \quad (7)$$

We add this additional optimization step to the evaluation to remove the effect of the encoder: Ultimately, we are only interested in how well D can approximate a given profile for some \mathbf{z} – which does not necessarily have to match the output of E for that profile perfectly. We have chosen nested sampling over gradient-based optimization – which is possible given that D is fully differentiable both with respect to p and \mathbf{z} – because the latter is generally not available during an AR, unless one uses a differentiable forward simulator (see, e.g., `Diff-τ` from Hou Yip et al. 2022). Nested sampling, therefore, should give us a more realistic idea of which profiles we can find during a retrieval. Our optimization procedure is based on *UltraNest* (Buchner 2021), with 400 live points and a truncated standard normal prior, limiting each z_i to $[-4, 4]$ (because $\mathcal{L}_{\text{norm}}$ limits $\|\mathbf{z}\|$ to $\tau = 3.5$).

For the polynomial baseline, we simply fit each profile in the test set using a polynomial with degree $n_{\text{poly}} - 1$ for $n_{\text{poly}} \in \{2, 3, 4, 5\}$. The minimization objective of the fit is again the MSE.

Finally, for the PCA baseline, we compute a PCA on all the temperature vectors in the training set, and then fit the temperature vectors in the test set as a linear combination of the principal components (PCs), for $n_{\text{PC}} \in \{2, 3, 4, 5\}$, once again minimizing the MSE. We note that this PCA baseline must be

taken with a grain of salt, for two reasons. First, it requires us to project all profiles onto a single common pressure grid to ensure that the i th entry of the temperature vectors always has the same interpretation (i.e., temperature at pressure p_i). We do this by linear interpolation. Consequently, the common pressure grid is determined by the intersection of the pressure ranges of all profiles, meaning that profiles may get clipped both at high and low pressures. In practice, this affects in particular the GOYAL-2020 data where, for example, the pressure in the deepest layer varies by more than an order of magnitude between different profiles, whereas for PYATMOS, the profiles all cover a similar pressure range. Second, combining principal components only returns vectors, not functions, meaning that if we want to evaluate a profile at an arbitrary pressure value, we again have to use interpolation.

Results. We are showing some example profiles and the best-fit reconstruction using our model in Fig. 4. The main results of this experiment – the distribution of the (root) mean square error (RMSE) on the test set – are found in Fig. 5. The RMSE is simply defined as the square root of the MSE defined in Eq. (3), that is, the root of the mean squared difference between a given temperature vector and its best-fit reconstruction. As Fig. 5 shows, our method achieves a significantly lower reconstruction error than both baseline methods, both for the PYATMOS dataset and for the GOYAL-2020 dataset (see Fig. D.2). We find that on the PYATMOS dataset, even the one-dimensional version of our model (i.e., each profile is parameterized as just a single number) outperforms the polynomial baseline with five fitting parameters, and is approximately on-par with the PCA solution using five components.

To account for the absolute scale of the data (i.e., Earth-like temperatures vs. hot Jupiters) and allow comparisons across the two datasets, we also compute the mean relative error (MRE):

$$\text{MRE} = \frac{1}{N} \cdot \sum_{i=1}^N \frac{|\hat{t}_i - t_i|}{t_i}. \quad (8)$$

We compute this quantity for every profile in the test set, and then take the median of this distribution. For PCA and our method, the medians are then also mean-averaged over the different runs. The final results are given in Fig. 6. Looking at the mean relative errors, we note our errors for the GOYAL-2020 dataset are systematically larger than for the PYATMOS dataset. We suspect that can be explained as the result of two factors: First, the PYATMOS training dataset is ten times bigger than the GOYAL-2020 training dataset, and, second, the GOYAL-2020 dataset covers a more diverse set of functions compared to the PYATMOS dataset.

4.3. Interpreting the latent representations of PT profiles in the context of the corresponding atmospheres

In this section, we take a closer look at the latent representations of PT profiles that our models learn and the extent to which they lend themselves to interpretation. We begin with Fig. 7, where we present an illustration of our entire method using a model with $\dim(\mathbf{z}) = 2$ that we trained on the GOYAL-2020 dataset. The upper part of the figure shows how the encoder E maps PT profiles – which are given by vectors \mathbf{p} and \mathbf{t} , which in this example are 51-dimensional – onto 2-dimensional latent representations \mathbf{z} . The three colored dots in the latent space correspond to the three PT profiles shown on top, which are examples from our test set (i.e., profiles that were not used for training). The gray dots in the latent space show the representations of all PT profiles in our test set. Their distribution approximately follows a 2D standard Gaussian (cf. the marginal distributions), which is precisely what we would

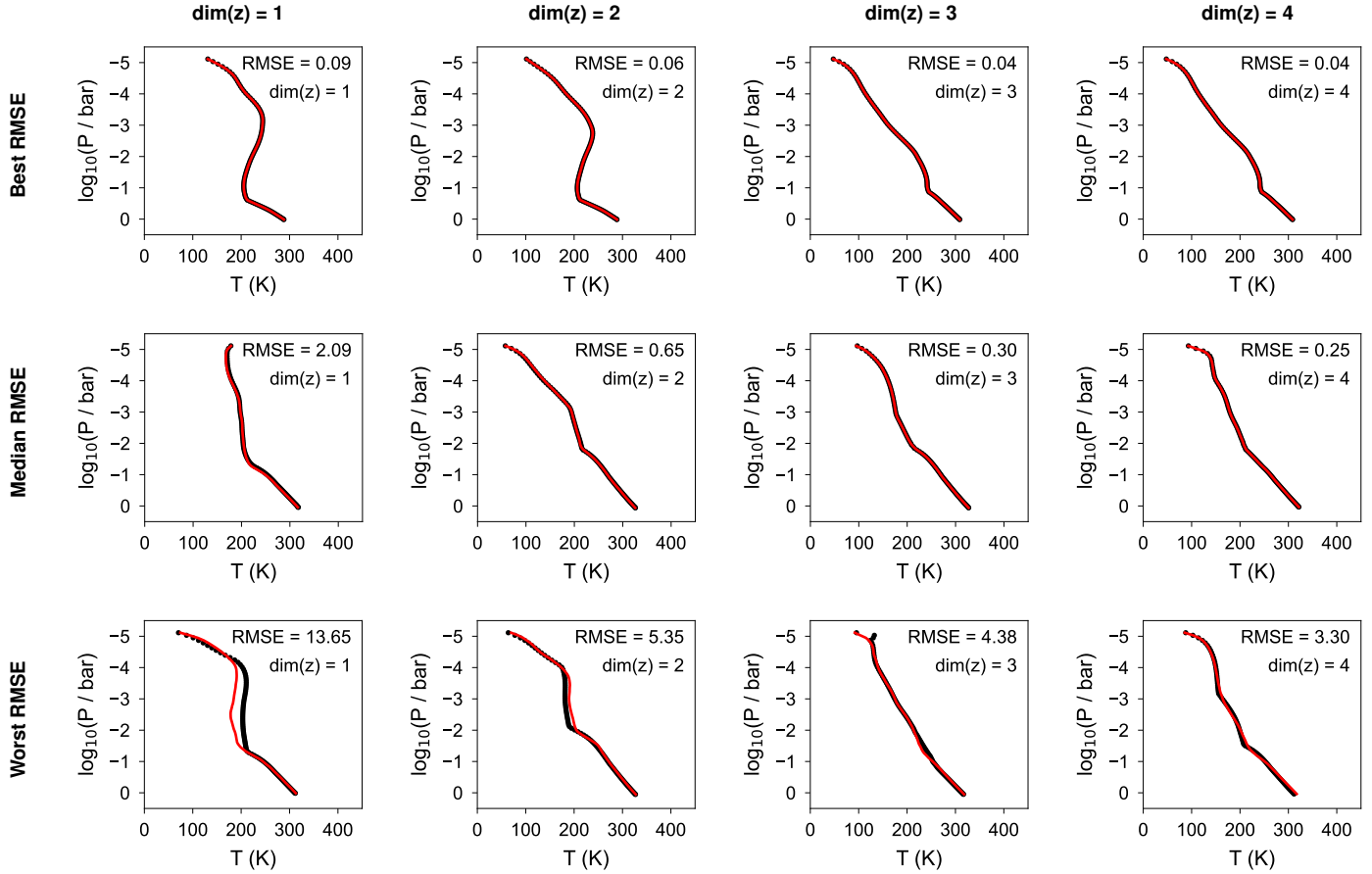


Fig. 4. Examples of pressure–temperature profiles from our test set (black) together with the best approximation using our trained model (red). Columns show different values of $\dim(z) \in \{1, 2, 3, 4\}$; rows show the best, median, and worst example (in terms of the RMSE). Results are shown for the PYATMOS dataset; additional plots for the GOYAL-2020 are found in Fig. D.1 (<https://github.com/timothygebhard/ml4pt/blob/main/scripts/plotting/fig-4-example-pt-profiles/plot-pt-profile.py>).

expect, as this was the reason we introduced the \mathcal{L}_{MMD} term to the loss function. The bottom part of the figure illustrates what happens if we place a regular grid on the latent space and, for each z on the grid, evaluate $D(\cdot | z)$ on a fixed pressure vector \mathbf{p}' , which does not have to match the \mathbf{p} of any profile in the training or test dataset. For example, \mathbf{p}' can have a higher resolution (i.e., more atmospheric layers) than the training data.

In both parts of the figure, we observe that PT profiles with different shapes – implying different physical processes – are mapped to different parts of the latent space, and that similar PT profiles are grouped together in the latent space. Here, “similar” refers not only to the shape of the PT profiles, but also to the properties of the corresponding atmospheres. To see this, we turn to Fig. 8. As mentioned above, both the PYATMOS and GOYAL-2020 datasets contain not only PT profiles, but also various other properties of the corresponding atmospheres, such as concentrations or fluxes of chemical species. In Fig. 8, we show several scatter plots of the latent representations of the test set (i.e., the z we obtained in the first experiment), color-coded according to some of these atmospheric properties – for example, the mean concentration of CO_2 in the atmosphere corresponding to a given PT profile. For ease of visualization, we limit ourselves to the case $\dim(z) = 2$. We observe that there are clear patterns in these scatter plots that show a strong correlation between our latent representations and the properties of the corresponding atmospheres, again demonstrating that PT profiles from physically similar atmospheres are grouped together.

With this in mind, we return to the lower half of Fig. 7, where we evaluate D on a grid of z values. We observe that not only do different shapes of PT profiles correspond to different parts of the latent space, but the PT profiles given $D(\cdot | z)$ also vary smoothly as a function of z . This means that the decoder not only reproduces the real PT profiles that were used to train it, but also allows smooth interpolation between them. Along the axes of the latent space, one can identify specific behaviors: For example, fixing $z_2 = 0$ and increasing z_1 leads to cooling in the upper atmosphere, while fixing $z_1 = 2$ and increasing z_2 leads to heating that extends further into the upper layers. Profiles around the center of the latent space (i.e., $z_1 \approx z_2 \approx 0$) are almost isothermal, indicating a relatively uniform temperature distribution throughout the atmosphere. A negative value of z_1 is generally associated with the presence of a thermal inversion, and for these cases, z_2 seems to encode the altitude where the inversion happens. For instance, for $z_1 = -2$ the profile shows an inversion in the mid-atmosphere (around $\log_{10}(P) = -2$) for $z_2 = 2$, whereas for $z_2 = -2$, the inversion only happens at very high altitudes/low pressures around $\log_{10}(P) = -6$.

Finally, we can also draw direct connections between the behavior of the PT profiles in latent space and the correlations of z with some of the atmospheric parameters that we show in Fig. 8. For example, a high concentration of TiO (panel 3 of Fig. 8) corresponds to the hot, inverted atmospheres in the upper left of the latent space. This is consistent with expectations from the literature, where TiO is indeed one of the canonical species

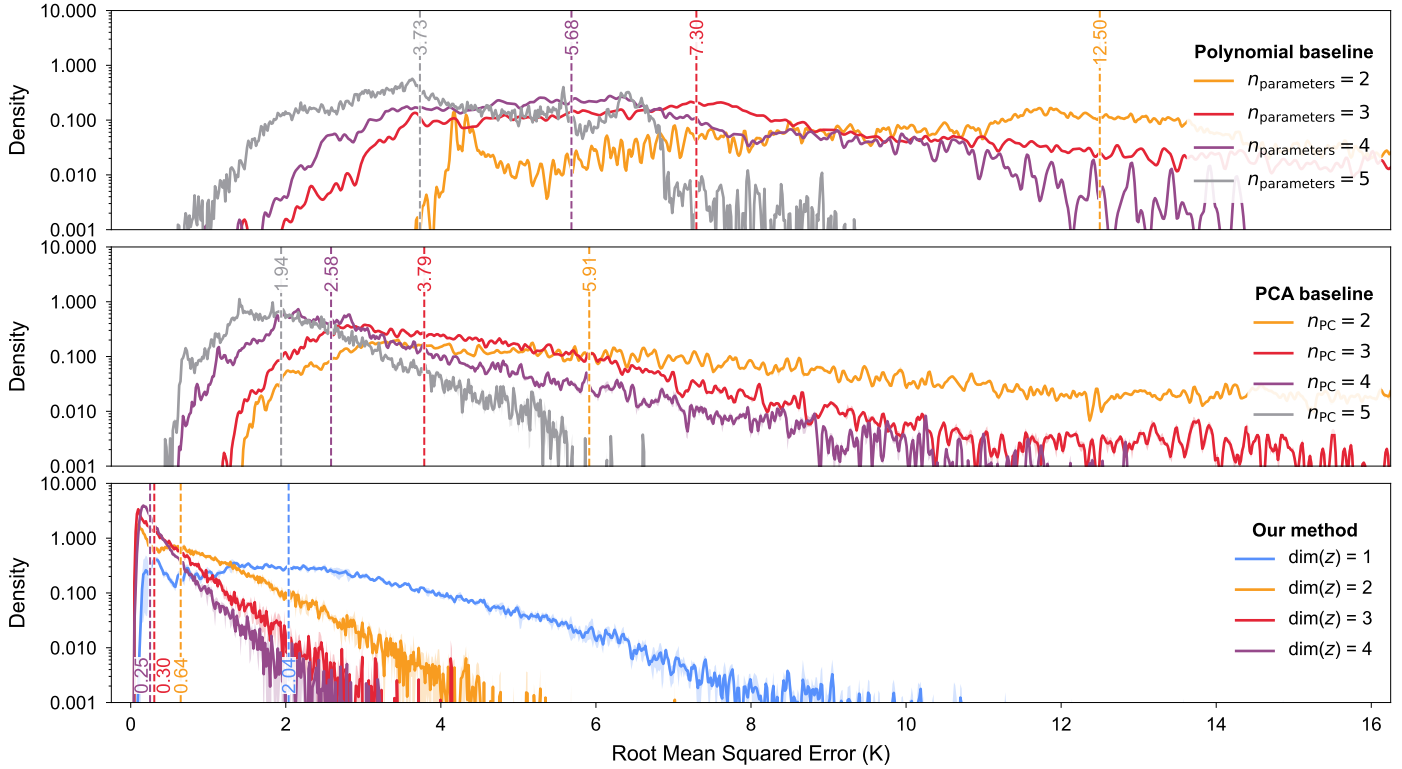


Fig. 5. Distributions of the reconstruction error (RMSE) for a polynomial baseline, a PCA-based baseline, and our method, for different number of fitting parameters. For our model, as well as the PCA baseline, each distribution is the average of three runs using different random seeds controlling the initialization and the train/validation split. The dashed lines indicate the respective medians. Results are shown for the PYATMOS dataset; additional plots for the GOYAL-2020 are found in Fig. D.2 (<https://github.com/timothygebhard/ml4ptp/blob/main/scripts/plotting/fig-5-error-distributions/plot-error-distributions.py>).

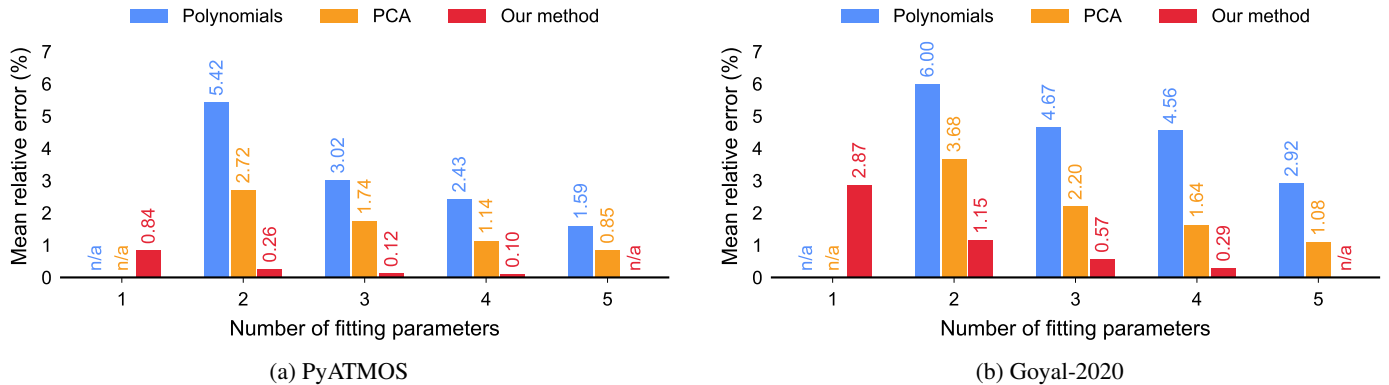


Fig. 6. Comparison of the median test set MRE for the different methods and datasets (lower is better). Reminder: For each profile, we compute the mean relative error (over all atmospheric layers), and then aggregate by computing the median over all profiles. Finally, for the PCA baseline as well as our method, the results are also mean-averaged over different random seeds (<https://github.com/timothygebhard/ml4ptp/blob/main/scripts/plotting/fig-6-mre-comparison/plot-mre-comparison.py>).

proposed to cause temperature inversions in hot Jupiters (e.g., Hubeny et al. 2003; Fortney et al. 2008; Piette et al. 2020).

Overall, this suggests that the latent space is to some extent interpretable: While the latent variables may not correspond directly to known physical parameters – as is the case, for example, for the analytical parameterizations of Madhusudan & Seager (2009) and Guillot (2010) – their relationships with other variables may provide insight into the behavior of exoplanet atmospheres.

Finally, a small word of caution: While we have just shown that the latent space is in principle interpretable, any specific interpretation – that is, a statement along the lines of “PT profiles with hot upper layers are found in the top left corner of

the latent space” – will only hold for one specific model. If we re-train that same model using a different random seed to initialize the neural network, or using a different dataset, we will end up with an equivalent but different latent space. This is because our model is, in general, not identifiable, and there exists no preferred system of coordinates for z : For example, we can rotate or mirror the latent space without losing information, and unless we add additional constraints to the objective function, there is no reason why the model should favor any particular orientation. However, this simply means that the value of z can only be interpreted for a given trained model. It does not, in any way, affect the ability of our model to be used for atmospheric retrievals.

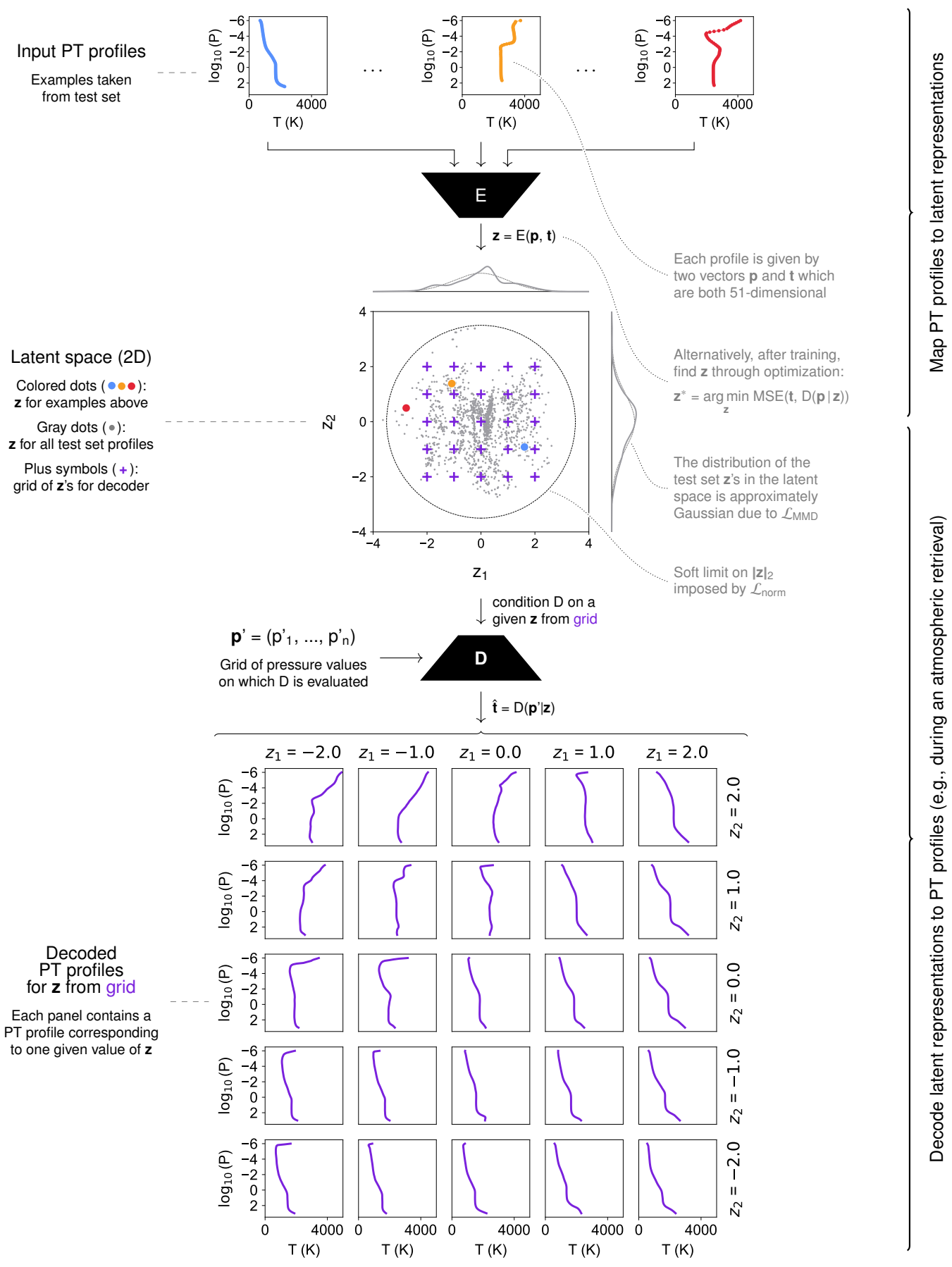


Fig. 7. A visual illustration of the encoder E , the decoder D , and their connection via the latent space. See Sect. 4.3 in the main text for a detailed explanation. This figure is best viewed in color (<https://github.com/timothygebhard/ml4ptp/blob/main/scripts/plotting/fig-7-overview/create-plots.py>).

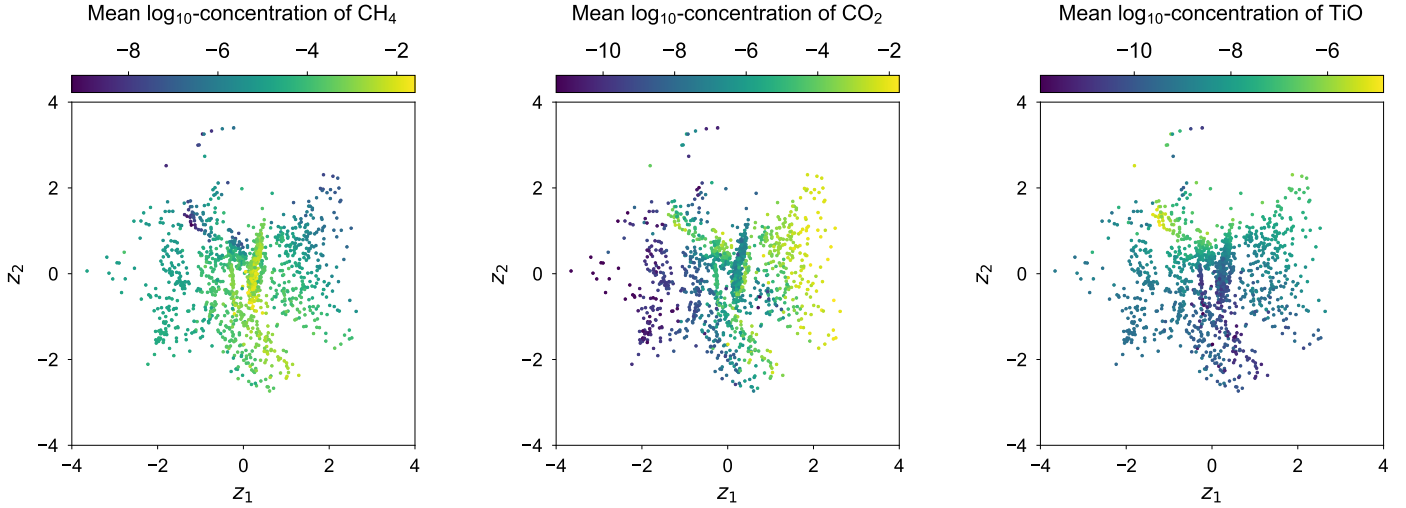


Fig. 8. Examples of 2D latent spaces, where each z from the test set is color-coded by a property of the corresponding atmosphere. Results are shown for the GOYAL-2020 dataset; additional plots for the PYATMOS are found in Fig. D.3 (<https://github.com/timothygebhard/ml4pt/blob/main/scripts/plotting/fig-8-color-coded-z/plot-color-coded-z.py>).

4.4. Usage in an atmospheric retrieval

A major motivation for the development of our method is the possibility of using it for atmospheric retrievals. In this experiment, we show how we can easily plug our trained model into an existing atmospheric retrieval pipeline. The target PT profile we use for this experiment is taken from the PYATMOS test set. We made sure that it is covered by the training data (i.e., there are similar PT profiles in the training set); however, in order not to make it too easy for our method, we chose a PT profile shape that is comparatively rare in the training data (see below). In Appendix C, we consider an even more challenging setting and show two additional results for PT profiles that were generated with a different code and that are not covered by our training distribution.

Setup. For our demonstration, we add our trained model to the atmospheric retrieval routine first introduced in Konrad et al. (2022) and developed further in Alei et al. (2022b) and Konrad et al. (2023), which is based on petitRADTRANS (Mollière et al. 2019), LIFEsim (Dannert et al. 2022) and PyMultiNest (Buchner et al. 2014). The goal of this experiment is to compare the PT profile parameterization capabilities of a version of our model (trained on the PYATMOS dataset using $\dim(z) = 2$) with the fourth-order polynomial (i.e., five fitting parameters) used in Konrad et al. (2022) and Alei et al. (2022b). We perform two cloud-free atmospheric retrievals (using both our PT model or the polynomial one) on the simulated thermal emission spectrum of an Earth-like planet orbiting a Sun-like star at a distance of 10 pc, assuming a photon noise signal-to-noise ratio of 10 at a wavelength of $\lambda = 11.2 \mu\text{m}$. The wavelength range was chosen as $[3 \mu\text{m}, 20 \mu\text{m}]$, and the spectral resolution was $R = \lambda/\Delta\lambda = 200$. The number of live points for PyMultiNest was set to 700. In addition to the parameters required for the PT profile, we retrieve 10 parameters of interest; see Table 1 for an overview. This setup closely matches Konrad et al. (2022) and Alei et al. (2022b).

For the retrieval with our model, we use a 2-dimensional uniform prior, $\mathcal{U}_2(-4, 4)$ for z . The reason for choosing this uniform prior instead of a Gaussian is that the shape of the ground truth PT profile is relatively under-represented in the PYATMOS dataset that we use for training. This means that our trained model, together with a Gaussian prior on z , assigns it a very small prior probability, making it difficult for the retrieval routine to find the

correct PT profile. Using a uniform prior is an easy way to give more weight to “rare” profiles. We discuss this choice, as well as potential alternatives, in more detail Sect. 5.3.

Results. We show the main results of this experiment in the form of the retrieved PT profile and spectrum residuals in Fig. 9. First, we find that the spectrum residuals demonstrate that both retrievals reproduce the simulated spectrum with sufficient accuracy, despite the differences in the PT parametrization. The residual’s quantiles are centered on the truth, and are significantly smaller than the photon noise-level. Second, for the retrieved PT profiles, we visually find that the result obtained with our model – unlike the polynomial baseline – is in good agreement with the ground truth, except at the highest layers of the atmosphere, where it tends to underestimate the true temperature. We believe this can be explained by the fact that the upper layers of the atmosphere (i.e., low pressures) have little to no effect on the simulated exoplanet spectrum (cf. the emission contribution functions in the right panel of Sect. 4.4), making it hard for the retrieval routine to constrain the PT structure in the upper atmosphere. Third, the retrieved constraints for the surface pressure P_0 and temperature T_0 , are much tighter and more accurate for our model than the polynomial baseline. Of course, this is also partly because our model represents a much narrower prior over the PT profiles that we are willing to accept as an explanation of the data (i.e., the spectrum) compared to the polynomial. This is an assumption, and in this case, it is justified by construction. In general, however, the decision of what is an appropriate prior for a given atmospheric retrieval problem is beyond the scope of the method as such. We discuss this in more detail in Sect. 5.2. Finally, we note that the retrieval using our model was significantly faster than the baseline: by decreasing the number of parameters required to fit the PT profile from five (polynomial) to two (our model), we reduced the total runtime by a factor of 3.2.

5. Discussion

In this section, we discuss the advantages and limitations of our method and suggest directions for future work.

5.1. Advantages

Our approach for parameterizing PT profiles has multiple potential benefits compared to existing approaches. First, our model

Table 1. Overview of all parameters in our simulated atmospheric retrieval.

Parameter	Description	Prior	True value	Retrieved (polynomial)	Retrieved (our method)
a_4	PT profile parameter (polynomial)	Uniform(0, 10)	–	1.33 ^{+0.39} _{–0.67}	–
a_3	PT profile parameter (polynomial)	Uniform(0, 100)	–	30.49 ^{+49.83} _{–27.32}	–
a_2	PT profile parameter (polynomial)	Uniform(0, 300)	–	106.78 ^{+159.09} _{–87.47}	–
a_1	PT profile parameter (polynomial)	Uniform(0, 500)	–	204.71 ^{+239.09} _{–118.24}	–
a_0	PT profile parameter (polynomial)	Uniform(0, 600)	–	384.86 ^{+178.40} _{–90.09}	–
z_1	PT profile parameter (our method)	Uniform(–4, 4)	–	–	–2.56 ^{+1.69} _{–1.34}
z_2	PT profile parameter (our method)	Uniform(–4, 4)	–	–	–0.04 ^{+1.19} _{–1.95}
$\log_{10}(P_0)$	Surface pressure (in bar)	Uniform(–2, 2)	0.013	–0.48 ^{+0.60} _{–0.61}	–0.05 ^{+0.14} _{–0.16}
R_{pl}	Planet radius (in R_{\oplus})	Normal(1, 0.2)	1.00	1.00 ^{+0.03} _{–0.03}	1.00 ^{+0.03} _{–0.03}
$\log_{10}(M_{\text{pl}})$	Planet mass (in M_{\oplus})	Normal(0, 0.4)	0.00	0.02 ^{+0.57} _{–0.59}	0.03 ^{+0.65} _{–0.64}
$\log_{10}(X_{\text{N}_2})$	N_2 mass fraction	Uniform(–2, 0)	–0.11	–1.04 ^{+0.93} _{–0.89}	–1.05 ^{+0.93} _{–0.88}
$\log_{10}(X_{\text{O}_2})$	O_2 mass fraction	Uniform(–2, 0)	–0.70	–1.10 ^{+0.96} _{–0.85}	–1.11 ^{+0.99} _{–0.84}
$\log_{10}(X_{\text{CO}_2})$	CO_2 mass fraction	Uniform(–10, 0)	–3.40	–2.72 ^{+1.31} _{–1.25}	–3.22 ^{+0.80} _{–0.75}
$\log_{10}(X_{\text{CH}_4})$	CH_4 mass fraction	Uniform(–10, 0)	–5.77	–5.06 ^{+1.26} _{–1.27}	–5.61 ^{+0.84} _{–0.84}
$\log_{10}(X_{\text{H}_2\text{O}})$	H_2O mass fraction	Uniform(–10, 0)	–3.00	–2.29 ^{+1.36} _{–1.32}	–2.82 ^{+0.77} _{–0.73}
$\log_{10}(X_{\text{O}_3})$	O_3 mass fraction	Uniform(–10, 0)	–6.52	–6.00 ^{+1.07} _{–1.04}	–6.39 ^{+0.72} _{–0.69}
$\log_{10}(X_{\text{CO}})$	CO mass fraction	Uniform(–10, 0)	–6.90	–7.47 ^{+2.93} _{–2.40}	–7.76 ^{+2.45} _{–2.13}

Notes. This table includes the respective priors, true values, and retrieved values for both PT profile parameterization schemes. For the latter, we report the median as well as the 2.5% and 97.5% percentiles.

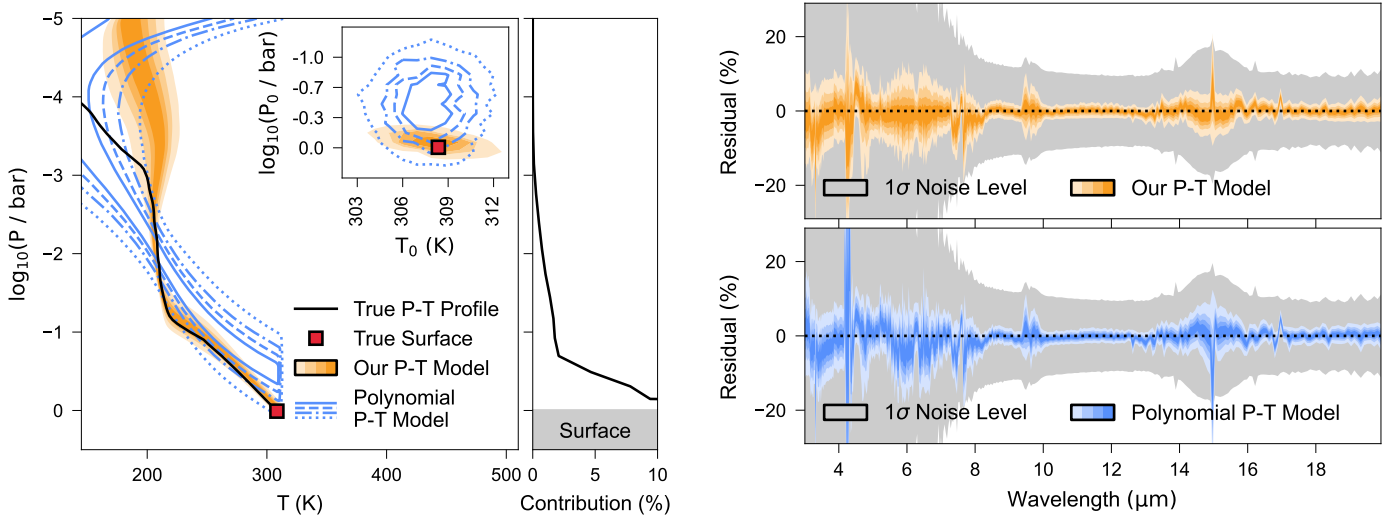


Fig. 9. Results for our simulated atmospheric retrieval of an Earth-like planet using *petitRADTRANS*, *LIFESim*, and *PyMultiNest*. *Left*: the retrieved PT profiles for the polynomial baseline and our model, together with the emission contribution function. *Right*: the relative fitting error of the spectrum, computed as (true spectrum – simulated spectrum)/true spectrum.

makes realistic, physically consistent PT profiles – which require a computationally expensive solution of radiative transfer equations – available as an ad hoc fitting function. It works, in principle, for all types of planets (Earth-like, gas giants, etc.) as long as we can create a suitable training dataset through simulations. In this context, “suitable” refers to, for example, the total size of the dataset and the density in parameter space (cf. the single out-of-distribution profile for *PyATMOS*). Our method then basically provides a tool which allows using the distribution of the PT profiles in that training set as a prior for the PT profile during a retrieval. Second, during an atmospheric retrieval, the parameterization of our model uses a simple prior for z that does not require hard-to-interpret fine-tuning (see also Sect. 5.3). Third, as we have shown in Sect. 4, our method can fit realistic PT profiles with fewer parameters than the baseline and still achieve a lower fit error. Limiting the number of parameters needed to express

a PT profile during retrieval can lead to faster processing times, or conversely, allow the retrieval of additional parameters when operating within a fixed computational budget. We would like to emphasize at this point that accelerating atmospheric retrievals is not only relevant for the analysis of existing observational data (e.g., from the *James Webb* Space Telescope), but can also be beneficial during the planning stages of future exoplanet science missions, such as LIFE (Quanz et al. 2022) or the recently announced Habitable Worlds Observatory (Clery 2023), which require the simulation of thousands of retrievals.

Finally, looking to the future, it seems plausible that approaches that replace the entire Bayesian inference routine with machine learning may gain further traction in the context of atmospheric retrievals (see, e.g., Márquez-Neila et al. 2018, Soboczenski et al. 2018, Cobb et al. 2019, Ardevol Martinez et al. 2022, Hou Yip et al. 2022, or Vasist et al. 2023). In this

case, reducing the number of retrieval parameters may no longer be a major concern – once trained, the inference time of these models is relatively independent of the number of parameters. However, even in this case, our approach will still be useful to provide a parameterized description of the posterior over physically consistent PT profiles.

5.2. The role of the training dataset

The practical value of our method stands and falls with the dataset that is used to train it. This should come as no surprise, but there are a few aspects to this that we will discuss in more detail here.

First, it is important to understand that the training dataset defines a distribution over the functions that our model learns to provide parameterized access to. When using our model in an atmospheric retrieval, this distribution then becomes the prior over the set of PT profiles to be considered. Compared to, for example, low-order polynomials, this is a much narrower prior: While polynomials can fit virtually anything (this is the idea of a Taylor series), our model will only produce outputs that resemble the PT profiles that it encountered during training. If this is a good fit for a given target profile, we can expect to significantly outperform the broader prior of polynomials. Conversely, if our prior does not cover the target profile, we cannot expect good results. However, this is not a limitation of our method in particular, but is true for all approaches that use a parameterization for the PT profile instead of, for example, a level-by-level radiative transfer approach. As [Line et al. \(2013\)](#) explain, using “[a] parameterization does force the retrieved atmospheric temperature structure to conform only to the profile shapes and physical approximations allowed by that parameterization.” Thus, as always in Bayesian inference, the decision to use our method as a prior for the PT profile (rather than, e.g., low-order polynomials) represents an assumption about the space of possible explanations for the data (e.g., the observed spectrum) that one is willing to accept, and it is up to the scientist performing an atmospheric retrieval to decide whether that assumption is justified.

Finally, it is important to keep in mind that our model will learn to replicate any anomalies and imbalances present in the training data. For example, if atmospheres without inversions are overrepresented in the training data (compared to atmospheres with inversions), our learned function distribution will also place a higher probability on PT profiles without inversions, which will obviously affect the results of a retrieval.

5.3. Choosing a prior distribution for \mathbf{z}

The last point in the previous section is closely related to the question of which prior one should choose for \mathbf{z} during a retrieval. The natural choice, of course, is to assume a d -dimensional standard Gaussian prior for \mathbf{z} as that is the distribution that we encourage during training by using the \mathcal{L}_{MMD} loss. However, there are a few additional considerations to keep in mind here. First, using a Gaussian prior for \mathbf{z} means that the induced distribution over PT profiles, given by $D(\cdot | \mathbf{z})$, will (approximately) match the distribution in the training dataset. This is a good choice if the training dataset itself constitutes a good prior for the PT profiles one wants to consider during a retrieval. On the other hand, consider now the case mentioned above where the training dataset contains all the expected shapes, but is highly biased towards one of the modes, not because this reflects the (assumed) true distribution of atmospheres, but as an artifact of the dataset generation process. In this case, it may make sense to choose a non-Gaussian prior (e.g., a uniform distribution) for \mathbf{z} to compensate for this

imbalance. When doing so, it is important to ensure that \mathbf{z} stays in the value range that the decoder D encountered during training (i.e., $\|\mathbf{z}\| < \tau$). Another (and perhaps more principled) way to solve this problem is to resample the training dataset to balance the different modes (e.g., “inversion” and “no inversion”). For our experiment in Sect. 4.4, we have validated that such a resampling-based approach does indeed work, but have chosen not to include all the details of this additional experiment so as not to distract too much from the main message of our paper.

Finally, there is the fact that our trained model does not only return physically consistent PT profiles, but also allows smooth interpolation (or, to some extent, extrapolation) between them. In cases where this is not desired, for example because one wants to be sure that only PT profiles very close to the training dataset are produced, one can choose a data-driven prior distribution for \mathbf{z} . A simple way to achieve this is to train a normalizing flow (see, e.g., [Kobyzev et al. 2021](#) or [Papamakarios et al. 2021](#) for recent introductions) to match the distribution of \mathbf{z} ’s obtained from the training dataset. If we then use the normalizing flow to sample \mathbf{z} , we will almost perfectly reproduce the distribution of the training data set. However, this can also exacerbate some of the problems we discussed earlier, for example, if the training data is unbalanced.

5.4. Directions for future research

We see several directions for future work. For example, to account for the fact that not all parts of the atmosphere contribute equally to the observed spectrum, one possible approach could be to use a modified reconstruction loss weighted by the respective contribution function for each point in the profile. In the case of emission spectra, this could focus the expressiveness of the model on regions of higher pressure, which have a greater influence on the spectrum, rather than regions at high altitudes, where the thermal structure of the atmosphere is difficult to constrain from the available observational data. For transmission spectra, which probe the upper layers of the atmosphere, one would do the opposite and give more weight to high altitudes.

Another promising direction could be to extend our method to parameterize not only temperature as a function of pressure, but also chemical abundance profiles (e.g., the concentration of CO_2 in each layer), to make them accessible during a retrieval.

Finally, given a sufficiently large grid of training data, our approach can also be used to parameterize the thermal structure of 3-dimensional atmospheres, where temperature depends not only on pressure but also on the longitude and latitude.

6. Summary and conclusion

In this work, we have introduced a new approach to parameterizing PT profiles in the context of atmospheric retrievals of exoplanets. Unlike existing approaches for this problem, we do not make any a priori assumptions about the family of functions that describe the relation between P and T , but instead learn a distribution over such functions from data. In our framework, PT profiles are represented by low-dimensional real vectors that can be used to condition a particular neural network, which, once conditioned, is the corresponding PT profile (i.e., a single-argument, single-output function that maps pressure onto temperature values).

We experimentally demonstrated that our proposed method works for both terrestrial planets and hot Jupiters, and when compared with existing methods for parameterizing PT profiles, we found that our methods give better approximations on average

(i.e., smaller fitting errors) while requiring a smaller number of parameters. Furthermore, we showed that our learned latent spaces are still interpretable to some extent: For example, the latent representations are often directly correlated with other properties of the atmospheres, meaning that physically similar profiles are grouped together in the latent space. Finally, we have demonstrated by example that our method can be easily integrated into an existing atmospheric retrieval framework, where it resulted in a significant speedup in terms of inference time, and produced a tighter and more accurate posterior for the PT profile than the baseline.

Given access to a sufficient training dataset, we believe that our method has the potential to become a valuable tool for exoplanet characterization: not only could it allow the use of physically consistent PT profiles during a retrieval, but it can also reduce the number of parameters required to model the thermal structure of an atmosphere. This can either reduce the overall computational cost or free up resources for other retrieval parameters of interest, allowing for more efficient and accurate atmospheric retrievals, which in turn could improve our understanding of exoplanetary habitability and the potential for extraterrestrial life.

Acknowledgements. We thank Sandra T. Bastelberger and Nicholas Wogan for pointing out the potential problems with the PyATMOS simulations and taking the time to discuss the issue with us. Furthermore, we thank Markus Bonse, Felix Dannert, Maximilian Dax, Emily Garvin, Jean Hayoz and Vincent Stimper for their helpful comments on the manuscript. Part of this work has been carried out within the framework of the National Centre of Competence in Research PlanetS supported by the Swiss National Science Foundation under grants 51NF40_182901 and 51NF40_205606. S.P.Q. and E.A. acknowledge the financial support of the SNSF. T.D.G. acknowledges funding through the Max Planck ETH Center for Learning Systems. B.S.K. acknowledges funding through the European Space Agency's Open Space Innovation Platform (ESA OSIP) program. This work has made use of a number of open-source Python packages, including `matplotlib` (Hunter 2007), `numpy` (Harris et al. 2020), `scipy` (Virtanen et al. 2020), `scikit-learn` (Pedregosa et al. 2011), `pymultinest` (Buchner et al. 2014), `pandas` (McKinney 2010), `petitRADTRANS` (Mollière et al. 2019), `pytorch` (Paszke et al. 2019), and `ultranest` (Buchner 2021). The colorblind-friendly color schemes in our figures are based on Petroff (2021).

References

- Alei, E., Konrad, B. S., Mollière, P., et al. 2022a, *Proc. SPIE*, 12180, 121803L
- Alei, E., Konrad, B. S., Angerhausen, D., et al. 2022b, *A&A*, 665, A106
- Amundsen, D. S., Baraffe, I., Tremblin, P., et al. 2014, *A&A*, 564, A59
- Ardevol Martínez, F., Min, M., Kamp, I., et al. 2022, *A&A*, 662, A108
- Arney, G., Domagal-Goldman, S. D., Meadows, V. S., et al. 2016, *Astrobiology*, 16, 873
- Barstow, J. K., & Heng, K. 2020, *Space Sci. Rev.*, 216, 82
- Blecic, J., Dobbs-Dixon, I., & Greene, T. 2017, *ApJ*, 848, 127
- Borgwardt, K. M., Gretton, A., Rasch, M. J., et al. 2006, *Bioinformatics*, 22, e49
- Buchner, J. 2021, *JOSS*, 6, 3001
- Buchner, J., Georgakakis, A., Nandra, K., et al. 2014, *A&A*, 564, A125
- Chen, X., Kingma, D. P., Salimans, T., et al. 2016, ArXiv e-prints [arXiv:1611.02731]
- Chopra, A., Bell, A. C., Fawcett, W., et al. 2023, ArXiv e-prints [arXiv:2308.10624]
- Christiansen, J. L. 2022, *Nat. Astron.*, 6, 516
- Chubb, K. L., & Min, M. 2022, *A&A*, 665, A2
- Clery, D. 2023, *Science*, 379, 123
- Cobb, A. D., Himes, M. D., Soboczenski, F., et al. 2019, *AJ*, 158, 33
- Čvorović-Hajdinjak, I., Kovačević, A. B., Ilić, D., et al. 2022, *Astron. Nachr.*, 343, e210103
- Dannert, F. A., Ottiger, M., Quanz, S. P., et al. 2022, *A&A*, 664, A22
- Drummond, B., Tremblin, P., Baraffe, I., et al. 2016, *A&A*, 594, A69
- Fortney, J. J., Marley, M. S., Saumon, D., et al. 2008, *ApJ*, 683, 1104
- Garnelo, M., Rosenbaum, D., Maddison, C. J., et al. 2018a, ArXiv e-prints [arXiv:1807.01613]
- Garnelo, M., Schwarz, J., Rosenbaum, D., et al. 2018b, ArXiv e-prints [arXiv:1807.01622]
- Goyal, J. M., Mayne, N., Drummond, B., et al. 2020, *MNRAS*, 498, 4680
- Gretton, A., Borgwardt, K. M., Rasch, M. J., et al. 2012, *JMLR*, 13, 723
- Guillot, T. 2010, *A&A*, 520, A27
- Hansen, B. M. S. 2008, *ApJS*, 179, 484
- Haqq-Misra, J. D., Domagal-Goldman, S. D., Kasting, P. J., et al. 2008, *Astrobiology*, 8, 1127
- Harris, C. R., Millman, K. J., Oliphant, T., et al. 2020, *Nature*, 585, 357
- Heng, K. 2017, *Exoplanetary Atmospheres: Theoretical Concepts and Foundations* (Princeton, NJ: Princeton University Press)
- Heng, K., Hayek, W., Sing, D., et al. 2011, *MNRAS*, 420, 20
- Hubeny, I., Burrows, A., & Sudarsky, D. 2003, *ApJ*, 594, 1011
- Hunter, J. D. 2007, *Comput. Sci. Eng.*, 9, 90
- Hou Yip, K., Changeat, Q., Al-Refaie, A., & Waldmann, I. 2022, *ApJ*, submitted, [arXiv:2205.07037]
- Jankov, I., Kovačević, A. B., Ilić, D., et al. 2022, *Astron. Nachr.*, 343, e210090
- Kaltenegger, L., & Sasselov, D. 2010, *ApJ*, 708, 1162
- Kaltenegger, L., & Traub, W. A. 2009, *ApJ*, 698, 519
- Kasting, J. F., & Ackerman, T. P. 1986, *Science*, 234, 1383
- Knoll, A. H., & Nowak, M. A. 2017, *Sci. Adv.*, 3, e1603076
- Kobyzev, I., Prince, S. J., & Brubaker, M. A. 2021, *IEEE Trans. Pattern Anal. Mach. Intell.*, 43, 3964
- Konrad, B. S., Alei, E., Quanz, S. P., et al. 2022, *A&A*, 664, A23
- Konrad, B. S., Alei, E., Quanz, S. P., et al. 2023, *A&A*, 673, A94
- Line, M. R., Wolf, A. S., Zhang, X., et al. 2013, *ApJ*, 775, 137
- Loshchilov, I., & Hutter, F. 2017, ArXiv e-prints [arXiv:1711.05101]
- Madhusudhan, N. 2018, in *Handbook of Exoplanets*, eds. H. Deeg, & J. Belmonte (Cham: Springer International Publishing), 1
- Madhusudhan, N., & Seager, S. 2009, *ApJ*, 707, 24
- Márquez-Neila, P., Fisher, C., Sznitman, R., et al. 2018, *Nat. Astron.*, 2, 719
- McKinney, W. 2010, in *Proceedings of the 9th Python in Science Conference*, eds. S. van der Walt, & J. Millman, 56
- Meadows, V. S., Arney, G. N., Schwietzman, E. W., et al. 2018, *Astrobiology*, 18, 133
- Mollière, P., Wardenier, J. P., van Boekel, R., et al. 2019, *A&A*, 627, A67
- Nixon, M. C., & Madhusudhan, N. 2022, *ApJ*, 935, 73
- Papamakarios, G., Nalisnick, E., Rezende, D. J., et al. 2021, *JMLR*, 22, 1
- Park, Y.-J., & Choi, H.-L. 2021, *Aerospace Sci. Technol.*, 113, 106672
- Paszke, A., Gross, S., Massa, F., et al. 2019, ArXiv e-prints [arXiv:1912.01703]
- Pavlov, A., & Kasting, J. 2002, *Astrobiology*, 2, 27
- Pavlov, A. A., Kasting, J. F., Brown, L. L., et al. 2000, *J. Geophys. Res. Planets*, 105, 11981
- Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, *JMLR*, 12, 2825
- Petroff, M. A. 2021, ArXiv e-prints [arXiv:2107.02270]
- Piette, A. A. A., Madhusudhan, N., McKemmish, L. K., et al. 2020, *MNRAS*, 496, 3870
- Quanz, S. P., Ottiger, M., Fontanet, E., et al. 2022, *A&A*, 664, A21
- Robinson, T. D., & Catling, D. C. 2012, *ApJ*, 757, 104
- Rugheimer, S., & Kaltenegger, L. 2018, *ApJ*, 854, 19
- Schreier, F., Städt, S., Wunderlich, F., et al. 2020, *A&A*, 633, A156
- Seager, S. 2010, *Exoplanet Atmospheres: Physical Processes* (Princeton, NJ: Princeton University Press)
- Sitzmann, V., Martel, J. N. P., Bergman, A. W., et al. 2020, ArXiv e-prints [arXiv:2006.09661]
- Skilling, J. 2006, *Bayesian Anal.*, 1, 833
- Soboczenski, F., Himes, M. D., O'Beirne, M. D., et al. 2018, ArXiv e-prints [arXiv:1811.03390]
- Teal, D. J., Kempton, E. M.-R., Bastelberger, S., et al. 2022, *ApJ*, 927, 90
- Traub, W. A., & Stier, M. T. 1976, *Appl. Opt.*, 15, 364
- Tremblin, P., Amundsen, D. S., Mourier, P., et al. 2015, *ApJ*, 804, L17
- Tremblin, P., Amundsen, D. S., Chabrier, G., et al. 2016, *ApJ*, 817, L19
- Vasist, M., Rozet, F., Absil, O., et al. 2023, *A&A*, 672, A147
- Vidaurri, M. R., Bastelberger, S. T., Wolf, E. T., et al. 2022, *PSJ*, 3, 137
- Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, *Nat. Methods*, 17, 261
- Zhang, Y., Snellen, I. A. G., Bohn, A. J., et al. 2021, *Nature*, 595, 370
- Zhao, S., Song, J., & Ermon, S. 2017, ArXiv e-prints [arXiv:1706.02262]
- Zingales, T., Falco, A., Pluriel, W., et al. 2022, *A&A*, 667, A13

Appendix A: Method

In this appendix, we provide additional background information for Sect. 2 (“Method”) of the main part of the paper.

Appendix A.1: Loss functions

To encourage the encoder to produce values of z that follow a given prior distribution (in our case: a d -dimensional Gaussian), we have chosen to add a term to the loss based on the Maximum Mean Discrepancy (MMD): \mathcal{L}_{MMD} . Our motivation for this choice came from [Zhao et al. \(2017\)](#), who have shown that for the case of variational auto-encoders (VAE) – a method that is conceptually related to our approach – the usage of the MMD maximizes the mutual information between the latent representation z and the output of the decoder D . Given our main task (i.e., atmospheric retrievals with simulation-based Bayesian inference), this seems desirable: After all, we do not only want to generate random PT profiles from the distribution given by the training data, but we would like to explore the latent space systematically (e.g., using nested sampling) to find the pressure–temperature profile that is the best match for our data.

There are, of course, also alternatives to the MMD which one could explore in future work. For example, one might choose to minimize the Kullback-Leibler (KL) divergence between z and a standard multivariate Gaussian. This corresponds to the loss function used in the original paper on neural processes ([Garnelo et al. 2018b](#)), who use the evidence lower bound (ELBO) as the objective function to train their models. However, at least in the case of variational auto-encoders, it has been shown (see, e.g., [Chen et al. 2016](#) and [Zhao et al. 2017](#)) that for sufficiently powerful models, using the ELBO objective can lead to a decoder that actually ignores the latent variable; a behavior that does not seem helpful for the application that we have in mind.

Appendix A.2: Maximum Mean Discrepancy (MMD)

The MMD is a kernel-based approach for estimating the distance between two probability distributions. It can be computed from two samples and works in arbitrary dimensions. Following [Borgwardt et al. \(2006\)](#) and [Gretton et al. \(2012\)](#), an unbiased estimator of the MMD of two samples $X = \{x_1, \dots, x_m\}$ and $Y = \{y_1, \dots, y_n\}$, with all x_i and y_i coming from the same vector space \mathcal{X} , is given by:

$$\begin{aligned} \text{MMD}^2(X, Y) &= \frac{1}{m \cdot (m-1)} \cdot \sum_{i=1}^m \sum_{j \neq i}^m k(x_i, x_j) + \\ &\quad \frac{1}{n \cdot (n-1)} \cdot \sum_{i=1}^n \sum_{j \neq i}^n k(y_i, y_j) - \\ &\quad \frac{2}{m \cdot n} \cdot \sum_{i=1}^m \sum_{j=1}^n k(x_i, y_j). \end{aligned}$$

Here, $k(x, y) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a kernel function. In our application, we use a Gaussian kernel:

$$k(x, y) = \exp \left\{ -\frac{\|x - y\|_2^2}{2\sigma^2} \right\},$$

where we estimate σ using the median heuristic from [Gretton et al. \(2012\)](#), that is, we set σ to the median of the distances between all points in the aggregate sample $X \cup Y$. We note that our loss term \mathcal{L}_{MMD} actually uses the squared MMD, as given by the above formula.

Appendix B: Implementation and training

In this appendix, we provide more detailed information about the neural networks that we use for the encoder and decoder, their technical implementation, as well as our training procedure.

Appendix B.1: Network architectures

Perhaps not surprisingly, when experimenting with different encoder and decoder network architectures, we found that there does not seem to exist a “one-size-fits-all” configuration that consistently yields the best results across all data sets and latent space dimensions. However, keeping in mind the proof-of-concept nature of this work, we decided to report only a single, simple experimental setup that gave reasonable results in all cases. Figure B.1 shows a visual illustration of our choices. For practical applications, where one is considering a specific data set and latent size, one can fine-tune the configuration to that particular setting.

Our encoder is a standard convolutional neural network (CNN), consisting of 4 convolutional layers followed by a multi-layer perceptron (MLP) with 5 fully-connected layers. It takes as input a PT profile where p and t are given as 2 channels. The convolutional layers have 64 channels, with the last layer returning only a single output channel. The kernel size of all layers was chosen as 1. The output of the convolutional part of the network is then passed to a multi-layer perceptron (MLP; i.e., a series of fully-connected layers) with 3 hidden layers and 512 units each, which outputs the latent representation $z \in \mathbb{R}^d$.

The decoder is even simpler than the encoder, not least to ensure that the evaluation during a retrieval is fast. It consists of a single MLP, with 3 hidden layers and 512 units. The “conditioning D on z ” part is achieved as follows: Each layer receives the output of the previous layer concatenated with $M_i z$ as its input, where the $M_i \in \mathbb{R}^{m \times d}$ are learnable matrices that map the d -dimensional z vector onto a vector in \mathbb{R}^m . For this work, we have set $m = 16$. We note that the number of layers and the activation function used for the decoder have an effect on which kind of functions (i.e., PT profiles) the model can learn. However, as real PT profiles are usually smooth, low-frequency functions with few inflection points, our simple choice seems to work well enough in practice.

All of our networks use LeakyReLU activation functions. No further regularization (e.g., dropout or batch norm) was used.

Appendix B.2: Implementation

We implement our models in Python using PyTorch ([Paszke et al. 2019](#)) and use the PyTorch Lightning wrapper⁵ to run the training. All of our code is available on GitHub.⁶

Appendix B.3: Data split and pre-processing

We split both the PYATMOS and the GOYAL-2020 dataset randomly into a training and a test set. For the PYATMOS data, the training set size is 100 000; the remaining 24 314 are held out for the evaluation. For the GOYAL-2020 data, we use 10 000 profiles for training and keep 1290 for testing. During training, the training set is again split into two datasets: one that is actually used to train the model, and one validation set that we use to monitor the training progress. We use a random 90% / 10% split here, and run all experiments 3 times with different random seeds to test how much the final model depends

⁵ <https://lightning.ai/pytorch-lightning>

⁶ <https://github.com/timothygebhard/ml4ptp>

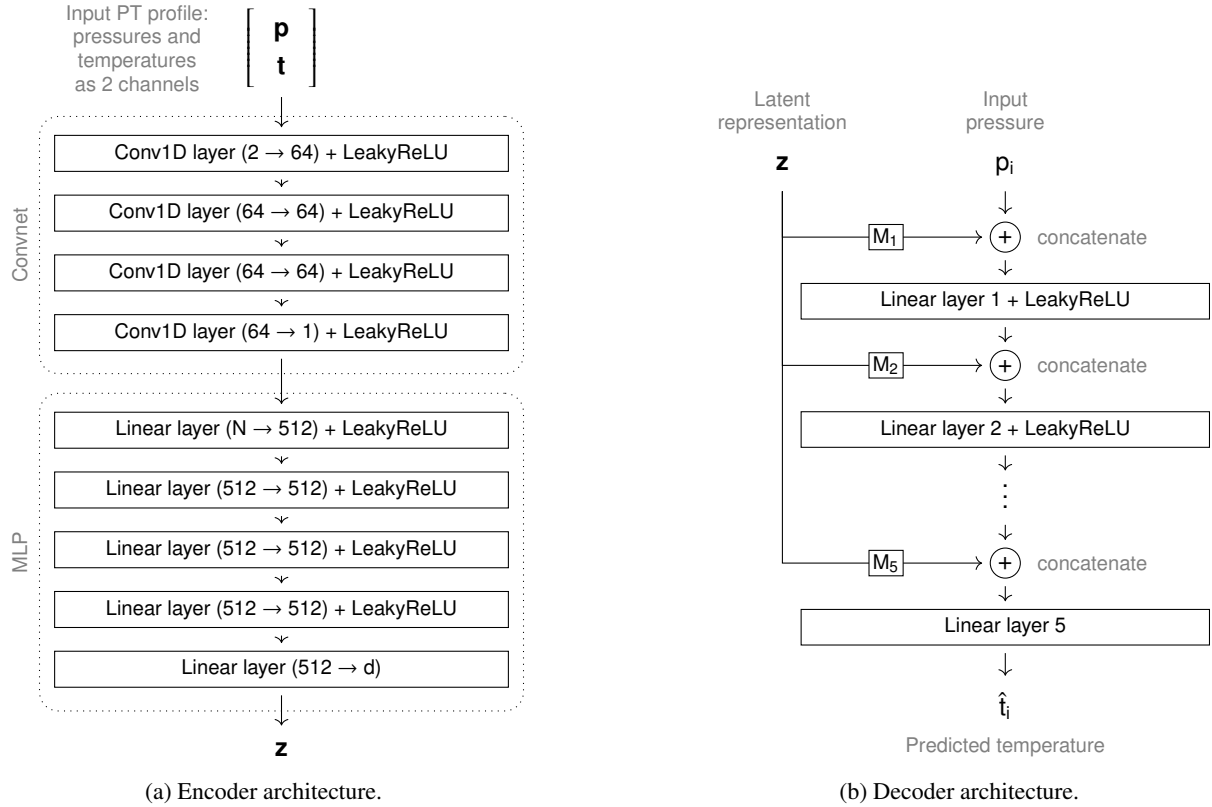


Fig. B.1: Schematic illustration of the architecture of our encoder and decoder networks. The numbers in parentheses denote the number of input and output features (or channels) of a layer.

on the exact training data and the initialization of the weights of the network.

We built our models to work with the decimal logarithm of the pressure rather than the pressure itself; otherwise, the scale of the pressure values would span several orders of magnitude, making it difficult for the neural net to learn anything. In addition, both (log)pressure and temperature are normalized using a standard score transform, where we subtract the mean and divide by the standard deviation (computed on the training set).

Appendix B.4: Training

We train our models for 1000 epochs using the AdamW optimizer (Loshchilov & Hutter 2017) with default values for the momentum and weight decay parameters. The batch size is set to 512, and we modulate the learning rate using a OneCycleLR scheduler, with 5% warm-up and a maximum learning rate of $\eta_{\max} = 5 \times 10^{-4}$. Additionally, we use gradient clipping with a threshold of 1.0 on the gradient norm. Again, these hyperparameters were not optimized. Finally, to reduce overfitting, we only save the model with the lowest validation loss (“early stopping”).

We run our training on modern GPUs (NVIDIA V100 and A100). Depending on the dataset, training a model to convergence usually takes between half an hour and two hours.

Appendix B.5: Exporting and running trained models

Trained models are exported using ONNX⁷, an open-source format for ML interoperability. Using the standard runtime engine⁸,

⁷ <https://onnx.ai>

⁸ <https://onnxruntime.ai>

evaluating our model for a given z on 100 pressure values takes about 1.6 ms on a standard laptop CPU (Intel Core i7). If necessary, this can be further improved, for example by pruning or quantizing the model, or by using a specialized execution engine.

Appendix B.6: Things that did (not) work

To inform further research, we disclose a few ideas that we have tried and that did not work for us:

- *Hypernet decoders*: In this setup, we used an additional neural network H that takes in z and predicts the weights of the conditioned decoder network $D(\cdot | z)$. While this may seem like the most principled way of conditioning D on z , we found that in practice, this seems to give no advantage over the simpler decoder architectures described above.
- *SIREN-style decoders*: We observed that when fitting only a single PT profile with a neural network, using sine activation functions (see Sitzmann et al. 2020) leads to faster convergence and a smaller MSE than other activation functions. However, this advantage did not seem to uphold when learning a distribution over PT profiles.
- Batch normalization, stochastic weight averaging (SWA), and larger batch sizes all seemed to decrease the performance.

Conversely, two things that we did find to be important are:

- *Normalization*: As mentioned before, we work with the logarithm of the pressure instead of the pressure itself, and normalize all PT profiles using a standard score transform.
- *Initialization*: In some cases, the initial weights of the encoder produce representations z that are all very close to 0. This usually causes the model to collapse and not learn anything. Our training procedure implements a sanity check for this case,

which re-initializes the encoder weights until the mean norm of z is above some arbitrarily chosen threshold.

Appendix C: Additional atmospheric retrievals

In Section 4.4, we demonstrated the applicability of our method in atmospheric retrieval for a scenario where the true PT profile is taken from the distribution of PT profiles that we used to train our model. Here, we consider a more challenging situation where the true PT profile is not covered by the training data. In this case, we cannot expect the retrieval to recover the ground truth PT profile perfectly; however, we can show that our method still behaves reasonably and produces interpretable results.

To this end, we perform cloud-free atmospheric retrievals for two PT profiles of planets around a Sun-like star published in [Rugheimer & Kaltenegger \(2018\)](#), representing (1) a present-day Earth and (2) a version of Earth 0.8 Ga ago, around the time of the Neoproterozoic Oxygenation Event (NOE), which is associated with the emergence of large and complex multicellular organisms (e.g., [Knoll & Nowak 2017](#)).⁹ Both profiles (and the corresponding spectra) were obtained using EXO-Prime ([Kaltenegger & Sasselov 2010](#)), a self-consistent coupled one-dimensional code developed for terrestrial exoplanets, which is different from the code that was used to generate our training data (ATMOS; see Sect. 3). EXO-Prime uses 1D models for the climate ([Kasting & Ackerman 1986](#); [Pavlov et al. 2000](#); [Haqq-Misra et al. 2008](#)), the photochemistry ([Pavlov & Kasting 2002](#)), and the line-by-line radiative transfer ([Traub & Stier 1976](#); [Kaltenegger & Traub 2009](#)). The emission spectra published alongside these PT profiles in [Rugheimer & Kaltenegger \(2018\)](#) were obtained using non-constant (i.e., pressure-dependent) abundance profiles for the different species. While this is obviously a more realistic approach, running our atmospheric retrieval procedure with pressure-dependent abundances would complicate things without

providing more insight into our method. Therefore, to keep things simple, we have recalculated the spectra using petitRADTRANS ([Mollière et al. 2019](#)) assuming constant abundance profiles (using the ground truth values from Table 1), and used these as targets for the atmospheric retrieval procedure. While this is, as noted above, a simplification compared to the self-consistent spectra of [Rugheimer & Kaltenegger \(2018\)](#), recomputing the spectra with petitRADTRANS (which we use for retrieval) has the added benefit of reducing the potential bias from comparing spectra computed with different codes or opacity tables (cf., e.g., [Alej et al. 2022a](#)), which is beyond the scope of this work. Finally, the general setup (i.e., tools, parameters, priors, etc.) for the atmospheric retrievals remains the same as in Section 4.4.

We present the results for both the Modern Earth case and the NOE Earth case in Figure C.1. Looking at these results, we find that – as expected – the retrieval routine is unable to fit the respective ground truth PT profile perfectly. This is because these PT profiles were not part of the training set (as illustrated by the top panel in Figure C.1) and thus the model did not learn to produce such PT profiles. However, if we look at the retrieved PT profiles in the middle panel of Figure C.1, we see that they match well with the set of PT profiles in the training data that are closest to the respective target profile. (“Closest” here refers to the smallest mean squared difference when comparing temperatures over the same grid of pressure values.) This behavior – recovering the best-matching PT profiles from the training set – is the best outcome one can hope for in cases where the model is confronted with data that it has never encountered before. After all, one of the advantages of our method is that it allows using a given grid of PT profiles as a prior for an atmospheric retrieval (in a parameterized, continuous fashion). A model that could “generalize out of the distribution” and also fit PT profiles that it never encountered during training would run counter to this idea.

⁹ We obtained the PT profiles through private correspondence with the first author of [Rugheimer & Kaltenegger \(2018\)](#).

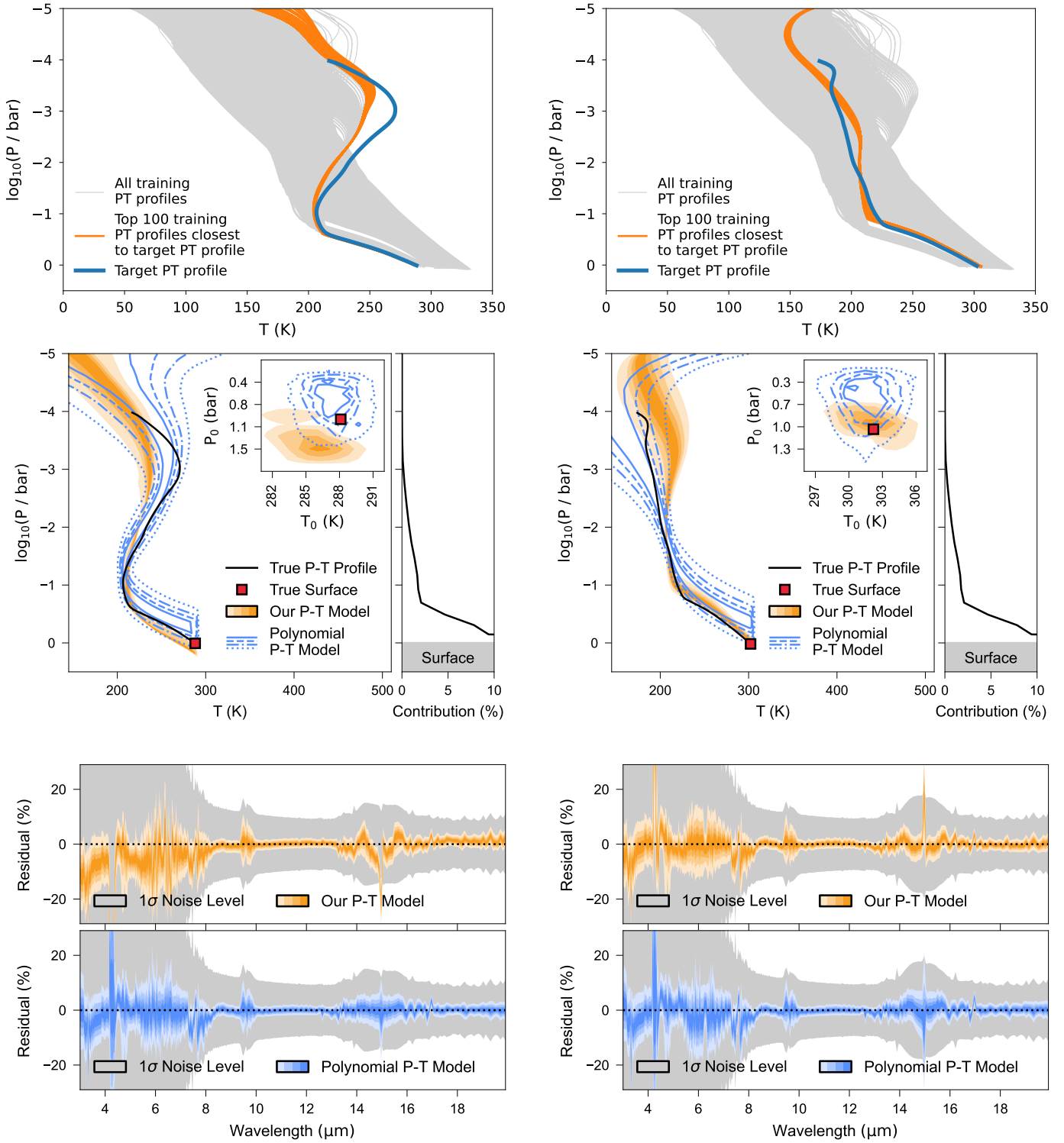


Fig. C.1: Retrieval results for both the Modern Earth and the NOE Earth case. The top panel in each subfigure shows the respective target PT profile together with the 100 best-matching PT profiles from the training set. (“Best-matching” here refers to the unweighted mean squared difference between the PT profiles, where the average is taken over all atmospheric layers.) The middle panel shows the PT profiles obtained by our simulated atmospheric retrieval, both for our model and for the polynomial baseline. Finally, the bottom panel shows the respective fitting errors on the spectrum; cf. the right side of Fig. 9.

Appendix D: Additional result plots

In Figure D.1, Figure D.2 and Figure D.3, we are showing additional result plots for the experiments described in the main text.

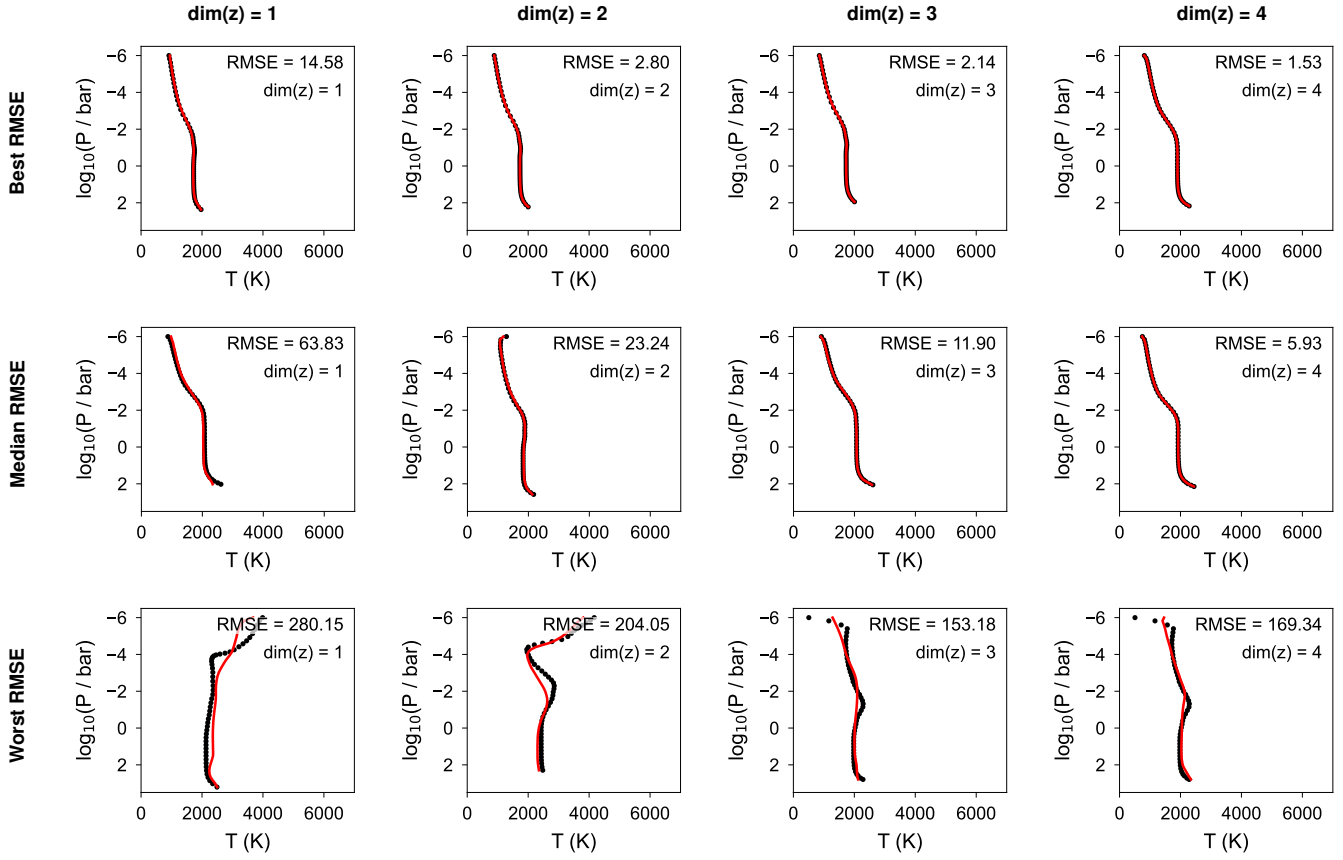


Fig. D.1: Same as Figure 4 (i.e., PT profiles from our test set, and the best fit with our model), but for the GOYAL-2020 dataset. [📄](#)

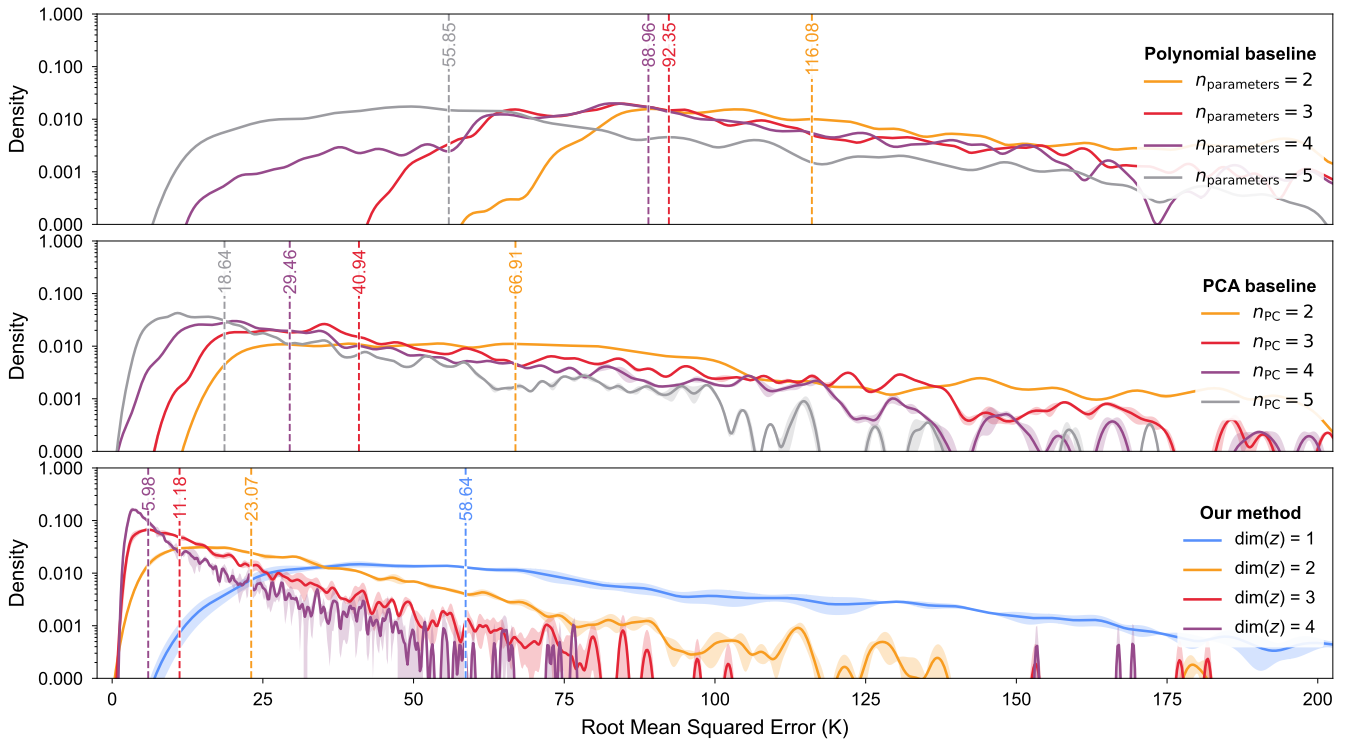


Fig. D.2: Same as Figure 5 (i.e., a comparison of the fitting error distribution on the test set), but for the GOYAL-2020 dataset. [📄](#)

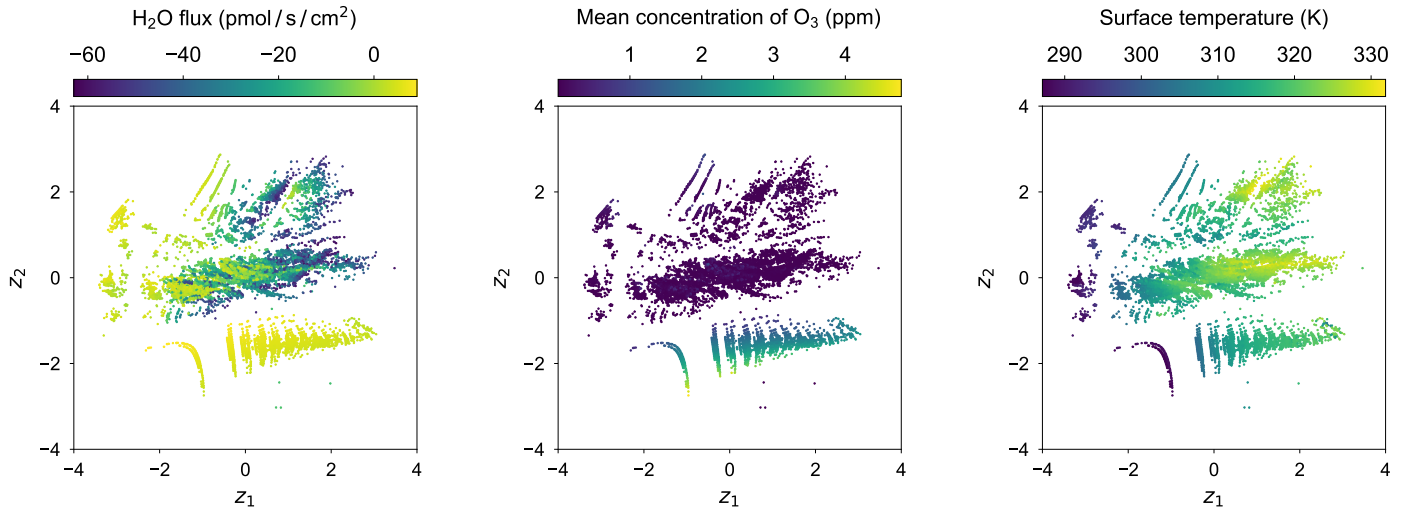


Fig. D.3: Same as Figure 8 (i.e., latent representations z color-coded with atmospheric properties), but for the PYATMOS dataset. [↗](#)