The Astrophysical Journal, 820:107 (8pp), 2016 April 1 doi:10.3847/0004-637X/820/2/107

# DREAMING OF ATMOSPHERES

I. P. Waldmann
Department of Physics & Astronomy, University College London, Gower Street, WC1E 6BT, UK; ingo@star.ucl.ac.uk
*Received 2015 November 20; accepted 2016 February 10; published 2016 March 28*

## ABSTRACT

Here, we introduce the *RobERt* (Robotic Exoplanet Recognition) algorithm for the classification of exoplanetary emission spectra. Spectral retrieval of exoplanetary atmospheres frequently requires the preselection of molecular/atomic opacities to be defined by the user. In the era of open-source, automated, and self-sufficient retrieval algorithms, manual input should be avoided. User dependent input could, in worst-case scenarios, lead to incomplete models and biases in the retrieval. The *RobERt* algorithm is based on deep-belief neural (DBN) networks trained to accurately recognize molecular signatures for a wide range of planets, atmospheric thermal profiles, and compositions. Reconstructions of the learned features, also referred to as the "dreams" of the network, indicate good convergence and an accurate representation of molecular features in the DBN. Using these deep neural networks, we work toward retrieval algorithms that themselves understand the nature of the observed spectra, are able to learn from current and past data, and make sensible qualitative preselections of atmospheric opacities to be used for the quantitative stage of the retrieval process.

*Key words:* methods: data analysis – methods: statistical – radiative transfer – techniques: spectroscopic

## 1. INTRODUCTION

The atmospheric retrieval of exoplanetary emission/transmission spectra is a complex undertaking (e.g., Madhusudhan & Seager 2009; Lee et al. 2011b; Line et al. 2012; Benneke & Seager 2013; Griffith 2014; Waldmann et al. 2015a, 2015b). Here, retrieval parameter dimensionality becomes an important factor to consider and, though desirable, most times allowing for all known atmospheric species to be fitted is too computationally expensive. Hence, a user-defined preselection of atmospheric absorbers/emitters must be made. A "seasoned user" would make this preselection based on previous experiences and a qualitative recognition of absorption/emission features present in the observed spectrum. Here, the human brain is very good in abstracting previously seen patterns to unseen circumstances, a desirable feature to be replicated by machines.

As we move to an era of largely automated retrievals, through the provision of open-source code to the community and future ground- and space-based spectroscopic surveys, it is important to strive toward universally applicable self-sufficient retrieval algorithms. In an ideal-case scenario, the retrieval suite would posses recognition and learning capabilities similar to the "seasoned user" and would not require any auxiliary user input aside from the observed spectrum itself. In other words, the program would understand what it is looking at and make a qualitative preselection of absorbing/emitting atmospheric species, followed by a quantitative retrieval.

In Waldmann et al. (2015b), we began working toward this end by introducing a pattern recognition algorithm, *Marple*. Based on principal-component analysis (PCA) facial-recognition approaches, *Marple* is able to rapidly sift through large molecular databases and return a list of the most probable absorbing species in the observed spectrum. This information can then be fed to the $\mathcal{T}$-REx atmospheric retrieval code (Waldmann et al. 2015a, 2015b) for a more quantitative analysis. Based on intrinsically linear coordinate transformations, *Marple* works well for transmission spectroscopy where the temperature–pressure (TP) profile can be assumed to be isothermal and the transmission approximated by a linear system.

The emission spectroscopy case is more complicated. Here, the shape of the spectral features strongly depends on the varying atmospheric thermal profile as well as varying molecular abundances. Such a nonlinear system is often poorly captured by a principal-component approach.

Consequently, we have developed a new neural-network-based spectroscopic pattern recognition framework, *RobERt* (Robotic Exoplanet Recognition), capable of learning and abstracting highly nonlinear systems and recognizing spectral features found in emission spectroscopy.

In this paper, we introduce the concept of deep-belief networks (DBNs) to the recognition of spectral features, describe the training set and algorithm used, and discuss *RobERt*'s recognition abilities using simulated spectra.

## 2. *RobERt*

*RobERt* mimics human recognition of spectroscopic features by using a pre-trained, deep-belief neural network (Hinton 2006, 2007; Bengio et al. 2007; Le Roux & Bengio 2010; Montavon et al. 2012; Bianchini & Scarselli 2014) at its core. DBNs are multi-layer, nonlinear transformations of the input data, the emission spectrum in this case, where each consecutive layer presents a progressively higher level of abstraction of the underlying features in the spectrum. These levels of abstraction are learned in an unsupervised (i.e., autonomous) fashion from a large catalog of input spectra. Once these features are learned from the data, a second, supervised learning stage is used to assign the learned features to their correct labels (e.g., $H_2O$, $CH_4$, etc.).

Neural networks are now commonly used in complex classification tasks such as image recognition (e.g., Krizhevsky et al. 2012; Liu et al. 2014; Wang et al. 2016; Shen et al. 2015), speech and music recognition (e.g., Hung et al. 2005; Jaitly & Hinton 2011; Zhang & Wu 2013; Pradeep & Kumaraswamy 2014), biology (e.g., Head-Gordon & Stillinger 1993; Plebe 2007; Wu & McLarty 2012; Spencer et al. 2015), and are finding increased use in the classification

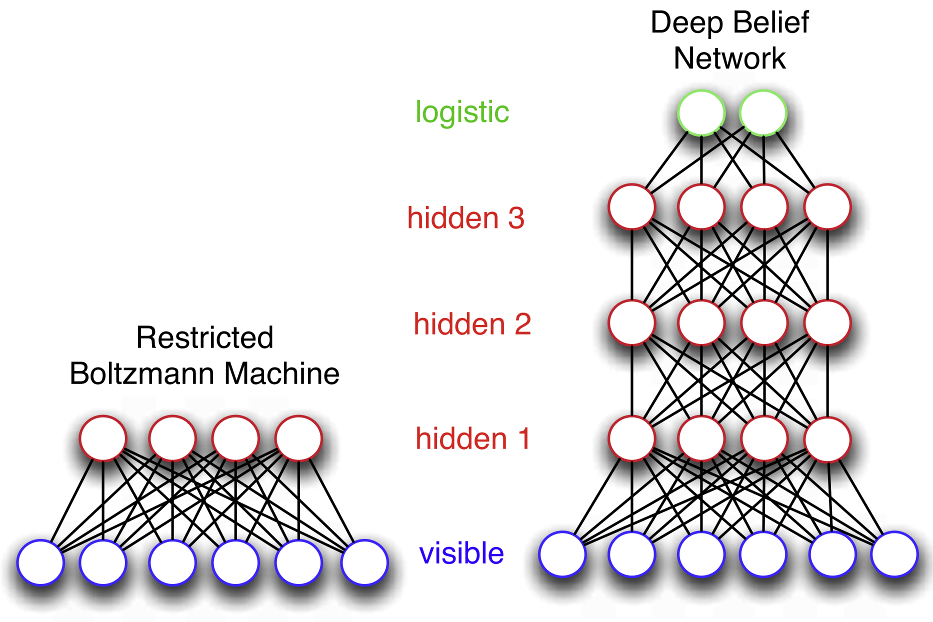

**Figure 1.** Schematic outline of a Restricted Boltzmann Machine (RBM) on the left and a full deep belief network (DBN) in the form of a Multi-layer Perceptron (MLP) on the right. The blue bottom layers are the "visible units" which are set to the input spectrum during training and recognition. The red layers are "hidden units" forming increasingly abstract representations of the input layer the further up the network they are. Green represents logistic units linking data labels to the top layer of hidden units. All of the units are connected (black lines) with all of the units in the layers above and below but no intra-level connections exist. It can be seen that the DBN can be built from three consecutive RBMs with the addition of a logistic regression layer.

of galaxies and cosmology (e.g., Collister & Lahav 2004; Agarwal et al. 2012, 2014; Reis et al. 2012; Karpenka et al. 2013; Dieleman et al. 2015; du Buisson et al. 2015; Ellison et al. 2015; Huertas-Company et al. 2015).

Whereas an in-depth derivation of DBNs is beyond the scope of this paper, we will briefly outline its underlying architecture and implementation. We refer the interested reader to Bengio (2009), Hinton (2012), and Fischer & Igel (2014) for detailed derivations.

### 2.1. *Restricted Boltzmann Machines*

Figure 1 shows a schematic of the the deep-belief network. The multi-layer DBN can be constructed from several Restricted Boltzmann Machines (Freund & Haussler 1992; Bishop 2006; Le Roux & Bengio 2008; Bengio 2009, 2012; Lee et al. 2011a; Hinton 2012; Montavon et al. 2012; Fischer & Igel 2014) with the addition of a logistic regression layer at the top of the network. The RBM is a two-layer neural network able to learn the underlying probability distribution over its set of input values. It represents a particular kind of Markov Random Field (Davison 2008) consisting of one layer of binary or Gaussian stochastic visible units (the input data) and one layer of binary stochastic hidden units. In RBMs, all of the hidden units are connected to all of the visible units but have no intra-layer dependence. Hence, all of the hidden units given the visible units are statistically independent and we can write the probability of all visible units given all hidden units and vice versa as the product of the individual probabilities:

$$P(\boldsymbol{v}|\boldsymbol{h}) = \prod_i P(v_i|\boldsymbol{h})$$
$$P(\boldsymbol{h}|\boldsymbol{v}) = \prod_j P(h_j|\boldsymbol{v}), \tag{1}$$

where $\boldsymbol{v}$ and $\boldsymbol{h}$ are the column vectors of the visible and hidden units, respectively, and $i$ and $j$ are their corresponding indices. We now want to find a configuration of the hidden layers, $\boldsymbol{h}$, that allows us to reconstruct the input, $\boldsymbol{v}$, with minimal error. Since $P(\boldsymbol{v}|\boldsymbol{h})$ and $P(\boldsymbol{h}|\boldsymbol{v})$ are factorial, we can write the activation functions of the individual visible and hidden binary units as

$$P(v_i = 1|\boldsymbol{h}) = \varsigma\left(b_i + \sum_j h_j w_{ij}\right)$$
$$P(h_j = 1|\boldsymbol{v}) = \varsigma\left(c_j + \sum_i v_i w_{ij}\right), \tag{2}$$

where $\varsigma$ is the sigmoid function

$$\varsigma(x) = (1 + e^{-x})^{-1}. \tag{3}$$

Assuming that both the visible and hidden units are binary, RBMs assign an energy term for each configuration of $\boldsymbol{v}$ and $\boldsymbol{h}$:

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{h} - \boldsymbol{h}^{\mathrm{T}}W\boldsymbol{v}, \tag{4}$$

where $\boldsymbol{b}$ and $\boldsymbol{c}$ are the bias vectors for the visible and hidden units, respectively, and $W$ is a matrix of connection weights between $\boldsymbol{v}$ and $\boldsymbol{h}$. The probability over all of the visible and hidden units $P(\boldsymbol{v}, \boldsymbol{h})$ is now given by

$$P(\boldsymbol{v}, \boldsymbol{h}) = \frac{e^{-E(\mathrm{v,h})}}{Z}, \tag{5}$$

where $Z$ is the partition function

$$Z = \sum_{\mathrm{v,h}} e^{-E(\mathrm{v,h})}. \tag{6}$$

The probability over the visible units as given by the RBM can now be calculated by summing over all of the hidden units:

$$P(\boldsymbol{v}) = \frac{1}{Z}\sum_{\mathrm{h}} e^{-E(\mathrm{v,h})}. \tag{7}$$

We now train the RBM by finding a set of parameters, $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{b}\}$, that maximizes the log-likelihood of the data, $\ln P(\boldsymbol{v}|\boldsymbol{\theta})$. The derivative of the log-likelihood with respect to the individual weights gives us the gradient on $\ln P(\boldsymbol{v}|\boldsymbol{\theta})$:

$$\begin{aligned}\frac{\partial \ln P(\boldsymbol{v}|\boldsymbol{\theta})}{\partial w_{ij}} &= -\sum_{\mathrm{h}} P(\boldsymbol{h}|\boldsymbol{v})\frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} \\ &\quad + \sum_{\mathrm{v,h}} P(\boldsymbol{v},\boldsymbol{h})\frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} &(8)\\ &= \langle v_i h_j\rangle_{P(\mathrm{h|v})} - \langle v_i h_j\rangle_{P(\mathrm{v,h})} &(9)\\ &= \langle v_i h_j\rangle_{\mathrm{data}} - \langle v_i h_j\rangle_{\mathrm{model}}, &(10)\end{aligned}$$

where $\langle v_i h_j\rangle_{\mathrm{data}}$ is the expectation value of all of the the hidden and visible unit activations given the training data and $\langle v_i h_j\rangle_{\mathrm{model}}$ is the same expectation under the reconstructed model distribution. The cost function for the optimization algorithm is now simply given by

$$\Delta w_{ij} = \epsilon\left(\langle v_i h_j\rangle_{\mathrm{data}} - \langle v_i h_j\rangle_{\mathrm{model}}\right), \tag{11}$$

where $\epsilon$ is a learning rate parameter.

Training can be performed using simple gradient descent. However, an exact calculation of $\langle v_i h_j\rangle_{\mathrm{model}}$ is highly computationally expensive. The likelihood gradient can be approximated by sampling the likelihood using Gibbs sampling (Press et al. 2007). Here, samples are iteratively drawn from $\langle v_i h_j\rangle_{\mathrm{data}}$ and $\langle v_i h_j\rangle_{\mathrm{model}}$ until the Markov Chain Monte Carlo (MCMC) sampling converges. Contrastive Divergence (CD; Hinton 2002) further simplifies the Gibbs sampling process by breaking the requirement for exact convergence and restricting the MCMC chain to a few (as few as one) iterations. This leads to significant gains in convergence speed. For an in-depth explanation of CD, we refer the reader to Hinton (2002), Bengio et al. (2007), and Bengio (2009).

### 2.2. *Deep Belief Networks*

We now construct the DBN using RBMs as building blocks. In agreement with convention and in accordance with Figure 1, we refer to the data input as being at the "bottom" of the network and increasing in abstraction as we go "up" the network.

The bottom RBM has the normalized emission spectrum as input (i.e., visible) units. Here, a binary representation of the observed data is not ideal and we replace $\boldsymbol{v}$ with Gaussian units. These better represent the continuous values found in spectroscopic data. The hidden units and all of the higher DBN layers remain binary. For the Gaussian RBM layer, the unit activations (Krizhevsky 2009; Wang et al. 2016) become

$$\begin{aligned} P(v_i|\boldsymbol{h}) &= \mathcal{N}\left(v_i; b_i + \sum_j w_{ij}h_j, \sigma_i^2\right) \\ P(h_j = 1|\boldsymbol{v}) &= \varsigma\left(c_j + \sum_i w_{ij}\frac{v_i}{\sigma_i^2}\right)\end{aligned}, \tag{12}$$

where $\mathcal{N}$ is the Normal distribution and $\sigma$ is the standard deviation of the spectrum. Furthermore, we substitute the energy term (Equation (4)) with

$$E(\boldsymbol{v},\boldsymbol{h}) = \sum_i \frac{(v_i - b_i)^2}{2\sigma_i} - \sum_j b_j h_j - \sum_{i,j}\frac{v_i}{\sigma_i}h_j w_{i,j}. \tag{13}$$

We now learn the RBM greedily until convergence and take the resulting hidden layer as input to the next higher up RBM. We repeat this process for three consecutive RBMs. This constitutes the unsupervised training stage as the DBN learns on un-labeled data.

Once the RBM layers are trained, we form a Multi-Layer Perceptron (MLP) by attaching a logistic regression layer to the top layer of the network (Equation (12)). This links the top-most hidden units to the data labels (e.g., $H_2O$, $CH_4$, $CO_2$, etc.). We now greedily learn the whole network using stochastic gradient descent by presenting a spectrum of a given composition and its corresponding data label to the network. This supervised learning has two purposes: (1) it fine-tunes the network and (2) it associates labels to the network. More specifically, in the supervised learning stage, the RBM layers are fixed and act as a feed-forward network. The logistic regression layer now learns the mapping between the high-level representations of the upper RBM layer and the associated data labels. We refer the interested reader to the standard literature (e.g., Bishop 2006; Hilbe 2009) for an in-depth treatment of logistic regression.

We learn the MLP using mini-batch stochastic gradient descent (Li et al. 2014). Mini-batches determine the number of training examples looked at simultaneously before updating the DBN weights. Looking at "chunks" of data simultaneously, allows us to vectorize the gradient computation and achieve higher convergence speeds than for standard stochastic gradient descent methods. We did not require the use of any regularizations during supervised learning, but employ "early stopping" criteria to avoid overfitting (see Section 3.2). It is worth mentioning that "dropout" algorithms (Hinton et al. 2012; Srivastava et al. 2014) have recently been shown to reach lower reconstruction errors than conventional supervised learning (with or without regularization) and are found to be highly robust against overfitting, hence avoiding the need for early stopping criteria.

## 3. IMPLEMENTATION AND TRAINING

*RobERt* is written in python using the scipy optimization toolbox and the theano[1] library. Theano is a very powerful graph and symbolic math toolbox with efficient parallelization (through the BLAS library) and native GPU support. The training data was generated using $\mathcal{T}$-REx run with OpenMP parallelization to produce the required grid of emission forward models.

### 3.1. *Training Data Set*

In the unsupervised training stage, *RobERt* requires a large set of example emission spectra to train with. Such a training set should include a broad range of planet types, atmospheric trace gasses, and TP-profiles. We considered a total of five planets ranging from warm Super-Earths (GJ1214b,

[1] https://pypi.python.org/pypi/Theano

**Table 1**
Summary of Training Set

| | |
|---|---|
| No. planets | 5 |
| Planets[a] | WASP-12b, HD189733b, HD209458b, HAT-P-11b, GJ1214b |
| No. molecules | 10 |
| Molecules | $H_2O$, HCN, $CH_4$, $CO_2$, CO, $NH_3$, NO, SiO, TiO, VO |
| Abundance range | $1 \times 10^{-7}$–$1 \times 10^{-2}$ |
| Compositions/planet | 5 |
| TP-profiles/planet | 7 |
| $\lambda$ range | 1–20 $\mu$m |
| Resolution | 300 (constant) |
| Points/spectrum | 900 |
| Spectra/planet | 17150 |
| Spectra total | 85750 |

**Note.**
[a] All parameters are from http://exoplanet.eu.

Charbonneau et al. 2009) to strongly insolated hot-Jupiters (e.g., WASP-12b, Hebb et al. 2009). In total, we simulated 17,150 emission spectra per planet and 85,750 spectra in total. Each spectrum contains only one trace gas species at a time and no mixtures are considered in the training set. Table 1 summarizes the training set parameters. The creation of the training set took ~3 hr on 96 Intel Xeon E5-2697v2 cpus.

The data set was then randomly divided into 80% training data and 20% test data. *RobERt* is only trained on the training data with random selection of spectra from the test data presented to *RobERt* at every *N*th iteration of the supervised learning to test *RobERt*'s prediction accuracy.

#### 3.1.1. Normalization

Before training *RobERt* on the catalog of input spectra, we first normalize the input to a zero mean and unit variance grid. Though this is not strictly necessary, the normalization significantly improves convergence properties of DBNs. The normalization consists of three steps.

1. We normalize the emission spectrum with the Planckian of the planet's host star to obtain the planetary intensity

$$\boldsymbol{I}_p = (\boldsymbol{F}_p/\boldsymbol{F}_*) \times \boldsymbol{BB}_* \left(\frac{R_*}{R_p}\right)^2, \quad (14)$$

   where $\boldsymbol{F}_p/\boldsymbol{F}_*$ is the column vector of the planetary/stellar flux ratio and $\boldsymbol{BB}_*$ is the Planck function at the stellar temperature. This normalization step ensures that the training process is not biased by the underlying stellar blackbody function.
2. We now convert $I_p$ into brigthness temperatures using

$$\boldsymbol{T}_p = \frac{\mathrm{hc}}{\lambda k} \times \left[\log\left(\frac{2\mathrm{hc}^2}{I_p(\lambda)\lambda^5} + 1\right)\right]^{-1}, \quad (15)$$

where $k$ is the Boltzmann constant, $h$ is the Planck constant, $c$ is the speed of light, and $\lambda$ is the wavelength.

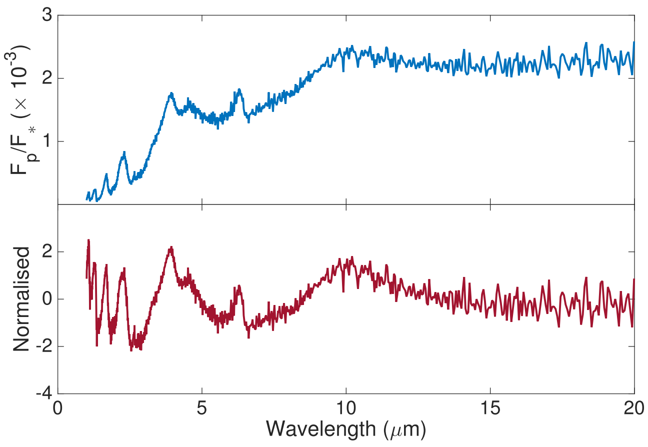

**Figure 2.** Top: example spectrum of a hot-Jupiter (water only) generated by $\mathcal{T}$-REx. Bottom: the normalized emission spectrum used for training *RobERt*.

3. Finally, we subtract the mean value of $\boldsymbol{T}_p$ and normalize to unit variance to give the normalized spectrum $\hat{\boldsymbol{T}}_p$:

$$\hat{\boldsymbol{T}}_p = \frac{\boldsymbol{T}_p - \bar{\boldsymbol{T}}_p}{\sqrt{\mathrm{var}(\boldsymbol{T}_p - \bar{\boldsymbol{T}}_p)}}. \quad (16)$$

   Figure 2 shows an example input spectrum of $H_2O$ before normalization (top, blue) and after normalization (bottom, red).

### 3.2. Training

*RobERt* is now set up to contain three RBM levels of 500, 200, and 50 neurons from bottom to top, respectively, with the input data vector containing 900 spectral points. As discussed in Section 6, we find that slightly smaller networks have similar performance levels but larger networks are too redundant.

The unsupervised training stage ran over 100 iterations per RBM level at a learning rate of $\epsilon = 0.01$. We find that for all of the layers, convergence is typically reached between the 80th–90th iteration. During the supervised training stage, we adopt a learning rate of $\epsilon = 0.01$ and mini-batch sizes of typically 100 training spectra. The reconstruction error of the DBN given the test data is computed at each training epoch. Convergence of the supervised learning is reached when no improvement in the reconstruction error is obtained over a maximum of 20 epochs and the iteration with the lowest reconstruction error is then taken as final result. This early stopping prevents significant overfitting during the supervised training stage.

The full training process takes ~1.5 hr on 6 cpu cores or <10 minutes using an Nvidia Tesla K40 card (2880 GPUs). *RobERt* completes the supervised training stage with a test data recognition accuracy of 99.7%.

## 4. RECOGNITION OF EMISSION SPECTRA

One major advantage of DBNs is their ability to generalize patterns over large ranges of parameter spaces, both seen and perviously unseen by the network. To demonstrate this behavior, we generated emission spectra of the hot-Jupiter WASP-76b (West et al. 2013), unknown to *RobERt*, for a variety of trace gas molecules, mixtures, and signal-to-noise ratios (S/N). The spectral recognition process then proceeds in three stages.

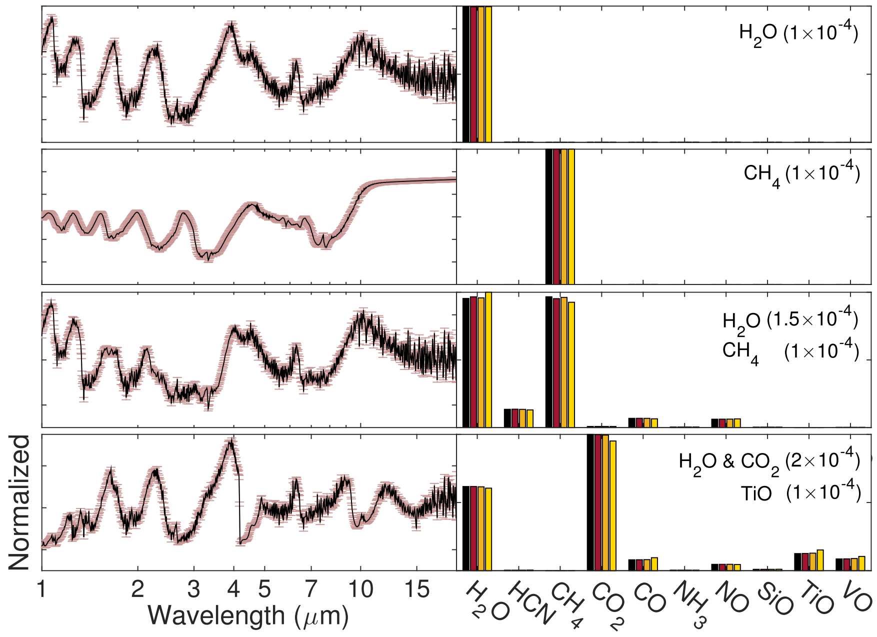

**Figure 3.** Left: example normalized emission spectra at S/N = 20. From top to bottom, the spectral compositions are as follow: (1) $H_2O$ ($1 \times 10^{-4}$); (2) $CH_4$ ($1 \times 10^{-4}$); (3) $H_2O$ ($1.5 \times 10^{-4}$) and $CH_4$ ($1 \times 10^{-4}$); (4) $H_2O$ ($2 \times 10^{-4}$), $CH_4$ ($2 \times 10^{-4}$), and TiO ($1 \times 10^{-4}$). Right: corresponding probability of the molecule being present in the spectrum to the left. All of the probabilities are normalized ($p(x)/\max[p(x)]$) for clarity and are color coded to represent four different S/N values of the input spectrum: 20 (black), 10 (brown), 5 (orange), and 2 (yellow).

1. The observed spectrum is normalized following the steps described in Section 3.1.1.
2. The mean of each spectral bin is randomly perturbed within the measurement error bar, resulting in a "noisy" spectrum.
3. The visible units of the DBN are set to the normalized, noisy spectrum and the DBN is run in the forward direction to obtain the label probabilities *P*(*label*).

Steps 2 and 3 are repeated 100 times and the label probabilities are recorded, summed, and normalized.

Figure 3 shows four normalized example spectra and the results of *RobERt*'s identification for S/Ns of 20, 10, 5, and 2. Spectra containing only one main trace gas component are recovered >99% of the time, across all of the planet types considered. This remains true for strongly saturated spectra with molecular abundances of $>1 \times 10^{-2}$ and very low S/N values. Surprisingly, even S/Ns of 0.5–1.0 allow *RobERt* to recognize the dominant trace gas component with good accuracy. *RobERt* was trained on only individual trace gases, i.e., pure water spectra or pure methane spectra, but not on mixtures of trace gasses. This is mainly due to the very large number of permutations required to represent mixtures of molecules accurately over varying abundances and TP-profiles in the training data. It is hence encouraging to see that *RobERt* understands mixtures well when presented with them. Figure 3 shows two examples of spectra containing $H_2O + CH_4$ and $H_2O$, $CO_2$, and TiO. In the three molecules example, *RobERt* identifies the main constituents, water and carbon-dioxide, with a high probability and the third constituent is either attributed to TiO, VO, CO, or NO with TiO having the highest probability of these candidates. In an automated retrieval context, the retrieval code would run a first pass with $CO_2$, $H_2O$, TiO, VO, CO, and NO as input and proceed to nested model down-selection in subsequent retrieval runs (Waldmann et al. 2015b).

### 4.1. Restricted Wavelength Ranges and Resolution Mismatches

Whereas it is more adequate to train the DBN with instrument-specific resolutions and wavelength ranges, e.g., for *Hubble Space Telescope (HST)*/WFC3, *James Webb Space Telescope (JWST)*/MIRI, and *JWST*/NIRSPEC, it is an intriguing exercise in itself to explore the effect of incomplete wavelength ranges on *RobERt*'s ability to recognize molecular species. As stated previously, in this example, *RobERt* was trained on a wavelength grid ranging from 1 to 20 $\mu$m with a constant resolution of 300. Figure 4 shows the normalized water-only emission spectrum for the *HST*/WFC3 G141 grism wavelength range (yellow spectrum). The remaining spectrum outside the wavelength range considered is padded with zeros on both sides. *RobERt* is clearly able to identify water as the dominant trace gas. We now consider increasingly restrictive wavelength ranges until the clear water detection breaks down at the 1.26–1.53 $\mu$m bandpass and *RobERt* attributes nearly equal probabilities to $H_2O$, $CH_4$, and NO. While initially surprising, upon closer inspection, all three molecular species have strong overlapping features in this wavelength range (blue and black lines in Figure 4 show the normalized spectrum of NO at $1 \times 10^{-2}$ and $CH_4$ at $1 \times 10^{-4}$, respectively) and a "visual" separation of molecules becomes very difficult.

We now investigate the effect of resolution mismatches between the observed data and the resolution with which the DBN is trained. As expected, downsampling from a higher resolution to the DBN resolution does not impair recognition efficiency. The effect of upsampling, i.e., interpolating the observed spectrum to the resolution of the DBN, is more case dependent. We find no degradation of the recognition efficiency upsampling broad absorbing species such as $H_2O$ or $CH_4$ from resolutions as low as $R = 30$ to the native

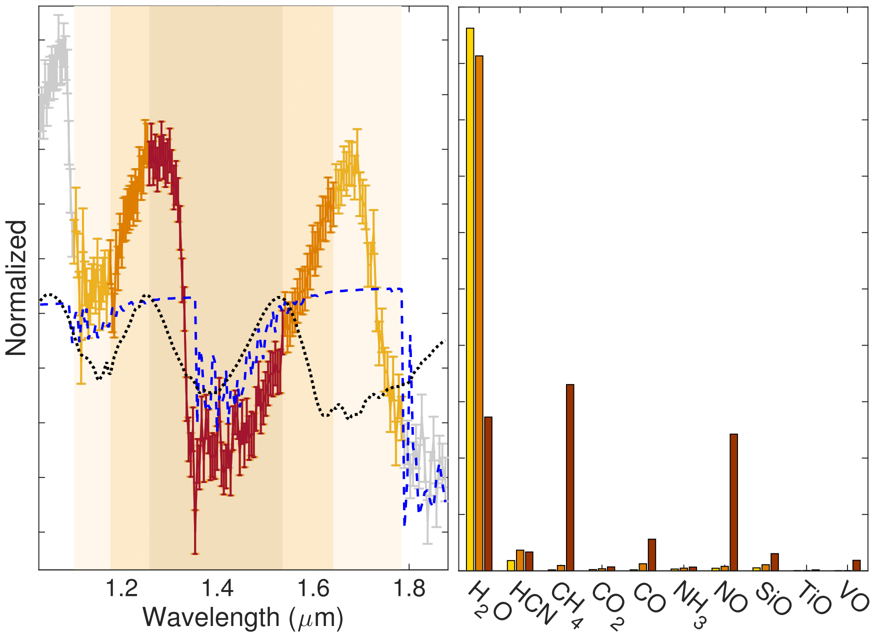

**Figure 4.** Similar to Figure 3. The left shows input spectrum at S/N = 20 for a normalized water spectrum in the *HST*/WFC3 G141 grism passband (yellow, 1.1–1.8 μm). Darker color shading represents progressively smaller passbands for which the recognition was performed. Blue dashed and black dotted lines show normalized spectra of NO and $CH_4$, respectively. The right shows the corresponding detection probability per molecule for the varying wavelength ranges. Water is readily recognized to be the main trace gas component except for the smallest bandpass considered where $H_2O$, $CH_4$, and NO are assigned roughly equal probabilities. As can be seen in the left plot, $H_2O$, $CH_4$, and NO normalized spectra all have very similar features when only the most restricted (darkest shaded) spectral range is considered.

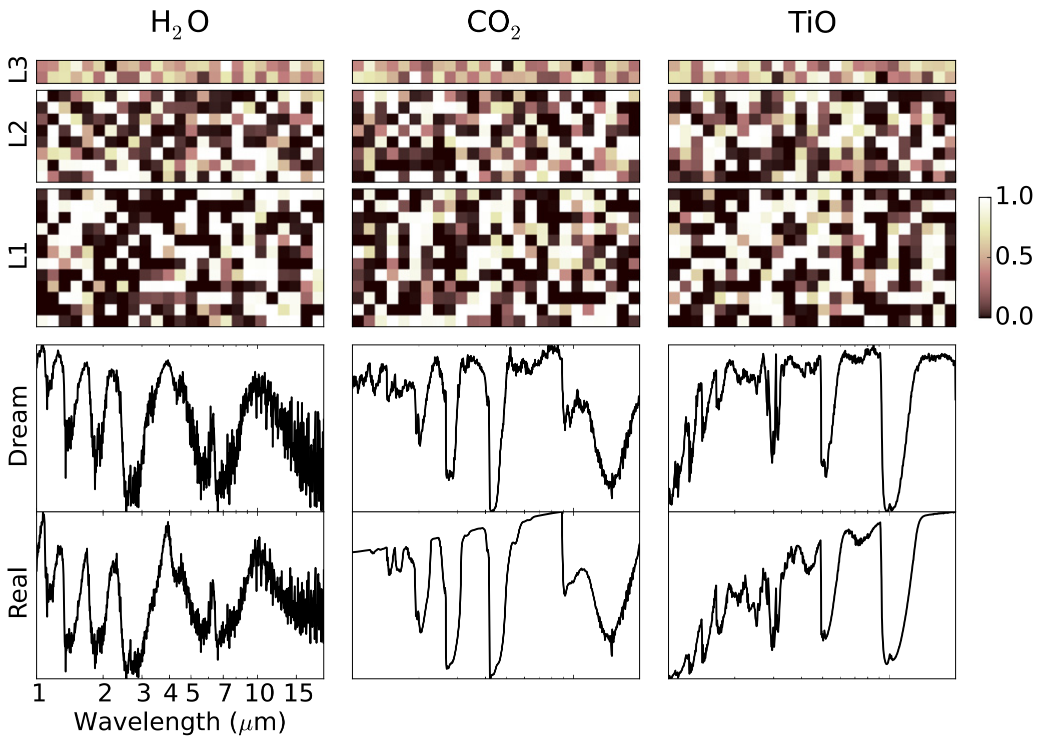

**Figure 5.** Spectral reconstruction (or "dreaming") of three molecules $H_2O$, $CO_2$, and TiO. The top three panels show neuron activations for the bottom (L1) to top (L3) Restricted Boltzmann Machine layers. The bottom two rows show normalized $H_2O$, $CO_2$, and TiO spectra reconstructed by the neural network and real data examples as comparison. The similarities between the "dreamed" and real spectral features are striking. This indicates a good representation of molecular features in the neural network.

resolution of the DBN. Here, the interpolation simply adds noise to the spectrum against which the DBN is very robust. Generally speaking, all of the molecules can be identified unless their features are strongly undersampled. Trace gases with more narrow emission/absorption bands (e.g., CO, NO) are hence more strongly affected. For the molecular mixtures considered here, we find a conservative lower limit of $R \sim 25$ (constant with $\lambda$) below which feature detection becomes difficult. It should be noted that a strongly undersampled spectrum will always be difficult to interpret independently of the methodology used.

## 5. DREAMING OF ATMOSPHERES

When *RobERt* is used for recognition purposes, we set the visible units to the values of the input spectrum and propagate the network forward (i.e., upwards) to obtain a classification label. Another approach to qualitatively check the convergence quality of the DBN is to reverse the network and propagate the network weights backwards (i.e., downwards) starting from a label. In other words, we activate the, e.g., $H_2O$ label, and *RobERt* will return what it "thinks" are the defining features of a water spectrum. This backwards propagation is commonly referred to as "dreaming" in the machine learning literature. Figure 5 shows dreams of three molecules, $H_2O$, $CO_2$, and TiO. We compare these dreams with real, normalized spectra with abundances of $1 \times 10^4$ underneath. The likeness of the dreamed spectra with real data is striking. L1, L2, and L3 represent the neural activations of the bottom, middle, and top RBMs, respectively. We find the neural activations in the dream state to be a useful indicator of the sparsity (i.e., number of units set to or close to zero) of the neural network and find networks with $\sim$10% average sparsity to yield the most accurate spectral reconstructions.

## 6. DISCUSSION

The size of the DBN is an important factor to be considered; *RobERt* consists of three RBMs of 500, 200, and 50 neurons from bottom to top, respectively. We find a three-layer DBN to work best but also find that networks with too many neurons per layer, particularly in the upper levels, lead to noisy reconstructions, low maximum likelihoods, and a poorer recognition performance. We attribute this effect to a high level of redundancy in the network, which introduces noise. As described above, by inspecting the neural activations during the dream state of *RobERt*, we can measure the sparsity of individual layers for individual states (i.e., molecule activations). Tests have shown that $\sim$10% in sparsity averaged across activation states produces the most robust and highest S/N networks. Smaller, simpler networks run the risk of not being able to differentiate between molecules correctly.

As stated previously, *RobERt* has only been trained on spectra containing one trace gas at a time. Despite this obvious limitation, in Section 4 we show that *RobERt* is indeed able to identify mixtures of molecules, though caveats to this capability should be mentioned. Similar to inspecting a spectrum by eye, *RobERt* is able to identify mixtures if the trace gas signatures are very different from one another (e.g., $H_2O$ and CO, Figure 5) or if sufficient wavelength coverage is provided (e.g., $CH_4$ and $H_2O$, Figure 3). The DBN struggles whenever either too little wavelength coverage is available (e.g., Figure 4) or the secondary trace gas is an order of magnitude less abundant than the primary absorber/emitter, i.e., secondary signatures imprint themselves as noise on the main absorber/emitter.

Though some of these limitations are fundamental (i.e., insufficient wavelength coverage, too low S/N, etc.), future work will investigate the use of convolutional deep-belief networks (e.g., Lee et al. 2011a) to boost recognition accuracy by learning the localized correlations in the observed spectra. Additionally, an updated supervised learning cost function is imaginable where not the identification of a single trace gas is rewarded but instead a "best ranking" of groups of molecules.

As pre-selector to the $\mathcal{T}$-REx retrieval suite, *RobERt* will provide rankings of the most likely molecules to be considered in the quantitative retrieval. This is an iterative process with the retrieval models increasing in complexity from the simplest atmospheres (containing only the few most likely molecular absorbers/emitters detected by *RobERt*) to more complex models (containing less likely opacities). The Bayes factor is the measure of convergence here (Waldmann et al. 2015b). In future implementations of *RobERt*, online learning will become important after its initial training phase is complete. With each new data set, *RobERt* will be able to update and improve its DBN, taking the $\mathcal{T}$-REx results as a labeled training set. Such an application is particularly suited as part of a larger data reduction/analysis pipeline for future large-scale ground- and space-based surveys.

## 7. CONCLUSION

In this paper, we present the use of deep belief networks in the identification and classification of exoplanetary emission spectra. We have shown that DBNs are well suited to identifying molecular signatures in extrasolar planet spectra. They are very robust to low S/Ns and are able to identify trace gases even when wavelength ranges are strongly restricted compared to the initial training setup. This property is important as training a DBN is relatively computationally intensive, and hence one would ideally want the trained DBN to be as universally applicable as possible. Their ability to abstract and generalize nonlinear systems very effectively makes DBNs an ideal tool for qualitative "preselection" of parameter spaces for spectral retrieval applications.

I.P.W. thanks G. Tinetti, R. Varley, M. Rocchetto, A. Tsiaras, and G. Morello for useful discussions. This work was supported by the ERC project 617119 (ExoLights) and STFC grant ST/P000282/1.

## REFERENCES

Agarwal, S., Abdalla, F. B., Feldman, H. A., Lahav, O., & Thomas, S. A. 2012, MNRAS, 424, 1409
Agarwal, S., Abdalla, F. B., Feldman, H. A., Lahav, O., & Thomas, S. A. 2014, MNRAS, 439, 2102
Bengio, Y. 2009, Learning Deep Architectures for AI, Vol. 2 (Boston, MA: Now Publishers)
Bengio, Y. 2012, Neural Networks: Tricks of the Trade (Berlin: Springer)
Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. 2007, Adv. Neural Inf. Process. Syst., 19, 153
Benneke, B., & Seager, S. 2013, ApJ, 778, 153
Bianchini, M., & Scarselli, F. 2014, Neural Networks Learn., 25, 1553
Bishop, C. M. 2006, Pattern Recognition and Machine Learning (Berlin: Springer)
Charbonneau, D., Berta, Z. K., Irwin, J., et al. 2009, Natur, 462, 891
Collister, A. A., & Lahav, O. 2004, PASP, 116, 345
Davison, A. C. 2008, Statistical Models (Cambridge: Cambridge Univ. Press)
Dieleman, S., Willett, K. W., & Dambre, J. 2015, ApJ, 450, 1441
du Buisson, L., Sivanandam, N., Bassett, B. A., & Smith, M. 2015, MNRAS, 454, 2026
Ellison, S. L., Teimoorinia, H., Mendel, J. T., & Rosario, D. J. 2015, MNRAS, 455, 370
Fischer, A., & Igel, C. 2014, Pattern Recognit., 47, 25
Freund, Y., & Haussler, D. 1992, in Advances in Neural Information Processing Systems 4, ed. J. Moody, S. Hanson, & R. Lippman (Cambridge, MA: MIT Press), 912
Griffith, C. A. 2014, RSPTA, 372, 30086

Head-Gordon, T., & Stillinger, F. H. 1993, PhRvE, 48, 1502
Hebb, L., Collier-Cameron, A., Loeillet, B., et al. 2009, ApJ, 693, 1920
Hilbe, J. H. 2009, Logistic Regression Models (London: Chapman and Hall)
Hinton, G. E. 2002, Neural Comput., 14, 1771
Hinton, G. E. 2006, Sci, 313, 504
Hinton, G. E. 2007, Trends Cognit. Sci., 11, 428
Hinton, G. E. 2012, Neural Networks: Tricks of the Trade (Berlin: Springer)
Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. 2012, arXiv:1207.0580
Huertas-Company, M., Gravet, R., Cabrera-Vives, G., et al. 2015, ApJ, 221, 8
Hung, J. C., Wang, C.-S., Yang, C.-Y., Chiu, M.-S., & Yee, G. 2005, in 19th Int. Conf. on Advanced Information Networking and Applications (AINA'05), ed. E. Kranakis (Heidelberg: Springer), 157
Jaitly, N., & Hinton, G. 2011, in ICASSP 2011—2011 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), ed. P. Tichavsky, H. Cernocky, & A. Prochazka (Piscataway, NJ: IEEE), 5884
Karpenka, N. V., Feroz, F., & Hobson, M. P. 2013, MNRAS, 429, 1278
Krizhevsky, A. 2009, Learning Multiple Layers of Features from Tiny Images, Tech. Rep. 04.08.2009 (Toronto: Univ. Toronto)
Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, Advances in Neural, 1097
Le Roux, N., & Bengio, Y. 2008, Neural Comput., 20, 1631
Le Roux, N., & Bengio, Y. 2010, Neural Comput., 22, 2192
Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. 2011a, Commun. ACM, 54, 95
Lee, J. M., Fletcher, L. N., & Irwin, P. G. J. 2011b, MNRAS, 420, 170
Li, M., Zhang, T., Chen, Y., & Smola, A. J. 2014, in Proc. 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining KDD'14, ed. S. Macskassy & C. Perlich (New York: ACM), 661
Line, M. R., Zhang, X., Vasisht, G., et al. 2012, ApJ, 749, 93
Liu, P., Han, S., Meng, Z., & Tong, Y. 2014, in 2014 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), ed. S. Dickinson, D. Metaxas, & M. Turk (Piscataway, NJ: IEEE), 1805
Madhusudhan, N., & Seager, S. 2009, ApJ, 707, 24
Montavon, G., Orr, G., & Müller, K.-R. 2012, Neural Networks: Tricks of the Trade, Vol. 7700 (Berlin: Springer)
Plebe, A. 2007, Neurocomputing, 70, 2060
Pradeep, R., & Kumaraswamy, R. 2014, in National Conf. on Communication, Signal Processing and Networking (NCCSN), ed. A. K. Chaturvedi (Piscataway, NJ: IEEE), 1
Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 2007, Numerical Recipes The Art of Scientific Computing (3rd ed.; New York: Cambridge Univ. Press)
Reis, R. R. R., Soares-Santos, M., Annis, J., et al. 2012, ApJ, 747, 59
Shen, S., Li, X., & Zhu, S. 2015, ElL, 51, 905
Spencer, M., Eickholt, J., & Cheng, J. 2015, IEEE/ACM Trans. Comput. Biol. Bioinf., 12, 103
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. 2014, J. Mach. Learn. Res., 15, 1929
Waldmann, I. P., Rocchetto, M., Tinetti, G., et al. 2015a, ApJ, 813, 13
Waldmann, I. P., Tinetti, G., Rocchetto, M., et al. 2015b, ApJ, 802, 107
Wang, N., Melchior, J., & Wiskott, L. 2016, A&A, 585, A126
West, R. G., Almenara, J. M., Anderson, D. R., et al. 2013, arXiv.org, 5607
Wu, C. H., & McLarty, J. W. 2012, Neural Networks and Genome Informatics (Amsterdam: Elsevier)
Zhang, X.-L., & Wu, J. 2013, IEEE Trans. Audio Speech Lang. Process., 21, 697