JARS Journal of Applied Remote Sensing

RESEARCH PAPER

HyperBlend leaf simulator: improvements on simulation speed, generalizability, and parameterization

Kimmo A. Riihiaho,* Leevi Lind, and Ilkka Pölönen
University of Jyväskylä, Faculty of Information Technology, Jyväskylä, Finland

ABSTRACT. In recent decades, remote sensing of vegetation by hyperspectral imaging has been of great interest. An important part in interpreting the remotely sensed spectral data is played by simulators, which approximate the connection between plants' biophysical and biochemical properties and detected spectral response. We introduce improvements and new features to recently published hyperspectral leaf model HyperBlend. We present two methods for increasing simulation speed of the model up to 200 times faster with slight decrease in simulation accuracy. We integrate the well-known PROSPECT leaf model into HyperBlend allowing us to use the PROSPECT parametrization for leaf simulation. For the first time, we show that HyperBlend generalizes well and can be used to accurately simulate a wide variety of plant leaf spectra. HyperBlend is available as an open-source Python project under MIT license in a GitHub repository available at: https://github.com/silmae/hyperblend.

Keywords: spectral; hyperspectral; leaf; simulation; efficient computing; forestry; neural network

Paper 220660G received Nov. 26, 2022; revised Jul. 24, 2023; accepted Aug. 23, 2023; published Sep. 12, 2023.

1 Introduction

Plant's growth and development can be negatively affected by biotic or abiotic stress factors.[1] Biotic stress is caused by other organisms, such as insects, pathogens, or other plants. Abiotic stress refers to physical and chemical factors in growth environment, such as pollution, drought, and nutrient availability. Remotely sensing vegetation condition and health by hyperspectral imaging (imaging spectroscopy) has been widely studied in recent decades.[2] High spectral resolution of hyperspectral images enables finding spectral bands that are most distinctly affected by a certain stressor.[2] In forestry, hyperspectral remote sensing has been used to detect pests and pathogens,[3–5] ozone damage,[6] effects of pollution,[7] and identifying tree species,[8] for example.

Analysis of remotely sensed spectral images relies largely on machine learning (ML) algorithms, although statistical tests have also been widely applied.[2] As training of ML algorithms requires large amounts of training data cumbersome to gather from real-world measurements, it is often generated with simulators. In practice, the simulators, or leaf optical properties models, mimic spectral response of plants with known biophysical and biochemical properties. Predicting the presence and type of stressors from a real hyperspectral image is done by finding spectral features that match the simulation, i.e., the simulation problem is inverted.

*Address all correspondence to Kimmo A. Riihiaho, kimmo.a.riihiaho@jyu.fi

A rough taxonomy of the leaf optical properties models can be established by examining the scale that they operate on: cell scale models depict the intricate internal structure of a plant leaf,[9–11] leaf scale models see the leaves as a whole,[12,13] and canopy scale models simulate the reflectance of groups of plants.[14–17] Canopy scale simulators are often statistical extensions of leaf models with some additional elements, such as soil reflectance. Canopy scale models can simulate remotely sensed hyperspectral images and thus, they are essential in training ML algorithms.

Most existing canopy scale simulators are developed to simulate high altitude imaging where an imager is mounted to a satellite or an airplane. The first satellite with spectral imaging equipment was the Landsat 1 mission (originally called Earth Resources Technology Satellite, ERTS) launched in 1972 that was equipped with four-band multispectral imager.[18] The first airplane-based hyperspectral imaging sensor capable of measuring continuous spectra with 10 nm bandwidth in the 400 to 2500 nm range commonly used in vegetation spectroscopy was the airborne visible/infrared imaging spectrometer (AVIRIS).[19] AVIRIS produced its first hyperspectral images in 1987.[19] With growing popularity and decreasing prices of unmanned aerial vehicles (commonly called drones) and hyperspectral imagers, there is need for leaf optical simulators aimed at low altitude imaging. For a comprehensive review of drone-based imaging, see Ref. 20.

One of the most noticeable differences between high and low altitude imaging is in the ground pixel size (spatial resolution), i.e., how many meters in ground a single pixel covers. The ground pixel size of airplane-based imagers range from 0.5 to 10 m and is even greater in space-borne imagers.[2] Utilizing statistical approximations for canopy structure, such as the commonly used average leaf inclination angle (LIA), is well-founded in these applications. However, in low altitude drone-based imaging, ground pixel size may be centimeters or even millimeters depending on the sensor and altitude. As such, the concept of average LIA is not applicable since a pixel can be filled by single leaf and the inclination angle of that leaf is what it is. Simulating low altitude imaging demands intricate description of plant geometry.

One of the most recent attempts to address the need for a low altitude vegetation simulator is project HyperBlend.[21] HyperBlend's working principle is shown in Fig. 1. The simulator requires reflectance and transmittance spectra of a leaf as an input; they are called target spectra. The target spectra can originate from real-world measurements, or they can be generated by a third party leaf simulator. In the following conversion step, HyperBlend creates a renderable three-dimensionl (3D)-model of a leaf that respects spectral reflectance and transmittance properties defined by the target spectra. Copies of the leaves are then placed on 3D-models of trees, which in turn are placed on 3D-model of a forest floor to form a forest canopy model. In Fig. 2, an example of a tree generated by this process is shown. Finally, a virtual camera renders an image cube of the forest accompanied by pixel labels that have information on which object was present in each pixel.

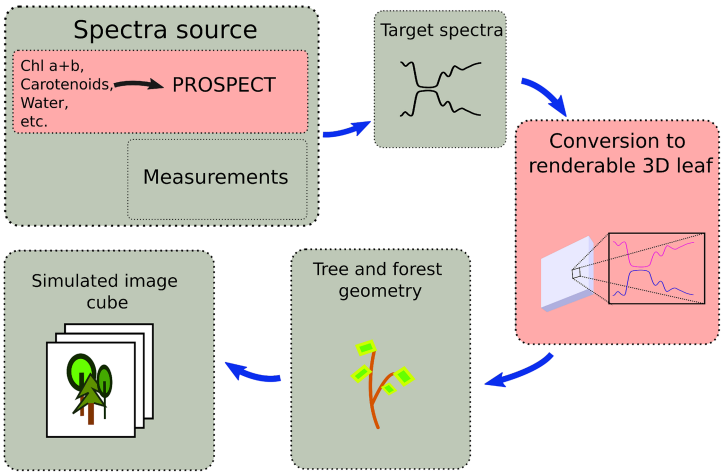
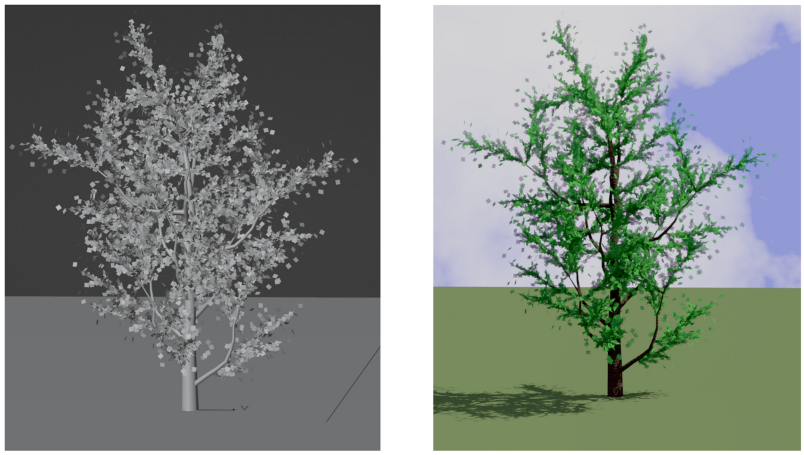


Fig. 1 Working principle of HyperBlend canopy model. Steps considered in this paper are highlighted in red.



(a) (b)

Fig. 2 An example of a tree generated with HyperBlend. (a) Pure geometry without shading. (b) Tree model shaded with arbitrary RGB colors.

Rendering is a computer graphics term that means the process of producing a synthetic image of a virtual scene.[22] HyperBlend utilizes 3D-modeling and rendering software Blender[23] and especially its path tracing rendering engine Cycles. Path tracing is a rendering technique that numerically approximates Kajiya's rendering equation.[24] Kajiya's path tracing only accounts for light interactions on object's surface.[24] Later efforts of Lafortune extended the algorithm to include interactions inside an object as well.[25] HyperBlend takes advantage of volumetric path tracing by depicting a leaf as translucent media that can scatter and absorb light.

In this study, we only consider HyperBlend's leaf scale operations, i.e., we exclude tree and forest geometry generation and construction of the image cube. HyperBlend's leaf model depicts a leaf as a flat box whose volume is filled with absorbing and scattering particles. Desired reflectance and transmittance properties can be achieved by adjusting absorbing particle density, scattering particle density, forward versus backwards scattering tendency, and mixing factor of the two particle types. For more detailed explanation of the working principle of HyperBlend's leaf model, see Ref. 21. HyperBlend is written in Python 3.7[26] and the source code is available in a GitHub repository at: https://github.com/silmae/hyperblend. The version that this paper refers to is tagged with version number 0.2.0.

While the conversion step (highlighted in red in Fig. 1) in the original version of HyperBlend produced highly accurate results with root mean squared error (RMSE) of ∼1%,[21] in this paper, we wish to address some of its limitations: simulation speed, parameterization, and generalizability.

The conversion step of the original version was very slow taking more than an hour to convert a pair of reflectance and transmittance spectra into renderable leaf material parameters on a consumer-grade computer.[21] We present two new methods for the conversion that are computationally more efficient: a surface fitting method in Sec. 2.2 and a neural network (NN) method in Sec. 2.3. Training data generation for the new methods is explained in Sec. 2.1.

HyperBlend relies on an external source of target spectra (top left corner in Fig. 1). Target spectra should include associated biophysical and biochemical parameters, such as water content and chlorophyll content, which explain the shape of the spectra. Although real-world datasets where measured spectra are accompanied by biophysical and biochemical data exist, it is often more convenient to produce such data with a leaf simulator. As HyperBlend's conversion step is agnostic to the source of such data, both methods can be used. In Sec. 2.4, we include an implementation of the well-known PROSPECT[13,27–31] leaf simulator into HyperBlend. The role of the PROSPECT model in HyperBlend is the same as, for example, in PROSAIL[17,29] where PROSPECT acts as an underlying leaf model that is combined with canopy model SAIL.[14] From the perspective of the user of HyperBlend, this also hides the unwieldy rendering parameters under much simpler PROSPECT parameterization.

Lastly, as the validation of the original model in Ref. 21 utilized leaf samples of only a single species, HyperBlend's generalizability was not verified. In this study, we run PROSPECT with

randomized parameters to generate a wide variety of target spectra to show that HyperBlend can accurately reproduce leaf spectra also in a general case.

We conduct two simulation experiments in Sec. 3. The first one is a repeat experiment of Ref. 21, which compares simulation speed and accuracy of the presented methods against the original simulation method. The second experiment tests how well HyperBlend can reproduce arbitrary leaf spectra. Results of the experiments are presented in Sec. 4. Future developement and limitations are discussed in Sec. 5 and Sec. 6 concludes the paper.

2 Materials and Methods

The leaf model in HyperBlend works by adjusting four leaf matter parameters (absorbing particle density ρ_a, scattering particle density ρ_s, scattering anisotropy α, and mixing factor of the two particle types β) until target reflection and transmission values are achieved. The target can be obtained from real world measurements or from another simulator. As Blender is not capable of working with spectral quantities, HyperBlend simulates each wavelength band separately before compiling them into a spectrum. Thus, we are mainly interested in which values of ρ_a, ρ_s, α, β can produce a certain reflectance-transmittance pair (RT pair).

Letting R be reflectance in closed interval [0, 1] and T transmittance in the same interval, we can denote the four leaf material parameters by four respective functions

$$f_i(r,t)\colon \mathbb{R}^2 \to [0,1], \tag{1}$$

where $i \in \{0,1,2,3\}$ and $r \in R$, $t \in T$. Values of leaf material parameters range from 0 to 1. Based on behavior of leaf material parameter spectra reported in Ref. 21, we can assume functions f_i to be smooth. If we ignore the fact that produced RT pairs are probabilistic due to Monte Carlo–process in path tracing and optimization process in the HyperBlend itself, certain leaf material parameters ρ_a, ρ_s, α, β should always produce the same RT pair. With these assumptions, we can generate an evenly spaced set of points in RT-space and interpolate f_i in-between. Assuming we can find functions that represent the phenomena accurately enough, this approach allows reducing the time-consuming optimization process of the original simulation method into four simple function evaluations.

In the rest of the paper, we will use three different methods in running HyperBlend's leaf model. The first is the original optimization-based method presented in Ref. 21 which we will call the "Original" method. Two new methods are presented below: "Surface" fitting method and NN method. These short names will be used for brevity throughout the rest of the paper.

All computations were done with a consumer-grade machine with Intel i9-9900K 8-core processor. Hereafter, all the plots in this paper are plotted with Matplotlib.[32]

2.1 Training Data Generation

In order to find functions f_i in Eq. (1), we generated evenly spaced RT pairs with following rules:

$$\begin{aligned} |r-t| &\le 0.25 \\ r+t &\le 1 \\ r,t &\ge 0. \end{aligned} \tag{2}$$

The first rule ensures that reflectance and transmittance are always close to each other as is usually the case in plant leaves. The threshold value of 0.25 was selected because the accuracy of the Original optimization method begins to decrease when reflectance and transmittance values are too dissimilar. The second rule ensures that no more photons can be reflected and transmitted than what is incident on the leaf, i.e., bioluminescence is not allowed. The last rule ensures that reflectance or transmittance cannot be negative.

After generating the RT pairs, Original HyperBlend leaf model was run for each of them to produce leaf material parameters ρ_a, ρ_s, α, β, which, in turn, produce simulated reflectance $\hat{r}$ and simulated transmittance $\hat{t}$. To avoid grossly bad data points, a point was removed from the training set if either $|r-\hat{r}| \ge 0.01$ or $|t-\hat{t}| \ge 0.01$.

From initial point generation runs, we noticed that points where $t > 3.8r + 0.02$ or $t < 0.5r - 0.035$ were almost always exceeding the pruning limit. We added those areas as

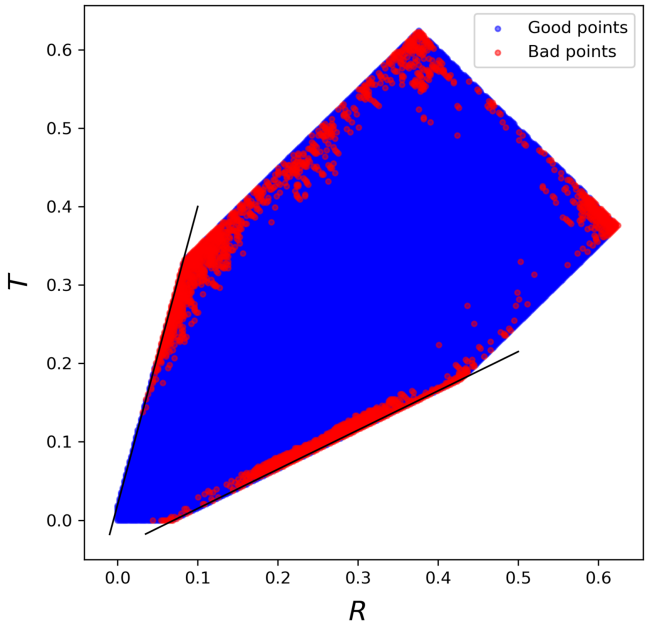


Fig. 3 Generated training data points. Black lines show additional restrictions. Points shown in red exceed the 0.01 error threshold and were pruned from the final training set. Only points shown in blue were used in final training. Axes R and T stand for reflectance and transmittance, respectively.

additional conditions where training points were not generated to avoid unnecessary computing of points that would be pruned later. Figure 3 shows generated training data, the additional conditions, and pruned points. One can notice from the figure that accuracy of the simulation decreases toward the edges of the area. The final number of accepted training points was 44,284. The training data set is available at https://osf.io/trhf8/.

2.2 Surface Fitting

Now that we have simulated leaf material parameters for each RT pair, we can plot them into R, T, z-space, where z-axis shows the value of each leaf material parameter for given r and t (see Fig. 4). We assumed that the leaf material parameters behave smoothly in that space so we can fit an analytical function to approximate each material parameter.

The first suggested method to find functions f_i in Eq. (1) is to fit a surface into training data. As shown in Fig. 4, the shape of the surfaces varies. We used exponential, logarithmic, and polynomial functions depending on the shape of the surface:

$$\begin{aligned}
\rho_a &= f_0(r,t) = a_0 \mathrm{e}^{b_0 r} + c_0 \mathrm{e}^{d_0 t} \\
\rho_s &= f_1(r,t) = a_1 \log(b_1 r) + c_1 \log(d_1 t) \\
\alpha &= f_2(r,t) = a_2 r^{b_2} + c_2 t^{d_2} \\
\beta &= f_3(r,t) = a_3 \mathrm{e}^{b_3 r} + c_3 \mathrm{e}^{d_3 t},
\end{aligned} \tag{3}$$

where a_i, b_i, c_i, d_i are constants to fit each function into the training data. In this case, function fitting is a non-linear least squares problem that minimizes the squared $L1$ distance of the prediction function $f_i(r,t)$ to the training data $z_i(r,t)$ at the same point

$$\min_{a_i,b_i,c_i,d_i} \sum_{k=1}^{n} (z_i(r_k,t_k) - f_i(r_k,t_k))^2, \tag{4}$$

where n is the number of training points, subscript i refers each of the four leaf material parameters. This kind of fitting problem can readily be solved with Levenberg–Marquardt algorithm.[33,34] For the code implementation, we used SciPy's[35] `optimize.curve_fit` method with the default options.

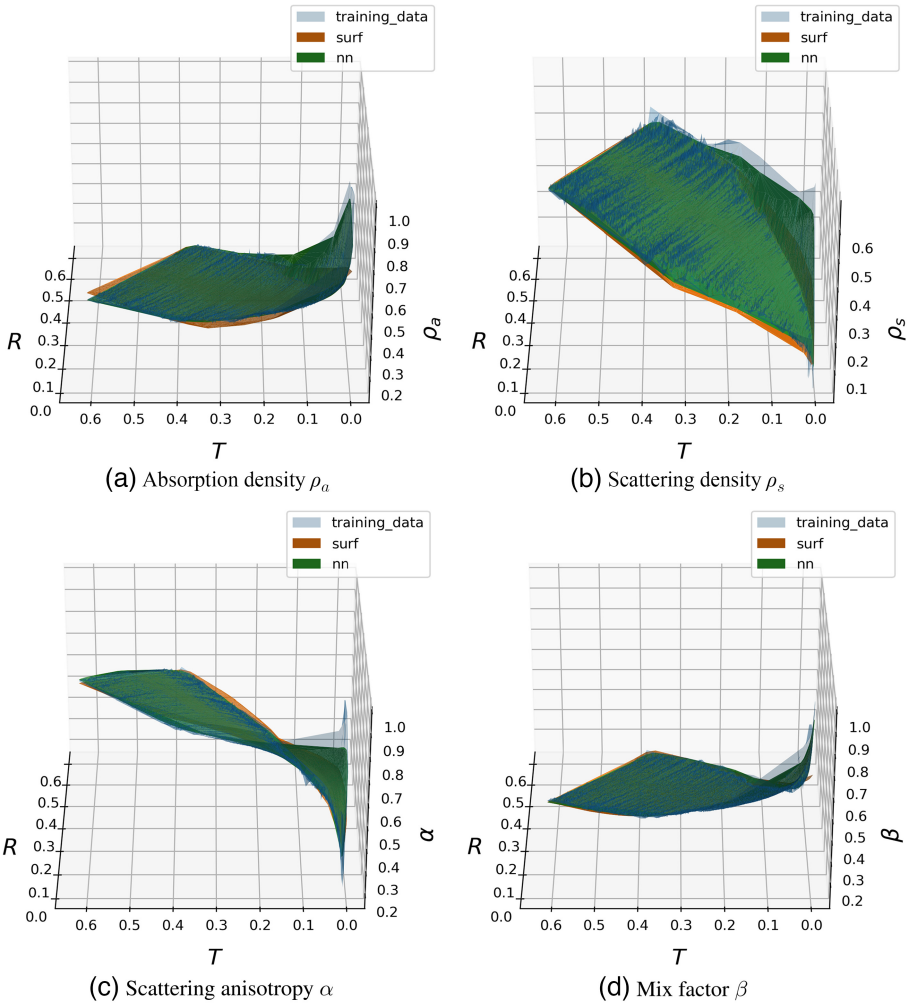


Fig. 4 Leaf material parameter surfaces shown as training data (light gray), surface fitting (orange), and trained NN (green). The differences between surface fitting and NN are easiest to see at the bottom right corner of the plots. For all parameters, the NN fits the training data better than the surface fit. (a) Absorption density ρ_a, (b) scattering density ρ_s, (c) scattering anisotropy α, and (d) mix factor β.

The surfaces align fairly well with most of the training data but problems do occur especially when both reflectance and transmittance are close to zero. This behavior is especially apparent in the plot of anisotropy parameter α in Fig. 4(c) where the surface twists into almost vertical orientation near the origin. The training data also have ripples in the direction of $r = t$ as shown in Fig. 4. These features appear too regular to be random noise, suggesting that they may contain some relevant information.

2.3 Neural Network

In theory, a Neural Network should be able to adapt into any smooth surface,[36] which makes it a good candidate for approximating the leaf material functions f_i. A conventional multilayer NN consists of an input layer, one or more hidden layers, and an output layer. The input layer has a node for each input value of the problem and the output layer has one node for each output value, respectively. Hidden layer(s) may have any number of nodes. Each input node is connected to each node of the next layer by a weight. The first hidden layer is similarly connected to the following layer until the output layer is reached. The network is trained with training data by adjusting the weights of the connections until it can predict the output from the input with sufficient accuracy. In the remainder of this work, we expect the reader to be familiar with neural networks. For more details, see Refs. 37 and 38, for example.

By experimenting with different architectures manually, we ended up with a network with five hidden layers that are 1000 nodes wide each. The input layer has two nodes to represent a single RT pair and output layer has four nodes representing the four leaf material parameters. Used activation function for each hidden layer was leaky rectified linear unit.[39] We employed mean squared error as a loss function. For local optimizer, we used Adam[40] with suggested default parameters. Changing the default parameters into either direction seemed to have none or negative effect to the result.

Training data were split into two parts where 10% of the data were reserved for testing. Validation of the network was done separately using real leaf data as shown later in Sec. 4. By test data, we refer to data used during training and by validation data we mean running the NN after training with data it has not previously seen.

The network was allowed to train maximum of 300 epochs. Early stopping criterion was set to require improvement in test loss at least once in 30 epochs. Most training loops achieved the lowest loss after few dozen epochs and did not improve even when disabling the early stopping and allowing to run for all 300 epochs. The NN was implemented and trained using deep learning library PyTorch.[41]

A low test loss at training time did not correlate with good validation performance with real leaf spectra, which made hyperparameter tuning difficult. For example, a very small network with 10 hidden layers that were 10 nodes wide produced very good test results, but validation showed significant and systematic errors.

It is very difficult to plot the parameter surfaces because they are intersecting and close to each other. If looking closely to plots in Fig. 4, one can see that the NN surface in green is a closer match to training data shown in gray than the Surface fitting plots shown in orange. The difference can best be seen in the right side of Figs. 4(a) and 4(d), where the orange surface does not follow the grey one at all. The NN method did better job at fitting the vertical parts of scattering density ρ_s and scattering anisotropy α in Figs. 4(b) and 4(c) but the fit is not perfect. Even the NN method did not catch the rippling effect in training data, which can be seen as intersecting of the green and gray surfaces.

2.4 Reparameterization

The first version of HyperBlend[21] was a proof-of-concept in nature and not much attention to practicality of the parameterization was given. HyperBlend's leaf material parameters, let us call them HB parameters for short, only make sense as volume shader parameters used for rendering images in blender. Although they have physical meaning in how light is scattered and absorbed inside a leaf, they are not meaningful as biophysical and biochemical properties that could be used in analyzing plant condition.

The HB parameters are impractical to use as four parameters for each wavelength band are needed. For a spectrum from 400 to 2500 nm with 5 nm resolution, 1680 HB parameters are required. In comparison, the original PROSPECT model used three input parameters to produce a fully continuous spectrum (though usually sampled at 1 nm intervals) with the same spectral range.[27] Subsequent PROSPECT versions use more parameters but still less than a dozen.

In the conversion step illustrated in Fig. 1, HyperBlend can autonomously find the HB parameters for a given spectrum, which allows parameter conversion from any external simulator that produces reflectance–transmittance spectra. Parameter conversion allows exposing meaningful biophysical and biochemical parameters, such as chlorophyll content and carotenoid content to the user while hiding the HB parameters used for rendering. In this study, we implemented the conversion for PROSPECT as an example because it is widely used in the field. Note that the conversion step is not equal to the parameter conversion discussed here. The conversion step can also be run with spectra that do not include associated biophysical and biochemical parameters, such as spectral measurements of real-world leaves used later in experiment 1 in Sec. 3.

The conversion from PROSPECT parameters to HB parameters is relatively simple because HyperBlend works with reflectance–transmittance pairs that can be obtained from PROSPECT simulation. First, the user gives a set of PROSPECT parameters to HyperBlend. PROSPECT simulation is then run inside HyperBlend and a (PROSPECT) leaf spectrum is generated. Then, HyperBlend uses one of the simulation methods presented in this study to find HB parameters that produce matching spectrum.

Because HyperBlend simulates a single wavelength band at a time, its spectral resolution can be set arbitrarily high by providing a discontinuous spectrum with arbitrarily high sampling frequency. The cost of higher frequency is longer simulation time. In addition, there is no specific reason to use constant intervals between samples, which makes it easy to simulate imagers consisting of multiple sensors with varying resolution or multispectral sensors with few channels that may have large and uneven spacing.

3 Experiments

To assess the performance of the new Surface and NN simulation methods, we conducted two experiments. The first experiment replicates the one in the original paper.[21] The second experiment tests HyperBlend's capability to generalize plant leaf spectra with varying characteristics. Both experiments were run with 5 nm spectral resolution.

Experiment 1 is essentially the same experiment that was conducted in the original HyperBlend paper.[21] The only difference is that now, in addition to the Original method, we also use presented Surface and NN methods. We use the exact same data set of 28 spectral measurements of silver birch (*Betula pendula*) leaves collected by Ref. 42. For more details on the original experiment, see Ref. 21. The main reason for this experiment is to compare the simulation time and accuracy of the old method to the new methods presented above.

Experiment 2 tests generalizability of HyperBlend. HyperBlend has previously been tested only against the data set used in experiment 1 that consists of a samples of single tree species, so HyperBlend's generalizability has not previously been demonstrated. This experiment also shows that the parameter conversion from PROSPECT parameters is valid. We assessed the accuracy of conversion by generating 100 random leaf spectra with PROSPECT. For random generation of the spectra, we used the standard PROSPECT leaf of Ref. 43 (p. 275) as a basis, and let each parameter vary independently by drawing from normal distribution centered at the standard value. Drawn values were clipped to range suggested by Ref. 44 whose PROSPECT implementation we adapted to our code base. Mean and variance for normal distributions along with clipping ranges for each PROSPECT parameter are shown in Table 1. Experiment 2 was not run with Original method due to the excessive computation time this would have required.

4 Results

Mean errors and simulation times for both experiments are shown in Table 2. Mean errors are shown over all samples and over the whole spectrum. Simulation time is the mean over all samples.

In the first experiment with birch leaf data, both new methods are roughly 200 times faster than the Original with 11 to 12 s per sample. For the whole 28 sample set in experiment 1, this means reducing simulation time from 16 h to less than 6 min. Simulation time of presented methods stay consistent in experiment 2, which shows that simulation speed is constant

Table 1 PROSPECT spectra were generated by drawing from normal distribution using mean and variance tabulated here. If necessary, drawn values were clipped to the shown range.

Parameter	Mean (μ)	Variance (σ^2)	Range	Unit of measure
N	1.5	0.34	0.8 to 2.5	N/A
Chlorophyll a+b	32.0	16.0	0.0 to 80.0	$\mu g/cm^2$
Carotenoids	8.0	4.0	0.0 to 20.0	$\mu g/cm^2$
Brown pigments	0.0	0.2	0.0 to 1.0	N/A
Water thickness	0.016	0.01	0.0 to 0.05	cm
Dry matter	0.009	0.004	0.0 to 0.02	g/cm^2
Anthocyanins	0.0	8.0	0.0 to 40.0	$\mu g/cm^2$

Table 2 Comparison of simulation speed and errors between Original method, Surface method, and NN method. Simulation speed is reported in seconds per sample and errors for reflectance (R) and transmittance (T) as RMSE. Experiment 2 was not run with Original method. Simulation times for Surface and NN are roughly 200 times smaller than that of the Original. While both sacrifice some accuracy, NN is consistently more accurate than Surface.

Experiment 1 (birch leaves)			
	Original	Surface	NN
s/sample	2096	11	12
R error	0.0018	0.0111	0.0036
T error	0.0012	0.0128	0.0024
Experiment 2 (PROSPECT leaves)			
		Surface	NN
s/sample		13	13
R error		0.0111	0.0031
T error		0.0197	0.0047

regardless of the shape of the simulated spectra. This is not true with the Original method, as the optimization process takes more time if the starting guess is not close to the optimal solution.

The cost of faster simulation is increase in simulation error. RMSE of the Surface method in experiment 1 is roughly 10 times larger than the Original method. The error in NN is roughly doubled. We consider doubling the error for 200 times speedup a good trade. The errors of experiment 2 show that the accuracy of NN method does not suffer even when more difficult data are used. The Surface method, on the other hand, has difficulties especially in simulating transmittance.

More detailed spectrum-wise results of the experiment 1 are shown in Fig. 5. Even if the mean error 0.01 of the Surface method does not seem much, it is clearly visible in several areas of mean spectrum in Fig. 5(b). Large difficulties can be seen in short wavelengths (<500 nm) and in near infrared (NIR) plateau (700 to 1300 nm) on both reflectance and transmittance and in long wavelengths (>2100 nm) transmittance. The NN method in Fig. 5(c) does not show similar difficulties.

Similar plots for experiment 2 are shown in Fig. 6. The accuracy of the NN method is similar to experiment 1, even though some over-estimation in transmittance can be seen in the NIR plateau in Fig. 6(b). The Surface method, on the other hand, has great difficulties in estimation of transmittance throughout the spectrum shown in Fig. 6(a). Estimate of the reflectance is nearly perfect with NN and on par with experiment 1 with Surface.

To get more insight on why the errors are greater in experiment 2 than in experiment 1, we examined the solitary samples. Simulation results for sample number 59 are shown in Fig. 7. Inspecting the target spectra shown in black, one notices that the difference of reflectance and transmittance is quite large. For example, on wavelength 1000 nm in NIR plateau, the reflectance is 0.26 and transmittance is 0.60. The difference 0.34 is much greater than 0.25 that was allowed in our training data. Lack of training data depicting this kind of situation explains why Surface and NN perform poorly with this and seven other samples with similar difference in reflectance and transmittance.

The similarity condition 0.25 was set in Eq. (2) because most of the erroneous points in generated training data were concentrated on the areas of greatest dissimilarity between reflectance and transmittance. For further verification that the similarity condition is indeed crucial, we simulated sample 59 with the Original method. The result of that simulation in Fig. 7(a) clearly shows that the Original method is unstable in the NIR plateau where some wavelengths are very poorly simulated while others are at least approximately correct. Based on that behavior,

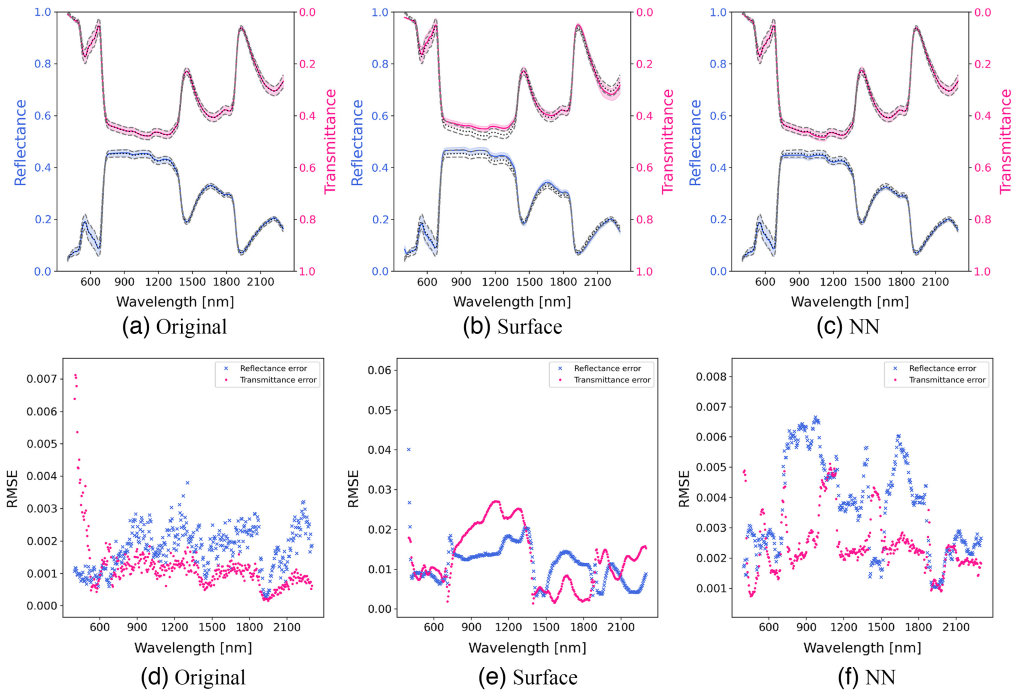


Fig. 5 Mean spectra (top row) and mean RMSE (bottom row) of the 28 silver birch samples of experiment 1. In the top row plots, target mean spectra plotted as black dotted line, target variance as gray dashed line, simulated mean as colored solid line, and simulated variance as colored shadow. Simulation error of Surface method is one order of magnitude greater than that of the other two methods. (a) Original, (b) Suface, (c) NN, (d) Original, (e) Surface, and (f) NN

it seems that the optimal solution does indeed exist, but it has not been reached in all cases. On the other hand, looking at long wavelengths, correct reflectance is never reached so the solution might not exist at all. Both Surface in Fig. 7(b) and NN in Fig. 7(c) can find approximately correct reflectance spectra but fail spectacularly with transmittance. The other seven poorly simulated spectra (not shown) exhibit the same behavior.

We assume that the reason why simulation of dissimilar reflectance and transmittance fails has to do with scattering anisotropy parameter of cycle's volume scattering shader. We have noticed that it can behave discontinuously when too small or too great values are used. Scattering anisotropy is mainly responsible for providing required asymmetry between reflectance and transmittance. For example, increasing scattering anisotropy favors forward scattering allowing more photons to pass the virtual leaf, which increases transmittance and decreases reflectance.

5 Discussion

With the two new methods presented, HyperBlend can now convert spectra to renderable leaf parameters with three different methods: Original optimization method, Surface fitting method, and NN method. If maximal accuracy is needed, we recommend using Original and for vastly faster evaluation when slight inaccuracy can be allowed, we recommend using NN. We do not recommend using Surface at all because while it is as fast as NN, its accuracy is significantly worse.

Regardless of the method, the most noticeable error is made when reflectance and transmittance are too dissimilar. This should not affect modeling real leaves too much as they tend to have fairly similar reflectance and transmittance spectra. As Ref. 43 points out, drawing PROSPECT parameters as independent variables does not represent natural interconnections of biophysical and biochemical parameters, which may result in spectra that does not occur in real leaves.

Regarding the failure to simulate dissimilar reflectance and transmittance with the presented methods, one solution could be to generate more dissimilar training data. As the Original method

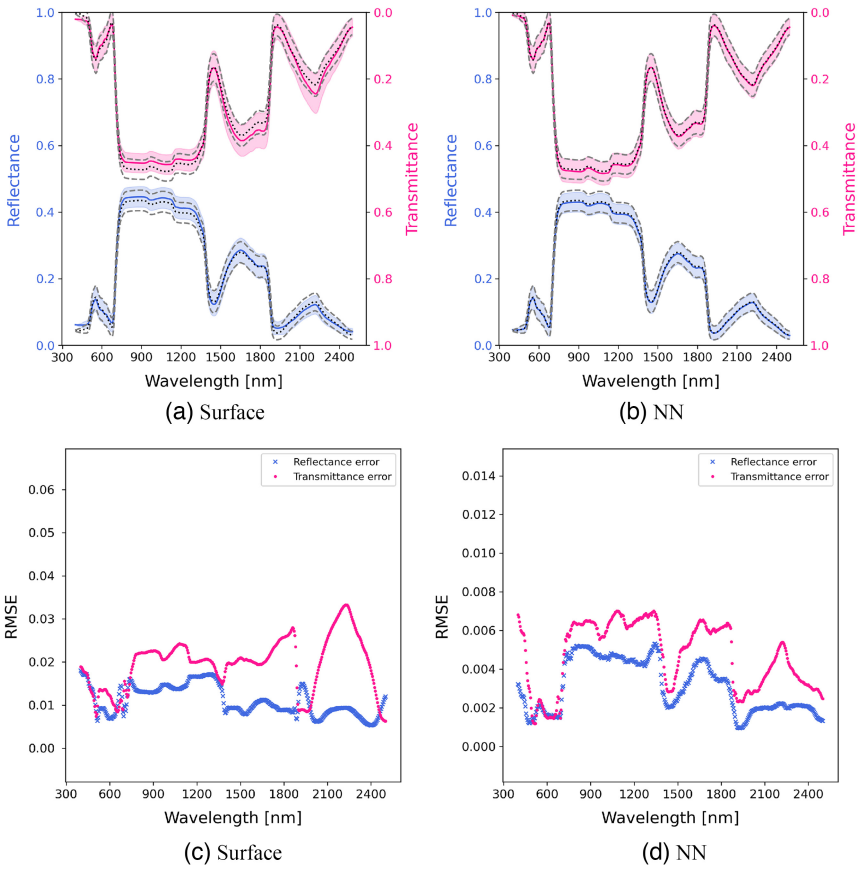


Fig. 6 Mean spectra (top row) and mean error (bottom row) of the of 100 PROSPECT-generated random leaf spectra samples of experiment 2. Target (i.e., PROSPECT-generated) mean spectra plotted as black dotted line, target variance as gray dashed line, simulated mean as colored solid line, and simulated variance as colored shadow. Simulation errors of both methods are slightly higher than in experiment 1. Again, the error of the Surface method is one order of magnitude higher than that of the NN method. (a) Surface, (b) NN, (c) Surface, and (d) NN.

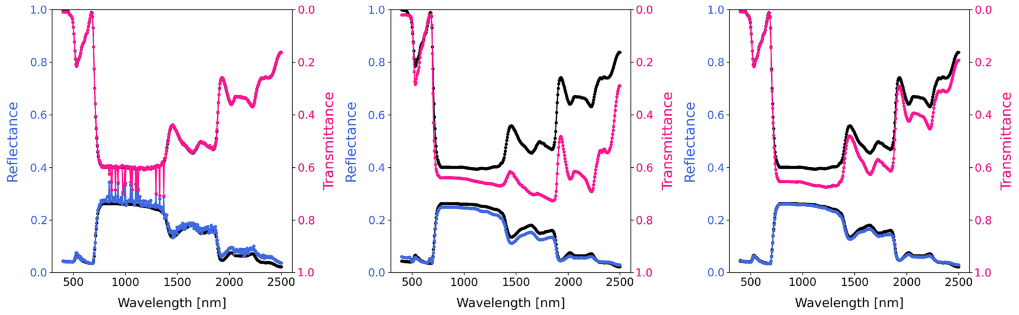


Fig. 7 Difficult PROSPECT-generated sample number 59 simulated with all three methods. Black points show the target spectra and colored ones the simulation results. This sample shows the instability of our model if difference between reflectance and transmittance is too high. The Original method is unstable in the NIR plateau and systematically overestimates reflectance on long wavelengths. Training data of Surface and NN did not contain a sample with a similar situation, so they are merely guessing the spectra. (a) Original, (b) Surface, and (c) NN.

often makes gross errors in these situations, a lot of points would have to be generated to get sufficient amount of usable data. Also, alleviating the 1% error threshold for acceptance should be considered, as even the best of dissimilar points are not very accurate. On the other hand, using less accurate training data would probably worsen the performance of the models in general.

Generating a lot of training data takes a lot of time. Generating the 44,284 training points used in this study took roughly 72 h with a consumer-grade machine with Intel i9-9900K 8-core processor.

The leaf model could still be improved by adding surface reflections, which would produce more realistic leaves and allow modeling some special cases such as waxy leaves. Using bidirectional reflectance distribution function and bidirectional scattering distribution function for the leaf surface would be possible in Blender when those functions are known from real-world measurements. Another improvement would be to model needle-like leaves, which would make modeling coniferous and mixed forests possible.

6 Conclusions

With the developments presented in this study, the conversion from target spectra to virtual renderable leaf material can now be done much faster in HyperBlend. While the conversion with the new methods is not as accurate as the original optimization method, the 200-fold increase in simulation speed makes HyperBlend more usable especially in quick drafting.

The new methods also produce more stable spectra. The original method is based on optimization, which has no guarantees to converge in a specific solution and may result in discrepancy in adjacent wavebands. As the functions used in the surface fitting method are smooth by definition, the resulting spectra are guaranteed to be smooth. The NN method does not have this kind of inbuilt guarantee, but the network trained in this study appears to be smooth enough to alleviate any concerns of jagged spectra.

HyperBlend is not a leaf optical simulator in the traditional sense that it could take some biophysical and biochemical parameters, such as chlorophyll content, as an input, and produce leaf reflectance and transmittance spectra as an output. However, by incorporating PROSPECT leaf optical model as a source of target spectra, from the user's perspective, the functionality is as if they were running PROSPECT simulator in a 3D virtual world. Because the underlying leaf simulator is not tightly connected to 3D-graphics, it can be easily replaced with another simulator or measured data if needed. This gives a high degree of flexibility in adjusting the configuration to fit different purposes.

Future studies will cover generation of tree and forest geometry as well as construction of hyperspectral image cubes and associated metadata needed for inversion of the canopy model. This and the previous study on HyperBlend have focused on necessary prerequisites for creating simulated hyperspectral images of forests with Blender. The following studies will highlight the advantages of our image generation scheme: procedural geometry, largely automated workflow, fast rendering of complex scenes, and vast heterogeneous forest canopies complete with understory.

Data Availability Statement

The program code of HyperBlend is available at https://github.com/silmae/hyperblend under MIT license. The training data set used in this study is available at https://osf.io/trhf8/ under CC BY 4.0 license.

Acknowledgments

We would like to thank anonymous reviewers for their valuable criticism and suggestions. We also thank Vilho Halonen for assistance with neural network architecture. This study was funded by the Academy of Finland (Grant No. 327862). The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. R. Othman, F. A. Mohd Zaifuddin, and N. M. Hassan, "Carotenoid biosynthesis regulatory mechanisms in plants," *J. Oleo Sci.* **63**(8), 753–760 (2014).
2. G. Lassalle, "Monitoring natural and anthropogenic plant stressors by hyperspectral remote sensing: recommendations and guidelines based on a meta-review," *Sci. Total Environ.* **788**, 147758 (2021).

3. J. A. Hatala et al., "Landscape-scale patterns of forest pest and pathogen damage in the greater yellowstone ecosystem," *Remote Sens. Environ.* **114**(2), 375–384 (2010).
4. R. Näsi et al., "Using UAV-based photogrammetry and hyperspectral imaging for mapping bark beetle damage at tree-level," *Remote Sens.* **7**(11), 15467–15493 (2015).
5. H. Abdullah et al., "European spruce bark beetle (*Ips typographus, L.*) green attack affects foliar reflectance and biochemical properties," *Int. J. Appl. Earth Obs. Geoinf.* **64**, 199–209 (2018).
6. S. C. Kefauver, J. Peñuelas, and S. Ustin, "Using topographic and remotely sensed variables to assess ozone injury to conifers in the Sierra Nevada (USA) and Catalonia (Spain)," *Remote Sens. Environ.* **139**, 138–148 (2013).
7. P. Arellano et al., "Detecting the effects of hydrocarbon pollution in the amazon forest using hyperspectral satellite images," *Environ. Pollut.* **205**, 225–239 (2015).
8. I. Pölönen et al., "Tree species identification using 3d spectral data and 3D convolutional neural network," in *9th Workshop on Hyperspectral Image and Signal Process.: Evol. in Remote Sens. (WHISPERS)*, pp. 1–5 (2018).
9. Y. M. Govaerts et al., "Three-dimensional radiation transfer modeling in a dicotyledon leaf," *Appl. Opt.* **35**, 6585–6598 (1996).
10. A. Kallel, "Leaf polarized BRDF simulation based on Monte Carlo 3-D vector RT modeling," *J. Quant. Spectrosc. Radiat. Transfer* **221**, 202–224 (2018).
11. A. Kallel, "Two-scale Monte Carlo ray tracing for canopy-leaf vector radiative transfer coupling," *J. Quant. Spectrosc. Radiat. Transfer* **243**, 106815 (2020).
12. B. D. Ganapol et al., "LEAFMOD: a new within-leaf radiative transfer model," *Remote Sens. Environ.* **63**(2), 182–193 (1998).
13. J. Jiang et al., "FASPECT: a model of leaf optical properties accounting for the differences between upper and lower faces," *Remote Sens. Environ.* **253**, 112205 (2021).
14. W. Verhoef, "Light scattering by leaf layers with application to canopy reflectance modeling: the SAIL model," *Remote Sens. Environ.* **16**, 125–141 (1984).
15. S. Jacquemoud, "Inversion of the PROSPECT + SAIL canopy reflectance model from AVIRIS equivalent spectra: theoretical study," *Remote Sens. Environ.* **44**, 281–292 (1993).
16. J. P. Gastellu-Etchegorry, E. Martin, and F. Gascon, "DART: a 3D model for simulating satellite images and studying surface radiation budget," *Int. J. Remote Sens.* **25**(1), 73–96 (2004).
17. S. Jacquemoud et al., "PROSPECT+SAIL models: a review of use for vegetation characterization," *Remote Sens. Environ.* **113**, S56–S66 (2009).
18. U.S. Geological Survay, "Landsat 1," https://www.usgs.gov/landsat-missions/landsat-1 (accessed 21 October 2022).
19. R. O. Green et al., "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)," *Remote Sens. Environ.* **65**(3), 227–248 (1998).
20. I. Colomina and P. Molina, "Unmanned aerial systems for photogrammetry and remote sensing: a review," *ISPRS J. Photogramm. Remote Sens.* **92**, 79–97 (2014).
21. K. A. Riihiaho, T. Rossi, and I. Pölönen, "Hyperblend: simulating spectral reflectance and transmittance of leaf tissue with blender," *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* **V-3-2022**, 471–476 (2022).
22. T. Akenine-Möller, E. Haines, and N. Hoffman, *Real-Time Rendering*, 3rd ed., A. K. Peters, Ltd., USA (2008).
23. O. C. Blender, "Blender: a 3D modelling and rendering package," Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018, http://www.blender.org.
24. J. T. Kajiya, "The rendering equation," in *Proc. 13th Annu. Conf. Comput. Graphics and Interact. Tech., SIGGRAPH '86*, Association for Computing Machinery, New York, pp. 143–150 (1986).
25. E. P. Lafortune and Y. D. Willems, "Rendering Participating Media with Bidirectional Path Tracing," in *Rendering Techniques '96*, X. Pueyo and P. Schröder, Eds., pp. 91–100, Springer Vienna, Vienna (1996).
26. G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*, CreateSpace, Scotts Valley, California (2009).
27. S. Jacquemoud and F. Baret, "PROSPECT: a model of leaf optical properties spectra," *Remote Sens. Environ.* **34**(2), 75–91 (1990).
28. T. Fourty et al., "Leaf optical properties with explicit description of its biochemical composition: direct and inverse problems," *Remote Sens. Environ.* **56**(2), 104–117 (1996).
29. S. Jacquemoud et al., "Estimating leaf biochemistry using the PROSPECT leaf optical properties model," *Remote Sens. Environ.* **56**, 194–202 (1996).
30. J.-B. Feret et al., "PROSPECT-4 and 5: advances in the leaf optical properties model separating photosynthetic pigments," *Remote Sens. Environ.* **112**, 3030–3043 (2008).
31. J.-B. Féret et al., "PROSPECT-D: towards modeling leaf optical properties through a complete lifecycle," *Remote Sens. Environ.* **193**, 204–215 (2017).
32. J. D. Hunter, "Matplotlib: a 2D graphics environment," *Comput. Sci. Eng.* **9**(3), 90–95 (2007).

33. K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quart. Appl. Math.* **2**(2), 164–168 (1944).
34. D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *J. Soc. Ind. Appl. Math.* **11**(2), 431–441 (1963).
35. P. Virtanen et al., "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nat. Methods* **17**, 261–272 (2020).
36. K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks* **2**(5), 359–366 (1989).
37. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature* **521**(7553), 436–444 (2015).
38. P. Sharma and A. Singh, "Era of deep neural networks: a review," in *8th Int. Conf. Comput., Commun. and Networking Technol. (ICCCNT)*, pp. 1–5 (2017).
39. A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, Vol. 30, p. 6 (2013).
40. D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization." arXiv:1412.6980 (2017).
41. A. Paszke et al., "Pytorch: an imperative style, high-performance deep learning library," in *Adv. Neural Inf. Process. Syst. 32*, H. Wallach et al., Eds., pp. 8024–8035, Curran Associates, Inc. (2019).
42. A. Hovi, P. Raitio, and M. Rautiainen, "A spectral analysis of 25 boreal tree species," *Silva Fennica* **51**(4) (2017).
43. S. Jacquemoud and S. Ustin, *Leaf Optical Properties*, Cambridge University Press (2019).
44. L. M. Domenzain, J. Gómez-Dans, and P. Lewis, "jgomezdans/prosail: Pip package bug fix release," https://doi.org/10.5281/zenodo.2574925 (2019).

Biographies of the authors are not available.