Using Random Forests to Compute Radiative Transfer

Paulina Czarnecki
Columbia University
EAEE 4000 Final Project

December 22, 2021

Contents

1 Introduction

1.1 Background: Radiative Transfer

Radiative transfer is the transfer of electromagnetic energy through matter–specifically, through the Earth's atmosphere. Though the physics of radiative transfer is well known, the calculations are often incredibly computationally expensive. In fact, in climate and weather models, radiation is often computed less frequently than other elements [8]. In order to predict how climate will change and what interventions will be effective in climate models, how climate evolves on other planets, or what the weather will be like tomorrow, we need algorithms to quickly compute radiative transfer with sufficient accuracy [4].

Monochromatic radiative flux is the measure of how much electromagnetic energy is passing through the atmosphere for each wavenumber of light, measured in W/m^2/cm^{-1}. The total flux at the top of the atmosphere (TOA) is simply the sum of all the individual monochromatic fluxes when it is assumed that the spectrum of light is a continuum. More precisely, this quantity can be characterized as the integral of monochromatic fluxes across all wavenumbers:

$$F = \int_{\nu} F_{\nu} d\nu,$$

where F_{ν} denotes the flux at wavenumber ν.

A typical radiation dataset may span tens of thousands of wavenumbers; thus, computing this integral using all the wavenumbers as quadrature points is not practical [4]. Additionally, monochromatic fluxes are highly nonlinear and not smooth, so standard numerical methods will not perform well (Fig. 1). Radiative transfer algorithms in use in climate models aim to streamline the calculation by making assumptions and approximations.

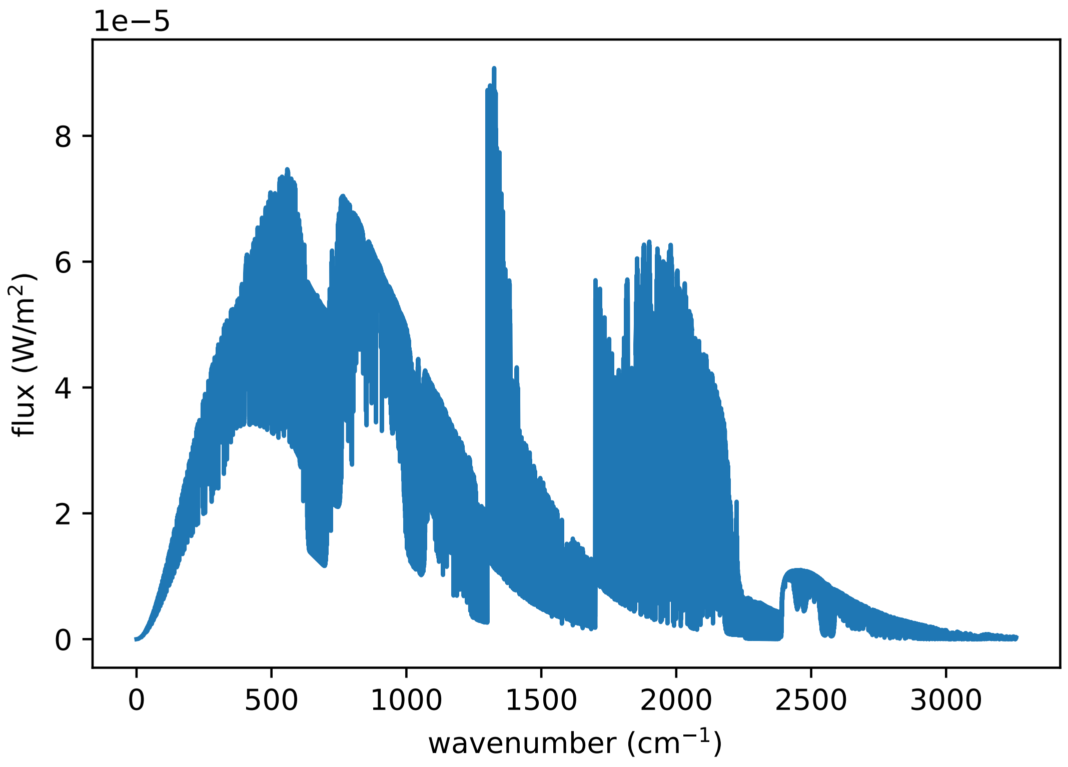


Figure 1: Monochromatic outgoing longwave flux at the top of the atmosphere. The figure illustrates how difficult integrating the flux would be due to the nonlinearity and nonsmoothness.

The modern state-of-the-art algorithm for computing radiative transfer is called the correlated k-distribution (CKD) algorithm. The variable k denotes the absorption coefficient, which characterizes how a certain wavelength of light is absorbed by a gas. The amount of light that is transmitted through a material is independent of the ordering of k, so radiation data can be sorted into a monotonically increasing function [4]. This function can be easily integrated using typical numerical methods, thus allowing for the calculation of radiative transfer while avoiding the nonsmoothness of the monochromatic fluxes.

There are several problems that make the correlated k-distribution method difficult to work with, impractical, or otherwise worth improving. First, in order to assign and sort data by k, it is assumed that the absorption coefficient is correlated with pressure and temperature, and that the mapping to the smooth monotonic function is independent of height [4]. These assumptions are not always true and make the algorithm inflexible to different scenarios [9]. The method is difficult to calibrate to a user's specifications as it is typically built into packages with outdated legacy codes [9]. Physically, the correlated k-distribution will not converge to line-by-line calculations (integration using each wavenumber as a quadrature point), and so behave as something of a black box [9]. Finally, as with many algorithms, the choice of the number of points to sort and integrate across leads to an accuracy/efficiency tradeoff [9]. Thus it is important to explore new parametrizations for radiation and work to incorporate fast, accurate, and flexible codes into climate models [9].

1.2 Existing Work

There have been many attempts at improving radiation algorithms in various ways. A popular approach is to try to find a subset of wavenumbers that can produce the radiative integral as a weighted sum across the monochromatic fluxes at these wavenumbers, rather than an exact sum across a huge number of data points [3, 5, 7]. Finding this subset of wavenumbers is often an optimization problem in minimizing the error between the weighted estimate and the real TOA integrated flux.

One method of finding this optimal subset is using simulated annealing [3, 7]. Simulated annealing is an algorithm that aims to find the global minimum of a function. The algorithm begins with a random subset of wavenumbers, denoted S [3, 7]. The linear regression across the monochromatic fluxes found at the wavenumbers in S and the reference integrated TOA flux is computed in order to find the weights W; then the dot product between the fluxes found at wavenumbers S and S is the estimated TOA integrated flux $F(S)$ [3, 7]. The error e between F and the reference value is computed [3] [3, 7]. This computation is described in more detail in the Methods section.

Next, a neighbor state of S is computed by swapping one of the wavenumbers in S with a random wavenumber not in S; the neighbor state is denoted S_n [3, 7]. An estimated integrated flux $F(S_n)$ is computed through a linear regression again, and a new error between this estimate and the reference flux E_n is calculated [3, 7]. This new state is accepted with a certain probability $p = e^{(E-E_n)/T}$, where T is the annealing temperature [3, 7]. The annealing temperature starts high, so that almost all moves are accepted, and decreases as the algorithm descends into a minimum, so that moves are less likely to be accepted [3, 7]. This prevents the algorithm from becoming stuck in a local minimum, in order to find the true global minimum. The process of generating new moves is repeated iteratively until there are no better moves or until a desired accuracy is reached [3].

Another popular method to find a subset of wavenumbers across which to compute a weighted sum is through Principal Component Analysis (PCA). First, monochromatic radiances are arranged in a matrix r with rows denoting each wavenumber or frequency and columns corresponding to distinct atmospheric profiles [5]. This matrix r can be decomposed using a singular value decomposition (SVD) in which $r = P\sigma S$; P and S are orthonormal eigenvectors of $r^T r$ and rr^T in columns and rows, respectively, and σ has principal components, sorted in descending order, derived from the eigenvalues of $r^T r$, on its diagonal [5]. Then, the wavenumbers in r corresponding to the n largest principal components are chosen as the representative subset [5]. The weights are found with a linear regression against reference integrated TOA flux [5].

As discussed in class, neural networks may be analogous to PCA or a nonlinear regression. Thus, they may help with the problem of computing radiative transfer with speed, accuracy, and flexibility, expanding on current popular methods.

1.3 Random Forests

Random forests were introduced by Leo Breiman in 2001 [2]. A random forest is a way to aggregate many decision or regression trees in order to improve the overall accuracy of a prediction algorithm [2]. Small portions of datasets are sampled, random trees are applied to the samples, and the results of the many trees are averaged in some way [1, 2]. While random forests of decision trees are often used in order to perform a classification, they can also be used for a regression where the predictor will be a numerical value instead of a class [2].

Regression random forests are easily implemented with packages in programming languages like Python or R [1]. They are a machine learning technique for prediction and aggregation of data, with Breiman showing they always achieve a lower mean squared error than bagging alone [2]. They also avoid overfitting the data due to the large number of trees they are usually made of, per the law of large numbers [2].

Similarly to other machine learning algorithms, regularizations are often applied in order to exert some control over the way the trees assign weights. As discussed in class, sometimes these regularizations include pruning branches at the end of training (analogous to dropout in a standard neural network), or early stopping. Another such method is L_1 regularization [10]. The L_1 regularization forces many of the weights of a weight vector to be close to zero, or, in other words, for the weights vector to be sparse [10]. This forces the most important features of the data to be highlighted by reducing the total number of important features.

2 Methods

2.1 Data

The data used was from the Correlated K-Distribution Model Intercomparison Project (CKDMIP) [6]. Specifically, the Evaluation-1 longwave radiation dataset was used. A script provided by the project was used to generate datasets of monochromatic radiative fluxes, such that for each of fifty atmospheres, each wavenumber in the radiative spectrum is attributed a radiative flux. These monochromatic fluxes are summed across the entire radiative spectrum to generate the reference calculations of top of atmosphere (TOA) total outgoing longwave radiation (OLR). Although other variables such as pressure and temperature are included in the dataset, they were omitted here. As we are interested in only the top of the atmosphere, other height levels were disregarded as well. Thus the predictors were the 7 million monochromatic fluxes in each of the 50 atmospheres in the top height level, and the labels were the reference calculations.

Data used for the regression tree algorithm was first cleaned. Using a `pandas` dataframe structure, the data was formatted to be acceptable to an `sklearn` algorithm. The feature names (wavenumbers, cm^{-1}) were separated from the data (the monochromatic flux at each of these wavenumbers, W/m^2) and the labels (the reference net TOA flux at the top of the atmosphere, W/m^2). Then, training and testing subsets were created. Here, 42 atmospheric profiles were used to train the model, and 8 were set aside to test the model.

2.2 Algorithm

Initially, the `RandomForestRegressor` from `sklearn` was used to explore the capabilities of the algorithm. However, due to relatively poor initial results and a desire to use L_1 regularization, I transitioned to `XGBRFRegressor` from `xgboost`. The `xgboost` package allows for more flexibility in setting various parameters and finding the best fit.

In order to tune the regression tree random forest to find the best root mean squared error as compared to the reference calculations, I coded a loop to emulate the built-in `GridSearchCV`. I created my own function in order to explore the results of the model in two different ways, as described below. The function loops through a provided list of parameters and reports the parameter set that provides the best (lowest) error. Important parameters for this project include the maximum depth of each regression tree (`max_depth`, default of 6), the number of trees in a forest (`n_estimators`), and the L_1 regularization parameter (`reg_alpha`, can be any nonnegative number). The number of estimators used in particular was important to the runtime of the program. In order to test the effect of these parameters on the performance of the forest, I explored the following parameter space:

```
params = {'max_depth': [6, 10],
          'n_estimators': [10, 100],
          'reg_alpha': [0, 5, 10, 20]
          }
```

To determine the accuracy of the model, I took the two approaches described below.

2.3 Prediction Error Measure

The first approach in determining the error produced by the random forest was to train the model on the 42 training profiles and use the `predict` function to estimate the top of the atmosphere total flux of the 8 testing profiles. Then, the absolute root mean squared error between these predicted values and the reference values was computed.

2.4 Feature Importance Error Measure

A different approach was to use feature importance. Algorithms such as simulated annealing or principal component analysis have been used to perform a dimension reduction on the 7 million contributing fluxes, or to choose a small subset of the "important" ones. Analogously, the feature importance function is part of both `xgboost` and `sklearn` regression forests. The function selects the most important features identified by the regression random forest. Here, I selected all features of nonzero importance, but I limited the set to at most 100 features, as preliminary results from my own research have shown that a set of about 60 individual fluxes is sufficient for high accuracy (Fig. 3). Those features could be used as a set of representative features, or monochromatic fluxes. I called the set S:

$$S = \{f_{\nu_1},\, f_{\nu_2},\, \dots\, ,\, f_{\nu_N}\},$$

where f_{ν_i} was the monochromatic flux at the wavenumber ν_i. Then, the linear regression of the top important fluxes was computed against reference calculations. The linear regression yielded a set of weights, along with an intercept, which is omitted here:

$$W_i = \{w_{\nu_1},\, w_{\nu_2},\, \dots\, ,\, w_{\nu_N}\}.$$

Individual weights were computed for each atmosphere, though the subset S was held constant for all atmospheres. A condition was placed on the linear regression weights: they must be positive. Negative weights indicated that the flux at that wavenumber was redundant and thus may be excluded [3]. The linear regression was performed in a loop until none of the weights were negative, removing any fluxes that yielded a negative weight at each iteration. Once a set of nonnegative weights was computed, the model's guess of the integrated TOA flux would be the dot product of the monochromatic subset of fluxes with their linear weights, and the error was computed between the estimate and the reference:

$$\text{estimate}_i = S \cdot W_i$$

and

$$e = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(\text{estimate}_i - \text{reference}_i)^2},$$

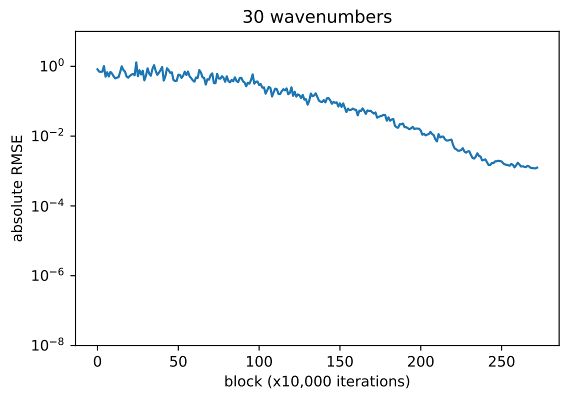
where i indexes over the available atmospheres.

Due to the different ways of looking at the error of the algorithm, I implemented my own grid-search function to optimize the parameters across the given parameter space. As with the built-in `GridSearchCV`, my function looped through the parameter space and trained a random forest of regression trees. Then it computed both forms of error and reported them.

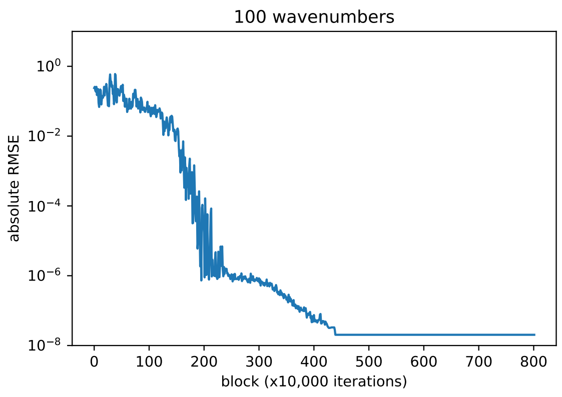
3 Results

3.1 Baseline Results

In my research, I have implemented the simulated annealing algorithm in order to find a representative subset of n wavenumbers. As outlined in the Introduction and Methods section, the simulated annealing algorithm was applied in order to derive an estimated TOA integrated flux from such a representative subset. The algorithm was executed in blocks of 10, 000 iterations, and the mean error in each block was reported. At the end of all iterations, the best subset is reported, as determined by the lowest absolute RMSE.



(a) The mean error over around 200 blocks of 10,000 algorithm iterations each for 30 representative fluxes.



(b) The mean error over around 200 blocks of 10,000 algorithm iterations each for 100 representative fluxes.

Figure 2

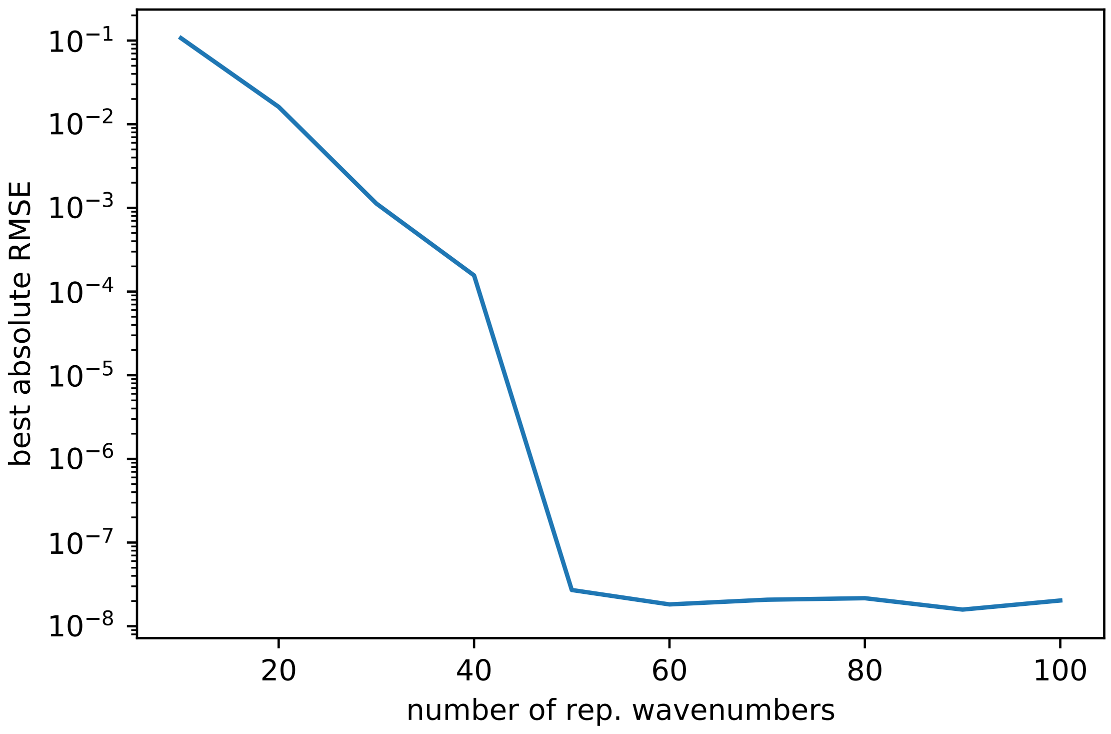


Figure 3: The final absolute root mean squared error obtained by the best state of the simulated annealing algorithm plotted against the size of the representative wavenumber subset. There are 10 independent runs represented in the figure, for representative subsets of size $\{10, 20, \ldots, 100\}$.

Preliminary results from the simulated annealing method show that for representative subsets of size 60, a linear regression is able to obtain an absolute root mean squared error of order 10^{-8}

(Fig. 3).

Figures 2 and 3 demonstrate the relationship between the number of representative wavenumbers in an approximation and the obtainable RMSE between that approximation and reference calculations, as determined by the simulated annealing algorithm. In Fig. 2a), the mean absolute error in a block of 10,000 iterations is plotted. The figure demonstrates that a representative subset of 30 wavenumbers is insufficient to optimize the error. The error only decreases only about one order of magnitude before the algorithm exits as it cannot find more appropriate moves. In contrast, in Fig. 2b), a subset of about 100 wavenumbers yields an absolute error of almost 10^{-8} W/m^2. Figure 3 summarizes these results. The size of the subset is plotted on the x-axis, with the best error achieved by the algorithm on the y-axis. The figure shows that after a subset size of 50 wavenumbers, the capabilities of the algorithm are saturated and significant further improvements are not made to the estimate. It is important to note that data given to the algorithm has 8 significant figures, so an error lower than 10^{-8} is not possible.

A machine learning algorithm should obtain a similar order of magnitude in error for representative subsets of a similar size in order to be useful.

3.2 ML Model Results

3.2.1 Errors

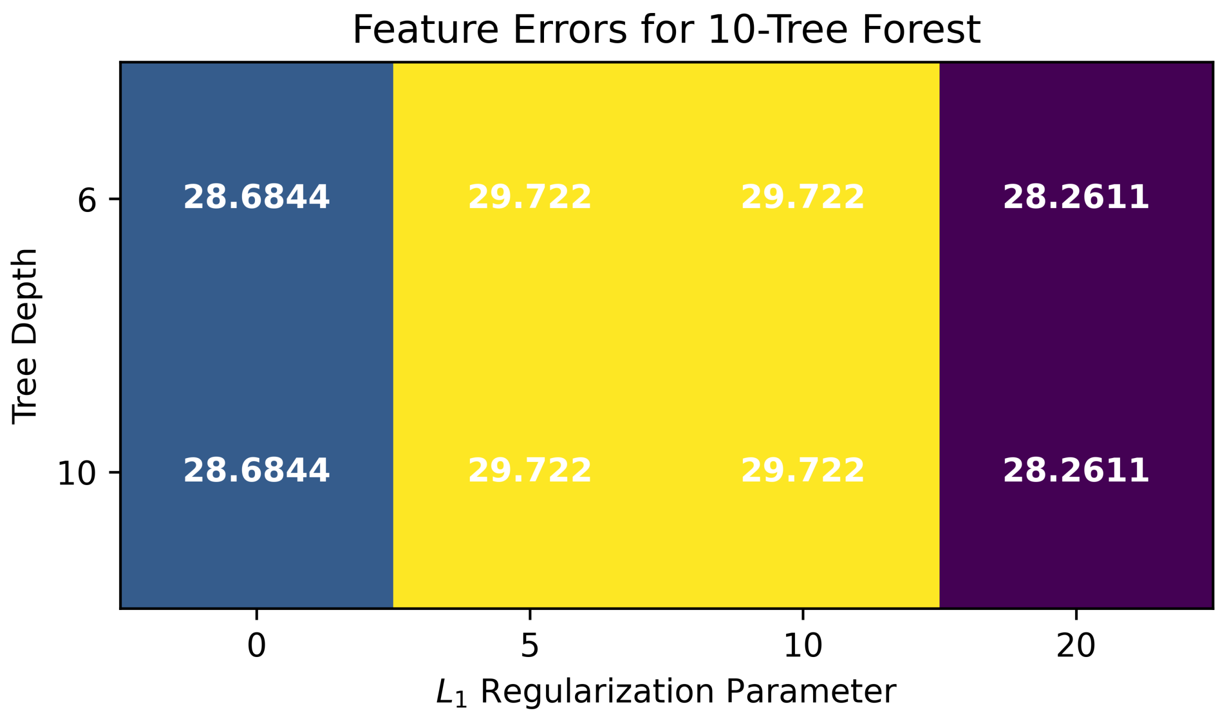


Figure 4: The model was run for tree depth values of 6 and 10, with the L_1 regularization parameter in $\{0, 5, 10, 20\}$. The error for each set of parameters in a forest of 10 trees, as determined by the feature importance measure, is reported here. Error values are color coded as well as printed in each square. The absolute RMSE measure results in errors in units of W/m^2.

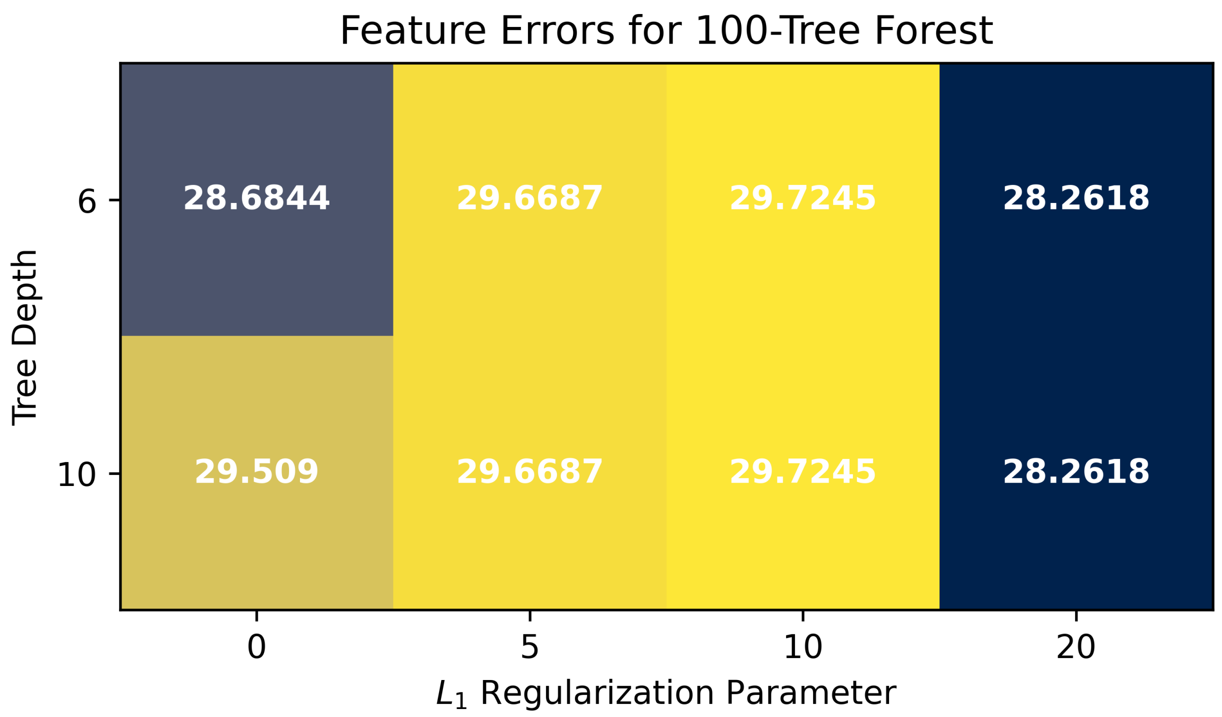


Figure 5: The model was run for tree depth values of 6 and 10, with the L_1 regularization parameter in $\{0,\ 5,\ 10,\ 20\}$. The error for each set of parameters in a forest of 100 trees, as determined by the feature importance measure, is reported here. Error values are color coded as well as printed in each square. The absolute RMSE measure results in errors in units of W/m^2.

As described in the Methods section, two different error measures (the feature importance measure and the prediction measure) were used to evaluate the performance of the model across several parameters: forest size in $\{10, 100\}$ trees; tree depth in $\{6, 10\}$, and the L_1 regularization parameter in $\{0, 5, 10, 20\}$. I first split the results into the two error measures, then forest sizes. Results for forests of the same size are shown with matching color schemes for convenience.

Results for the feature error measure are reported in Figs. 4 and 5. In Fig. 4, feature errors are reported for a random regression forest of 10 trees. Errors across tree depth are identical to four significant figures, indicating that tree depth did not affect the accuracy of the program. Regularization parameter errors are lowest for $L_1 = 20$, followed closely by $L_1 = 0$. The error doesn't seem to improve uniformly with increasing L_1 parameter, or differ significantly across the L_1 parameter.

In Figure 5, errors for a 100-tree forest are reported. Again, errors do not increase or decrease uniformly with the L_1 parameter, and tree depth has a small effect for $L_1 = 0$ but no effect elsewhere. In fact, a deeper tree results in a worse error here. Finally, as Fig. 9 summarizes, a bigger representative subset size (or a smaller L_1 parameter) is uncorrelated with a better feature importance error.

Next, I evaluate the error as determined by the model prediction function. Each trained forest is asked to predict the integrated TOA flux based on the previous models it has seen. In both Figures 6 and 7, error increases monotonically with the L_1 parameter. As for the feature importance error, tree depth seems to make a minimal difference, if at all.

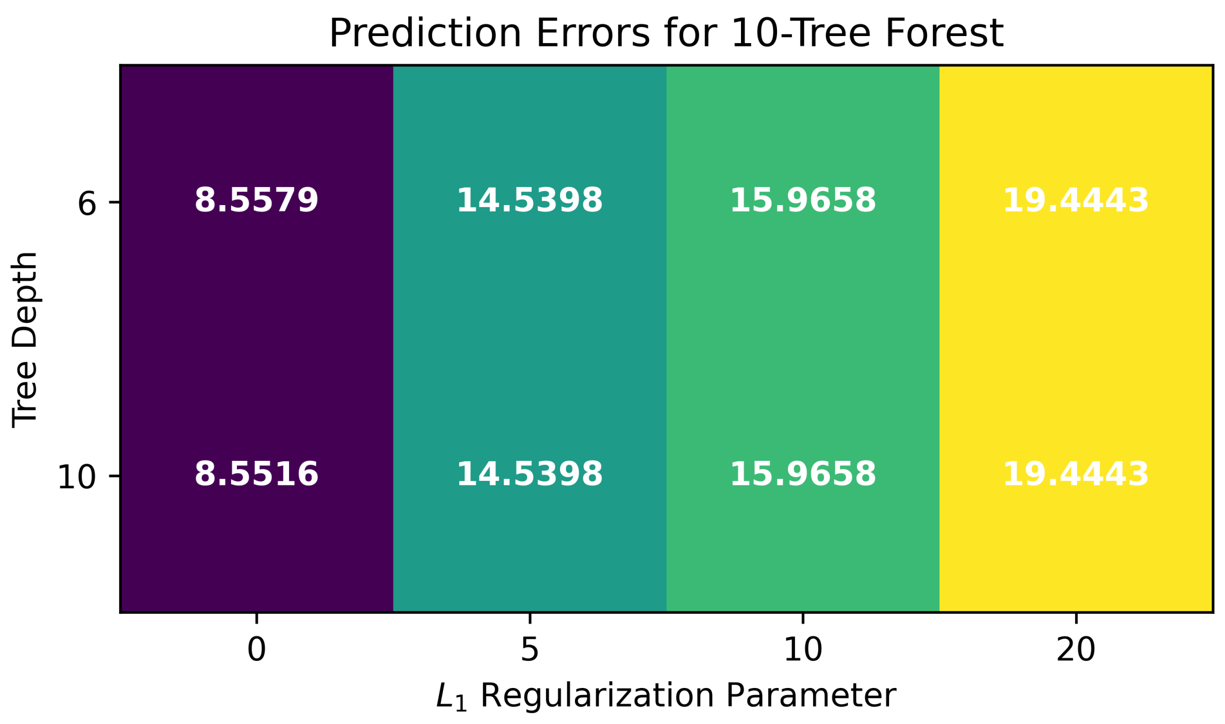


Figure 6: The model was run for tree depth values of 6 and 10, with the L_1 regularization parameter in $\{0,\ 5,\ 10,\ 20\}$. The error for each set of parameters in a forest of 100 trees, as determined by the model prediction measure, is reported here. Error values are color coded as well as printed in each square. The absolute RMSE measure results in errors in units of W/m^2.

Disappointingly, the feature importance method of calculating the integrated TOA flux yields error worse than any iteration of the prediction method. The average error as computed by the feature importance method is 29.143 W/m^2, while the average of the error computed using `model.predict()` is 14.113 W/m^2. This shows that a random forest may not be the best method for finding a representative subset from which to compute the TOA integrated flux via linear regression.

The depth of the tree does not seem to make a difference. Random forests with trees that have different maximum depths while all other parameters are fixed are always within 1 W/m^2 in error, and often they are identical to at least 4 significant figures. L_1 regularization monotonically increases the error for the model prediction method, while having little noticeable effect on the feature importance method. Finally, the effect of size of the forest on error is negligible. The mean error for 10- and 100-tree forests is within 1 W/m^2 for both the prediction method and the feature importance method independently.

The parameter set that produced the best error overall is a tree of maximum depth 10, with no L_1 regularization applied, and with TOA integrated flux computed by the random forest's prediction feature (Fig. 6). However, the resulting error is still 8.2917 W/m^2, many orders of magnitude higher than the absolute RMSE computed by prior work (Figs. 2, 3).

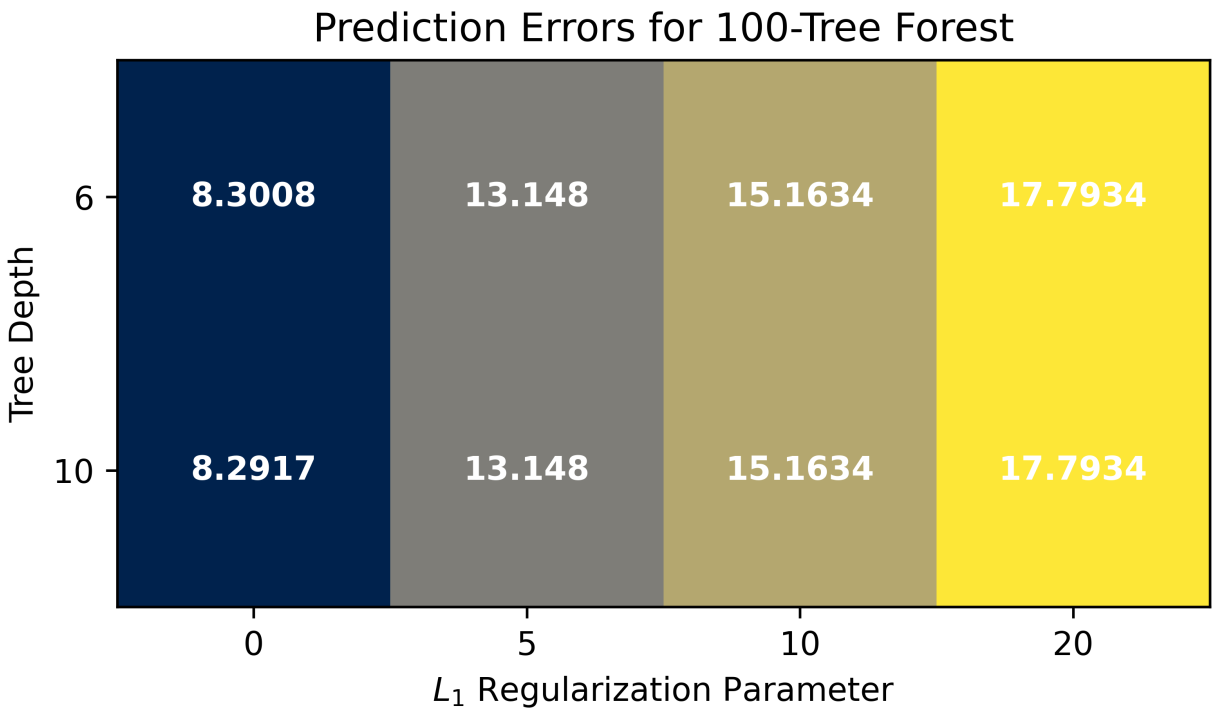


Figure 7: The model was run for tree depth values of 6 and 10, with the L_1 regularization parameter in $\{0,\ 5,\ 10,\ 20\}$. The error for each set of parameters in a forest of 100 trees, as determined by the model prediction measure, is reported here. Error values are color coded as well as printed in each square. The absolute RMSE measure results in errors in units of W/m^2.

3.2.2 Effects of L_1 Regularization

The features of nonzero importance given by the model determine the representative subset of wavenumbers in the given atmospheres. In order to understand the effect of L_1 regularization on the results of the model, I plot the size of the resulting representative subset against the L_1 regularization parameter (Fig. 8). Results from forests with 10 trees are denoted in blue, while orange dots represent results from forest of size 100. From Fig. 8, a higher L_1 regularization yields a smaller representative subset. This is expected because L_1 regularization forces weights to be zero, thus choosing the most important features. Larger forests also yield larger representative subsets compared to smaller forests.

From Fig. 3, we know that a representative subset of around 60 wavenumbers is sufficient to compute an error of 10^{-8} W/m^2 via a weighted sum of fluxes found at those wavenumbers. Based on this information and Fig. 8, it is clear that a small forest will not produce an appropriate representative subset. Furthermore, any forest with a large L_1 regularization will also produce a subset too small to accurately represent the variability in the spectrum in order to compute an accurate weighted sum.

In order to investigate the effect of the subset size on feature importance error, I plot them against each other and fit a line to the scatter plot in Figure 9. Based on Fig. 3, a larger subset size should yield a smaller error – in other words, the line of best fit should have a negative slope.

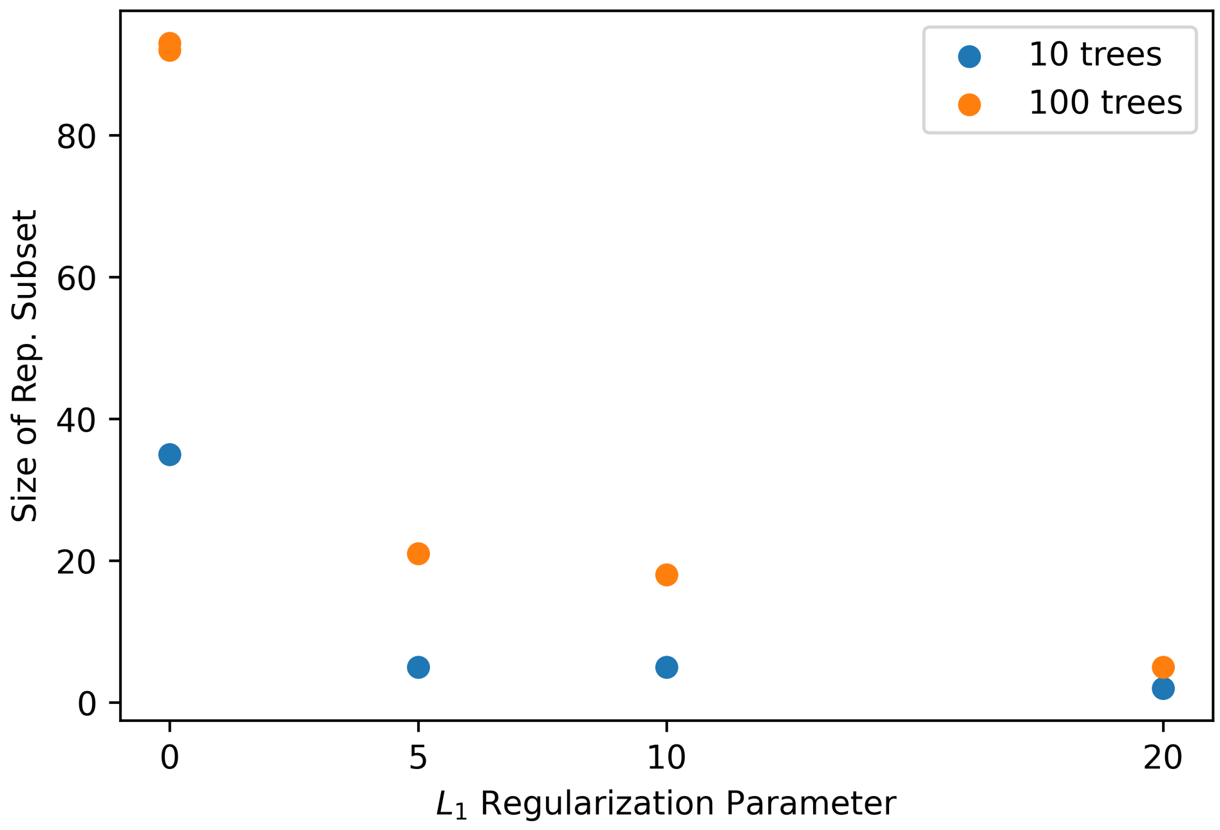


Figure 8: The number of representative wavenumbers chosen by the random forest's feature importance function across L_1 regularization parameters in $\{0, 5, 10, 20\}$ and forest sizes in $\{10, 100\}$. Forests of 10 trees are represented by blue dots, while forests of 100 trees are orange.

However, in Fig. 9, the line of best fit does not seem to slope, and the size of the representative subset is uncorrelated with the feature importance error. This demonstrates that the model is not choosing the best features in order to represent the top of atmosphere integrated flux in the way desired.

It is possible that I simply do not have enough data to train a machine learning model to perform the prediction accurately. Another potential reason for the discrepancy is the lack of information the regression forest has about the physics of the problem. It is clear that the features it is choosing as "most important" are not the correct features needed to compute a weighted sum. As evidenced by Fig. 2, even a random guess of about 100 representative wavenumbers may yield an absolute RMSE of less than 1. From Fig. 9, we see that independent of subset size, the feature importance error is around 29 W/m^2. This indicates that the best wavenumbers as submitted by the random forest may be worse than a random guess. Thus, the random forest does not choose the most important features for computing the TOA integral via a weighted sum.

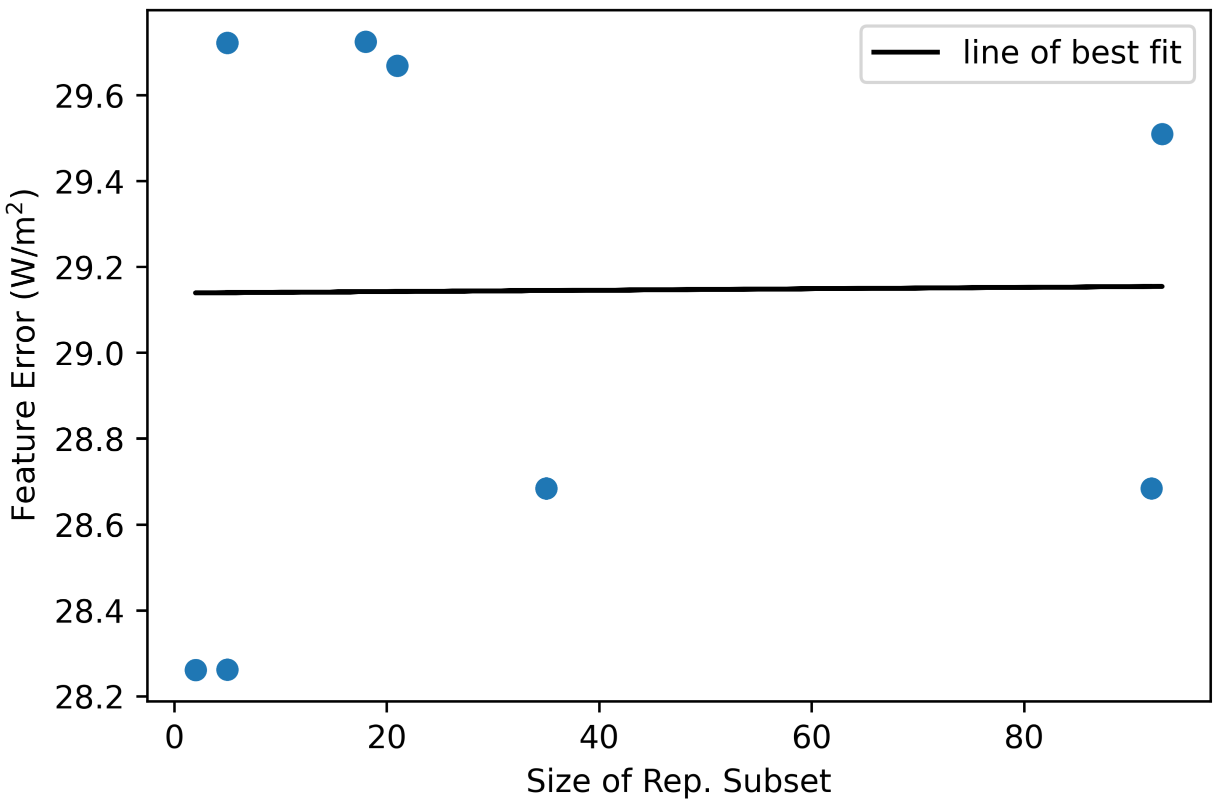


Figure 9: The size of the representative subset chosen by the feature importance function is plotted against the feature importance error for that representative subset. A line of best fit is calculated and plotted in black. The slope of the line is $m = 0.00016$, indicating that the size of the subset is uncorrelated with the error as derived from that subset, in contrast from the previously used simulated annealing method.

4 Conclusion

Computing integrated flux at the top of the atmosphere involves integrating across millions of monochromatic fluxes, which is often computationally expensive. In order to simplify this computation, a dimension reduction, such as choosing a smaller representative subset of these monochromatic fluxes, may be implemented. In order to apply machine learning to this problem, I use a random regression forest to compute the TOA flux in two ways. Firstly, I train a random forest and ask it to predict the TOA integrated flux based on example atmospheres it is shown. In a different approach, I choose the most important monochromatic frequencies as identified by the model, and compute a weighted sum in order to represent the TOA integrated flux. For both methods, I perform a parameter sweep across tree depth, L_1 regularization parameter, and forest size.

Despite parameter tuning, the best error either method was able to produce was of $\mathscr{O}(1)$. This is many orders of magnitude larger than previous work. A possible reason for the underperformance of the machine learning model may be insufficient data, as it was trained only on 42 example atmospheres. Another possible reason may be the blindness the regression forest has to the physics of the problem. The regression forest is not told that it is computing a weighted sum, but is given monochromatic fluxes and asked to predict the TOA flux. In the future, I hope to address some of these shortcomings in order to use machine learning to improve algorithms of radiative transfer.

4.1 Project Reproducibility

All code and figures used in this project can be found at: `https://github.com/pczarnecki/ML-Project`. Though I saved the models and error output created in the parameter sweep for analysis, I did not include them in the repository due to space concerns; they may be generated by running `runML.py` and used to create figures in `Result Analysis.ipynb`. Data is also not included for the same reasons, but may be found here: `https://confluence.ecmwf.int/display/CKDMIP/CKDMIP%3A+Correlated+K-Distribution+Model+Intercomparison+Project+Home`. I used Prof. Lorenzo Polvani's computer `galileo` to perform the computations rather than my personal machine due to its superior computational power on such a large dataset.

References

[1] G. Biau and E. Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.

[2] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] S. Buehler, V. John, A. Kottayil, M. Milz, and P. Eriksson. Efficient radiative transfer simulations for a broadband infrared radiometer—combining a weighted mean of representative frequencies approach with frequency selection by simulated annealing. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 111(4):602–615, 2010.

[4] Q. Fu and K. Liou. On the correlated k-distribution method for radiative transfer in nonhomogeneous atmospheres. *Journal of Atmospheric Sciences*, 49(22):2139–2156, 1992.

[5] S. Havemann, J.-C. Thelen, J. P. Taylor, and R. C. Harlow. The havemann-taylor fast radiative transfer code (ht-frtc): A multipurpose code based on principal components. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 220:180–192, 2018.

[6] R. J. Hogan and M. Matricardi. Evaluating and improving the treatment of gases in radiation schemes: the correlated k-distribution model intercomparison project (ckdmip). *Geoscientific Model Development*, 13(12):6501–6521, 2020.

[7] J.-L. Moncet, G. Uymin, A. E. Lipton, and H. E. Snell. Infrared radiance modeling by optimal spectral sampling. *Journal of the atmospheric sciences*, 65(12):3917–3934, 2008.

[8] R. Pincus. Radiation: fast physics with slow consequences in an uncertain atmosphere. In *Workshop on Representing Model Uncertainty and Error in Numerical Weather and Climate Prediction Models, 20-24 June 2011*, pages 65–76, Shinfield Park, Reading, 2011. ECMWF, ECMWF.

[9] R. Pincus, E. J. Mlawer, and J. S. Delamere. Balancing accuracy, efficiency, and flexibility in radiation calculations for dynamical models. *Journal of Advances in Modeling Earth Systems*, 11(10):3074–3089, 2019.

[10] N. Schaaf, M. Huber, and J. Maucher. Enhancing decision tree based interpretation of deep neural networks through l1-orthogonal regularization. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 42–49. IEEE, 2019.