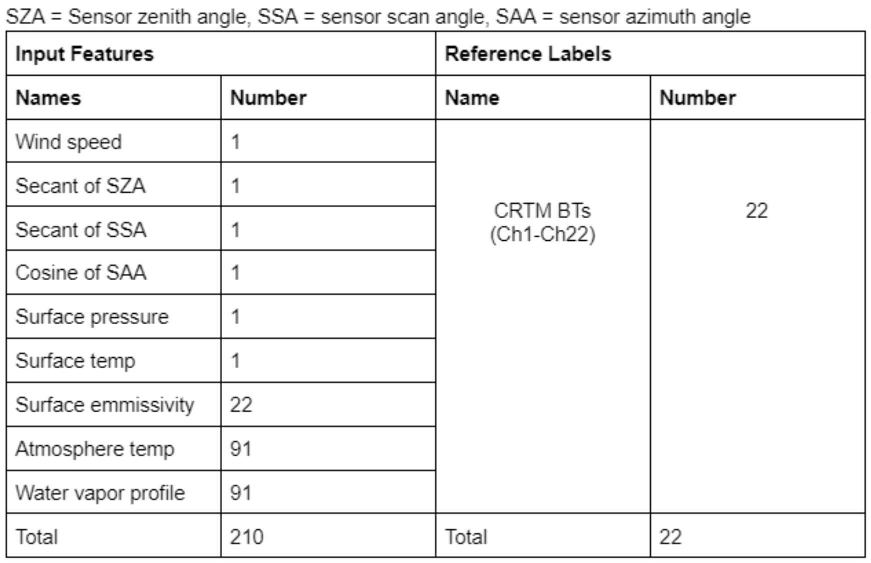
Developing a Deep Learning-Based Machine Learning Model to Predict Brightness Temperatures on Land

By Rida Khan

In my internship this summer I worked with my mentor, Dr. Xingming Liang, to develop a deep learning-based microwave forward radiative emulator for land. The objective was to develop a fully-connected deep neural network algorithm to emulate the Community Radiative Transfer Model (FCDN_CRTM) simulation of brightness temperatures (BTs) from the Advanced Technology Microwave Sounder (ATMS) channels for clear-sky cases over land surfaces (including ice and snow).

The CRTM is used as the reference for our model, and is represented in 22 ATMS bands at different frequencies. As outlined in figure one, the input for our model is 210 different measures: wind speed, sensor zenith angle, sensor scan angle, sensor azimuth angle, surface pressure, surface temperature, surface emissivity (represented in 22 bands), atmosphere temperature (91 different measures), and water vapor profiles (91 different measures).



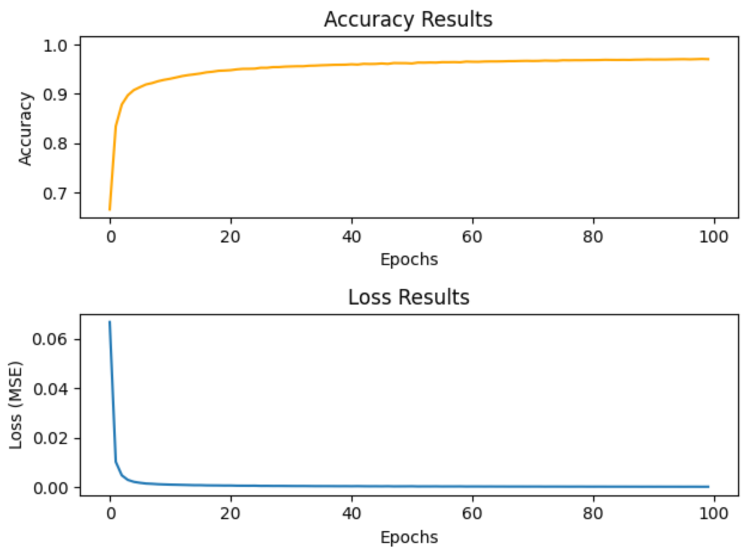
SZA = Sensor zenith angle, SSA = sensor scan angle, SAA = sensor azimuth angle

Input Features		Reference Labels	
Names	**Number**	**Name**	**Number**
Wind speed	1	CRTM BTs (Ch1-Ch22)	22
Secant of SZA	1		
Secant of SSA	1		
Cosine of SAA	1		
Surface pressure	1		
Surface temp	1		
Surface emmissivity	22		
Atmosphere temp	91		
Water vapor profile	91		
Total	210	Total	22

(Figure 1)

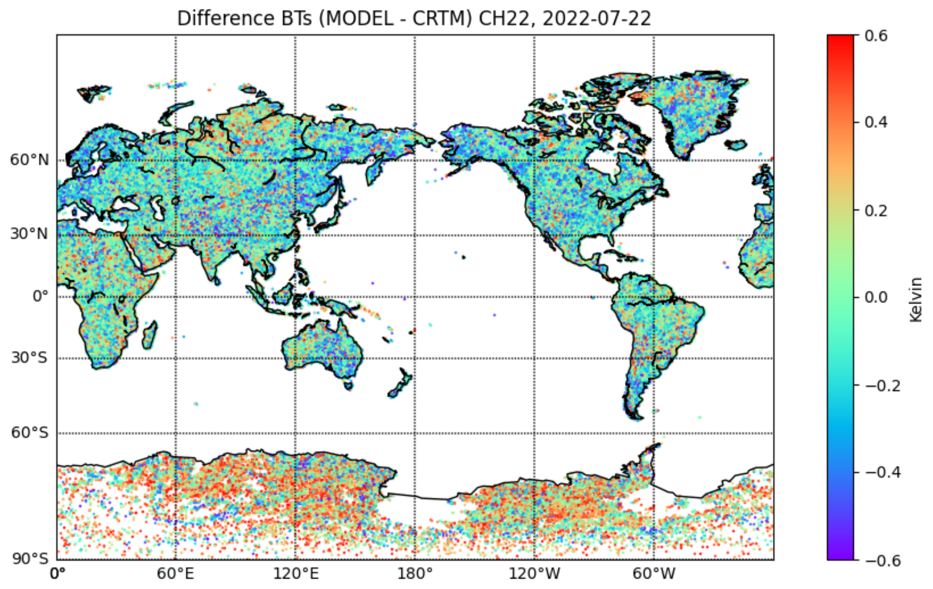
The primary reasoning for a majority of the model architecture was based on and adapted from a similar and heavily researched model that Dr. Liang worked on that was meant for predictions over water surfaces rather than land. The model is a sequential model created in Python using the Tensorflow and keras modules. The activation function used was ReLu for the hidden layers and linear was used for the output layer. The learning rate used was exponential decay, and used the Adam optimizer. This means that the model starts at a learning rate, in this case the starting learning rate was 0.001, and will decrease exponentially as the model trains. The model used 5 layers: an input layer with 210 nodes, three hidden layers with 512, 384, and 64 nodes respectively, and an output layer with 22 nodes. As noted earlier, the 210 input nodes represent all of the measures needed and the 22 output nodes make up the 22 ATMS channels.

The dataset used contains around 3 million records before quality control. In order to improve the model's accuracy and retain relevant data, I removed certain records by converting the data into a numpy array and used the numpy module to manipulate the records. I removed records that were over ocean surfaces, as the model is intended to predict over land and ice and snow, which narrowed the data set down to around 1.2 million records. Next, I removed records that contained NaN values for any of the input measures, as they would be invalid values in the regression analysis and render the record unusable. This only removed about 9000 records, but is a very important step because of how measures like loss, one of the metrics used to assess the model, are calculated. I also removed reference data that was too extreme, i.e. outliers. Brightness temperatures that were above 350 Kelvins or below 100 Kelvins were considered too extreme to be used in the dataset. Furthermore, there were three input values that were angles: sensor zenith angle, sensor scan angle, and sensor azimuth angle, and in the previous ocean model it was found that performing trigonometric operations on the values yielded greater accuracy, so I took the secant of sensor zenith angle and sensor scan angle, and the cosine of sensor azimuth angle before the model took it as input. The data also included ozone information, but that wasn't necessary for our model as the values were the same across all records and wouldn't have helped the model make more accurate predictions, so this data was scrubbed from all records. Before the data was used for training it was also normalized, and then it is denormalized again to yield the predicted absolute brightness temperatures.



(Figure 2)

After employing all of these quality control and preprocessing steps, the model was able to achieve 97% accuracy and 0.018 loss in 100 epochs seen in Figure 2. To further evaluate the model's quality, longitude and latitude data was used to see how the model fared for different locations. This was quantified by subtracting the model's predicted brightness temperatures from the reference values for brightness temperatures and using basemap and matplotlib modules to create a scatterplot projected onto a world map where each data point is placed based on longitude and latitude and the color is put on a scale in order to describe the difference value (Figure 3). The scale used was -0.06 to 0.06 Kelvins, so for example if the difference between the model and the reference was 0.06 in Antarctica, there would be a red data point in Antarctica to represent that. Generally, we can see that the model overestimated more in the Antarctic region and underestimated around coastlines, but there was also a good amount of areas with cyan colors which represented 0 or close to 0 difference. A global scatterplot was made for each of the 22 channels, and we could also see that the most variance was found in the later channels, CH18-CH22, possibly because of the water vapor profiles at those frequencies. Additionally, the differences between the model and reference means and standard deviations were also very low, ranging between -0.2 to 0.2 for means and 0.1 to 0.4 for standard deviations.



(Figure 3, global map for CH22)

After looking at these results, we had more time beyond just developing the model so we worked towards improving the accuracy of the model, as 97% is very high but 99% would be the ideal accuracy. Because there was a lot of overestimation in the Antarctic area, it seemed that the low brightness temperatures there were an anomaly compared to the rest of the dataset, so I trained the model to remove ocean as well as ice and snow data to see how it affected the models performance. I also made changes in quality control and preprocessing with batch normalization, using different trigonometry operations or no trigonometry on the angle inputs, removing certain emissivity and water vapor values, and removing outliers for the different measures after plotting histograms for each of the 210. Changes were also tested with decreasing the learning rate and using the LeakyReLu activation function rather than normal ReLu, making changes to the number of nodes in the hidden layers and number of hidden layers, and simply training the model more with a shuffled dataset. However, despite testing all of these methods, the accuracy and loss did not change significantly with the highest accuracy only matching 97%, and in some cases even resulted in even worse accuracy.

There is definitely more room for improvement in the model and this could be in many areas, from the data preprocessing to the model architecture and hyperparameters. While the model architecture was based on that used for the ocean model, it is possible that the ocean model itself could be improved more, let alone the possible improvements when adapting it to land surfaces. It is also possible that there is a better way to process the input data, with the sensor angles being the most prominent examples. The resulting model is still very accurate at 97%, and I was able to learn a lot about machine learning and deep learning and get experience with the Tensorflow, keras, numpy, matplotlib, and basemap modules as well as experience communicating and presenting the process and outcomes of my work.