**CSC–480: Final Report**

## 1. Project title and team members

- Title: TSI Prediction Model
- Group members: Nate Groves and Henry Goodwin

## 2. Problem description

We designed our model in response to Kaggle Project 1, which tasked us with building a machine learning model to predict total solar irradiance (TSI) using images of the sun. TSI is defined as the total amount of solar energy that reaches the top of Earth's atmosphere. According to the Kaggle project description, "Accurate TSI prediction is essential for understanding climate change, optimizing solar power generation, and studying solar dynamics."

The project included three datasets and one data folder: a training dataframe, a test dataframe, a private test dataframe, and a folder containing images of the sun. The training and test dataframes included four columns: an index column, an HMI_img column containing the file name of one of the solar images, a timestamp for when that image was taken, and "tsi_pert_0" (the target variable, TSI). The private test dataframe contained the same columns except for tsi_pert_0. There were 3,288 observations in the training dataframe, 274 observations in the test dataframe, and 275 observations in the private test dataframe.

The solar images were grayscale and 224x224 pixels. Their intensities varied from 0 to approximately 70,000. They contained different solar features, including sunspots.

The goal of the project was to predict tsi_pert_0 (TSI) based on one or both of the available features: the solar images and their corresponding timestamps. There were two leaderboards to evaluate the resultant model: a public leaderboard, which was calculated based on a model's combined accuracy on the test dataframe and the private test dataframe, and a private leaderboard, which tested the model with unseen test data. Accuracy was measured by mean absolute error (MAE).

## 3. Approach

We decided to design an ensemble model to tackle Project 1. Both of us independently created a model, which we then combined using a linear regression. We opted to use an ensemble model for two reasons. First, ensemble models typically outperform single models because they can incorporate the predictions of diverse models. Second, using an ensemble would allow us to work independently on our models over Thanksgiving break. Each model was submitted individually for scoring before being integrated into the ensemble. The approach for each of our independent models, and the resultant ensemble model, is discussed below.

### A. Model 1 (Henry)

Model 1 is a convolutional neural network (CNN). A CNN was chosen because its architecture excels at learning the spatial hierarchies within images like the solar images.

However, this model also incorporated the timestamps associated with the images using a multilayer perceptron (MLP). The results of the MLP on the image timestamps were combined with the results of the CNN on the solar images, which were then fed into a single output layer. An overview of the model layout is included on the right side of this page.

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_layer (InputLayer) | (None, 224, 224, 1) | 0 | – |
| conv2d (Conv2D) | (None, 224, 224, 16) | 160 | input_layer[0][0] |
| max_pooling2d (MaxPooling2D) | (None, 112, 112, 16) | 0 | conv2d[0][0] |
| conv2d_1 (Conv2D) | (None, 112, 112, 32) | 4,640 | max_pooling2d[0]… |
| max_pooling2d_1 (MaxPooling2D) | (None, 56, 56, 32) | 0 | conv2d_1[0][0] |
| flatten (Flatten) | (None, 100352) | 0 | max_pooling2d_1[… |
| dropout (Dropout) | (None, 100352) | 0 | flatten[0][0] |
| dense (Dense) | (None, 300) | 30,105,900 | dropout[0][0] |
| dense_1 (Dense) | (None, 100) | 30,100 | dense[0][0] |
| input_layer_1 (InputLayer) | (None, 1) | 0 | – |
| dense_2 (Dense) | (None, 50) | 5,050 | dense_1[0][0] |
| dense_3 (Dense) | (None, 50) | 100 | input_layer_1[0]… |
| concatenate (Concatenate) | (None, 100) | 0 | dense_2[0][0], dense_3[0][0] |
| dense_4 (Dense) | (None, 1) | 101 | concatenate[0][0] |

Model 1 pre-processed the data by normalizing the image values and timestamps. However, the image values were normalized incorrectly. Model 1 employed two convolutional and pooling layers, and one 50% dropout layer after the flatten layer to mitigate overfitting. Dropout was selected largely by trial-and-error: an earlier version of the model, which added 30% dropout layers after each convolutional/pooling layer, performed worse. The output was fed into three dense layers with decreasing numbers of neurons. As stated above, the output of the dense layers was concatenated with the output of the timestamps, which were run through one dense layer to extract important features. Ultimately, the use of timestamps improved the model's MAE, so it was included in the Model 1 that was integrated into the ensemble. It should be noted that an earlier version of Model 1 employed data augmentation. However, this model performed worse, so no data augmentation was used in the final version of Model 1.

Model 1 was trained with early stopping and patience = 5. The validation split was approximately 7.7%, which is lower than what is typically used. However, this was a conscious decision. Because our final model is an ensemble, it was important that our models be designed differently (i.e. they be diverse) in order to extract the most value possible from the ensemble. In its independent submission, Model 1 achieved its best MAE on epoch 3 (and was cut off by early stopping on epoch 8). Its results on the test data are discussed in the results section below.

B. Model 2 (Nate)

Model 2 is also a CNN, chosen for its strength in determining patterns and successful image processing. In an effort to create variability for the ensemble model, this model did not include timestamp data. This was a slightly deeper model, with two additional convolutional layers and more dense layers for

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 221, 221, 32) | 544 |
| max_pooling2d (MaxPooling2D) | (None, 110, 110, 32) | 0 |
| conv2d_1 (Conv2D) | (None, 107, 107, 32) | 16,416 |
| max_pooling2d_1 (MaxPooling2D) | (None, 53, 53, 32) | 0 |
| conv2d_2 (Conv2D) | (None, 50, 50, 64) | 32,832 |
| max_pooling2d_2 (MaxPooling2D) | (None, 25, 25, 64) | 0 |
| conv2d_3 (Conv2D) | (None, 22, 22, 128) | 131,200 |
| max_pooling2d_3 (MaxPooling2D) | (None, 11, 11, 128) | 0 |
| flatten (Flatten) | (None, 15488) | 0 |
| dense (Dense) | (None, 256) | 3,965,184 |
| leaky_re_lu (LeakyReLU) | (None, 256) | 0 |
| dense_1 (Dense) | (None, 128) | 32,896 |
| leaky_re_lu_1 (LeakyReLU) | (None, 128) | 0 |
| dense_2 (Dense) | (None, 64) | 8,256 |
| leaky_re_lu_2 (LeakyReLU) | (None, 64) | 0 |
| dense_3 (Dense) | (None, 32) | 2,080 |
| leaky_re_lu_3 (LeakyReLU) | (None, 32) | 0 |
| dense_4 (Dense) | (None, 1) | 33 |

the image MLP than Model 1. A summary is to the right.

The process for building the model was filled with trial and error. The process consisted of lots of submissions and lots of results. The first version was just the CNN model, with no MLP layers. This, unsurprisingly, did not end up with great results. The MAE for the test and private data were both above 0.25.MLP layers were added, and that immediately improved the model, decreasing the MAE for the public data by over 2 hundredths of a point. The hyperparameters were selected by trial and error, and the best results observed were 4 convolutional and pooling layers with a kernel size of 4 * 4, a max pooling layer in between each one. This was fed into an MLP model with 4 fully connected layers using a LeakyReLU function and an output layer using a linear function.

The training of the data used a 90/10 validation split, slightly higher than the other model to introduce more variability to the ensemble model.

C. Ensemble

For our ensemble model, we used a multiple linear regression which took each independent model's predictions as inputs. We opted to use a multiple linear regression instead of other ensemble techniques, such as averaging, in recognition that one model might be better than another: we wanted the ensemble model to learn appropriate weights for each of the model's predictions. The linear regression was trained off of the independent models' predictions from the training dataframe, which only included 274 observations. (For more discussion on this and other opportunities for future improvement, see the analysis section below.) Our final ensemble model was our first attempt at an ensemble, and we anticipated making alterations after getting its score. However, we were pleasantly surprised with its MAE and its score on the public leaderboard, and decided not to tweak the ensemble model further.

**4. Results**

NOTE: Some data and graphs in this section may not exactly match the results discussed in earlier sections or in the competition. This is due to the randomness inherent in training the models when we re-ran to do analysis after the competition.

We used a three prong approach to experimenting with the models. Each of us experimented with our own models and then would create the best model possible by experimenting with the ensemble method. Both models used trial and error to find the best hyperparameters once their structure was determined. Model 1 used an early stopping methodology with patience 5 to find the best epoch to stop the training of the model so it was not

overfitted. (see below).

```
Epoch 1/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 99s 933ms/step - loss: 139.3716 - mae: 139.3716 - val_loss: 0.2160 - val_mae: 0.2160
Epoch 2/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 96s 928ms/step - loss: 0.2550 - mae: 0.2550 - val_loss: 0.2381 - val_mae: 0.2381
Epoch 3/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 95s 925ms/step - loss: 0.2421 - mae: 0.2421 - val_loss: 0.2070 - val_mae: 0.2070
Epoch 4/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 96s 929ms/step - loss: 0.2141 - mae: 0.2141 - val_loss: 0.2546 - val_mae: 0.2546
Epoch 5/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 96s 931ms/step - loss: 0.2003 - mae: 0.2003 - val_loss: 0.2795 - val_mae: 0.2795
Epoch 6/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 95s 926ms/step - loss: 0.1966 - mae: 0.1966 - val_loss: 0.2083 - val_mae: 0.2083
Epoch 7/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 96s 930ms/step - loss: 0.1796 - mae: 0.1796 - val_loss: 0.2141 - val_mae: 0.2141
Epoch 8/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 142s 927ms/step - loss: 0.1726 - mae: 0.1726 - val_loss: 0.2034 - val_mae: 0.2034
Epoch 9/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 95s 923ms/step - loss: 0.1697 - mae: 0.1697 - val_loss: 0.1993 - val_mae: 0.1993
Epoch 10/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 98s 947ms/step - loss: 0.1644 - mae: 0.1644 - val_loss: 0.1900 - val_mae: 0.1900
Epoch 11/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 96s 931ms/step - loss: 0.1643 - mae: 0.1643 - val_loss: 0.1927 - val_mae: 0.1927
Epoch 12/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 96s 930ms/step - loss: 0.1583 - mae: 0.1583 - val_loss: 0.2220 - val_mae: 0.2220
Epoch 13/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 95s 925ms/step - loss: 0.1519 - mae: 0.1519 - val_loss: 0.2198 - val_mae: 0.2198
Epoch 14/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 95s 919ms/step - loss: 0.1545 - mae: 0.1545 - val_loss: 0.2301 - val_mae: 0.2301
Epoch 15/50
103/103 ━━━━━━━━━━━━━━━━━━━━ 142s 916ms/step - loss: 0.1536 - mae: 0.1536 - val_loss: 0.2842 - val_mae: 0.2842
```

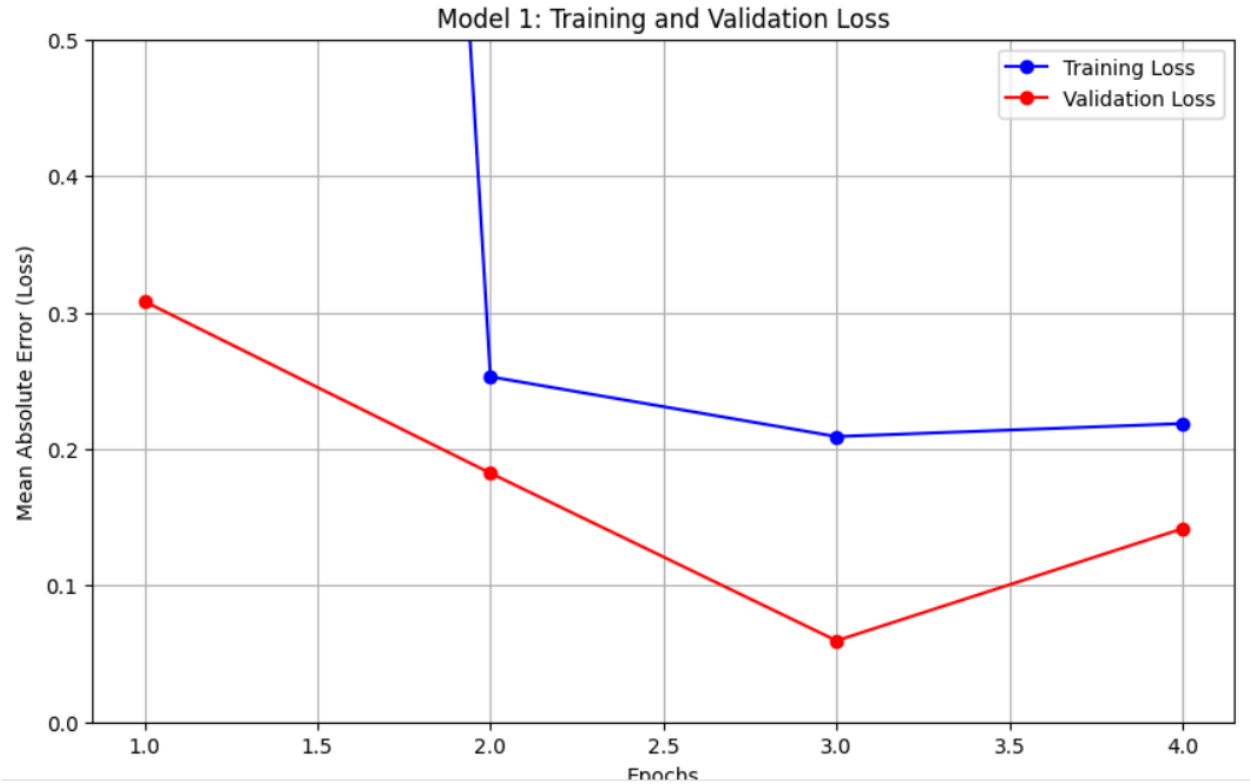
In this history above, the best validation loss was found at Epoch 10, and then when a better validation loss is not within 5 epochs, the model is stopped, saving time and preventing overfitting, as shown in the graph.

One thing we noticed in both the models was the immediate, dramatic drop in loss after the first epoch. This was due to the model going from having no concept of what it was describing, to quickly learning. This model also used the data from the timestamps to add another feature of analysis, and this improved the models performance from an MAE of 0.185 to 0.171. The most important features of this model that it brought to the ensemble were the use of the timestamps, the use of early stopping, and the validation data split.

The second model was hard coded to use 4 epochs based on trial and error. Testing began at 10 epochs and the best validation performance was found at 3 epochs. This was surprising, as

```
Epoch 1/4
93/93 ━━━━━━━━━━━━━━━━━━━━ 141s 1s/step - loss: 4.4391 - mean_absolute_error: 4.4391 - val_loss: 0.3081 - val_mean_absolute_error: 0.3081
Epoch 2/4
93/93 ━━━━━━━━━━━━━━━━━━━━ 138s 1s/step - loss: 0.2529 - mean_absolute_error: 0.2529 - val_loss: 0.1823 - val_mean_absolute_error: 0.1823
Epoch 3/4
93/93 ━━━━━━━━━━━━━━━━━━━━ 139s 1s/step - loss: 0.2091 - mean_absolute_error: 0.2091 - val_loss: 0.0596 - val_mean_absolute_error: 0.0596
Epoch 4/4
93/93 ━━━━━━━━━━━━━━━━━━━━ 139s 1s/step - loss: 0.2186 - mean_absolute_error: 0.2186 - val_loss: 0.1417 - val_mean_absolute_error: 0.1417
```

when running with 4 epochs, the validation loss was much lower, however when tested on the test data, the results were not better than with four epochs. Here are the graphs of the training data.




When testing the model, the use of the Leaky ReLU activation function caused an improvement in MAE, which was why it was included in the final model. In the individual submission, the MAE for this model was 0.22, which was not as good as Model 1, but this was acceptable, as it would add variability, and would potentially compensate for any overfitting in the first model. This model had some important distinctions when compared to the first model. First, it used slightly more validation data with a 90/10 split, which potentially avoided overfitting which was

important as this model was deeper, which could lead to overfitting. The model also used a LeakyReLU activation function in the MLP layers, which allowed for negative neurons to be included in the final model. These decisions added even more variability to the model.

When deciding which of the individual models to use in the ensemble, we used the best performing models on the public data. We used the linear regression ensemble method to determine the best weights to put on each of the models during the training and implementation of the model and found that the ensemble model outperformed both of the individual models on the training data. Model 1 was able to get a MAE of 0.171, and Model 2 was able to achieve a MAE of 0.223. The ensemble model was then able to get a MAE of 0.148, an improvement on both models, proving that our original hypothesis combining two good models would create a better one is correct. This was good enough to get the top score on the public dataset.

**5. Analysis, discussion, and conclusion**

We were able to create our ensemble model which improved on both individual models, which was our original goal. An MAE of 0.148 on the public dataset was something we could be proud of. We also know that our model was consistent. The model received a MAE of 0.15 on the public dataset, which was a very slight change compared to the public dataset. This is evidence that our model is not overfitted or under fitted, but will likely fit well regardless of the input. However, there are still certainly ways we could improve the model moving forward.

The regularization of the model was incorrect in both individual models. Model 1 had a correct regularization for the timestamps, but incorrectly divided the values of the images by 255, when the channel was expressed by over 70000 values. Model 2 failed to regularize the data altogether. If these were done correctly, the model could have been trained much more effectively. This could make the convolution layers more effective in finding patterns in the images and decreasing the MAE for the model.

The second way we could improve the model is tinkering with the ensemble methodology more. When we created the ensemble model, it immediately performed very well, which we did not feel to work further on. With more time, we could use different ensemble methods, which we discussed, and determined a Lasso model likely would have improved the results. It also would have been worth considering if adding more models to the ensemble would improve performance.

We were not surprised by the performance of our model on the private data versus the public data because the MAE was nearly identical. However, we were surprised to find that the models we had included for the ensemble were outperformed by other models we had built on the private data. This points to the fact that the models which improved on the private data were likely under-fitted, but is not clear if using those models instead of the ones we did end up using would be more effective. More testing on more data would be a useful step forward.

Concerning future work, there are certainly other ways to improve the model. One way which might help would be to increase the number of data samples. There were 3836 samples in the training data, but building on the convolutional model with more data, either through data augmentation or adding data beyond the training set in the competition could add to the model

becoming more accurate based on more training data. We also could add regularization techniques in the convolution layers. While we both tried dropout layers in the MLP layers, the model did not improve, but adding regularization techniques to the model in other places could have improved the performance. Finally, as previously mentioned, if we were to add other models to the ensemble, we may see an improvement.

Overall, we're very happy with the results of our model, which was consistent, accurate, and proved that the techniques we used were useful.