An Application of RNN: Crop Yield Prediction Using Planetary Data

Amy Liu
University of California, San Diego
Yil126@ucsd.edu

Abstract

Recurrent Neural Networks are well suited to model time-series data and are resistant to sequences of different lengths. In this study, we capitalize on unidirectional RNN, GRU, and LSTM algorithms to build an estimator for crop yield in the An Giang province in Vietnam. We train the model using the Sentinel-1 RTC dataset from the Microsoft Planetary Computer Catalog. We show that RNN-based models have a better performance compared to an extra-trees regressor and that GRU and LSTM predict with higher accuracy than RNN.

1. Introduction

This problem is adapted from the 2023 EY Open Science Data Challenge, and the provided notebooks and data can be found on the official website [2].

The goal of this study is to build a machine learning model that estimates the rice crop yield for the An Giang province in the Mekong Delta in Vietnam, using satellite time series data. Specifically, Vietnam is among the world's leading rice producers, but also among the most vulnerable countries to climate change. Thus, an accurate prediction model has a profound real-world impact.

Rice crop yield data was collected for the period of late-2021 to mid-2022 over the Chau Phu, Chau Thanh, and Thoai Son districts. This region is a dense rice crop region with a mixture of double and triple cropping cycles. For this study, we assume triple cropping for all data and focus on the Winter-Spring 2021-2022 season and the Summer-Autumn 2022 season, according to the challenge guidelines. When accessing the data, we define the Winter-Spring season as 2020/12/01 to 2021/04/30 and the Summer-Autumn season as 2021/04/01 to 2021/08/30.

2. Data

2.1. Target

Data for building the model comes from two main sources. First, the challenge website provides a table of crop yield data, which contains 557 locations in the district of interest, the geometric coordinates of that location, and information about the rice yield amount, which is the target. The only information from this table allowed for training is the location and the rice yield.

2.2. Features

The second source for building the model is the Microsoft Planetary Computer Data Catalog. Specifically, we consider the radar data from the Sentinel-1 RTC dataset and optical data from Landsat-8 and Landsat-9 datasets, which can be found in the Microsoft Planetary Computer Catalog [1].

2.2.1 Radar data

The Sentinel-1 mission uses C-band radar frequency, which corresponds to a 5.6 cm wavelength, at a 10-meter spatial resolution with a single mission revisit every 12 days. The radar signal is transmitted and received at either horizontal or vertical polarization, recorded as "H" or "V". The features provided in the Sentinel-1 dataset are VV and VH, with the first letter indicating the transmitted and the second the received polarization. These two radar signals contain information about surface backscatter on the earth's surface, which measures the crops' growth progress from small to large and then to bare soil after harvesting [3].

Using the VV and VH features, two other features can be generated: VV/ VH and the Radar Vegetation Index (RVI). According to the Sentinel Hub, the RVI is calculated by:

$$RVI = 4\sqrt{\left(1 - \frac{VV}{VV+VH}\right)} \cdot \frac{VH}{VV+VH} \qquad (1)[4]$$

Noticeably, since the lengths of the time windows for the two seasons are different, the resulting sequences for each point have different lengths. For easier training, all sequences shorter than the maximum sequence length, which is 26, are padded with zeros, so that the final training data has the size (557, 4, 26) with 557 samples, 4 features, and 26 observations. Since RNN is resistant to sequences of different lengths, it should take no information from the padded points.

2.2.2 Optical data

The two Landsat datasets combined contain data at a 30-meter per pixel spatial resolution with a revisit every 8 days. They include spectral bands such as Red, Green, Blue, Near Infrared (NIR), and Short-wave Infrared (SWIR), and can be combined to measure the greenness of the earth's surface, thus predicting crop yield [5].

However, one major problem with optical data is cloud filtering. Unlike radar data, optical waves cannot penetrate clouds, and considering the weather of the region of interest in this study, the raw data from Landsat satellites are highly contaminated. Hence, another band, the Quality Assessment band (qa_pixel), needs to be accessed to reduce the effect. It contains binary-encoded information on features such as whether the pixel is covered by snow/ice, whether the pixel is covered by cloud shadow, and whether the pixel is water [5]. Hence, a filter can be created to mask out contaminated regions [6]. Using the cleaned data, new features can be generated, including the Normalized Difference Vegetation Index (NDVI), the Enhanced Vegetation Index (EVI), and the Soil Adjusted Vegetation Index (SAVI) [5].

Due to limited computational resources, only Red, Green, and Blue bands were accessed from the Microsoft Planetary Computer API, so no cloud filtering was applied. The experiments section focuses on the radar variables from the Sentinel-1 dataset, which potentially suffers from having too few features. For further improvements of the model, optical data should be added to the training process.

2.3. Justification for extra credit

While accessing data, extra effort was devoted to three aspects. First, working knowledge of satellite data and agriculture was required. Understanding what each band stands for, how they can be combined to generate new features, and choosing the features required extra time and consideration. Second, data loading was accomplished through the Microsoft Azure Machine Learning workspace, rather than the Jupyterhub resource provided, because the required packages cannot be installed on Jupyterhub. Setting up the workspace and especially the API took extra effort.

Finally, the most challenging part was downloading the data in torch format and time window from the Planetary Computer Hub. Although a benchmark notebook was provided on the challenge website, it is customized to generate single-value features rather than a time sequence for each point. It is also customized for the Sentinel-1 RTC dataset. Hence, creating a new pipeline to get the time-series data from both Sentinel and Landsat datasets at any given location took extensive effort. Although the eventual training was done using the first dataset, this was only due to limited computation time on the Azure workspace with a free student account. The pipeline is finished, and more features can be easily added given a working Microsoft Azure subscription. The notebooks used to load the data and the Landsat data can be found at *https://github.com/YiyaoL/COGS181_project*.

3. Experiments

Three types of algorithms were tested in this section: Recurrent Neural Network, Gated Recurrent Unit, and Long-Short Term Memory. All models follow a similar architecture with an RNN layer, a batch normalization layer, and a linear function. An additional gradient clipping layer was added for models that suffer from the problem of exploding gradients. All models are unidirectional because the satellite signal of the preceding day should not be impacted by the signal of the succeeding day.

For each of the architectures, multiple models were generated with different combinations of hyperparameters. Each model was tuned using the first 500 data points and tested on the remaining 57 data points. The hyperparameters and losses are listed in Table 1.

3.1 Recurrent neural networks

First, the model was trained using the criterion Mean Squared Error. When tuning the second model, the resulting loss using MSE is infinite, so we switched to the criterion Mean Absolute Error.

Eight models were fully trained using a combination of batch size in [5, 22, 50, 100], number of layers in [5, 6, 10, 15, 20], the hidden layer size in [64, 128, 132, 200], nonlinearity in ReLU and tanh, optimizer in Adam and Stochastic Gradient Descent with different learning rates, and dropout percentage in [0, 0.1, 0.2, 0.3]. The batch sizes were chosen considering the size of the training sample size. The hidden layer size was chosen because the data had 4 features and more layers would hypothetically better characterize a more complex relationship between the time sequence. Model 7 was tested on the last 100 samples for a complete batch.

Except for the first model that used MSE, all other models were able to converge within 3000 epochs.

Noticeably, the final training losses of models 2,4,5,6,7, and 8 all converged to a number very close to 674. This was likely due to the limited number of training samples, because model 3, which was trained on the first 550 data points, converged to a training loss of 579, significantly smaller than the other training losses.

Model	**Batch Size**	**Number of Layers**	**Hidden Layer Size**	**Non-linearity**	**Optimizer**	**Learning Rate, Momentum**	**Dropout**	**Final Epoch Loss**	**Testing Loss**
RNN1	50	5	64	tanh	Adam	0.001	0.1	625831.8	6257591.3
RNN2	50	5	64	tanh	SGD	0.0009, 0.9	0.1	674.3603	673.9315
RNN3	22	15	128	ReLU	SGD	0.0015, 0.8	0.3	597.0633	N/A
RNN4	50	5	64	tanh	Adam	0.008	0.1	674.3603	673.834
RNN5	50	6	132	ReLU	SGD	0.003, 0.9	0.1	674.2168	673.9553
RNN6	50	20	64	tanh	Adam	0.008	0.2	673.9252	673.9213
RNN7	100	10	64	ReLU	Adam	0.008	0.2	673.7693	722.7276
RNN8	5	10	200	ReLU	Adam	0.008	0	676.2511	674.2165
GRU1	50	5	200	default	Adam	0.8	0.2	674	729.212
GRU2	50	7	200	ReLU	SGD	0.045, 0.9	0.2	864	880
GRU3	25	5	100	default	SGD	0.035, 0.9	0.1	673.7685	725.9825
GRU4	10	5	100	default	SGD	0.03, 0.9	0.1	N/A	N/A
GRU5	10	10	100	default	SGD	0.01, 0.9	0.1	673.764	725.9494
LSTM1	N/A	5	300	default	Adam	0.05	0	673.7953	725.9503
LSTM2	N/A	8	300	default	SGD	0.03, 0.9	0	673.7756	725.944
LSTM3	N/A	6	400	default	SGD	0.03, 0.99	0	673.7812	725.9059
LSTM4	100	5	100	default	SGD	0.5, 0.09	0.2	674.0472	674.1071
LSTM5	50	5	200	default	Adam	0.05	0.1	674.398	720.1632
LSTM6	20	10	100	default	Adam	0.05	0	N/A	N/A

Table 1: Trained models with hyperparameters and losses
Models with N/A final epoch loss failed to converge. RNN4, GRU5, and LSTM2 are used for cross-validation.

3.2 Gated recurrent unit

Next, five GRU models were trained using a combination of batch size in [10, 25, 50], number of layers in [5, 7, 10], hidden layer size in [100, 200], optimizer in Adam and Stochastic Gradient Descent with different learning rates, and dropout percentage in [0.1, 0.2]. For model 2, an additional activation layer was added using the ReLU function, but it resulted in poorer performance. Model 4 failed to converge. Models 1, 3, and 5 all converged to a final training loss at around 674, which is the same as for RNN. However, models 1 and 3, which used a smaller hidden layer size and the SGD optimizer generated better to the testing dataset.

3.3 Long-short-term memory networks

Finally, six LSTM models were trained using a combination of batch size in [10, 50, 100], number of layers in [5, 6, 8, 10], hidden layer size in [100, 200, 300, 400], optimizer in Adam and Stochastic Gradient Descent with different learning rates, and dropout percentage in [0, 0.1, 0.2]. For models 1, 2, and 3, no batch normalization layer was used. Model 3 was trained with the dataset scaled between 0 and 1, so the padded zero values were also transformed. However, this didn't have a significant impact on the training results. An additional gradient clipping step was added because the exploding gradient problem was encountered while tuning. Models 4 and 5 were tested on the last 100 samples for complete batches.

Further, the training time for LSTM models was longer every epoch as compared to the previous two algorithms, but it took much fewer epochs for the LSTM models to converge. Model 4 converged within 50 training epochs, and model 5 converged within 20 training epochs. However, model 6 had 25 batches in each epoch, so it could not converge in realistic training time.

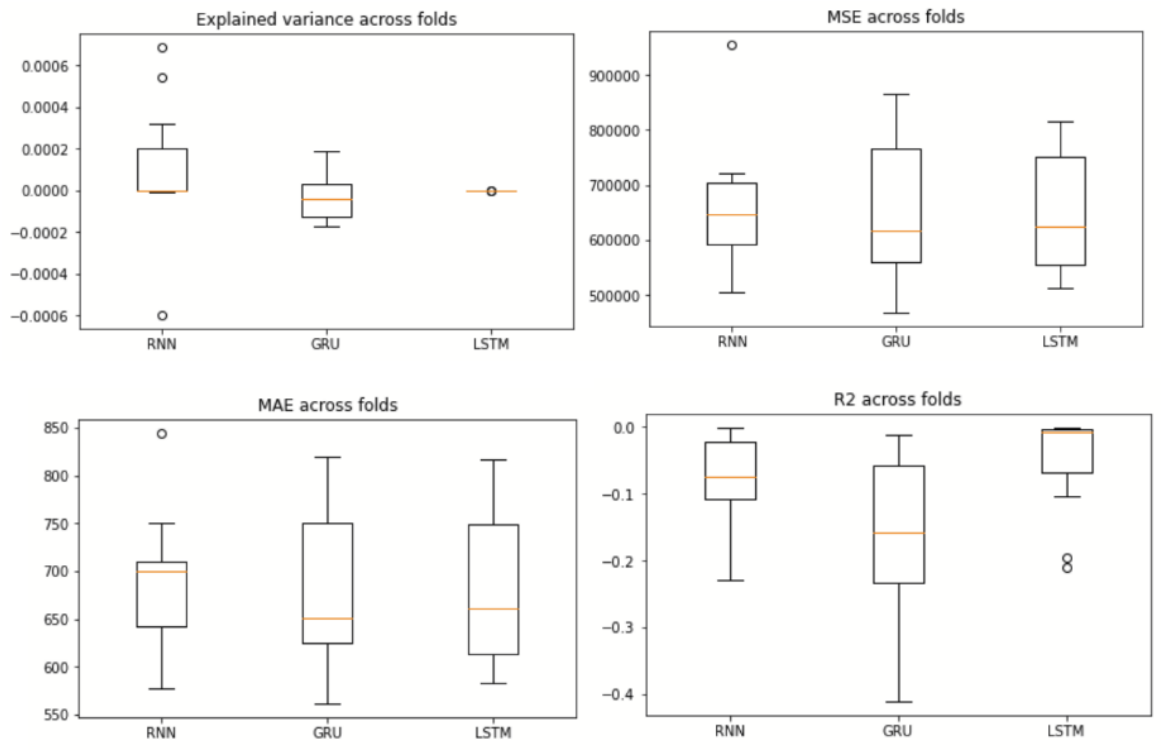
4. Discussion

One major problem present for each of the three categories was that none of the models converged to 0, and 674 was likely a local minimum. Model 3 in RNN indicates that this is probably due to a small training data size. A local minimum can be acceptable because the goal was to generate the model to different time ranges, not to fit the training data to the most. However, we want to achieve a better model that has a smaller training loss, so we will further explore this result in two aspects: cross-validation and comparison to a tree-based model.

4.1. Cross-validation

The first aspect we further explore is cross-validation. One model from each category that has the lowest training and testing error is chosen, and since many models had similar testing losses, we choose the one that converges within relatively fewer training epochs. These models are RNN model 4, GRU model 5, and LSTM model 2, as listed in Table 1. The learning rates for the GRU and the LSTM model are increased to 0.03 and 0.09 respectively for faster convergence.

A randomly selected and randomly ordered 550 samples from the dataset was divided into 11 folds for cross-validation. For each fold, four metrics are recorded: the total variance explained, the mean squared error, the mean absolute error, and the R2 score. The results are shown in graph 1. The results show that the LSTM model has the lowest average MSE and MAE, and the GRU model has the lowest median MSE and MAE, although both with larger variance compared to the RNN model. Overall, the LSTM model performs best. Further, none of the three models perform well in terms of the explained variance and the R2 score. Since MSE and MAE were used when training the model, this suggests that the model's performance can possibly be improved using the R2 score or the explained variance as the optimization criterion.



Graph 1: Cross-validation results

4.2. Extra-trees regressor

The second aspect we investigate is to compare the RNN-based model with a tree-based model. If this model generates a better result, it implies that RNN models may not be the most suitable for solving this problem, or that preprocessing should be improved.

For this purpose, an extra-trees regressor is trained using the same data but with different preprocessing, since it is incapable of processing a series as input. Instead of using the raw data, multiple features were generated. For each VV, VH, and VV/VH, the minimum, maximum, range, mean, correlation, and permutation entropy were calculated. The RVI was calculated using the mean. Thus, the data set has the shape (557, 19).

The samples were split with 10% in the testing set and 90% in the training set in consistency with the previous models. A regressor was fitted and multiple combinations of hyperparameters were tested, including the minimum samples per leaf in [3, 5, 10, 20], alpha in [0.01, 0.001], and the number of estimators in [50, 100, 200]. The training criterion was set to MAE in consistency with the previous models. The best-performing regressor on the testing set had an alpha value of 0.001, maximum depth of 30, minimum samples per leaf of 20, minimum samples per split of 5, estimators as 200, and all other hyperparameters as default.

This regressor had a training R2 score of 0.09, a training MAE of 663.52, a testing R2 score of -0.06, and a testing MAE of 753. In comparison, the average testing MAE of the best RNN, GRU, and LSTM models were 689, 681, and 676 during cross-validation, and 673, 725, and 725 during training respectively. Therefore, RNN-based models seem to perform better when predicting time-series data.

5. Conclusion

Three types of RNN-based models are proposed to estimate the rice crop yield for the An Giang province in Vietnam in the Winter-Spring and Summer-Autumn seasons. Compared to an extra-trees regressor, all three models generate better performances in terms of the mean squared error and the mean absolute error. The LSTM model and the GRU model perform better than the RNN model in cross-validation, resulting in a smaller median and average loss.

Further improvements and extensions can be made in three aspects. First, all three models were trained with 4 features generated from the Sentinel-1 RTC dataset, using radar information. Additional features generated from the optical information from the Landsat datasets should complement the current models. Second, current tuning is based on the MSE and MAE, but more criteria can be explored while tuning the hyperparameters, such as the R2 score and the explained variance. Third, more complex architectures, such as stacked RNNs can be used to characterize the relationship between the data.

References

[1] *Microsoft Planetary Computer*. Retrieved March 25, 2023, from https://planetarycomputer.microsoft.com/catalog.

[2] *Open Science Data Challenge*. Retrieved March 25, 2023, from https://challenge.ey.com/challenges/level-2-crop-forecasting-qEk17wFWyq/data-description.

[3] Rosenqvist, Ake & Killough, Brian. *A Layman's Interpretation Guide to L-band and C-band Synthetic Aperture Radar data, v2.0.* 2018.

[4] Sentinel Hub. *Radar Vegetation Index for Sentinel-1 SAR data - RVI4S1 Script*. Sentinel-Hub Custom Scripts. Retrieved March 25, 2023, from https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-1/radar_vegetation_index/.

[5] United States Geological Survey. *What are the band designations for the Landsat satellites? | U.S. Geological Survey*. Www.usgs.gov. https://www.usgs.gov/faqs/what-are-band-designations-landsat-satellites

[6] United States Geological Survey. *Landsat 8-9 Collection 2 Level 2 Science Product Guide | U.S. Geological Survey*. Www.usgs.gov.https://www.usgs.gov/media/files/landsat-8-9-collection-2-level-2-science-product-guide