IFT6759 - Project 1
Solar Irradiance Prediction
Team 1

Chih-Chao Hsu
Mila, Université de Montréal
chih-chao.hsu@umontreal.ca

Carolyne Pelletier
Mila, Université de Montréal
carolyne.pelletier@umontreal.ca

Nitarshan Rajkumar
Mila, Université de Montréal
nitarshan.rajkumar@umontreal.ca

Akshay Singh Rana
Mila, Université de Montréal
akshay.singh.rana@umontreal.ca

1 Introduction

This project is realized in the context of the IFT6759 machine learning project course at Mila, Université de Montréal. In this paper, we aim to predict **Global Horizontal Irradiance (GHI)** values at seven stations of interest located in the continental United States, called SURFRAD stations. The stations measure GHI values from the ground and are located in Bondville (IL), Table Mountain (CO), Desert Rock (NV), Fort Peck (MT), Goodwin Creek (MS), Penn. State University (PA), and Sioux Falls (SD). GHI values (measured in W/m^2) are important since they can be thought of as the 'usable' solar power per surface area, which is of interest to entities that care about the availability of solar energy. Being able to accurately predict GHI values could be important for Hydro-Québec for example, who might want to provide solar power to remote locations that are too far to be connected to the conventional grid, or to plan for surges or gluts of electricity from solar sources connected to the grid.

This task is a regression problem where the multi-modal inputs to the model are satellite images and station metadata, and the outputs are GHI values predicted at specific points on the imagery for both present and future times. More precisely, the task is called **nowcasting** and is defined as predicting GHI values at four timesteps into the future (T_0, T_0 + 1h, T_0 + 3h, T_0 + 6h) per station. We used 8-bit JPEG compressed versions of the original dataset for efficiency reasons. We used the Helios cluster with K80 GPUs for computing resources and TensorFlow 2.0 as the machine learning library for all experiments.

Baseline GHI prediction models rely on physical and mathematical models and have difficulty modelling spatial and temporal interactions using only satellite imagery. The main contribution of our paper is that we show that we can use a basic Convolutional Network (CNN) to look at present and past satellite images to predict GHI values at seven different locations, which outperforms baseline methods (ie. ClearSky Model). In more detail, our best performing model is a CNN with 3D convolutions with sequential images, batch normalization, and max pooling layers among other components. We also perform extensive data processing on the satellite images and concatenate spatial and temporal metadata as inputs to our model. Lastly, we manipulate our labels using ClearkSky GHI (residual_label = real_GHI - ClearSky_GHI) where the residual label (the difference between real GHI and ClearkSky GHI) helps our model learn. Note that since the task is to predict GHI values (and not the difference between real_GHI and ClearSky_GHI) we must add back the ClearSky_GHI value before recording our predictions. Inspired by [8], we hypothesize that adding the ClearSky GHI values to our model helps it learn the affects of cloudy versus clear days on GHI values.

We first start off the paper with a data analysis, where we cover important statistical findings that direct our methodology and results. We follow this with a literature review, covering three relevant papers that inspire our choice of models. Next, we go into much detail about our methodology which includes our extensive data processing pipeline, description of models, and learning procedure among other sections. Lastly, we present our experimental results and discuss them.

All code developed for this project can be found at https://github.com/leohsuofnthu/Solar-Irradiance-Prediction.

2 Data Analysis

2.1 GOES-13 Imagery

GOES-13 (Geostationary Operational Environmental Satellite) is a geostationary satellite that has a collection of sensors measuring atmospheric properties at multiple wavelengths. There are five bands, each measuring a specific event (i.e. central emission peak of CO2). This means that each image has 5 channels. GOES-13 images are available from December 15, 2008 to December 31, 2016, where the data is acquired at every 15th minute of every hour in UTC time. Three versions of the GOES imagery was provided in the project, and we used the lowest resolution option for computational reasons: **repackaged, 8-bit, JPEG-compressed (lossy) HDF5 (.h5) archives, in 1-day chunks (approx. 150 Gb in size)**.

2.2 SURFRAD Stations

SURFRAD stations consists of 7 stations at various locations in the continental United States. Each station's latitude and longitude coordinates are given so that one can locate the stations on the GOES-13 images. Each station measures solar irradiance and the data is available for the period of 2010-2016.

The teaching staff integrated both datasets into one, where the total size of the dataset is **210,336 by 33**. See Figure[1] for a sample of an image with its five channels represented separately, along with its corresponding station locations.

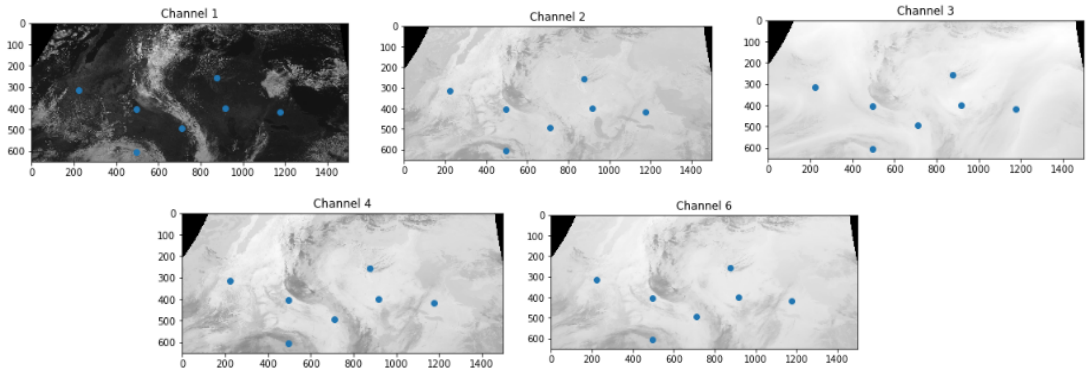

Figure 1: Sample of an image with its five channels represented separately, along with its corresponding station locations (seven blue dots). GOES-13 covers all of the continental United States.

We performed an extensive data analysis in order to guide our methodology. We first looked at the percentage of missing data from the images and GHI values. Of the total 210,336 datapoints, 82.59% of the datapoints had images, therefore **17.41%** datapoints were missing images. Next, we looked at the difference between the real GHI values and ClearSky GHI values as seen in Figure[2]. One can see that the real GHI values are lower than the ClearSky values, and we hypothesis that this is due to the fact that ClearSky does not incorporate cloudy images therefore their GHI values will be higher. Following this, we plotted with GHI values averaged over all years and stations for each month Figure[3]-Left. The underlying distribution hinted that we should undergo a cosine transformation on the monthly time data before we give it as input to our model Figure[3]-Right. This allows us to tell the model which months contribute to higher/lower GHI values. It is no surprise that the months of January and December (Winter months in the US), have low GHI values for example. We took a

similar approach and looked at the GHI values averaged over all years and stations per hour in a day Figure[7]-Left. The underlying distribution hinted that we should undergo a sine transformation on the hourly time data before we give it as input to our model Figure[7]-Right. This allows us to tell the model which hours in a day contribute to higher/lower GHI values. Finally, we took at look at the distribution of groundtruth GHI data Figure[5]-Left and ClearSky GHI data Figure[5]-Right using Bondville, IL station as an example. One can see that the histograms are skewed to the left, meaning that most GHI values are in the 0-50 range.

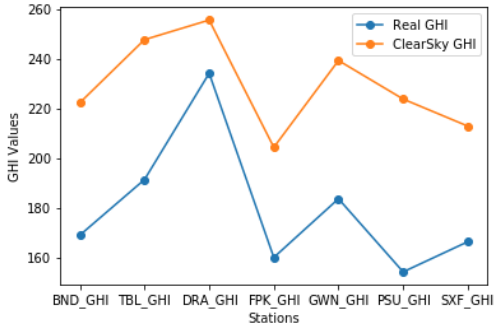

Figure 2: The difference between the real GHI values (lower values) and ClearSky GHI values (higher values). Both GHI values are averaged over all years and stations.

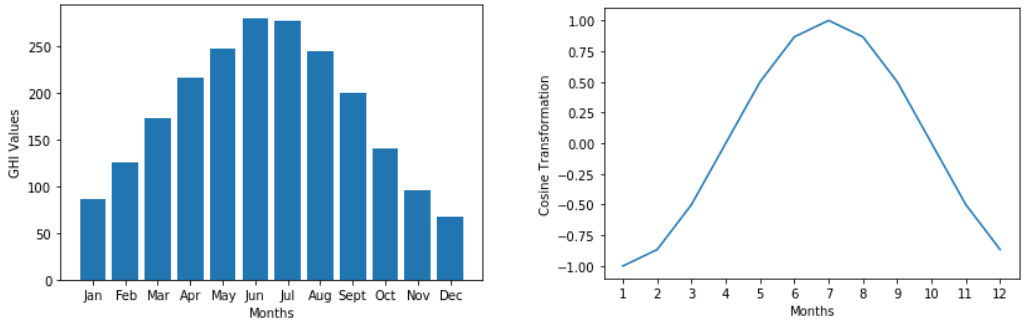

Figure 3: The left graph represents the real GHI values (y-axis) per month (x-axis) averaged over all years and stations. The right graph represents the real GHI values after applying a cosine transformation (y-axis) per month (x-axis)

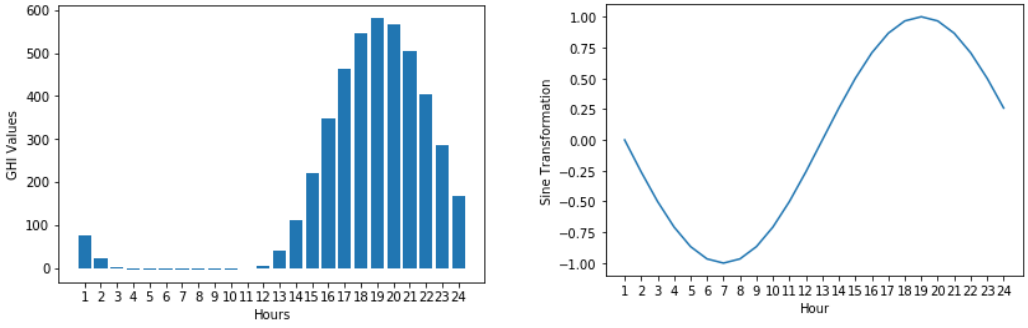

Figure 4: The left graph represents the real GHI values (y-axis) per hour in a day in UTC (x-axis) averaged over all years and stations. The right graph represents the real GHI values after applying a sine transformation (y-axis) per hour in a day in UTC (x-axis)

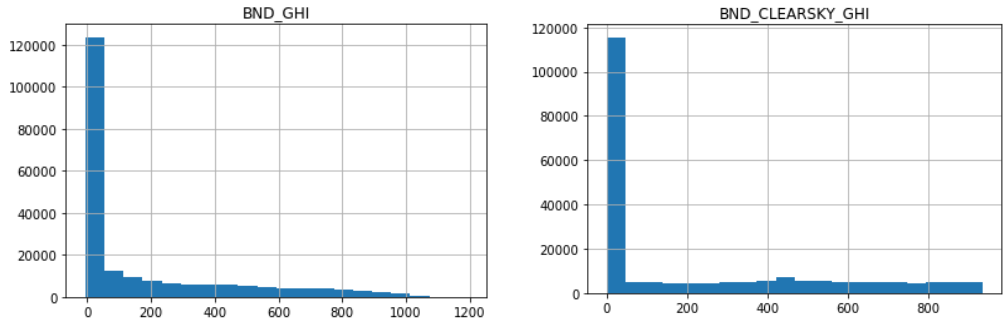

Figure 5: Histograms (bins=20) representing the distribution of data using the real GHI values (left) and ClearSky GHI values (right) which are represented on the x-axes, and where the y-axis is count. Data from the Bondville, IL station is used as an example.

3 Literature Review

In general, two approaches for improving upon a machine learning problem involve either getting higher quality data, or using better models (either in structure for the problem, or in parameter size, etc.). Palani et al. [6] approach the data quality problem, by making the observation that surface irradiance depends on local atmospheric conditions, even on perfectly clear sky days. Common approaches such as Ineichen and Perez [2] for clear sky modelling take a one-size-fits-all approach to capturing these conditions globally, and as such don't make use of the local variations that do occur and which are being measured anyways. The "Blue Sky" model of Panani et al is a location-aware estimate of the maximum GHI possible with clear skies, taking into account historical measurements of GHI for an area to construct better maximum irradiance bounds. They do so via first filtering out GHI values from the readings for a station which don't match the extraterrestrial irradiance curve (corresponding to cloud cover during the day, or other artifacts) and then filtering out readings from fully cloudy days via an unsupervised method to arrive at data from only truly cloud-free readings, and then finally fitting a model to these remaining data points. This method significantly improves upon existing clear sky models as measured by RMSE to ground truth GHI readings in multiple locations.

Many of the recent advances in model architectures for the problem of nowcasting have come through the use of convolutional neural network architectures applied to images. Siddiqui et al [8] make use of CNNs as feature extractors for the task of nowcasting and forecasting, given ground-based images of the sky from the SRRL [9] and MMTO [7] sky video datasets from Golden, Colorado and Tucson, Arizona respectively. These features are then used in an LSTM model for the forecasting task of predictions up to 4 hours ahead, to combine temporal information of the weather. A Key insight from their work is the use of dilated convolutions early on (a so-called atrous layer) to increase the receptive field of this convolution, which is important as spatially close features of cloud cover are relatively unimportant compared to the overall structure of the cloud. As well, the use of auxilliary non-image features (clear sky ghi, weather, wind, location information) was shown to greatly improve convergence and error of the trained model, especially in cases where image quality was poor (due to occlusion, shadows, weather effects, and other impacts on the camera lenses). The use of the clear sky ghi as an auxiliary input is found to be particularly helpful, as it allows the model to focus less on learning the periodic nature of the ghi, and to focus more on the impact of cloud and atmospheric conditions on reducing the maximum irradiance. These auxilliary inputs are concatenated with the features extracted by the CNN before being passed into a regular MLP for nowcasting, or the LSTM for forecasting. Early morning and late evening data is found to be unreliable (again due to shadows and occlusion), and due to the unimportance of these times for energy production they do not train on these times, instead focusing on the mid-day period when energy production is feasible.

Similarly to the previous approach, Zhao et al [10] make use of CNN models for feature extraction, but seek to develop better representations through the use of 3D convolutional kernels. Prior work [4] has demonstrated the ability of 3D kernels to extract both spatial and temporal features, and here they are applied to the task of forecasting DNI metrics 10-30 minutes ahead, using just a current image and the previous 2 images from a station. No recurrent architectures are employed here, and both the

nowcasting component (which sees the GHI range split into discrete tasks to train against) and the forecasting components make use of regular MLPs jointly trained with the CNN features. Due to severe class imbalance across the discretized targets of nowcasting, image augmentation is employed via contrast augmentation of the image. As with Siddiqui et al's method [8], these models are trained on ground-based images from weather stations.

From these methods we can see some general findings. Clear Sky models provide a good upper bound but there can be significant systemic variation from these physical models, which may vary by location. Separately, using CNNs for capturing image features has clearly worked on ground-based sky imagery with minimal domain knowledge and imperfect imagery, and the addition of auxiliary metadata greatly improves the performance of regression models built off of the CNN features as well. The temporal aspect required for forecasting can be approached with either CNNs or LSTMs. As well, the same model and feature space used for nowcasting can also be used for forecasting hours into the future.

4 Methodology

4.1 Data processing pipeline

Data Processing is an important and challenging task due to the sheer volume of the data (GOES-13 is about 960 GBs in size, but we used the compressed version using 8-bit jpeg compression which brought it down to about 150 GBs). Therefore preprocessing this data in order to extract only the useful information in order to speed up training is a crucial task. More efficient training allows us to test more models and perform extensive hyperparameter tuning. Our high level data processing strategy involves the following steps: Save data on team's drive, ignore/replace missing data, remove night-time image data, normalize images and metadata, crop images using h5 files, and finally save this preprocessed data on team's drive in mini-batches, where the size of the mini-batch is a hyperparameter.

4.1.1 Saving data on team's drive

Due to the high data regime, data processing was the most time consuming and challenging part of the project. First, since the data could only be accessed through a common path on the Helios server which everyone in the course accesses at different times, we decided it would be best to save our data following our data processing strategy on our personal team's drive on the Helios cluster. This decreased the number of times we had to access the slow shared folder path, as once we figured out our data processing strategy, we could save our processed images in our own personal folder which was more efficient to use.

4.1.2 Missing data, nighttime strategy, and normalization strategy

We ignored missing images, which accounted for 17.41% of the datapoints. For the missing GHI values, we found the neareast non-NAN GHI value before and after the missing value (using the timestamp) and linearly interporlated a GHI value. We removed all nighttime data from the dataset. For example, if two of the seven stations on the same timestamp was in nighttime, we would remove those two images, and kept the remaining five (this was done when making the minibatches). To include temporal information and its affect on GHI values, we incorporated hourly and monthly metadata as input to our model. We normalized this metadata via their respective sine and cosine transformations as seen in Figure 7 and Figure 3. As for the images, we normalized them in the following manner: for each channel (five in total), we subtracted the mean from the pixel value and divided this by the standard deviation (pixel - mean / std)

4.1.3 Image Cropping

Since we are only interested in the GHI values at SURFRAD stations, we crop regions centered on the coordinates of those stations. The crop size is a hyperparameter where we settled on 64X64 sized patches. While a larger patch might capture more information, it will slow down the training and testing of a model. Each crop was consistent across the 5 channels of each GOES-13 image.

4.1.4 Saving minibatches: non-sequential and sequential images

When saving our minibatches to the team's drive, we saved two versions: a non-sequential version where the sequence of the images were not kept, and a sequential version, where a look-back window of size four was used, meaning we took the present image and three previous images as input to the model (T_0, T_0 - 30mins, T_0 - 60mins, T_0 - 90mins). The minibatch size and look-back windown size

are both hyperparamters. Note that we kept the seven stations from the same timestamp in the same minibatch, and the data within the minibatch is shuffled.

4.2 Description of Models

For all models described below (Sections 4.2.1-4.2.4), we followed the following training procedure. First, since the task is to provide GHI values for both present and future times, we split the training and validation data over time periods as follows: training: years 2010-2014 and validation: year 2015. Images have a crop size of 64X64 and the look-back window size in our sequential data images is 4 timesteps, meaning we took the present image and three previous images as input to the model (T_0, T_0 - 30mins, T_0 - 60mins, T_0 - 90mins). Adam, which is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments, is the optimizer where its learning rate is a hyperparameter. Mean Squared Error is the loss and Root Mean Squared Error is the evaluation metric ; the lower the RMSE, the better. We kept the seven stations from the same timestamp in a minibatch (they are however shuffled), where the minibatch size is a hyperparameter. Lastly, we manipulate our labels using ClearkSky GHI (residual_label = real_GHI - ClearSky_GHI) where the residual label (the difference between real GHI and ClearkSky GHI) helps our model learn. Note that since the task is to predict GHI values (and not the difference between real_GHI and ClearSky_GHI) we must add back the ClearSky_GHI value before recording our predictions.
After the literature review, we decided to adapt the model from [8] to our dataset. The models we have tried are all the variant form of this model.

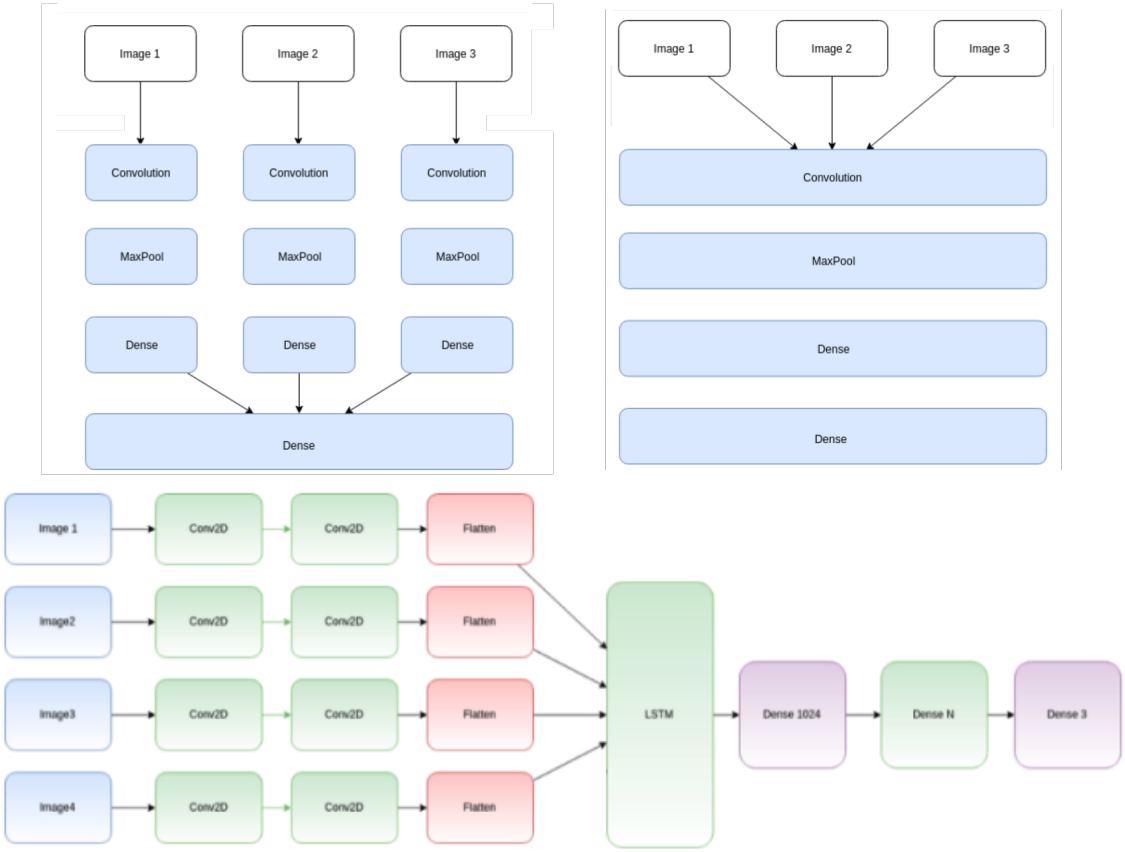

Figure 6: Top-Left: Conv2D with sequential images. Top-Right: Conv3D with sequential images. Note that the sequential look-back window in these images are of 3, whereas in our model we had a window size of 4. Bottom: Conv2D with sequential images and an LSTM.
Images are adapted from [1].

4.2.1 2D Convolutions with non-sequential images

In this model, we start with 128 dilating filters with the initial filter size of 7 x 7 and the dilation rate of 4 x 4. It is then followed by 64, 3 x 3 convolution filter and downsampled by max-pooling 2 x 2. After this, there are two layers of 128 filters, 3 x 3 convolution and three layers of 256 filters, 3 x 3 convolution. Each layers is followed by the same max-pooling setting as mentioned above. The final layer is a dense layer that reduced the feature to length 512. Only T_0 image are used here since this is non-sequential images. With this model, we observed overfitting, and the best validation RMSE was near 180. Thus we reduce the number of 128 filters and 256 filters convolution layer to 1 but increase the dense layer unit to 4096 with random contrast as data augmentation, because we assume that contrast could help model to learn and generalize the coverage of cloud so as to predict more accurate GHI value. The results shows that even reduce the capacity the best validation did not improve significantly with this model.

4.2.2 3D convolutions using sequential images

The 3D convolution approach convolves multiple images together across the temporal dimension by using a 3D kernel. The first convolution layer is connected to all consecutive images across our sequence. The images in our sequence are separated by 30minutes and can help the model learn the patterns in the cloud movements. The fusion of consecutive satellite images enables the network to detect cloud motions and accurately predict the irradiance. Our architecture consists of 3 convolution layers, 3 pooling layers followed by another 2D convolution layers. We also use 2 fully-connected layers in the end. All the layers are initialized using Xavier and the activation function is ReLU. The loss function is the mean-squared error and is optimized using the Adam optimizer. All the experiments of the model were performed using K-80 GPU.

4.2.3 2D convolutions using sequential images with fully-connected layer

Instead of feeding all the images together in one layer, we convolved each image in the sequence independently and connected all the feature maps together using a fully-connected layer in the end. This makes it easier to identify patterns among each image since we use different weight kernels for every image in the same sequence. Although it didnt really improve the model and instead it seemed to overfit on the training data. We used 2 convolution layers, 2 pooling layers followed by a dense layer in the end to concatenate all the features.

4.2.4 2D convolutions using sequential images with stacked LSTM

This model is quite similar to the previous model where we use separate convolutions on each image and then pass each feature map as a sequence to the stacked LSTM followed by a Dense layer. The output from the second LSTM is concatenated with the metadata to produce a 4 length vector of outputs. As we are working on frames of images that are chronologically ordered, LSTM could help understand the relation between each frame. Due to a large possibility of hyper parameters, we could not explore this network completely and it seemed to overfit on the training data. The results for this model are encouraging and this model can be used with high training data.

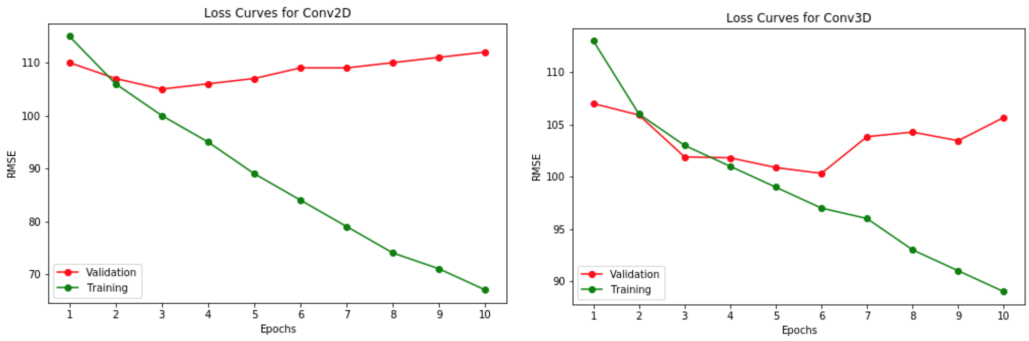

Figure 7: The left graph represents the training and validation curves on the Conv2D model with sequential images and the right graph represents the training and validation curves on the Conv3D model with sequential images, which was our best performing model.

5 Results and Discussion

Algorithms	BS	LR	$RMSE_1$	$RMSE_2$
ClearSky Baseline	n/a	n/a	205	-
Conv2D with non-sequence	128	0.01	142.36	109.21
	128	0.001	139.74	107.56
	256	0.001	136.14	104.59
	512	0.01	133.83	106.52
	512	0.001	122	105.24
Conv3D with sequence	128	0.001	-	102.81
	128	0.01	-	103.16
	256	0.001	128.32	**99.01**
	512	0.01	147.02	101.10
	512	0.001	145.28	100.88
Conv2D with sequence + LSTM	128	0.001	139.12	105.22
	128	0.01	-	107.12
	256	0.001	142.06	103.09
Conv2D with sequence + Dense	256	0.001	149.51	109.06
	128	0.001	154.11	106.99

Table 1: Results from five different models with their respective best hyperparameters, where **BS** stands for **batch size** and **LR** stands for **learning rate**. Validation RMSE was trained over time periods as follows: training: years 2010-2014 and validation: year 2015. $\mathbf{RMSE}_1$ represents the RMSE on the validation set **without residual labels**. $\mathbf{RMSE}_2$ represents the RMSE on the validation set **with residual labels**.

5.1 High Data Regime

Working in a setting of a high data regime, we encountered some unique difficulties. We ran into Out Of Memory (OOM) errors quite often. For examples, we wanted to work with TimeDistributed layers from Keras [5], but we were unable to do so due to OOM. We hypothesize that perhaps TimeDistributed layers are parallelizing convolutions across a batch for each sequence. Therefore, our data with dimension (512, 4, 64, 64, 5) will give you 2048 convolutions for (64, 64, 5) in parallel. In a lower data regime, the TimeDistributed layers would have been nice to use since they seamlessly transform our non-sequential images that are fed into the CNN, into sequentional ones, that we can then use to feed into an LSTM, which as we know, is typically used to predict sequential data. See Figure for a visualization of this model. Another limitation of the high data regime is that we had to choose a batch size while saving our data on the team's drive which made it difficult to conduct a proper hyperparameter search for batch size.

5.2 Estimating the Test Error

As explained in section 4.2, we split the training and validation data over time periods as follows: training: years 2010-2014 and validation: year 2015. A better approach to estimating the test error is inspired by K-Fold Cross-Validation (K-Fold CV), where rather than randomly divides the training data into K folds of approximately the same size, we would divide the folds by year. For example, the first fold is treated as the validation set (year 2015), and the model is trained on the rest k-1 folds (years 2010-2014) and compute the error. This procedure is then repeated K times, where at each time, a different fold is used as the validation set. One then takes an averages of the K errors. When compared to our original approach, this method gives more accurate estimates of the test error rate due to the bias-variance trade-off [3, p.183]. Since our final model is to be tested on a hidden test set, the test error estimate strategy is an important component of the project. Also, creating better estimations of the test error avoids overfitting. However, while this strategy inspired by K-Fold Cross Validation is statistically more accurate, due to its increase in computation, we could not implement it on this volume of data.

6 Conclusion

In this paper, we predicted **Global Horizontal Irradiance (GHI)** values at seven stations of interest located in the continental United States, when given multi-channel satellite imagery of these locations. Our best performing model was a 3D-Convolutional Neural Network using sequential images with a look-back temporal window of 4 timesteps, using dilated filters, batch normalization, and residual (from Clear Sky GHI) targets. The high data regime was our biggest challenge, and our extensive data processing pipeline, which includes image cropping and normalization techniques, helped tackle this challenge. Injecting temporal metadata (hour and month) to our model helped with capturing time-based effects along with the satellite imagery which captured spatial interactions. While this work focuses around predicting GHI values at specific stations, we expect our model to generalize well to other locations as long as latitudinal and longitudinal information is given since we can evaluate the ClearSky value and predict the offset from our model. In future work, we could choose to use higher resolution images which could result in better RMSE values. We could also experiment with different window look-back sizes for the images (we only experimented with a window of size 4), to see if it affects the results. Also, we could perhaps look into getting a better ClearSky model in order to get better residual labels, which would most certainly improve the results.

7 Acknowledgement

Thanks to our TA Krishna Murthy for his weekly guidance.

References

[1] *https://medium.com/smileinnovation/how-to-work-with-time-distributed-data-in-a-neural-network-b8b39aa4ce00*. https://medium.com/smileinnovation/how-to-work-with-time-distributed-data-in-a-neural-network-b8b39aa4ce00. Accessed: 2020-02-27.

[2] Pierre Ineichen and Richard Perez. "A new airmass independent formulation for the Linke turbidity coefficient". In: *Solar Energy* 73 (Sept. 2002), pp. 151–157. DOI: 10.1016/S0038-092X(02)00045-2.

[3] Gareth James et al. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. URL: https://faculty.marshall.usc.edu/gareth-james/ISL/.

[4] Shuiwang Ji et al. "3D Convolutional neural networks for human action recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013). ISSN: 01628828. DOI: 10.1109/TPAMI.2012.59.

[5] *Keras Time Distributed Layer Wrapper*. https://keras.io/layers/wrappers/. Accessed: 2020-02-27.

[6] Kartik Palani et al. "Blue Skies: A Methodology for Data-Driven Clear Sky Modelling". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 3777–3783. DOI: 10.24963/ijcai.2017/528. URL: https://doi.org/10.24963/ijcai.2017/528.

[7] T. E. Pickering. "The MMT all-sky camera". In: *Ground-based and Airborne Telescopes*. 2006. ISBN: 0819463329. DOI: 10.1117/12.672508.

[8] Talha Ahmad Siddiqui, Samarth Bharadwaj, and Shivkumar Kalyanaraman. "A deep learning approach to solar-irradiance forecasting in sky-videos". In: *CoRR* abs/1901.04881 (2019). arXiv: 1901.04881. URL: http://arxiv.org/abs/1901.04881.

[9] T. Stoffel and A. Andreas. "NREL Solar Radiation Research Laboratory (SRRL): Baseline Measurement System (BMS); Golden, Colorado (Data)". In: (). DOI: 10.7799/1052221.

[10] Xin Zhao et al. "3D-CNN-based feature extraction of ground-based cloud images for direct normal irradiance prediction". In: *Solar Energy* 181 (2019). URL: https://doi.org/10.1016/j.solener.2019.01.096.