GHI Projections: A Timely Matter

Maxime Daigle
DIRO
Université de Montréal
Montréal, QC

Annabelle Martin
DIRO
Université de Montréal
Montréal, QC

Kun Ni
DIRO
Université de Montréal
Montréal, QC

Marc-André Ruel
DIRO
Université de Montréal
Montréal, QC

1 Introduction

In the context of the transition towards renewable energy it is necessary to overcome new technology specific challenges. In the case of solar energy, it is necessary to be able to predict Global Horizontal Irradiance (GHI) in order to predict output of solar panels. That knowledge would enable power grids to adapt to the fluctuations in power production, and optimize the choice of energy at any given moment.

In this paper, we will use satellite imagery to predict GHI, understood as "the power per unit area (watt per square meter, W/m2), received from the Sun in the form of electromagnetic radiation." To obtain a successful prediction, different models that could apply were identified and were applied in a similar fashion. The main data source was satellite imagery coming from geostationary satellites (GOES-13) capturing atmospheric properties based on the Mid-Infrared (MIR) visible electromagnetic spectrum. This main data source was complemented by data coming from seven SURFRAD stations located in the United States. This meta data provided us with solar irradiance measures on a minute per minute bases, and was used for the benchmark against which to test the predictive model.

The Root Mean Square Error (RMSE) was used as the main model evaluation metric by comparing between predicted GHI and measures from the stations. RMSE was chosen as it is a frequently used to measure the difference between observed values and predictive model generated values. RMSE measures the predictive power of a model by aggregating the error in a single value that compares forecasting error in the output of different model running on the same data set. Being scale dependant, it cannot do the same thing for different data sets. Starting at 0 to represent perfect fit, the value of RMSE will go up to represent a higher error rate. RMSE was chosen in part also because of how it is sensitive to outliers since the weight of each error on the score is proportional to its squared size.

2 Data

2.1 Sattelite image

The main dataset used to train the models were preprocessed GOES-13 (Geostationary Operational Environmental Satellite) imagery with five channels of usable data spanning the electromagnetic spectrum that can be detected by Mid-Infrared (MIR). Covering fixed coordinates, the imagery were separated by 15 minute intervals from April 2010 to December 2015. It correspond to approximately 175,000 timestamps. The dataset contained missing dates and absent pixels that were pretagged as missing or unavailable. When preparing the data all dates past December 2014 were kept for the final test set. Since we are only interested to predict the amount of solar irradiance and there is some only

when it's daytime, only daytime hours were kept in an effort to limit the size of the data used and improve the balance and usefulness of the training data.

At certain times, the images are always unavailable (i.e. at 0:00, 0:30, 3:00, 6:00, 9:00, 12:00, 15:00, 15:30, 18:00 and 21:00). However, there is also multiple time intervals where the images are missing for long period (e.g. 4 months at the start of 2010). Some images have missing/corrupted pixels.

The raw images are quite large (toward 1 TB) and contains 16-bit channels. To speed up training and data management, we use the lossy repackaged 8-bit compressed images. Furthermore, for this task, we only need to focus on small area in the image. So, the images were cropped with the areas of interest at the center, decreasing the size even more.

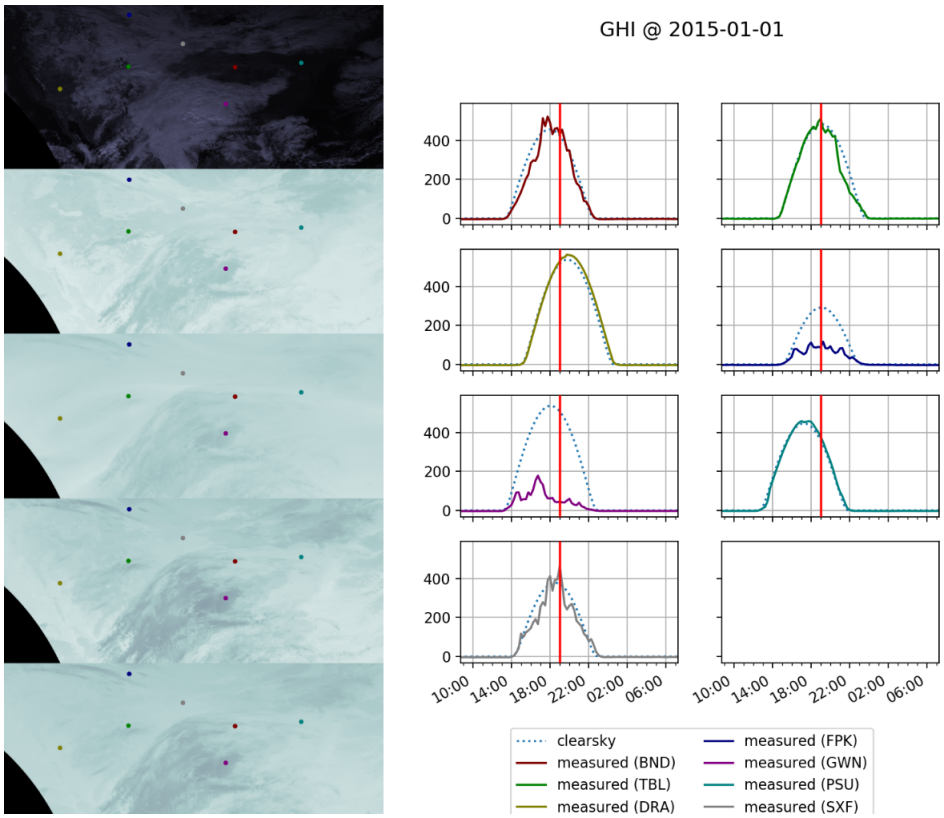

Figure 1: Sattelite imagery and GHI values of a day

2.2 Metadata

The supplemental metadata, the SURFRAD data, was comprised of solar irradiance measures collected in 1 minutes intervals from 2010 to 2015 at 7 stations across North America. The data was smoothed with moving averages that aligned the data with the GOES-13 data on a 15 minute time scale. Understood as the "ground truth" this is that data used to test the models. The data set also contained clearsky GHI estimates based on the pvlib package that uses the model defined by Ineichen and Perez (2002). A binary flag determining day time status could also be found in the data set. The figure below easily demonstrate how the metadata can be useful to project actual GHI at different location. Each of the station represented here shows a different distribution even though the data collection was done through the same program. The latitude, longitude and elevation can definitely impact the GHI throughout the day since we know that each SURFRAD stations were established at clearly distinct location. The GHI is regularly unavailable around midnight and at some specific hours, roughly every three hours from midnight.

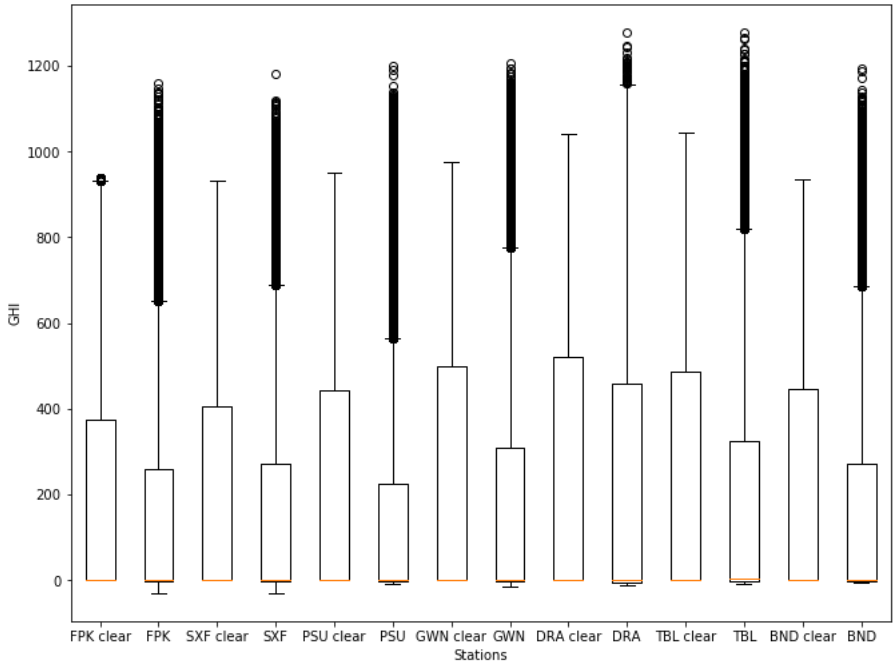

Figure 2: Clearsky vs Measured GHI

The data is skewed toward zero because most of the time the solar irradiance is low (morning, evening, night). We can see that Clearsky tends to overestimate the GHI and that there is a high volatility with numerous high GHI.

3 Related work

We reviewed papers with similar objectives or similar data to identify potential solutions to our problem. Some well known solutions for processing video data or image related to each other through time are CNN 3D and a CNN-LSTM.
Zhao et al. [2019] was able to successfully apply a CNN-3d to predict direct normal irradiance (DNI) over a short timespan. They used ground-based cloud images to try and predict the DNI over a timeframe of 10, 20 and 30 minutes. The temporal resolution they use is 10 minutes and they found through cross validation that getting image going back 3 time steps gave the best results. We assumed that this would translate to our task as that we would probably obtain better results by selecting image as far back in the past as we plan to project in the future, in this case 6 hours. However, obtaining multiple past images to process through the CNN 3D slowed down our model drastically. Zhao et al. [2019] also found that using the CNN 3D coupled with an MLP gave better results than using each one separately. The MLP they trained gave better prediction during broken-sky days. This helped us selecting a type of data combination for our own models. Since we have metadata that we believed could help with the prediction, we decided to include them in a similar way and concatenate the metadata to our processed satellite image layer and create a final small MLP to obtain the final predictions. We applied this method to most models we tested.

We reviewed another paper from Ribeiro et al. [2017] where the author used CNNs trained on different input channels, like RGB and LUV, to obtain better results on a classification task by combining the CNNs. Since we work with satellite image that contains 5 channels instead of the usual 3 channels in most classification task, we couldn't apply the same idea directly but we might be able to obtain better results by training more CNN on different inputs and combining their results afterwards. We combined this idea with what we read in Guo and Yang [2018] where the authors trained different ResNets on images that were preprocessed in a different manner for each ResNet. We didn't apply multiple preprocessing of each image, however, we did already have five channels that contained different information. Therefore, we might be able to use multiple ResNet but instead

of different preprocessing method, we use different channels.

We also reviewed paper Xingjian Shi and Woo [2015] which uses a convolutional LSTM to predict rainfall over a period of 6 hours. The authors propose to use convolution layer through an LSTM to take advantage of both model which takes image features and sequencing information into account to project the expected rainfalls. The authors determined that this type of model should be suitable for spatiotemporal problem which is the base of our task for this project. However, we were unsuccessful in applying this type of model to our task at hand.

4 Dataloader

Our first objective was to create a dataloader which would be able to support multiple different model-architectures to limit the amount of preprocessing and duplicating of large amounts of data. This came at a cost of programming time and some performance. At first, simply getting an image, decompressing and cropping it, would take more time then a single batch to train a simple model even with the 'prefetch' option. To remedy this, we leveraged the 'interleave' option which allowed parallel calls to different dataloaders (i.e. different image day files) to populate a single batch. It seemed to have fixed the performance issue until past images were also required by more complex models.

The tensorflow cache has its good and bad traits. Once the cache is build, it is saved and can be reused for next runs and epochs, just like a preprocessing phase. On the other hand, it would also mean that the data at that point would be stale and hurt the model's capacity to generalise better if it's trained on the same cache for too long.

5 Models and Training

Except MLP baseline model, all models share same training setup. Same dataloader to return image and meta data. The optimizer is Adam with initial learning rate of 0.0001. The loss function is MSE. Every 100 steps, training losses are recorded in summary log file. After each epoch, validation is executed. If validation RMSE gets lower than previous lowest RMSE, best checkpoints will be saved. MLP baseline model is trained without image data, so it is trained separately. The Adam optimizer with learning rate 1e-5 is used. Regarding regularization, early stopping is used which will stop the training if validaion score does not improve more than 0.0001for 10 consecutive epochs.

5.1 Preprocessing

We trained the models on random subsets of days sampled between the years 2010 and 2014. Most of the models used around 20% of the training data and we reserved the year 2015 strictly for validation. Most of the reprocessing is done on the fly while the data loader gets the values. First off, we skip over entries that have no GHI values and no image at T0. If the model require past values and the image was missing, we copied the last valid image used (e.g. T0 if T-1 has no image). All the models were trained on cropped images centered around the station locations making 64x64 pixel images. The images are then preprocessed. If the images are used in a pretrained model, the images are simply preprocessed the same way that they were when the original model was trained. Otherwise, images are normalized.

One important step was to set cyclic time encoding. It is to make it easier for model to interpret the closeness of values. For example, 23h59 and 00h01 are very close time wise, but numerically they are harder to link without this encoding. We compute the cos and sin value of normalized months and 15 minutes offset since midnight. This encoding is performed after each station has it's own time converted to it's own timezone. The model could learn that the sun goes up a certain time of day and help the prediction to other locations in the country. With those encoding, the metadata used becomes the time encoding, latitude, longitude, altitude, and the clearsky estimate.

If one of the GHI values were missing, we would use the corresponding clearsky GHI value. As a last resort, we would use the last valid GHI value (worst case it's T0's value) if no information was available.

5.2 Multilayer Perceptron without image data

To quickly build the simple baseline without any image data, we train the Multilayer Perceptron model. The model is trained without any image data and only predict current timestamp. The features used are [Station_index, daytime,sin_month,sin_minute,cos_month, cos_minute] . It is a very naive model as clearsky model. Dataset is split as percentage of 20/80. The reported RMSE is 120.244. So we believe with image data of T0 and past images, we should be able to get lower RMSE.

5.3 Vanilla CNN2D

To train basic model with image data, we first come up with simple CNN2D model with 3 dense layer and 2 dense layer. The result is bad as expected. But it provides the simplest model to validate the data loader. Based on this model, we think of two directions to design our new model 1) advanced network structure which can provide more expressive power to capture image features, 2) Integrating meta data other than image data to provide richest info to model. According to above guidelines, we experiments on different advanced network architectures, for example, ResNet, CNN3D, ResNet-LSTM, Double ResNet-LSTM, ResNet-Seq2Seq.

5.4 ResNet

5.4.1 Vanilla ResNet-50

To improve the performance of the CNN, the next step is to use an architecture known to be effective with images. ResNet (He et al. [2015]) is modified by removing the top layers. The output of ResNet is concatenated to some metadata and is fed to two fully connected layers with ReLU (respectively of size 1000 and 4).

5.4.2 Pretrained ResNet-50

As stated in Huh et al. [2016], it is frequently beneficial to use ImageNet to learn general features only to then transfer the weights to the actual task wanted. Therefore, transfer learning is an attractive possibility. However, ImageNet contains 3 channels and our task has 5. It implies that the architecture of the pretrained ResNet is the same as Vanilla ResNet except that it must have only 3 channels. To catch the widest range of information possible with the limited channels, We select the first, third, and last channel to have the widest spectrum of wavelength (550 to 750 nm, 5800 to 7300 nm, and 13,0 to 13,7 μm).

5.4.3 Double Pretrained ResNet-50

Since it is not ideal to completely ignore some channels, to further improve the use of pretrained ResNet, we use the idea from Guo and Yang [2018]. They use multiple ResNet in parallel and each receives a different version of the original image. Here, we use two ResNet in parallel and each receives a different combination of channels. The first ResNet receives the first, third, and last channel for each images and the second ResNet receives the second, third, and fourth channels. Both output are then concatenated to the meta data and fed into 3 fully connected layers with ReLU activation (respectively of size 2048, 1000, and 4). With this architecture, we combine the advantage of pretraining, ensemble method, and use all channels.

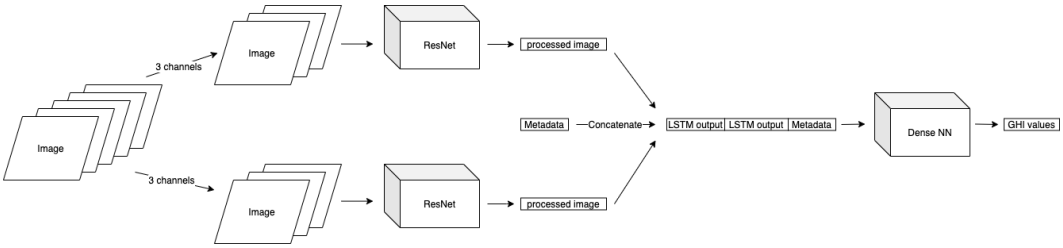

Figure 3: Double Pretrained ResNet architecture

5.5 CNN3D

The CNN3D is the classical approach when thinking about analysing videos. The hope is that it will pickup moving cloud formations and be able to better predict future GHI values. A small CNN3D proved to have the capacity to learn, which was our first step before adding more layers to the model. The final model has 10 convolution layers and 3 fully connected layers at the end with leaky relues, two batch norms and dropout layers. The dropout layers will help the model not to overfit as much and the batch normalization layers help improve the learning performance and stability.

An important distinction about this model is it does not use any other meta information for predicting GHI values. Only the satellite images were used to train and validate. We used 5 images from the past with 1h intervals.

5.6 ResNet-LSTM

In Mathe et al. [2019], CNN and LSTM is used to predict the GHI value of given periods. CNN is used to extract the spatial features while LSTM is used to extract temporal features. With the past observations, LSTM can capture the progression of clouds and predict the future timestamp whether the station is going to be affected by clouds. However, we encountered OOM erros after training 1 epoch of the model. We thought that the model had too many parameters and didn't fit into the GPU memory. So we reduced the batch size and number of past timestamps. However, the problem was still persisting. At last, we realized that it is a known bug in tensorflow 2.0.0. It is fixed in latest version; the TimeDistributed class from Tensorflow 2.0 has a memory leak. Therefore, building a CNN-LSTM is problematic with Tensorflow 2.0. Thus, it is necessary to manually import the fix to be able to use the model on calculquebec cluster. Once the fix imported, we reused the well working ResNet from above and feed it to our LSTM layer at the end.

5.6.1 Double ResNet-LSTM

Double ResNet-LSTM tries to improve the model by mixing both ideas of CNN-LSTM and Double Pretrained Resnet. The architecture is similar except that after the ResNet, instead of directly concatenating the processed images, they are fed into a LSTM. Optionally, there is a fully connected layer between the ResNet and LSTM.

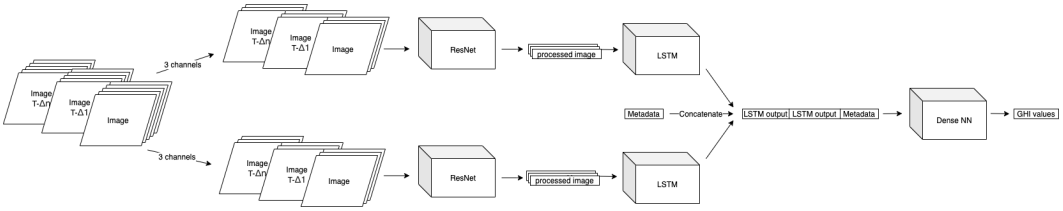

Figure 4: Double ResNet LSTM architecture

5.6.2 ResNet-Seq2Seq

In ResNet-LSTM and Double Resnet-LSTM, only the last hidden states is used to predict the target GHI values. With LSTM, we are able to reason over the past images, but the reasoning is not good as expected. So we wonder if attention mechanism can help to reason better. So we create this model with GRU encoder over past images, attention vector to store info about context and GRU decoder to predict the four GHI values.We train the model with past 3 images of [30mins, 1hour, 1hour30mins]. The result is better than ResNet-LSTM.The best validation RMSE is 199.4.

6 Results

Here is the summary of best validation RMSE from all models we have been trained. And in below sections we will discuss the result in two different groups. First group doesnot use past images, while second group uses past images.

Table 1: Best validation RMSE of all models

Model	RMSE
MLP	120
ResNet	217
Pretrained ResNet	174
Double Pretrained ResNet	115
ResNet-LSTM	272
Double ResNet-LSTM	302
smaller Double ResNet-LSTM	259
CNN3D	185
ResNet-Seq2Seq	199

6.1 ResNet models

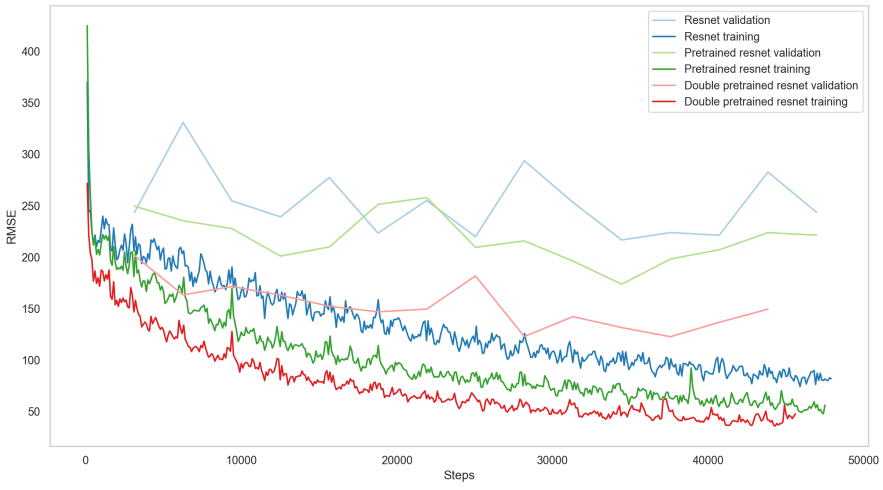

Figure 5: Comparison of different models using ResNet

To quickly find the best model, each model are trained on a small random sub-sample of the training dataset. The model with the highest RMSE is ResNet with 5 channels and without pretraining. By transferring the weights obtained from training on ImageNet, it improves the performance even if it means not using all channels. As mentioned before, using ImageNet for transfert learning is known

to be effective and, thus, is a de facto standard for solving a wide range of problems (Huh et al. [2016]). Finally, by stacking two ResNet, it significantly improves the performance by combining the advantages of using an ensemble method, all channels, and pretrained weights.

Table 2: Best validation RMSE

Model	RMSE
ResNet	217
Pretrained ResNet	174
Double Pretrained ResNet	122

By using Double Pretrained ResNet, it decreases the RMSE on the validation set by almost half when compared to the simple ResNet. However, even though Double Pretrained ResNet obtains the best result, there is a not insignificant gap between the training and validation. To reduce the gap and obtain a model that generalize more, the capacity of Double Pretrained ResNet is slightly reduced by reducing the number of hidden neurons in the dense layers and the model is trained on more data (40% of the training set).

Figure 6: Reducing the capacity of ResNet

The smaller capacity increases the training loss, but the RMSE on validation set is more under control. The bigger training set and the smaller capacity reduce the best validation RMSE from 122 to 118 which is an improvement even if it's not by much. Furthermore, training on 100% of the training data lower the RMSE to 115.

6.2 CNN3D

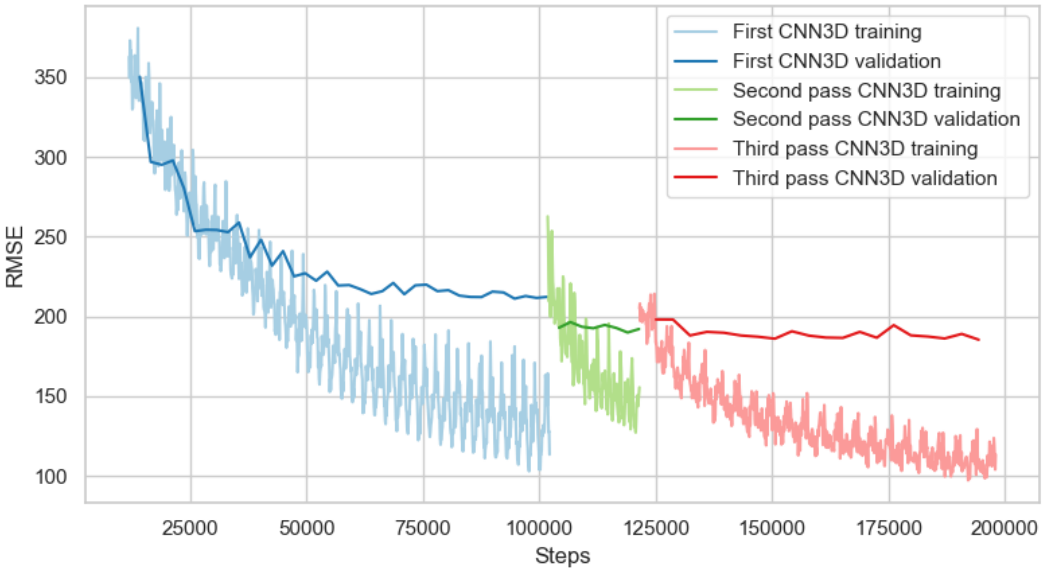

Figure 7: CNN3D training and validation curves

From Figure 7, we can clearly see the effect of the stale data when using the cached data from our dataloader. When lunching a new training training from a previous checkpoint on new data, we see a big improvement step between validation curves. Even so, the third time we lunch the model to train on another sub-sample of data, the validation seems to plateau at the same place as the previous run. The validation curve seems to plateau from that point on.

The best validation achieved was **185** RMSE. We were expecting this model to actually perform better on future predictions then the Double Resnet since it leverages past images. The pretrained weights make a big difference for the pretrained Resnet models. If we train the Resnet from scratch, the CNN 3D perferms better faster. But transfer learning and good general features already learnt from Imagenet help to fine tune the model so it can perform well on other tasks such as these. Data augmentation and fine tuning would probably result in better results for the CNN 3D, but would end up being more time consuming.

6.3 ResNet-LSTM

We were not able to make the ResNet-LSTM performs well. It is likely that the right architecture was not found.

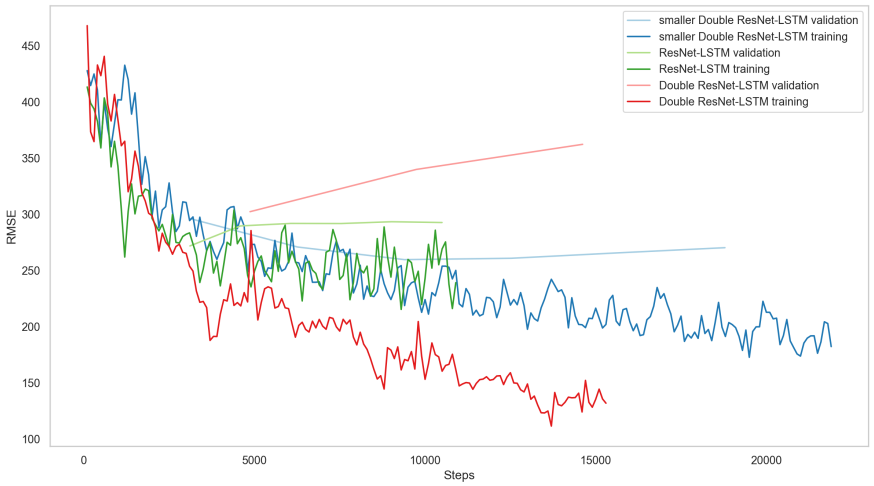

Figure 8: ResNet-LSTM and Double ResNet-LSTM

The single ResNet-LSTM does not seems to overfit. Unfortunately, the learning is saturating around 250 RMSE. As for the Double ResNet, it is learning rapidly, but is overfitting to a great extent. After, reducing the capacity by reducing the number of hidden units, the smaller Double ResNet-LSTM doesn't seem to overfit. However, it has similar validation results as the ResNet-LSTM at around 250 RMSE.

Table 3: Best validation RMSE

Model	RMSE
ResNet-LSTM	272
Double ResNet-LSTM	302
smaller Double ResNet-LSTM	259
ResNet-Seq2Seq	199

The Double ResNet seems to help improve the performance, however only slightly, and the performance was at first pretty poor. Therefore, the improvement is not significant and some important architecture changes would need to occur to fix it.

6.4 ResNet-Seq2Seq

We experiment about adding attention into past images and predict with last hidden state and attention vector state. From the training curve Figure 9, we can see model learns really quick in the first 1000 steps and then saturates. From the validation curse, we can see that model doesn't generalize well. The best validation RMSE is 199. In this training, dropout layer with dropout rate of 0.5 is added before the final output. However, it does not help well on the final result. So attention mechanism may not help to reason in context better in this specific task.

7 Conclusion

The final results shows that we were able to leverage the potential of pretrained models even though the number of channels of our input was different. The features learned by ResNet on the classification task of ImageNet are also useful for the regression problem of predicting solar irradiance. Our final model is very good at predicting GHI at time 0 and an hour later. However, it does progressively worse the farther ahead of time we project. The cropped image we selected

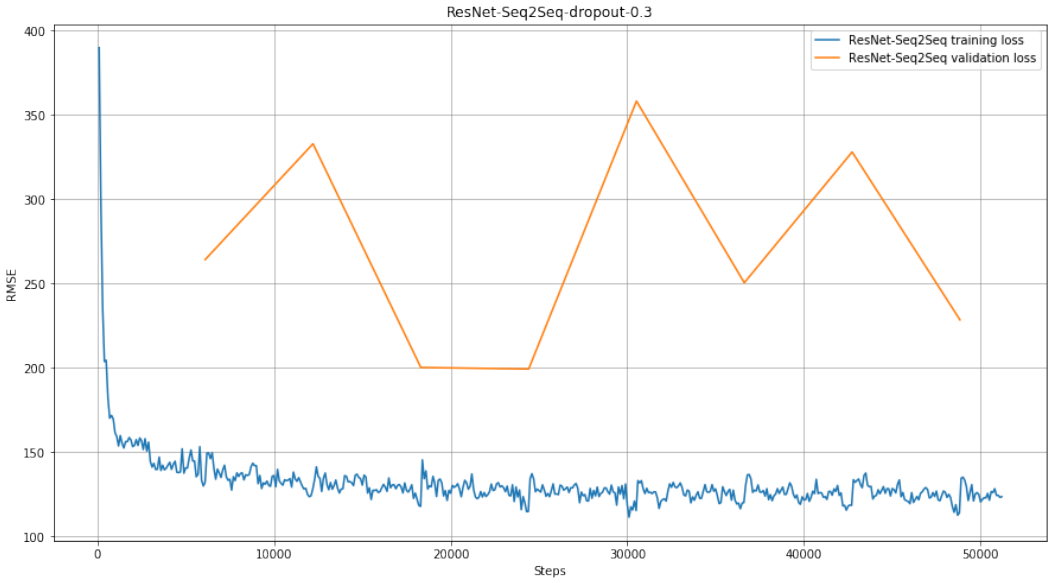

Figure 9: ResNet-Seq2Seq training and validation curves

might not contained all the information that would be useful to correctly project far in the future since weather can change and wind speed over time can affect the irradiance level if clouds or storms appears which would most probably change the expected level.

Although we obtain good results on our validation set, we have some ideas that could improve our results. Our best model uses ResNet. However, it dates from 2015 and, since then, there is multiple architecture that performs better on ImageNet (Inception-ResNet from Szegedy et al. [2016], Xception from Chollet [2016], NASNet from Zoph et al. [2017], etc.) and simply swapping ResNet for one of them (or multiple) could improve our results. Also, we are using a cropping of 64x64 and images compressed with 8 bits only to be able to experiment faster. Trying images of better quality and bigger cropping could be a way to improve the quality of information fed to the model. Additionally, we did not have time to do a hyperparameter search and try different meta data configurations. Finally, we did not succeed to make a CNN-LSTM work, but it would be unsurprising that the LSTM could improve the result by adding a temporal component to the model.

References

F. Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. URL `http://arxiv.org/abs/1610.02357`.

S. Guo and Z. Yang. Multi-channel-resnet: An integration framework towards skin lesion analysis. *Informatics in Medicine Unlocked*, 12:67 – 74, 2018. ISSN 2352-9148. doi: https://doi.org/10.1016/j.imu.2018.06.006. URL `http://www.sciencedirect.com/science/article/pii/S2352914818300868`.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

M. Huh, P. Agrawal, and A. A. Efros. What makes imagenet good for transfer learning? *CoRR*, abs/1608.08614, 2016. URL `http://arxiv.org/abs/1608.08614`.

J. Mathe, N. Miolane, N. Sébastien, and J. Lequeux. Pvnet: A LRCN architecture for spatio-temporal photovoltaic powerforecasting from numerical weather prediction. *CoRR*, abs/1902.01453, 2019.

D. Ribeiro, G. Carneiro, J. Nascimento, and A. Bernardino. Multi-channel convolutional neural network ensemble for pedestrian detection. pages 122–130, 05 2017. ISBN 978-3-319-58837-7. doi: 10.1007/978-3-319-58838-4_14.

C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. URL http://arxiv.org/abs/1602.07261.

H. W. D.-Y. Y. W.-K. W. Xingjian Shi, Zhourong Chen and W.-C. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *NIPS*, 2015. URL https://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowcasting.

X. Zhao, H. Wei, H. Wang, T. Zhu, and K. Zhang. 3d-cnn-based feature extraction of ground-based cloud images for direct normal irradiance prediction. *Solar Energy*, 181:510–518, 03 2019. doi: 10.1016/j.solener.2019.01.096.

B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017. URL http://arxiv.org/abs/1707.07012.