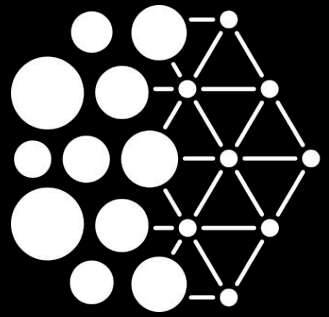
Quebec
Artificial
Intelligence
Institute



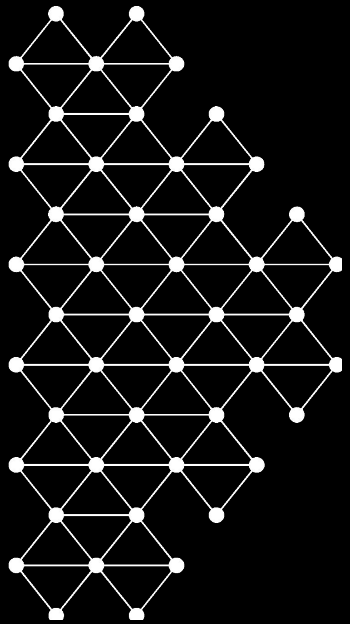
Mila

IFT6759 - Project 1
Solar Irradiance Prediction

Jeremy.Pinto@mila.quebec
PierreLuc.StCharles@mila.quebec

Quebec
Artificial
Intelligence
Institute

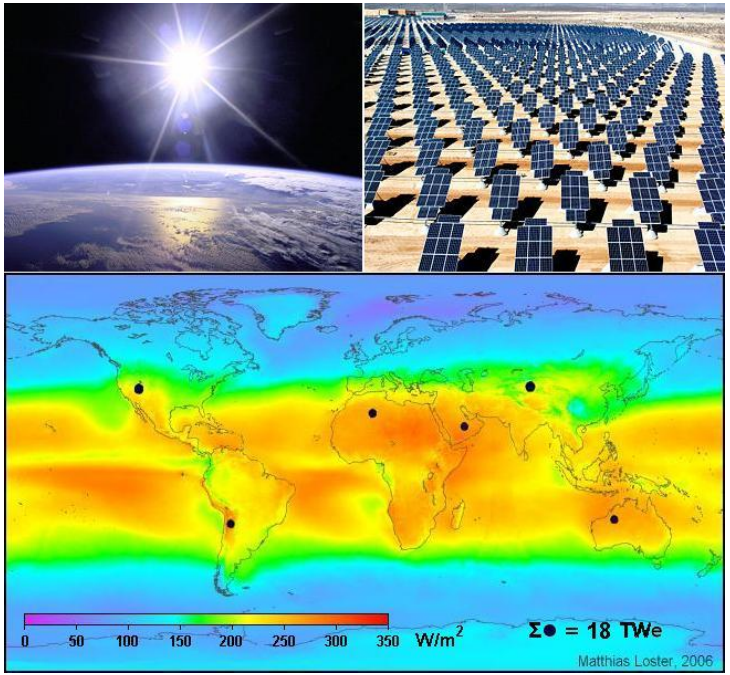
Mila



Project Description

Solar Irradiance

"**Solar irradiance** is the power per unit area (watt per square meter, W/m^2), received from the Sun in the form of electromagnetic radiation" - Wikipedia



https://commons.wikimedia.org/wiki/File:Solar_energy.jpg

Solar Irradiance and GHI

- Physical factors (elevation, latitude, longitude, etc.) impact how much solar irradiance is available at a given location on earth.
- External factors (atmospheric conditions, clouds, etc.) can affect how much “effective” solar irradiance reaches a horizontal surface on earth.
- This “effective” irradiance is called **Global Horizontal Irradiance (GHI)**. It is measured in [W/m^2]. It takes into account **both** physical factors and external factors and can be thought of as the “usable” solar power per surface area.
- **GHI** can be measured by stations on the ground.

The Desert Rock SURFRAD station

Use cases for GHI

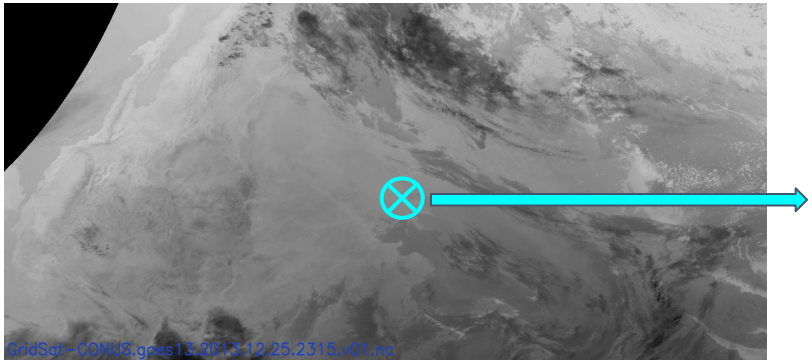
Use cases for GHI values:

- Forecasting the availability of solar energy
- Climate modeling
- ...

Goal of the project

Using satellite imagery, we are looking for robust predictive models to provide **GHI** values at specific points on the map for both present and future times. Specifically, we are interested in the task of **nowcasting**, i.e. predicting the GHI up to 6 hours in the future. For this project, we are interested in the GHI at four timesteps:

$$T = T_0, \quad T = T_0 + 1h, \quad T = T_0 + 3h, \quad T = T_0 + 6h.$$



GHI values:

T_0 = 433.2 W/m^2
T_0+ 1h = 443.5 W/m^2
T_0+ 3h = 263.1 W/m^2
T_0+ 6h = -3.2 W/m^2

Left - GOES-13 satellite while it was being built
Right - Image taken from GOES-13 satellite

Existing solutions

Existing GHI prediction solutions rely on **physical and mathematical** models.

These can work pretty well in some cases, but don't always capture **complex temporal phenomena** (clouds, winds, etc.). To improve their performance, some models use **satellite imagery**, but still have difficulty modeling spatial and temporal interactions. Also, some models only do instantaneous observations and cannot make predictions **in the future**. Finally, some models are behind paywalls.

We hypothesize that **satellite imagery** can be used to help model these complex phenomena and seek to develop models leveraging **open data sources**.

Existing solution: Clear Sky Model (CSM)

The clear sky model accounts only for **some** effects of the atmosphere on solar irradiance. It predicts the solar irradiance when the **sky is "clear" of clouds**. This can be thought of as an upper-bound to the actual GHI value. It takes into account surface pressure, zenith angle, turbidity, etc.

There are many open-source implementations of the clear-sky model. Given latitude, longitude and a datetime, one can compute the clear-sky value of GHI for a given point in time.

Remember: this is an **imperfect** model! You might find observed GHI measures that are higher than this supposed "upper bound"...

Clear sky — pvlib-python
A Physics-Based GOES Satellite Product for Use in NREL's National Solar Radiation Database: Preprint
Bird Clear Sky Model | Grid Modernization

Existing solution: Persistence Models

Persistence models are inherently naive models. They consider that the GHI at the next time step is the same as the observed GHI at the previous timestep, i.e.

$$GHI(T_0 + \Delta) = GHI(T_0)$$

While the predictions obviously degrade with time, they might be reasonable approximations for timesteps of ≤ 1 hour. They however require a known GHI at a given timestep, which isn't always available.

Persistence models can also be combined with clear sky models to take into account shift in the sun's position:

$$GHI(T_0 + \Delta) = GHI(T_0) * [\ GHI^{clear\text{-}sky}(T_0 + \Delta) / GHI^{clear\text{-}sky}(T_0)\ ]$$

Persistence Model - an overview

Existing solutions: Shortcomings

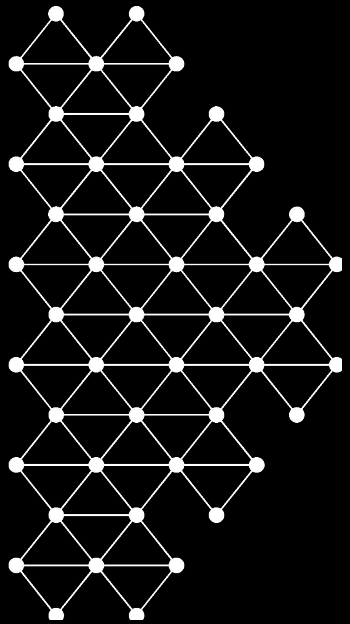
We have only presented a few of the existing models out there. We expect you to survey the literature as part of your project. Some of the shortcomings of other models you might run into are:

- “Observations” only - i.e. GHI(T_0) is available but not GHI($T_0 + \Delta$)
- Some models may not be open (i.e. are behind paywalls)
- Some models assume previous GHI values are known

This is still an active area of research. We are ultimately looking for models that are capable of doing nowcasting under various conditions with open data. We are also looking for models that can make predictions for every point on a map, and not be limited to points where reference GHI values are available. Although crucial, this latter requirement is relaxed in the scope of this project. We will only evaluate our models where stations on the ground are available.

Quebec
Artificial
Intelligence
Institute

Mila



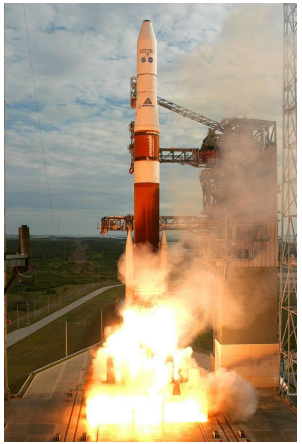
Data Sources

GOES-13

GOES-13 (Geostationary Operational Environmental Satellite) is part of a constellation of satellites measuring **atmospheric properties**. It is a **geostationary** satellite.

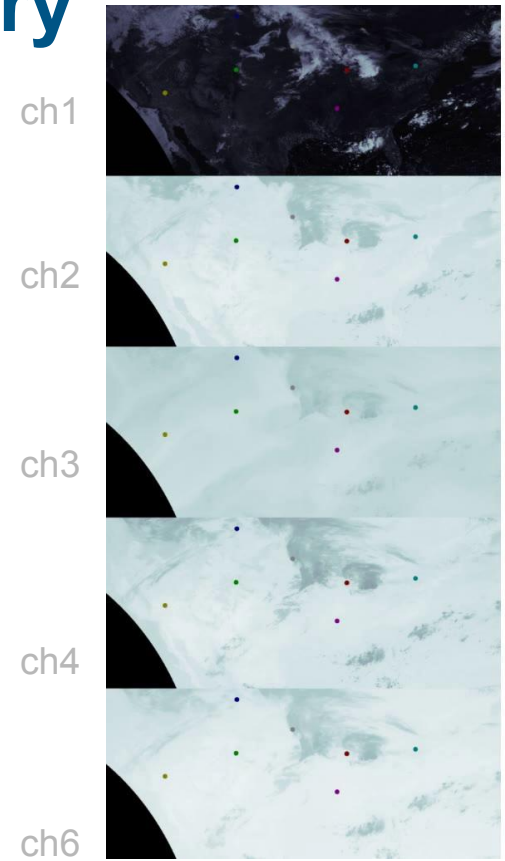
It has a collection of sensors capable of gathering data at multiple wavelengths. For example, one band of the satellite corresponds to the central emission peak of CO_2.

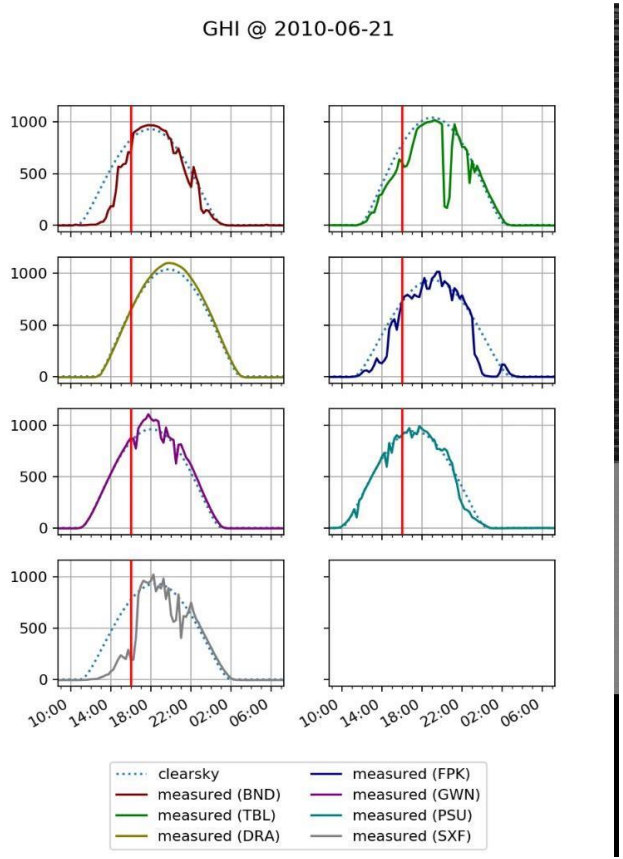
Channel	Wavelength	Description
1	550 to 750 nm	Red (visible spectrum)
2	3800 to 4000 nm	Infrared: Smaller wavelengths
3	5800 to 7300 nm	Infrared: Water vapor
4	10,2 to 11,2 μm	Infrared: Bigger wavelengths
5	Not Available	-
6	13,0 to 13,7 μm	Infrared: CO_2



GOES-13 Imagery

Here we visualize each available GOES-13 channel as well as how the GHI **evolves** over the course of a day at different GHI measurement stations (SURFRAD, detailed later).





Mila

GOES-13 Imagery

GOES-13 images are available from December 15, 2008 to December 31, 2016. Data is acquired at every 15th minute of every hour:

HH:00, HH:15, HH:30, HH:45

Where HH can be in the interval [0,23] and corresponds to UTC time.

Data is always unavailable at specific times:

0:00, 0:30, 3:00, 6:00, 9:00, 12:00, 15:00, 15:30, 18:00 and 21:00.

Data at other times can also sometimes be unavailable. It will be up to you to handle these cases when feeding data to your model.

SURFRAD

SURFRAD was originally put in place in 1993. Today, it consists of **7 stations** at various locations in North America. Each station measures solar irradiance every minute.

We have preprocessed & smoothed the SURFRAD data using moving averages to be aligned with the 15-minute acquisition frequency of GOES-13.

SURFRAD data is available for the period of 2010-2016.

SURFRAD stations

SURFRAD

Station Acronym	Station Name	Latitude	Longitude	Elevation (m)
BND	Bondville, IL	40.05192	-88.37309	230
TBL	Table Mountain, CO	40.12498	-105.2368	1689
DRA	Desert Rock, NV	36.62373	-116.01947	1007
FPK	Fort Peck, MT	48.30783	-105.1017	634
GWN	Goodwin Creek, MS	34.2547	-89.8729	98
PSU	Penn. State Univ., PA	40.72012	-77.93085	376
SXF	Sioux Falls, SD	43.73403	-96.62328	473

The Desert Rock solar tracker at sunset, December, 1999.

Missing Data (SURFRAD & GOES-13)

This is **not** ImageNet. Data can sometimes be missing for various reasons, for both SURFRAD and GOES-13. It can also possibly sometimes be corrupted. There are hundreds of thousands of entries, so be sure to keep that in mind when preparing your code.

Also keep in mind that these are two **distinct**, complementary datasets that we are using for this task. We have done our best to clean and align the data accordingly for you. Missing values are made explicit.

Data Indexing

We have done some preprocessing to streamline and simplify some requirements of the project. We are providing a pandas dataframe containing all the ground truth and metadata you will need for this project.

iso-datetime	ncdf_path	SXF_GHI	...
2010-04-15 04:00:00	/project/cq-training-1/project1/data/netcdf/GO...	-5.006667	
2010-04-15 04:15:00	/project/cq-training-1/project1/data/netcdf/GO...	-4.560000	
2010-04-15 04:30:00	/project/cq-training-1/project1/data/netcdf/GO...	-2.826667	...
2010-04-15 04:45:00	nan	-3.820000	
2010-04-15 05:00:00	nan	-4.053333	

This dataframe is described in more detail on GitHub.

Data Indexing

The dataframe **index** is a datetime. There is a valid index for every possible 15 minute interval from 2010-04-01 until 2015-12-31. All indices are valid ISO datetimes in UTC time.

iso-datetime	ncdf_path	SXF_GHI	...
2010-04-15 04:00:00	/project/cq-training-1/project1/data/netcdf/GO...	-5.006667	
2010-04-15 04:15:00	/project/cq-training-1/project1/data/netcdf/GO...	-4.560000	
2010-04-15 04:30:00	/project/cq-training-1/project1/data/netcdf/GO...	-2.826667	...
2010-04-15 04:45:00	nan	-3.820000	
2010-04-15 05:00:00	nan	-4.053333	

Data Indexing

For each datetime, the associated full path to the filename for the GOES-13 satellite image is given. If a value is missing, it is reported as NaN. These paths will point to files in a shared directory on Helios.

iso-datetime	ncdf_path	SXF_GHI	...
2010-04-15 04:00:00	/project/cq-training-1/project1/data/netcdf/GO...	-5.006667	
2010-04-15 04:15:00	/project/cq-training-1/project1/data/netcdf/GO...	-4.560000	
2010-04-15 04:30:00	/project/cq-training-1/project1/data/netcdf/GO...	-2.826667	...
2010-04-15 04:45:00	nan	-3.820000	
2010-04-15 05:00:00	nan	-4.053333	

Data Indexing

For each datetime, all ground truth values for GHI per station are given. We have also included values indicating whether it is daytime or nighttime at that given time for the given station. Predicting GHI at night time is easy since there is (usually) no sun. Notice here that values can be below 0. This is from the raw data and can be attributed to instrument calibration. This is typical for nighttime values.

iso-datetime	ncdf_path	SXF_GHI	SXF_DAYTIME
2010-04-15 04:00:00	/project/cq-training-1/project1/data/netcdf/GO...	-5.006667	0
2010-04-15 04:15:00	/project/cq-training-1/project1/data/netcdf/GO...	-4.560000	0
2010-04-15 04:30:00	/project/cq-training-1/project1/data/netcdf/GO...	-2.826667	0
2010-04-15 04:45:00	nan	-3.820000	0
2010-04-15 05:00:00	nan	-4.053333	0

Data Indexing

We have also included values of the clear-sky model so that you can compare your predictions to this model as well as a “cloudiness” factor based on the measured GHI and clear-sky model.

iso-datetime	SXF_GHI	SXF_CLOUDINESS	SXF_CLEARSKY_GHI	SXF_DAYTIME	…
2010-04-25 14:00:00	42.313333	cloudy	385.538429	1	
2010-04-25 14:15:00	55.313333	cloudy	433.321301	1	
2010-04-25 14:30:00	85.033333	cloudy	479.790734	1	…
2010-04-25 14:45:00	143.040000	cloudy	524.661381	1	
2010-04-25 15:00:00	226.306667	cloudy	567.677339	1	

Clear sky — pvlib-python
https://www.sciencedirect.com/science/article/pii/S0196890412002786?via%3Dihub
(PDF) Global horizontal irradiance clear sky models : implementation and analysis

Cloudiness Factor

We estimate the “cloudiness” on a given day using the delta between the GHI computed using the Clear Sky Model (CSM) and values measured at the ground.

“Cloudiness” is in itself a heuristic and is based on moving averages. It is a good approximation but not necessarily accurate.

This data should only be used for validation purposes, not as inputs to your model. It could help you evaluate whether your results are good even when the “clear sky” assumption fails...

$$k(t) = \frac{|x_{measured}(\Delta t) - x_{CSM}(\Delta t)|}{x_{CSM}(\Delta t)}$$

$$MA = \frac{1}{Nm} \cdot \sum_{s=0}^{Nm-1} k(\Delta t - s)$$

$$MF = \sum_{s=0}^{Nm-1} |MA - k(\Delta t - s)|$$

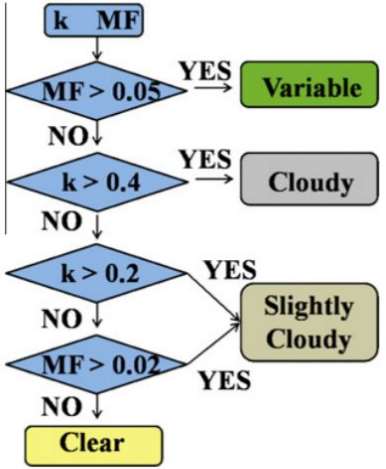


Fig. 1. Block diagram of the algorithm used to classified each time step considered during a day.

https://www.sciencedirect.com/science/article/pii/S0196890412002786?via%3Dihub

Data Indexing

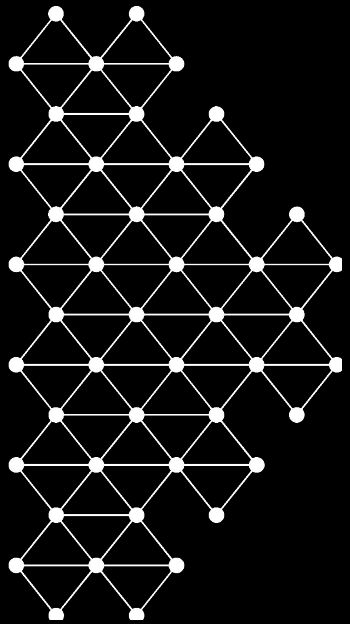
Here are all the fields you can find in the dataframe:

```
['ncdf_path',
 'hdf5_8bit_path',
 'hdf5_16bit_path',
 'hdf5_8bit_offset',
 'hdf5_16bit_offset',
 'BND_DAYTIME',
 'BND_CLEARSKY_GHI',
 'BND_CLOUDINESS',
 'BND_GHI',
 'TBL_DAYTIME',
 'TBL_CLEARSKY_GHI',
 'TBL_CLOUDINESS',
 'TBL_GHI',
 'DRA_DAYTIME',
 'DRA_CLEARSKY_GHI',
 'DRA_CLOUDINESS',
 'DRA_GHI',
 'FPK_DAYTIME',
 'FPK_CLEARSKY_GHI',
 'FPK_CLOUDINESS',
 'FPK_GHI',
 'GWN_DAYTIME',
 'GWN_CLEARSKY_GHI',
 'GWN_CLOUDINESS',
 'GWN_GHI',
 'PSU_DAYTIME',
 'PSU_CLEARSKY_GHI',
 'PSU_CLOUDINESS',
 'PSU_GHI',
 'SXF_DAYTIME',
 'SXF_CLEARSKY_GHI',
 'SXF_CLOUDINESS',
 'SXF_GHI']
```

...for more information on the dataframe, refer to the documentation on **GitHub**.

Quebec
Artificial
Intelligence
Institute

Mila

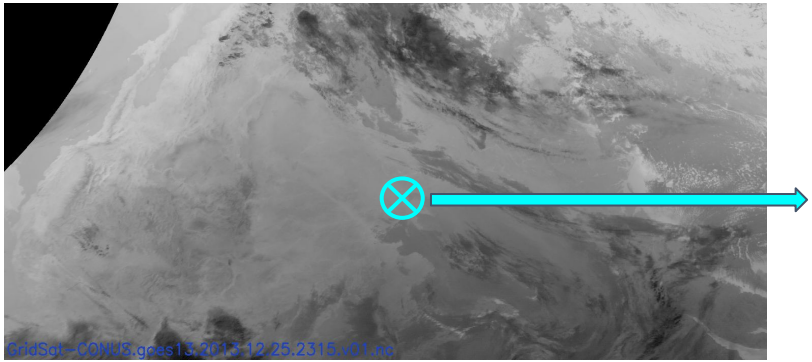


Task Specifications

Reminder: Goal

Using satellite imagery, we are looking for robust predictive models to provide **GHI** values at specific points on the map for both present and future times. Specifically, we are interested in the task of **nowcasting**, i.e. predicting the GHI up to 6 hours in the future. For this project, we are interested in the GHI at four timesteps:

$$T = T_0, \quad T = T_0 + 1h, \quad T = T_0 + 3h, \quad T = T_0 + 6h.$$



GHI values:

T_0 = 433.2 W/m^2
T_0+ 1h = 443.5 W/m^2
T_0+ 3h = 263.1 W/m^2
T_0+ 6h = -3.2 W/m^2

Left - GOES-13 satellite while it was being built
Right - Image taken from GOES-13 satellite

Limitations

Ultimately, the goal should be to **generalize** a GHI prediction model to the **entirety** of the map. However, since we only have 7 stations with an associated ground truth, we will limit our evaluation of the generalization to those 7 stations.

To evaluate our models, we will separate our train, validation and test sets over **time periods**. You will have 2010-2015 data for training and validation. We will keep 2016 data as a separate test set for the ultimate performance evaluation.

DO NOT DOWNLOAD the year of 2016. **While it is available online, we assume you will not cheat. If we suspect you are cheating, we will audit all of your code. In the case of reasonable doubt, we reserve ourselves the right to attribute zero to the project.**

Limitations

Keep in mind that you **cannot** use “past” GHI values as input data, since it would not be available for all points on the map. The same goes for the cloudiness factor since it is derived using the measured GHI. You can use the clear sky model since it is a general model.

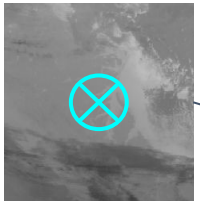
Using only previous GHI values in a predictive model could work well but would not be a scalable solution since we would need a station at every point we would like to survey.

We ask you to use only data that is available for the entirety of the map (i.e. datetime, latitude, longitude, elevation/altitude, etc.).

TL;DR this is NOT a timeseries problem. If you are in doubt about a data source, simply ask us.

Example inputs/outputs

This is a sketch of what your model inputs/outputs might look like:



Crop centered around
a SURFRAD station

Metadata = {
datetime,
latitude,
longitude,
elevation,
…
}

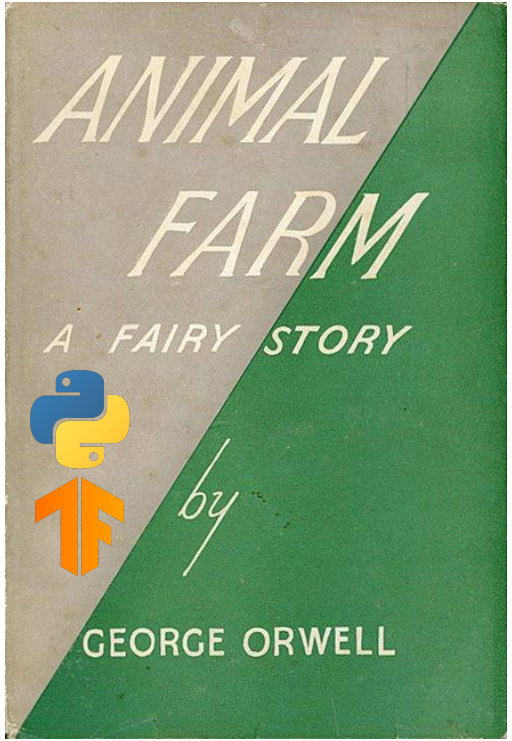
(optional)

GHI values:

T_0 = 433.2 W/m^2
T_0+ 1h = 443.5 W/m^2
T_0+ 3h = 263.1 W/m^2
T_0+ 6h = -3.2 W/m^2

Framework & Environment

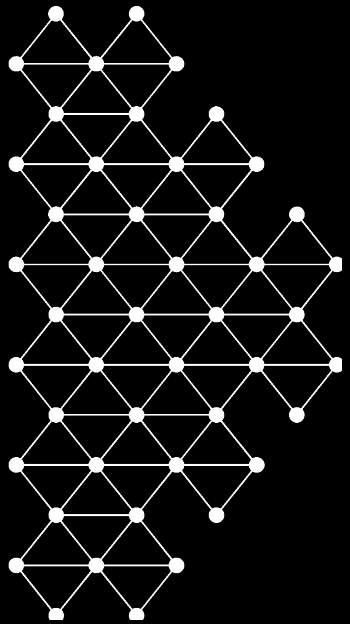
- You need to build your project using TensorFlow2.0
- **This is a mandatory requirement.**
- You will be using Helios with K80 GPUs.



“All modules are created equal, but some are more equal than others” - TensorflOrwell

Quebec
Artificial
Intelligence
Institute

Mila



Data Ingestion/Preprocessing

Data ingestion

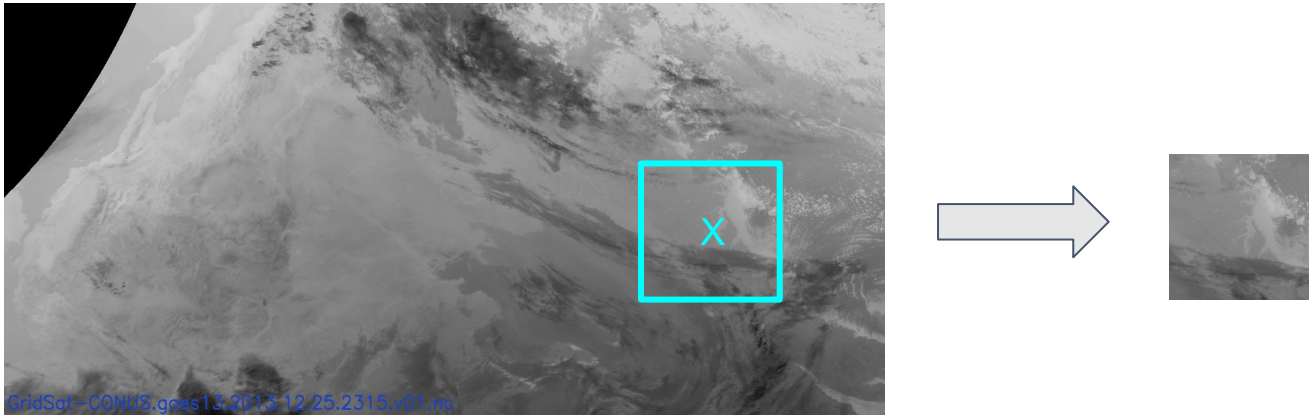
The data we provide is not actually “ready” to just hand into a model.

- The raw data is **hundreds** of GBs! Select what you need...
  - We focus on stations, so you ***might*** not need to use the entire continental US;
  - Even near a station, remember: one “pixel” = **multiple kilometers (16km^2)**;
  - In any case, what size of area would you like to see? (surrounding clouds)
- You need to handle the (uncommon) cases where data is missing (“skip” or “fill in”?);
- You need to decide what to do with **night time** data (keep or ignore?);
- You need to decide how to create **minibatches** (combine stations and/or timestamps?).

Preprocessing example: cropping

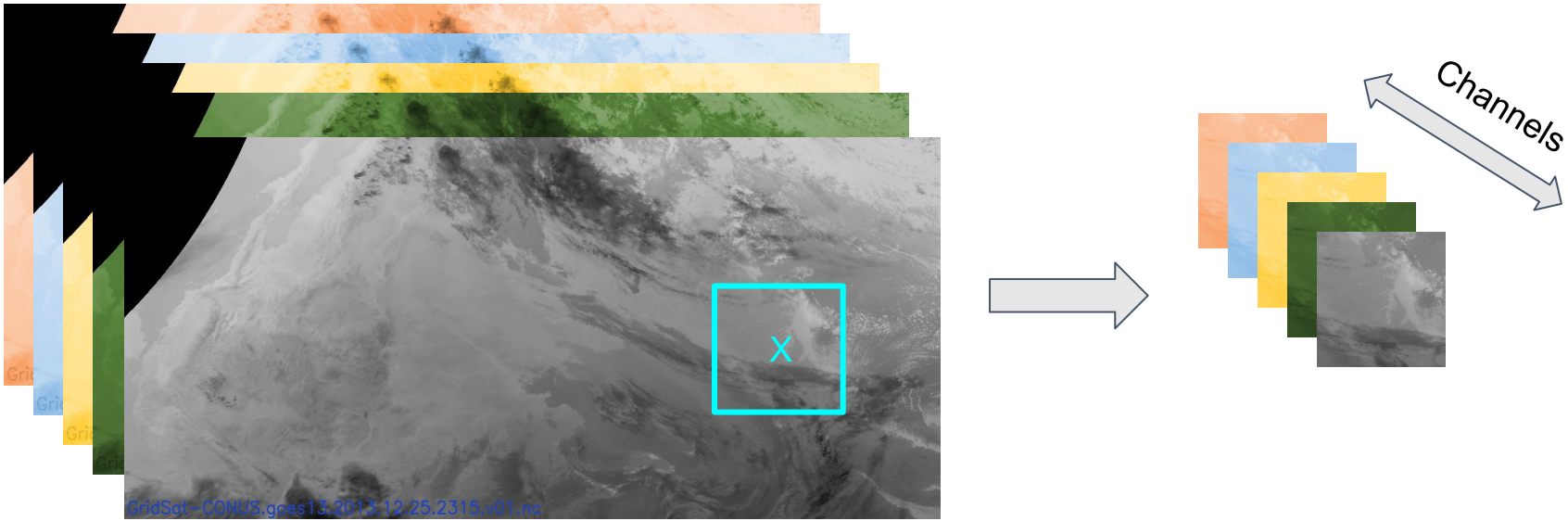
We are interested in GHI values at SURFRAD stations. You might decide to **crop** regions centered on the coordinates of those stations.

If you do so, you will have to **decide** what patch size to use. This is a **hyperparameter** among others. Larger patches capture more information but can also lead to slower models.



Preprocessing example: cropping

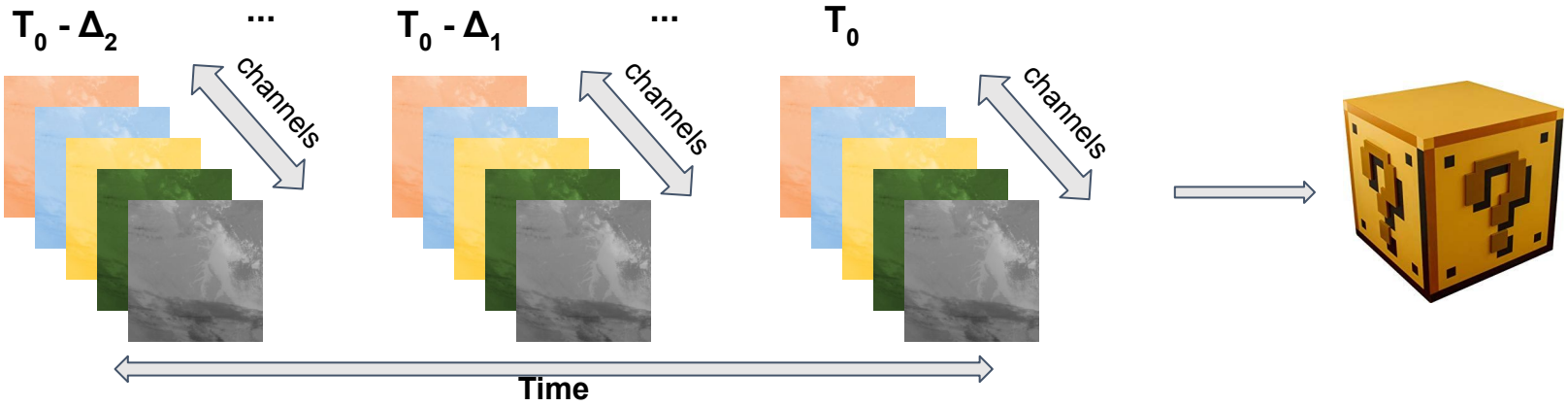
Remember that each GOES-13 image consists of 5 channels.
Each crop would thus consist of 5 channels as well.



Using temporal information

You may want to use past & present information to make predictions in the present/future. This means potentially using multiple frames to predict GHI values.

The model should capture the **multi-channel**, **multi-temporal** aspect of the problem.

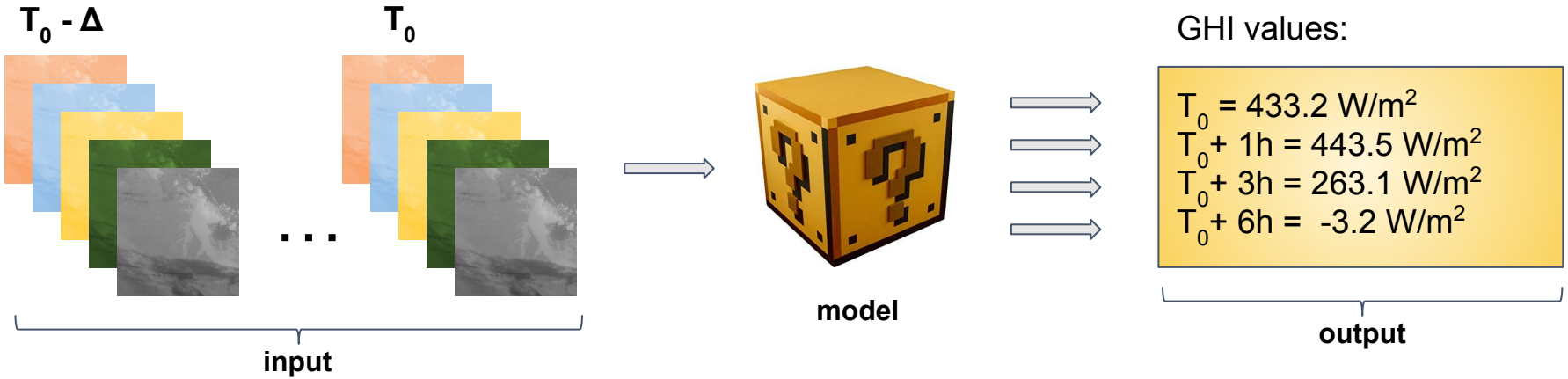


A sequence of images will allow you to track the dynamics of **atmospheric phenomena** (mainly: clouds).

Using temporal information

Remember: no matter what input you use, we always expect to receive 4 GHI values at:

$$T = T_0, \quad T = T_0 + 1h, \quad T = T_0 + 3h, \quad T = T_0 + 6h.$$

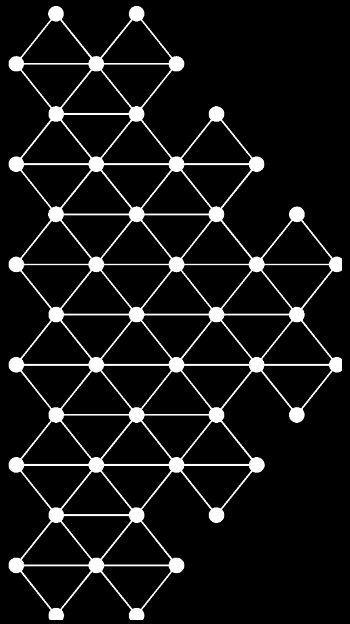


You are free to use alternative formulations for your inputs!

(for example, instead of using image sequences, you could use optical flow maps…)

Quebec
Artificial
Intelligence
Institute

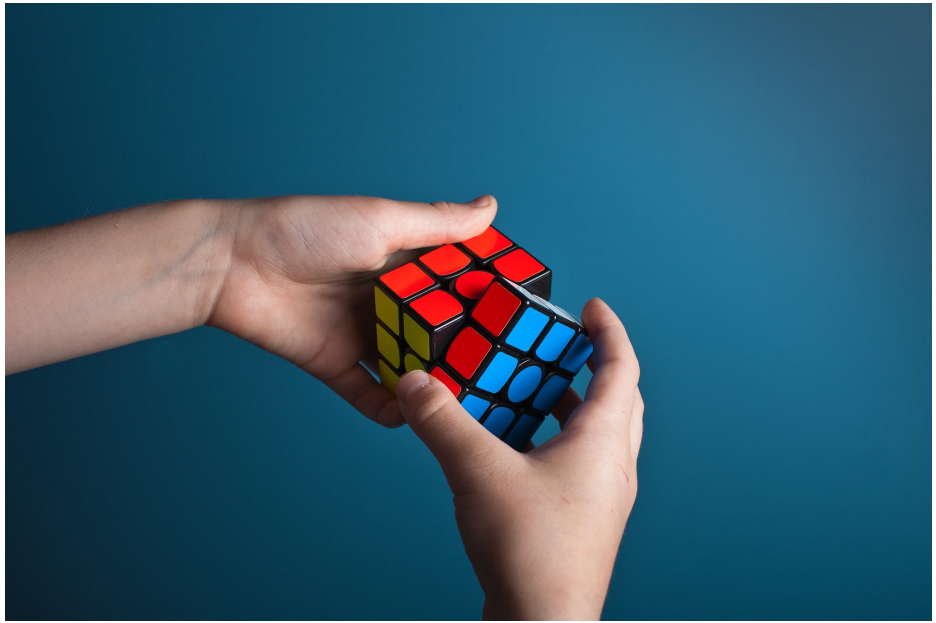
Mila



Models + Architectures

Models

You can be as creative as you'd like with your models. We will present a few that we think might work but there are many approaches that you can and should attempt.

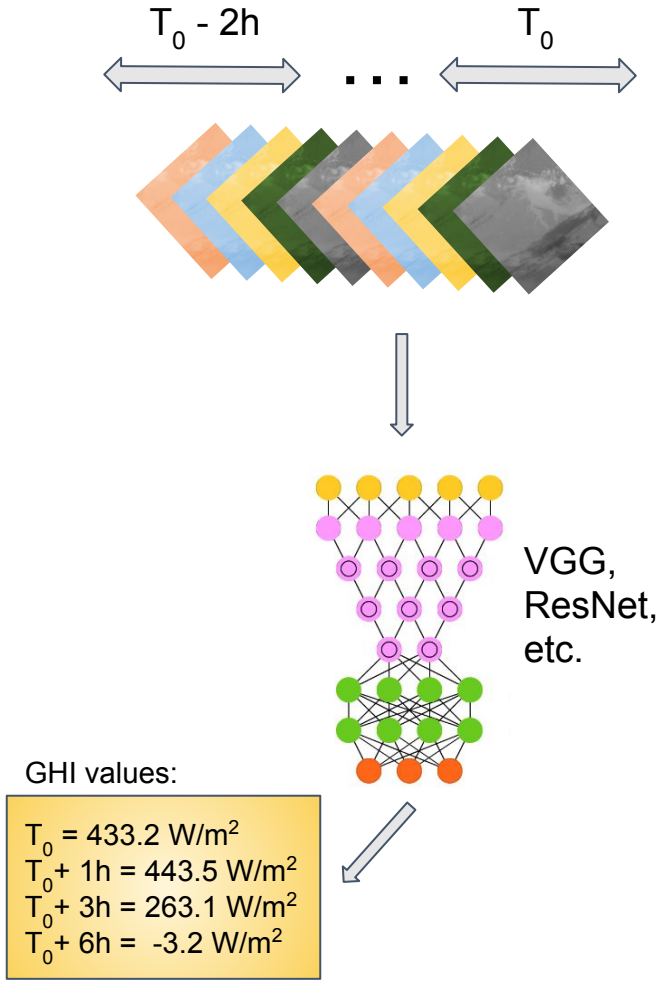


Conv2D

Simply **concatenate** past & present imagery to have an input of **HxWx(CxT)**, where **H** and **W** are the height and width of the input crops, **C** is the number of channels, and **T** is the length of the past "horizon" (or number of previous timesteps used).

The model outputs the (four) requested GHI values. While this might be a naive approach, it should be easier to train. It is a good idea to use it to debug your training loops.

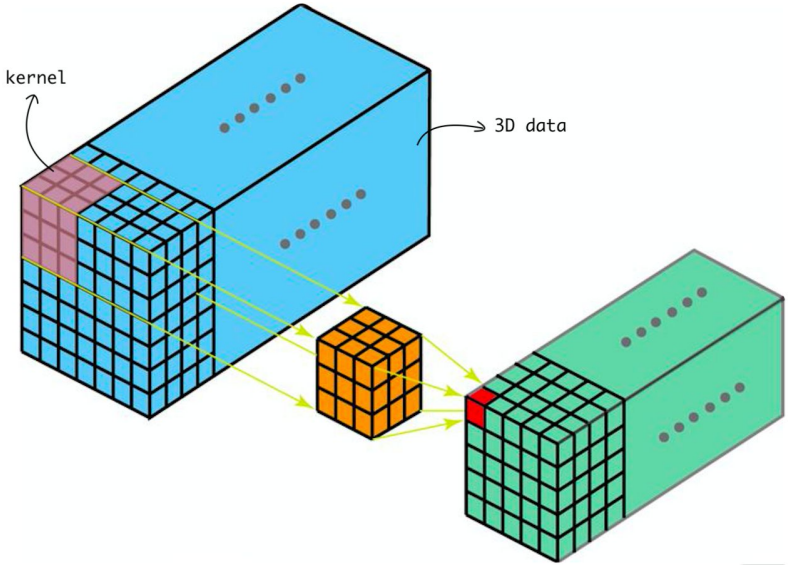
You could adapt almost any generic CNN classifier architecture for this approach (VGG, ResNet, etc.)



Conv3D

While the kernel in **Conv2D** operates across channels, the kernel in **Conv3D** operates across channels and across time. This kind of architecture is better suited for sequences of images, especially with regular intervals.

In this case, you do not need to reshape your tensors as in the Conv2D example, but instead treat time as a separate dimension.



Source

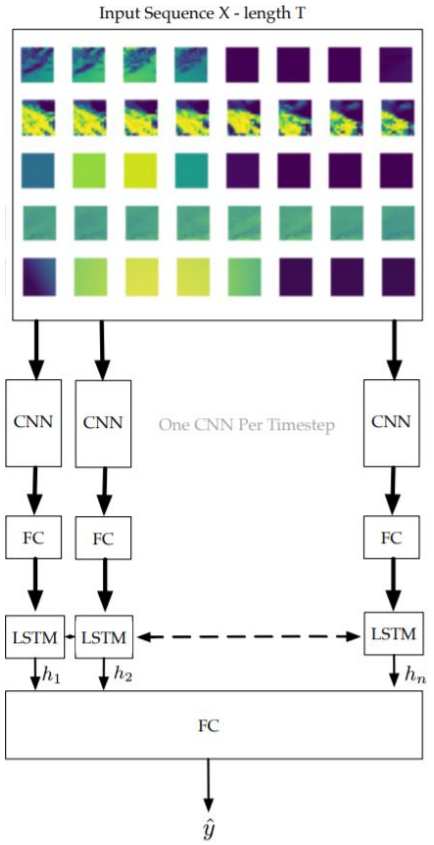
Helpful links:

https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610
https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv3D
https://arxiv.org/pdf/1412.0767.pdf
https://www.sciencedirect.com/science/article/abs/pii/S0038092X19301082

CNN + LSTM

You can implement a pipeline using a CNN on patches of satellite images and use the intermediate feature representations as input to an LSTM.

This combines both the spatial information extracted from the convolutional neural networks with the temporal aspect of the LSTM while sticking to common deep learning building blocks.



https://arxiv.org/abs/1902.01453

ConvLSTM

In this type of architecture, 2D convolutions are used on images which are then fed through a modified **LSTM** which accepts matrices as input (instead of vectors). The model was proposed to predict rainfall in the near future (sounds familiar?).

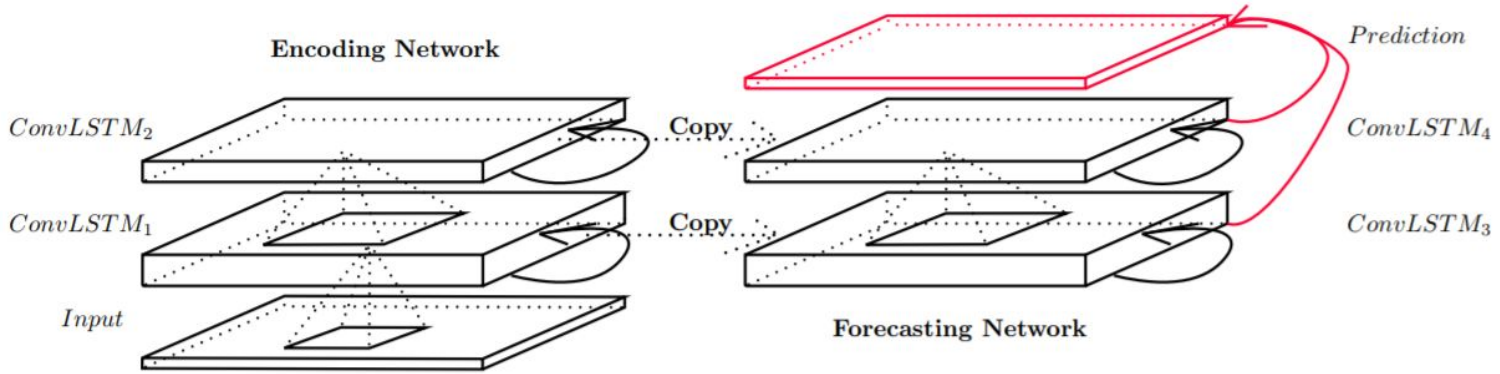


Figure 3: Encoding-forecasting ConvLSTM network for precipitation nowcasting

https://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowcasting.pdf

Autoencoder + LSTM

There are many ways this solution can be implemented. An advantage is that autoencoders can be trained in an **unsupervised** fashion on the entirety of the data available (i.e. even away from SURFRAD stations).

One approach would be to train an autoencoder on arbitrary patches of the full-sized images. Once the autoencoder is properly trained, you could freeze the encoder and use the latent representations to train an LSTM to predict the GHI sequences.

You can also experiment with training the autoencoder and LSTM jointly in many different flavors. Get creative :)

References:

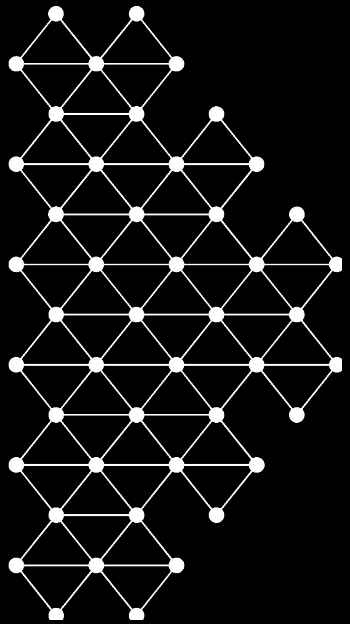
https://arxiv.org/pdf/1502.04681.pdf
https://machinelearningmastery.com/lstm-autoencoders/
https://ieeexplore.ieee.org/abstract/document/7844673

Additional Ideas

- **Think outside the box**! You don't have to focus on what we provide...
- You can supplement your models with **metadata**:
  - Date/time information (month, season, etc.)
  - SURFRAD station coordinates
  - Sunrise/Sunset
  - Different patch sizes for different time offsets
  - etc.

https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/

Quebec
Artificial
Intelligence
Institute



Evaluation

Evaluation

General notes:

- Every team will be evaluated on the same **test set**.
- The test set will be **withheld data (2016)** that you will not have access to.
- We will provide an **evaluation script** and instructions to submit your best models.
- You must adapt **two sections** of that script so that it can use your data loader/model.
- You must ensure that the evaluation script **works properly** before submitting.
- If we cannot run our script with your modifications, you **will be given zero** for the performance of your model.

...for more information on the evaluation, refer to the documentation on **GitHub**.

Evaluation

Important remarks on the final evaluation (1/2):

- We provide a list of "**datetimes**" that correspond to the **T_0's** to generate predictions for.
- You can assume that imagery for **T_0** will be available (**no guarantee beyond that**).
- You can use all the imagery **up to T_0** to infer current/future GHI values:
  - **OK:** [$IMG(T_0\text{-}1h)$, $IMG(T_0\text{-}30min)$, $IMG(T_0)$] to predict [$GHI(T_0)$, $GHI(T_0+1h)$, …]
  - **NOT OK:** [$IMG(T_0)$, $IMG(T_0+1h)$, $IMG(T_0+2h)$] to predict [$GHI(T_0)$, $GHI(T_0+1h)$, …]
- Code will be tested to ensure that no such "**future imagery**" is used for predictions
  - finding cheaters will be much easier than you think… **so don't do it**.

Evaluation

Important remarks on the final evaluation (2/2):

- During the evaluation, we will only be focusing on datetimes **during the day**, since the GHI at night is always constant. This will avoid skews in scores due to overfitting.
    - Specifically: we will only evaluate **T** in **[T_0, T_0+ 1h, T_0+ 3h, T_0+ 6h]** when **T** falls between that day's sunrise and sunset.
- Your model should **never** predict NaNs, but the groundtruth might sometimes be NaNs
    - the SURFRAD stations are sometimes unavailable, and that produces "NaN" measurements
    - these will be ignored on our side when evaluating your performance
- To create the final model ranking, all predictions for all stations and all GHI horizons will be aggregated without any specific weighting.

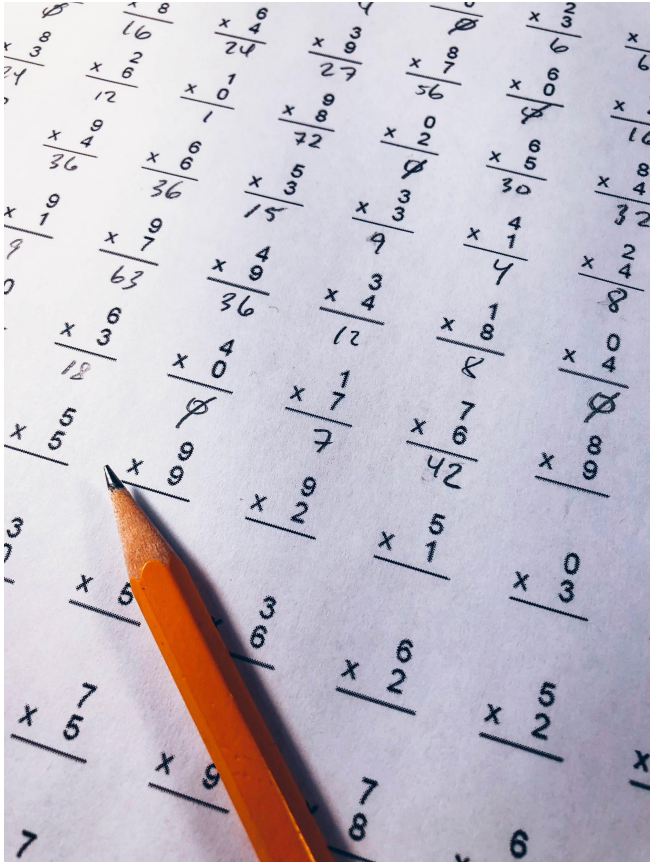
Model Ranking

Your models will be **compared** against each other and against our baselines. We will be analyzing the Root Mean Square Error (**RMSE**) between the measured GHI at the stations and the model predictions.

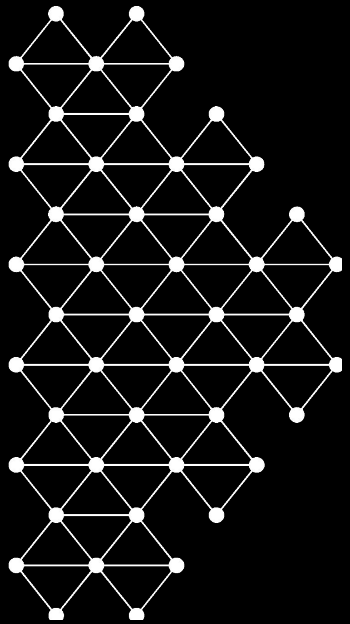
Grading scheme:

Statistically **best** model: 10 points
Statistically **better** than baseline model: 8 points
Statistically **equivalent** to baseline model: 6 points
Statistically **worse** than baseline model: 0 points

Refer to the evaluation grid for more details



Quebec
Artificial
Intelligence
Institute



Technical Details

Helios

You will have access to the Helios cluster for the project for computing resources.

Helios and chariot depicted on the dome of the entrance hall of the Széchenyi Bath, Budapest.

Personal Drives: Your personal drives, i.e. `$HOME` and `$SCRATCH` have a maximum capacity of 500 GB.

Additional information

Helios

You will have access to the Helios cluster for the project for computing resources.

Team Drives: Your team will have access to shared drives on which you will have "unlimited" storage space. Here you can share models, code, data, etc. within your team. We ask you to be mindful with how much space you actually use. All shared drives will be in:

```
/project/cq-training-1/project1/teams/
/project/cq-training-1/project2/teams/
```

We created a shared directory for each team at that location called `teamXX`, where `XX` is the team number you are assigned to for the project. `XX`=00 is reserved for the TAs as an example. You will be able to create any folder structure you like inside the "`teamXX`" directory.

Additional information

Helios

Submission Drives: Your team will have access to a submission drive for your final code submission. Your code repository should be a clone of your master branch on github. It should contain all the code that is relevant for the evaluation.

```
/project/cq-training-1/project1/submissions/
/project/cq-training-1/project2/submissions/
```

There should be one and only one submission per team.

Additional information

Helios

You will have access to the Helios cluster for the project for computing resources.

Shared Data: We will be providing you with data that we have prepared specifically for this project. This includes metadata that has been aggregated from various sources and formatted in a specific way for this course. All the data that we will provide will be in a shared read-only folder accessible to all:

```
/project/cq-training-1/project1/data/
/project/cq-training-1/project2/data/
```

You are free to copy this data to your shared drives or use it directly. Be mindful that the original datasets can be very large in size.

Additional information

GOES-13 Data

The entirety of the raw data associated to GOES-13 is available on Helios at:

```
/project/cq-training-1/project1/data/
```

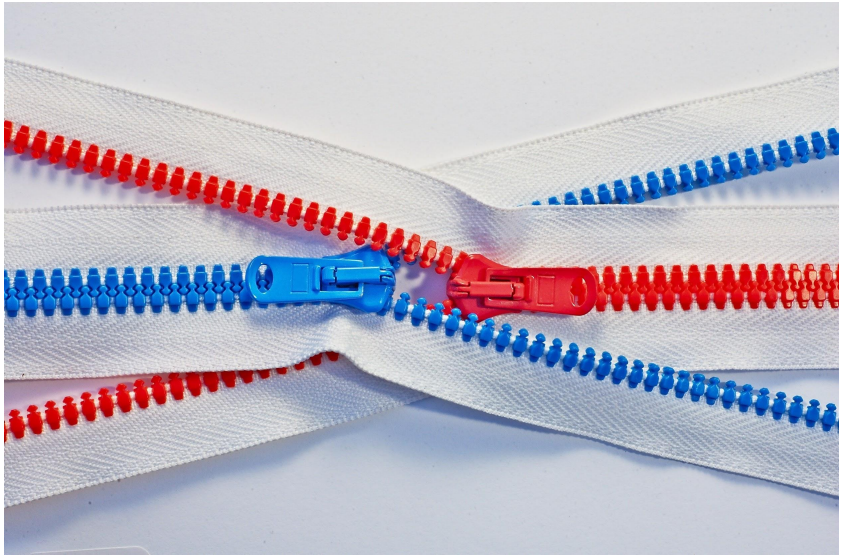
It consists of netcdf files (.nc) for each acquisition. Each .nc file contains metadata concerning the GOES-13 acquisition process, the geolocation of each pixel in the imagery channels, and the raw channel data itself. The names of the files are structured as follows:

```
GridSat-CONUS.goes13.YYYY.MM.DD.HHMM.v01.nc
```

GOES-13 Data

The entirety of the original data is approximately 960 Gb. You are free to use it, but be mindful of the size and your space limitations working with the cluster.

We will also make available to you a compressed version of the dataset that will be about 400Gb in size, as well as a highly compressed version using 8-bit jpeg compression at about 150Gb.

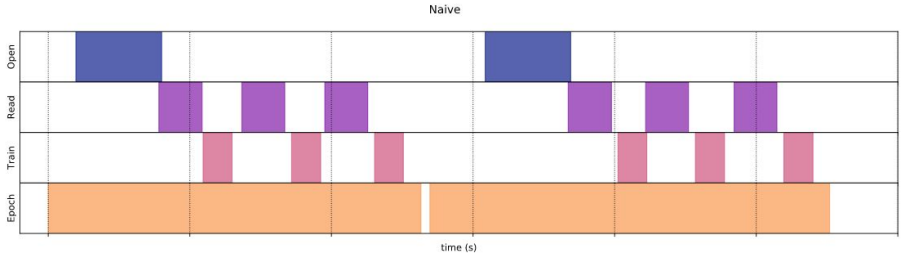


...for more information on the datasets, refer to the documentation on **GitHub**.

Data Loading

You will have too much data to simply load it all in RAM. One of the bottlenecks you will face is efficiently implementing data loaders. If your CPU cannot load data fast enough for your GPU to process, you will have models that will take too long to train. This is especially critical in this task as we have hundreds of thousands of potentially unique minibatches per training epoch.

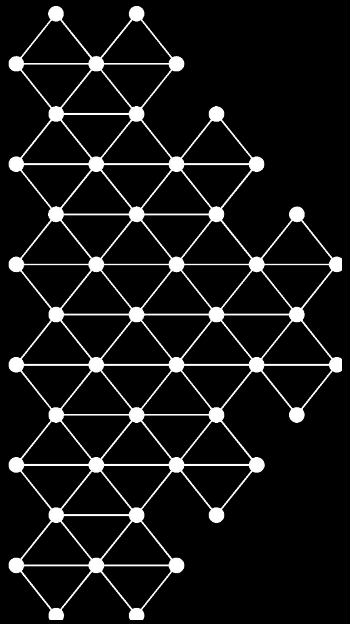
We recommend you start thinking early on about efficient ways to load your data.



https://www.tensorflow.org/guide/data_performance

Quebec
Artificial
Intelligence
Institute

Mila



Project Management

Managing Projects

Colab and **Jupyter Notebooks** are great tools for quick iteration, data visualization and sharing ideas. **They are not ideal** for scaling and deploying **large projects** and can get cluttered very quickly. Here are some tips for good python code and projects:

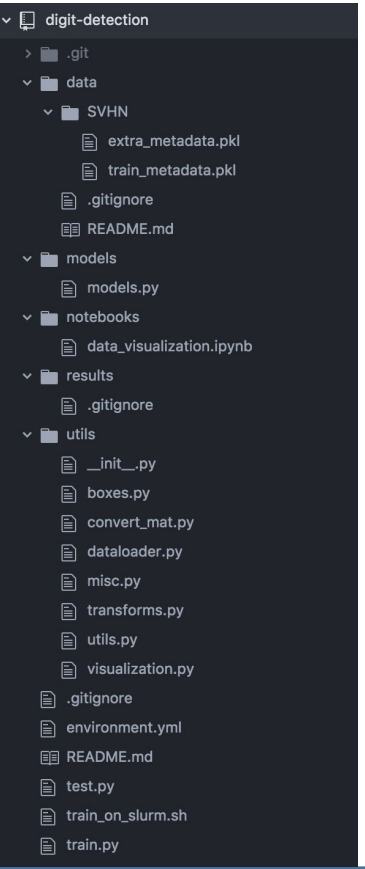
- Maintain an organized codebase
- Use Version Control for your code (Git)
- Use Virtual Environments
- Use Unit Tests

Managing Projects

Colab and **Jupyter Notebooks** are great tools for quick iteration, data visualization and sharing ideas. **They are not ideal** for scaling and deploying **large projects** and can get cluttered very quickly. Here are some tips for good python code and projects:

- **Maintain an organized codebase**
- Use Version Control for your code (Git)
- Use Virtual Environments
- Use Unit Tests

Organized Codebase



- Organize your code in a logical hierarchical structure
- Use logical names for variables, filenames, folders etc.
- Avoid code duplication
  - Have the same training/data loading routines regardless of experiments
  - Use object oriented programming paradigms
    - https://realpython.com/python3-object-oriented-programming/
- Follow PEP guidelines (lint your code, think “pythonic”, etc.)
  - https://www.python.org/dev/peps/pep-0008/#a-foolish-consistency-is-the-hobgoblin-of-little-minds
  - http://flake8.pycqa.org/en/latest/#
- Document your code https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_numpy.html

Managing Projects

Colab and **Jupyter Notebooks** are great tools for quick iteration, data visualization and sharing ideas. **They are not ideal** for scaling and deploying **large projects** and can get cluttered very quickly. Here are some tips for good python code and projects:

- Maintain an organized codebase
- **Use Version Control for your code (Git)**
- Use Virtual Environments
- Use Unit Tests

Version Control Systems

Version Control Systems (VCS) allow users to keep **track** of all **changes** that ever happened in a project. It is great for working collaboratively on large codebases. A very popular (and open source) VCS is **git**.

Managing Projects

Colab and **Jupyter Notebooks** are great tools for quick iteration, data visualization and sharing ideas. **They are not ideal** for scaling and deploying **large projects** and can get cluttered very quickly. Here are some tips for good python code and projects:

- Maintain an organized codebase
- Use Version Control for your code (Git)
- **Use Virtual Environments**
- Use Unit Tests

Virtual Environments

Since python relies on external libraries, versions matter. Libraries might get updated with **breaking changes** which can be dangerous for legacy code. The solution to this is **isolating** each project with their own virtual environment. We recommend using **virtualenv** and **pip** on Helios.

Here is how you would install tensorflow2.0 using python3.7:

```
module load python/3.7
virtualenv --no-download ~/ift6759-env
source ~/ift6759-env/bin/activate
pip install --no-index --upgrade pip
pip install --no-index tensorflow_gpu
pip install -r /path/to/your/project/requirements.txt
```

https://realpython.com/effective-python-environment/
https://docs.computecanada.ca/wiki/Python#Creating_and_using_a_virtual_environment
https://virtualenv.pypa.io/en/latest/

Managing Projects

Colab and **Jupyter Notebooks** are great tools for quick iteration, data visualization and sharing ideas. **They are not ideal** for scaling and deploying **large projects** and can get cluttered very quickly. Here are some tips for good python code and projects:

- Maintain an organized codebase
- Use Version Control for your code (Git)
- Use Virtual Environments
- **Use Unit Tests**

Unit tests

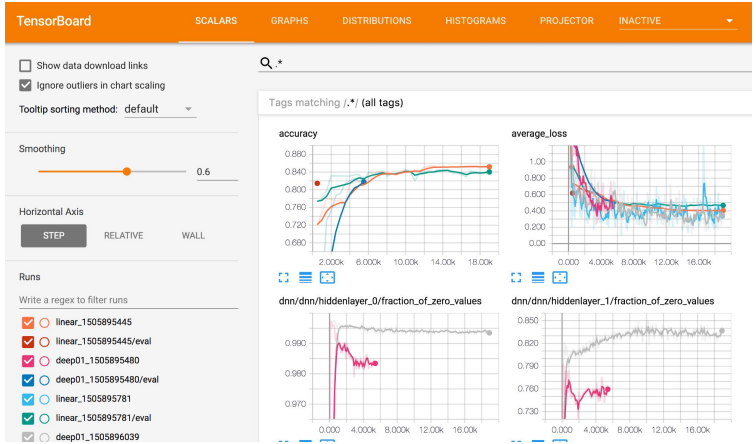
It is good practice to write unit tests that you expect your code to pass every time you run it. This is a great way to catch bugs early and ensure that code works as expected in an explicit way.

```
1  def test_tensorflow():
2      '''Basic test to make sure tensorflow is properly installed'''
3
4      import tensorflow as tf
5
6      graph = tf.constant(4) + tf.constant(3)
7
8      with tf.Session() as sess:
9          out = sess.run(graph)
10
11     assert out == 7
```

https://docs.pytest.org/en/latest/
https://docs.pytest.org/en/latest/getting-started.html

Experiment Management

Deep learning is inherently **empirical**. When solving a task, you will want to try many **variations** of a network, i.e. use different hyperparameters, models, processing, etc. You will therefore need tools to **organize** and **monitor** your experiments. Two popular experiment management libraries that are compatible with Helios are **mlflow** and **tensorboard**.

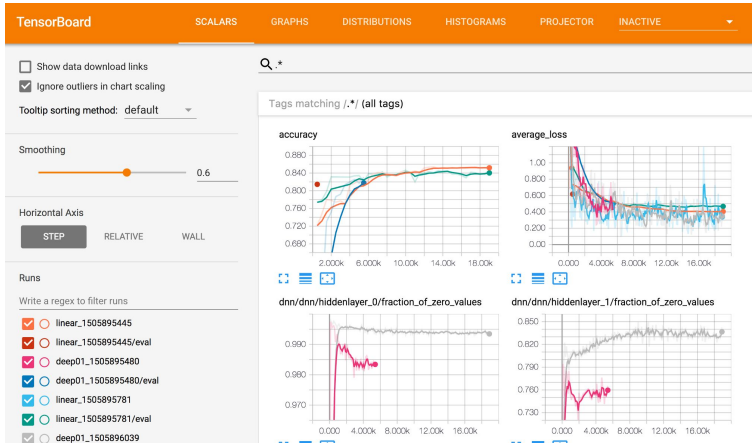


					Parameters		Metrics		
	Date	User	Source	Version	alpha	l1_ratio	mae	r2	rmse
☐	2018-06-04 23:00:10	mlflow	train.py	05e956	1	1	0.649	0.04	0.862
☐	2018-06-04 23:00:10	mlflow	train.py	05e956	1	0.5	0.648	0.046	0.859
☐	2018-06-04 23:00:10	mlflow	train.py	05e956	1	0.2	0.628	0.125	0.823
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	1	0	0.619	0.176	0.799
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	1	0.648	0.046	0.859
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0.5	0.628	0.127	0.822
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0.2	0.621	0.171	0.801
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0	0.615	0.199	0.787
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0	1	0.578	0.288	0.742
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0	0.5	0.578	0.288	0.742
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0	0.2	0.578	0.288	0.742
☐	2018-06-04 23:00:08	mlflow	train.py	05e956	0	0	0.578	0.288	0.742

https://www.tensorflow.org/guide/summaries_and_tensorboard
https://mlflow.org/docs/latest/tutorial.html

Experiment Management (on Helios)

To **visualize** results from both tensorflow or mlflow from Helios, you will need to copy the saved results to your **local** computer. You can do this using e.g. rsync and running the appropriate servers locally on your machine.

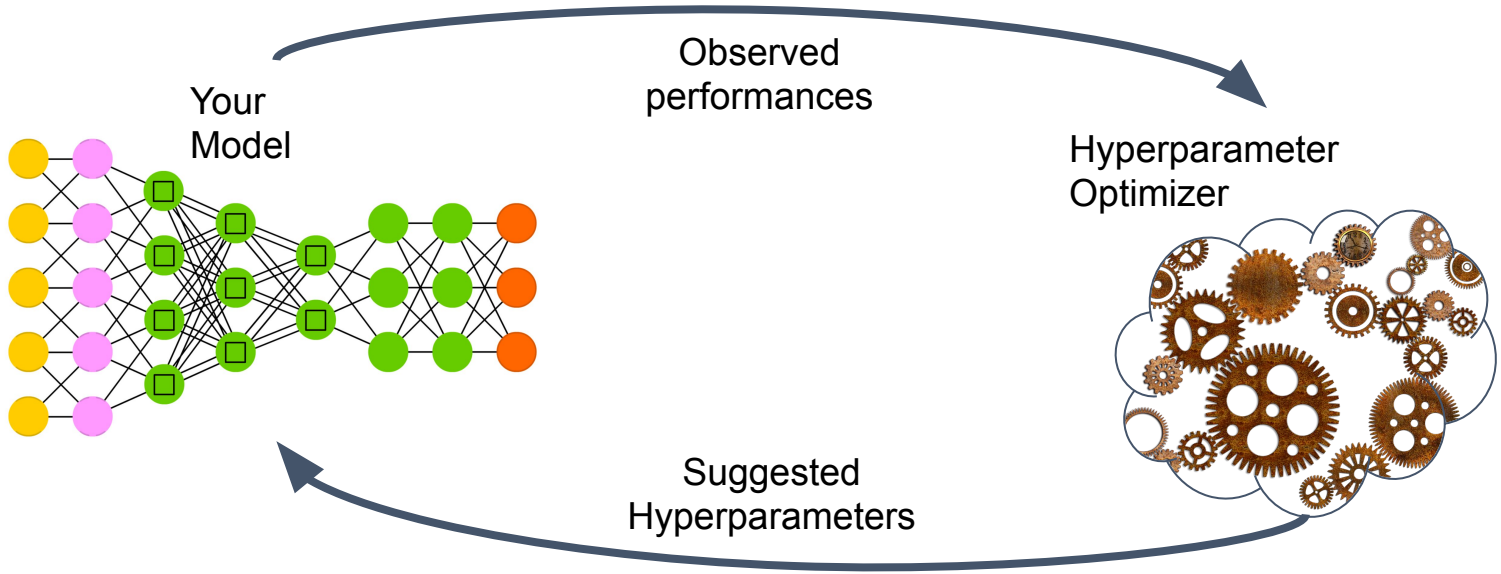


	Date	User	Source	Version	Parameters		Metrics		
					alpha	l1_ratio	mae	r2	rmse
☐	2018-06-04 23:00:10	mlflow	train.py	05e956	1	1	0.649	0.04	0.862
☐	2018-06-04 23:00:10	mlflow	train.py	05e956	1	0.5	0.648	0.046	0.859
☐	2018-06-04 23:00:10	mlflow	train.py	05e956	1	0.2	0.628	0.125	0.823
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	1	0	0.619	0.176	0.799
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	1	0.648	0.046	0.859
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0.5	0.628	0.127	0.822
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0.2	0.621	0.171	0.801
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0	0.615	0.199	0.787
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0	1	0.578	0.288	0.742
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0	0.5	0.578	0.288	0.742
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0	0.2	0.578	0.288	0.742
☐	2018-06-04 23:00:08	mlflow	train.py	05e956	0	0	0.578	0.288	0.742

https://www.tensorflow.org/guide/summaries_and_tensorboard
https://mlflow.org/docs/latest/tutorial.html

Hyperparameter Optimization

Manually tweaking parameters can be exhausting and time consuming. Many libraries also offer the possibility to suggest automatically how to update parameters based on empirical results.



http://www.asimovinstitute.org/neural-network-zoo/

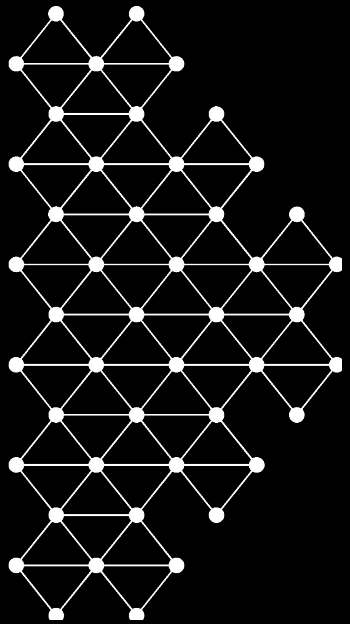
Tips

- Don’t reinvent the wheel!
- Use Python **datetime** structures for handling dates, but be aware of timezones!
- Don’t forget to **standardize** your data.
- Regression problem (MSE/RMSE) - these values won’t mean much other than lower is better. Try to have good visuals to help you debug your models along the way.
- Don’t be afraid to have different people on different tracks for model exploration!
- **Reproducibility** - ensure that you can reproduce results if necessary.
- Remember the zen of python:

```
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
```

Quebec
Artificial
Intelligence
Institute

Mila



Questions?