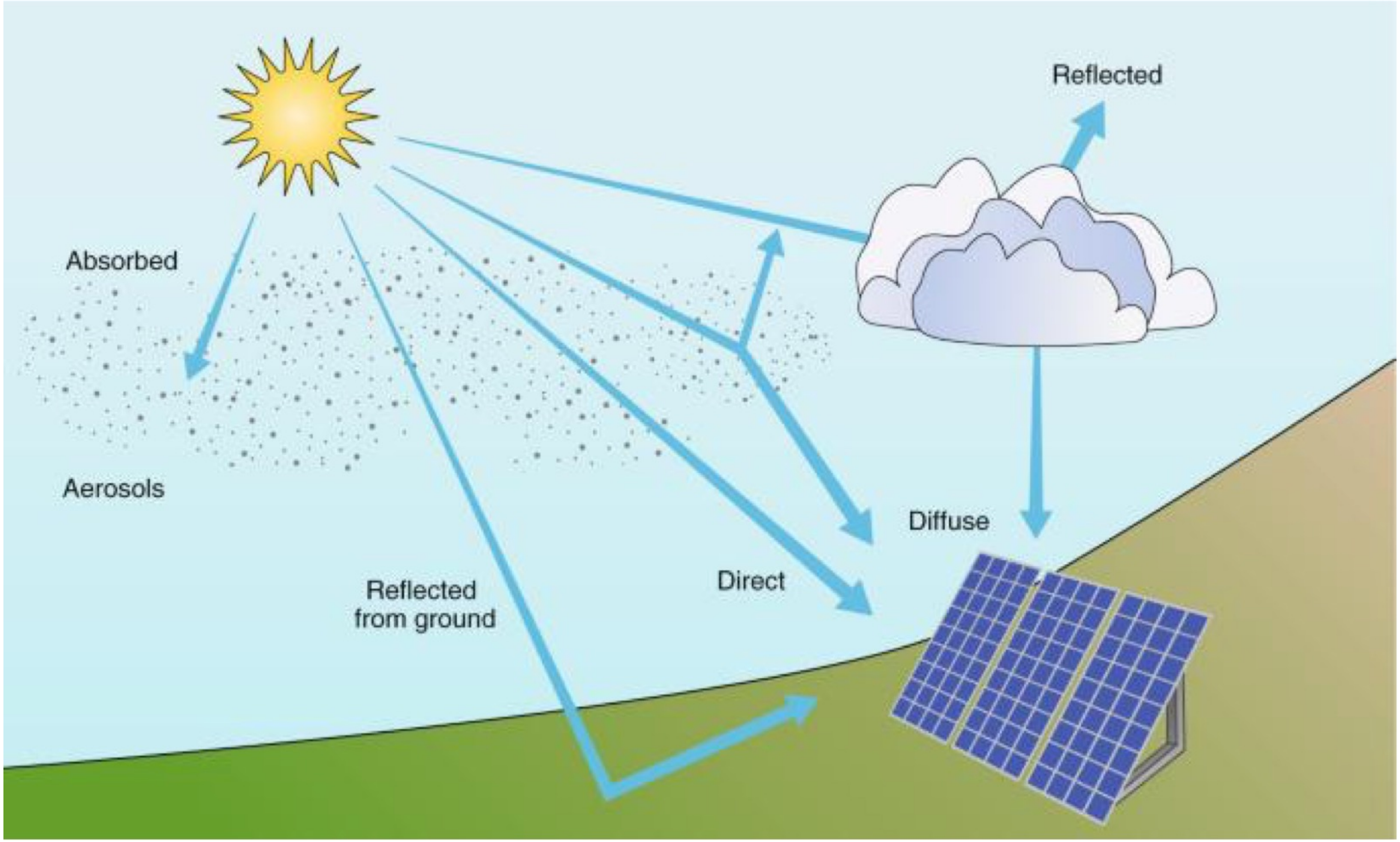
Intraday GHI Prediction Using Webcams

Roy Sarkis
Roy.sarkis@epfl.ch

May 10, 2023

Global Horizontal Irradiance (GHI)

- GHI is the total solar radiation incident on a horizontal surface



Importance of GHI Prediction

- Photo-voltaic (PV) generation is experiencing significant growth thanks to the decreasing costs of installations and reduced carbon footprint.

Importance of GHI Prediction

- Photo-voltaic (PV) generation is experiencing significant growth thanks to the decreasing costs of installations and reduced carbon footprint.
- However, the progressive displacement of conventional power generation in favor of renewables like PV, requires having an adequate capacity of regulating power to ensure reliable operation and grid stability.

Importance of GHI Prediction

- Photo-voltaic (PV) generation is experiencing significant growth thanks to the decreasing costs of installations and reduced carbon footprint.
- However, the progressive displacement of conventional power generation in favor of renewables like PV, requires having an adequate capacity of regulating power to ensure reliable operation and grid stability.
- This relies on the availability of high-performance solar irradiance forecasting tools at day-ahead and intraday time horizons.

Importance of GHI Prediction

- Photo-voltaic (PV) generation is experiencing significant growth thanks to the decreasing costs of installations and reduced carbon footprint.
- However, the progressive displacement of conventional power generation in favor of renewables like PV, requires having an adequate capacity of regulating power to ensure reliable operation and grid stability.
- This relies on the availability of high-performance solar irradiance forecasting tools at day-ahead and intraday time horizons.

Table 1 - Summary of use cases for PV power forecasting (Alet et al., 2016)

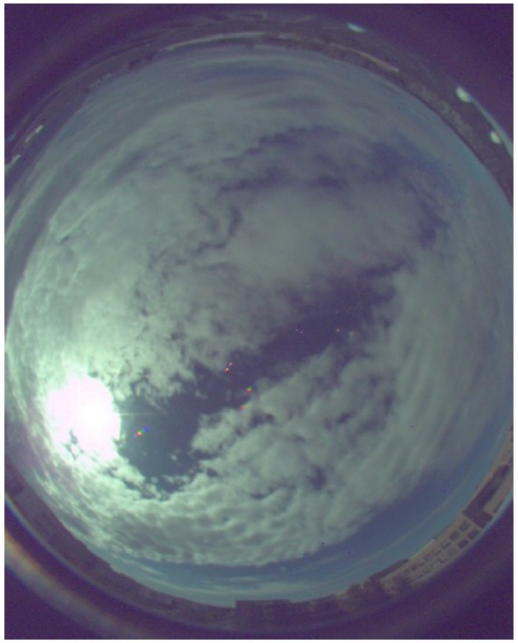
Time horizon	Single site (10 m – 100 m) **PV plant owners** **PV plant operators**	MV distribution grid (1 km – 10 km) **DSOs** **Microgrid operators**	Transmission system (100 km – 1000 km) **TSOs** **Market operators**
15 min	Management of storage system	Management of active/reactive power	Activation of reserves
1 h	Management of storage system Intra-day trades	Storage and load management	Intra-day trades
24 h	Management of storage system Compliance with regulations Day-ahead trades	Storage and load planning	Booking of reserves Transmission scheduling Day-ahead trades
1 year	O&M contract	Planning of maintenance operations	Long-term trades
20+ years	Investment case	Infrastructure planning	Infrastructure planning

GHI Prediction

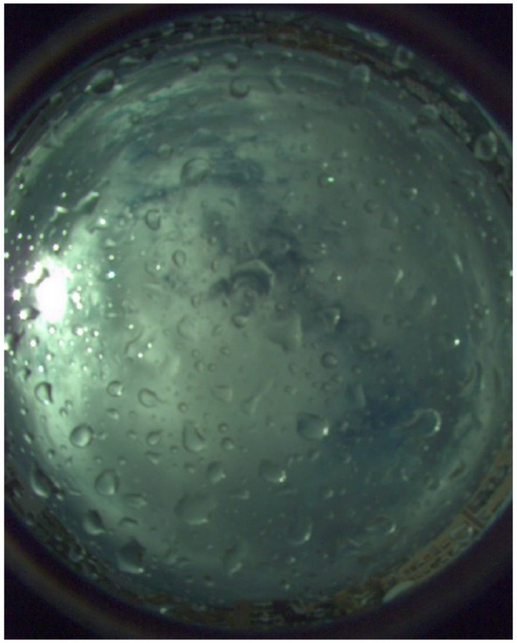
- GHI could be predicted via:
  - Time series of historical GHI (could also include meteorological data)

- GHI could be predicted via:
  - Time series of historical GHI (could also include meteorological data)
  - Feature extraction from all-sky images images with neural networks

(a) Sunny



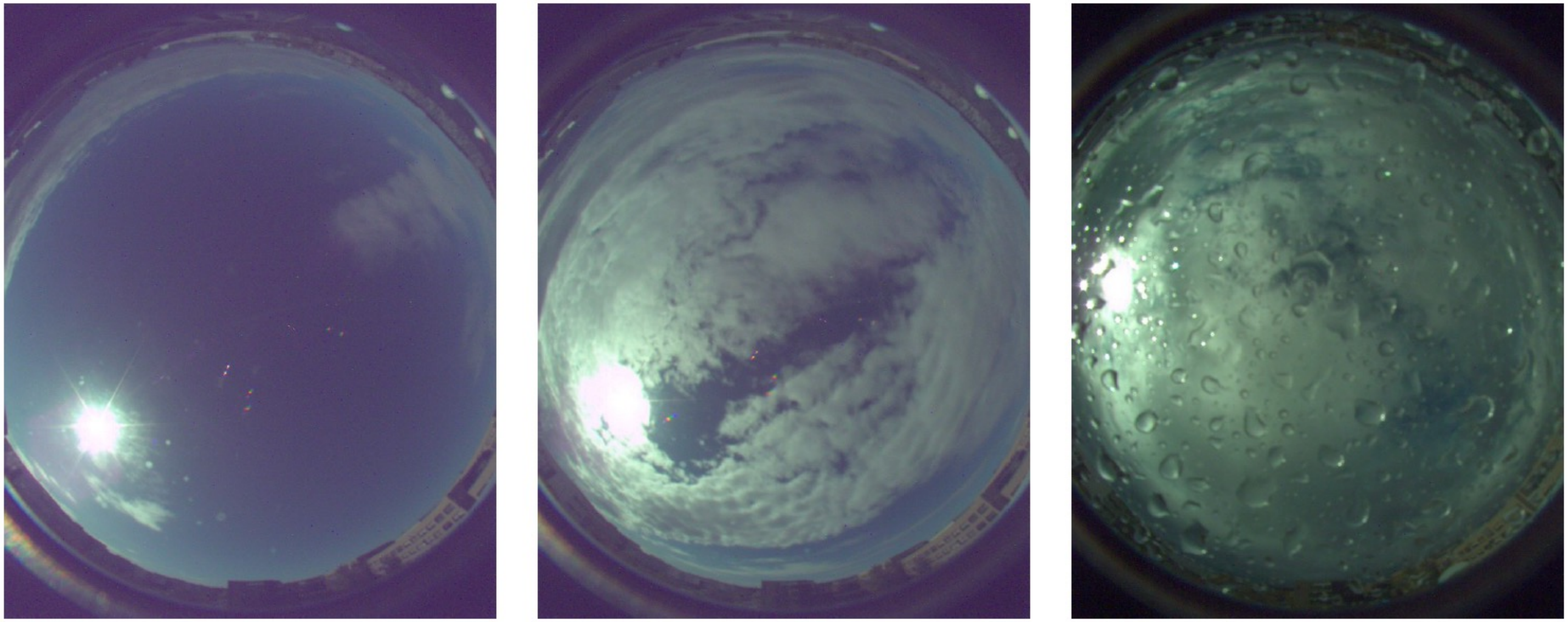
(b) Cloudy



(c) Rainy

GHI Prediction

- GHI could be predicted via:
  - Time series of historical GHI (could also include meteorological data)
  - Feature extraction from all-sky images images with neural networks

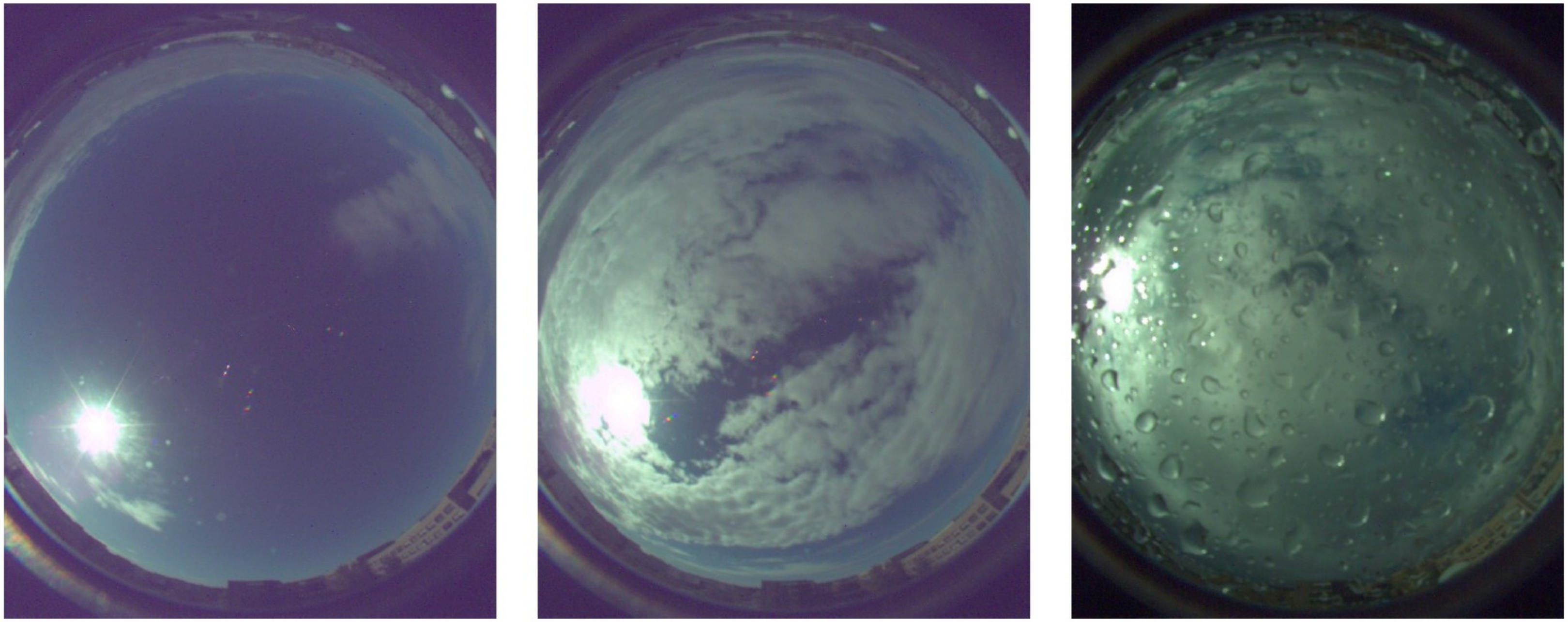


(a) Sunny (b) Cloudy (c) Rainy

- However, time series models are not suitable for GHI predictions in the 1 – 6 hour range.

GHI Prediction

- GHI could be predicted via:
  - Time series of historical GHI (could also include meteorological data)
  - Feature extraction from all-sky images images with neural networks



(a) Sunny (b) Cloudy (c) Rainy

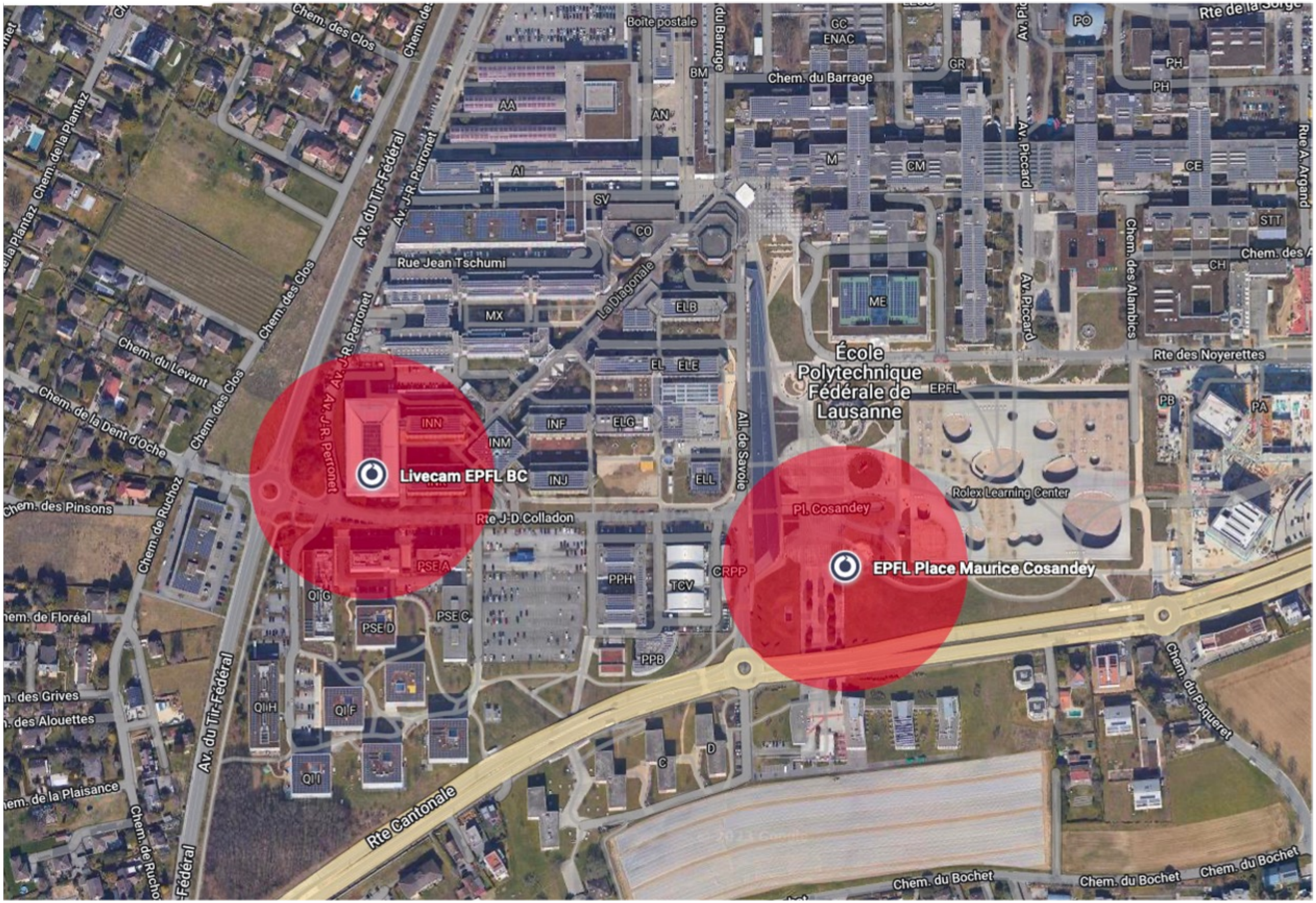
- However, time series models are not suitable for GHI predictions in the 1 – 6 hour range.
- Problem with all-sky cameras?

Our Approach

- Use images from public webcams

Data Description

1. **X1.npy** and **X2.npy**: two sets of 10000 RGB images (250x250x3) collected from two different cameras located on campus. The images are 360°.



Data Description

1. **X1.npy** and **X2.npy**: two sets of 10000 RGB images (250x250x3) collected from two different cameras located on campus. The images are 360°.

2. **common_time.npy**: time of the collected images. So the first element of the array common_time corresponds to the first images in X1 and in X2 were collected and so on.

```
array([datetime.datetime(2021, 12, 6, 8, 50),
       datetime.datetime(2021, 12, 6, 9, 10),
       datetime.datetime(2021, 12, 6, 9, 20), ...,
       datetime.datetime(2023, 1, 4, 12, 0),
       datetime.datetime(2023, 1, 4, 12, 10),
       datetime.datetime(2023, 1, 4, 12, 20)], dtype=object)
```

Data Description

1. **X1.npy** and **X2.npy**: two sets of 10000 RGB images (250x250x3) collected from two different cameras located on campus. The images are 360°.
2. **common_time.npy**: time of the collected images. So the first element of the array common_time corresponds to the first images in X1 and in X2 were collected and so on.
3. **ground_truth.npy**: measured GHI value at time of image collection. Example: the first element of this array is the GHI value collected at the first element in the array 'common_time'.

	Time	Ground Truth
0	2021-12-06 08:50:00	21.0
1	2021-12-06 09:10:00	34.0
2	2021-12-06 09:20:00	52.0
3	2021-12-06 09:30:00	70.0
4	2021-12-06 09:50:00	82.0
...	...	...
9995	2023-01-04 11:40:00	191.0
9996	2023-01-04 11:50:00	171.0
9997	2023-01-04 12:00:00	166.0
9998	2023-01-04 12:10:00	165.0
9999	2023-01-04 12:20:00	172.0

10000 rows × 2 columns

Data Description

1. **X1.npy** and **X2.npy**: two sets of 10000 RGB images (250x250x3) collected from two different cameras located on campus. The images are 360°.

2. **common_time.npy**: time of the collected images. So the first element of the array common_time corresponds to the first images in X1 and in X2 were collected and so on.

3. **ground_truth.npy**: measured GHI value at time of image collection. Example: the first element of this array is the GHI value collected at the first element in the array 'common_time'.

4. **labels.npy**: GHI value 2 hours in advance from the time of image collection. In general, it will be a forward version (+2 hours) of the ground_truth. It won't always be the case as we stop collecting images at night so you don't have the ground_truth values at night and hence you won't be able to get the GHI value 2 hours in advance. That is why it is provided to you.

Data Description - Labels

	Time	Ground Truth	labels
0	2021-12-06 08:50:00	21.0	132.0
1	2021-12-06 09:10:00	34.0	126.0
2	2021-12-06 09:20:00	52.0	136.0
3	2021-12-06 09:30:00	70.0	143.0
4	2021-12-06 09:50:00	82.0	134.0
5	2021-12-06 10:30:00	121.0	116.0
6	2021-12-06 10:40:00	129.0	128.0
7	2021-12-06 10:50:00	132.0	208.0
8	2021-12-06 11:00:00	136.0	305.0
9	2021-12-06 11:40:00	143.0	137.0
10	2021-12-06 11:50:00	134.0	293.0
11	2021-12-06 12:00:00	137.0	142.0
12	2021-12-06 12:10:00	126.0	381.0
13	2021-12-06 12:20:00	115.0	305.0
14	2021-12-06 12:30:00	116.0	230.0
15	2021-12-06 12:40:00	128.0	238.0
16	2021-12-06 13:10:00	337.0	86.0
17	2021-12-06 13:30:00	119.0	94.0
18	2021-12-06 13:40:00	137.0	63.0
19	2021-12-06 13:50:00	293.0	51.0

Data Description

1. **X1.npy** and **X2.npy**: two sets of 10000 RGB images (250x250x3) collected from two different cameras located on campus. The images are 360°.

2. **common_time.npy**: time of the collected images. So the first element of the array common_time corresponds to the first images in X1 and in X2 were collected and so on.

3. **ground_truth.npy**: measured GHI value at time of image collection. Example: the first element of this array is the GHI value collected at the first element in the array 'common_time'.

4. **labels.npy**: GHI value 2 hours in advance from the time of image collection. In general, it will be a forward version (+2 hours) of the ground_truth. It won't always be the case as we stop collecting images at night so you don't have the ground_truth values at night and hence you won't be able to get the GHI value 2 hours in advance. That is why it is provided to you.

5. **meteo_data.csv**
   a. **Time**: Matches time of images provided
   b. **Air_temp**: Air temperature (°C) 2 meters above ground
   c. **Wind_speed**: Wind speed (m/s), ten minutes mean
   d. **Wind_dir**: wind direction (°), ten minutes mean

- **Example: Observation interval for 10-minute values:**
  At "06.12.21 08:50", Wind_speed = 1 m/s means that the wind speed from 08:40 to 08:50 had a mean speed of 1 m/s. Similarly for wind direction.

Data Description - Meteo

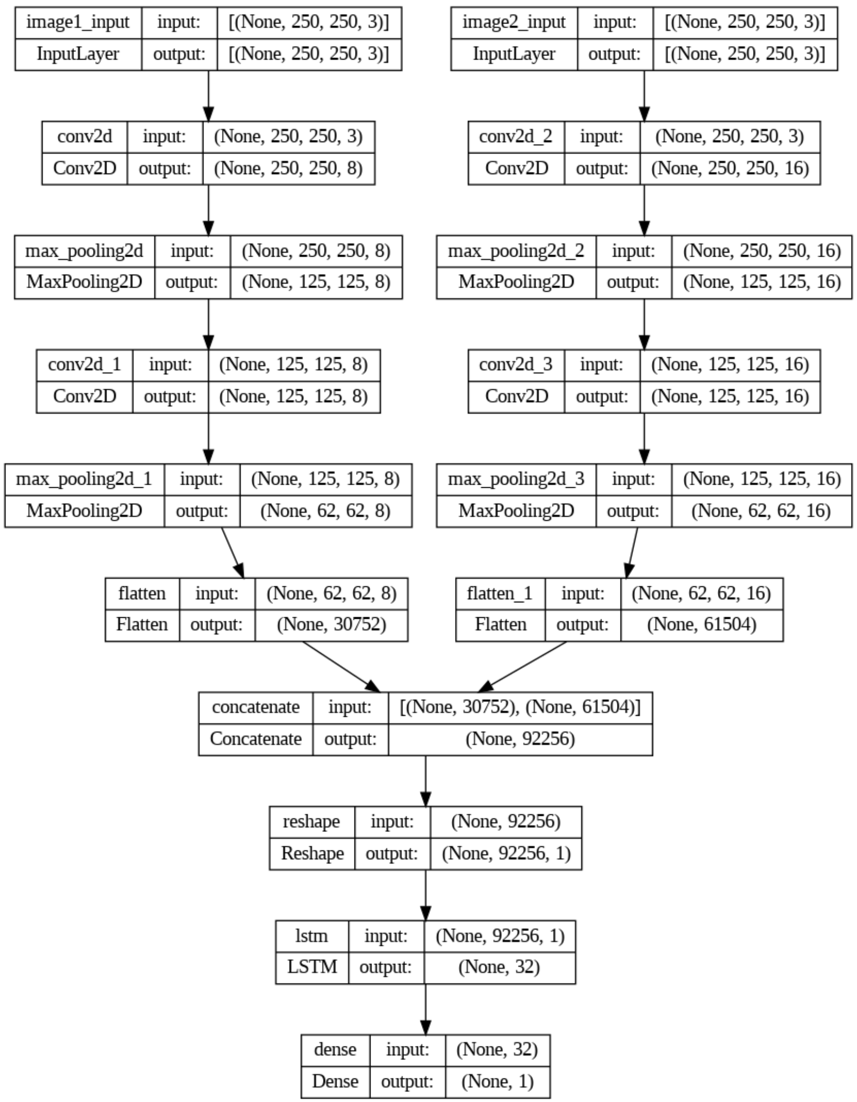
	time	Air_temp	Wind_speed	Wind_dir
0	06.12.21 08:50	1.6	1.0	324
1	06.12.21 09:10	1.8	1.2	331
2	06.12.21 09:20	1.7	1.7	329
3	06.12.21 09:30	1.8	1.3	325
4	06.12.21 09:50	1.9	1.0	326
...	...	...	...	...
9995	04.01.23 11:40	7.3	0.5	288
9996	04.01.23 11:50	7.3	0.7	165
9997	04.01.23 12:00	7.5	1.0	250
9998	04.01.23 12:10	7.6	1.2	261
9999	04.01.23 12:20	7.7	0.8	218

10000 rows × 4 columns

Network Architecture

- We asked Chat GPT: “We want a neural network in python that has CNN and LSTM layers to predict solar irradiance from an input of 2 RGB images”

```
# Define input shape
input_shape = (250, 250, 3)

# Define inputs
input1 = Input(shape=input_shape, name='image1_input')
input2 = Input(shape=input_shape, name='image2_input')

# CNN layers for image 1
x1 = Conv2D(8, (3, 3), activation='relu', padding='same')(input1)
x1 = MaxPooling2D((2, 2))(x1)
x1 = Conv2D(8, (3, 3), activation='relu', padding='same')(x1)
x1 = MaxPooling2D((2, 2))(x1)
x1 = Flatten()(x1)

# CNN layers for image 2
x2 = Conv2D(16, (3, 3), activation='relu', padding='same')(input2)
x2 = MaxPooling2D((2, 2))(x2)
x2 = Conv2D(16, (3, 3), activation='relu', padding='same')(x2)
x2 = MaxPooling2D((2, 2))(x2)
x2 = Flatten()(x2)

# Concatenate the outputs of the CNN layers
x = concatenate([x1, x2])
# (batch_size, time_steps, seq_len)
x = tf.keras.layers.Reshape((-1, 1))(x)
# LSTM layer
x = LSTM(32, activation='relu', return_sequences=False)(x)

# Output layer
output = Dense(1, activation='relu')(x)

# Define the model
model = tf.keras.Model(inputs=[input1, input2], outputs=output)

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
```

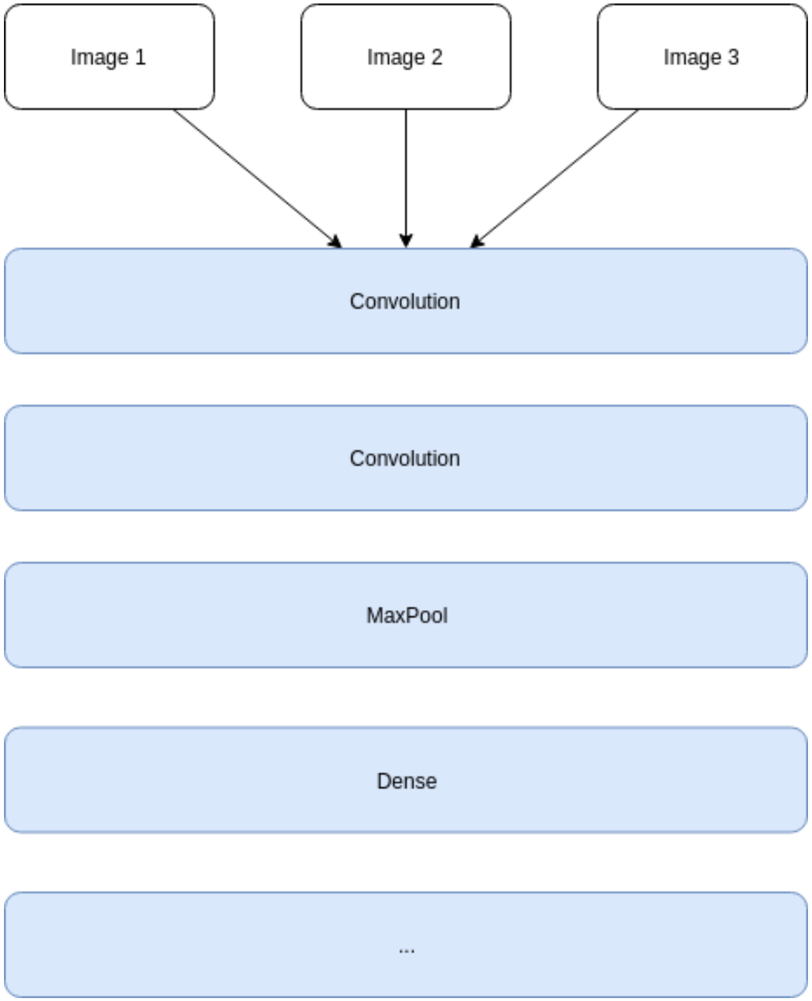


Network Architecture – Time Distributed

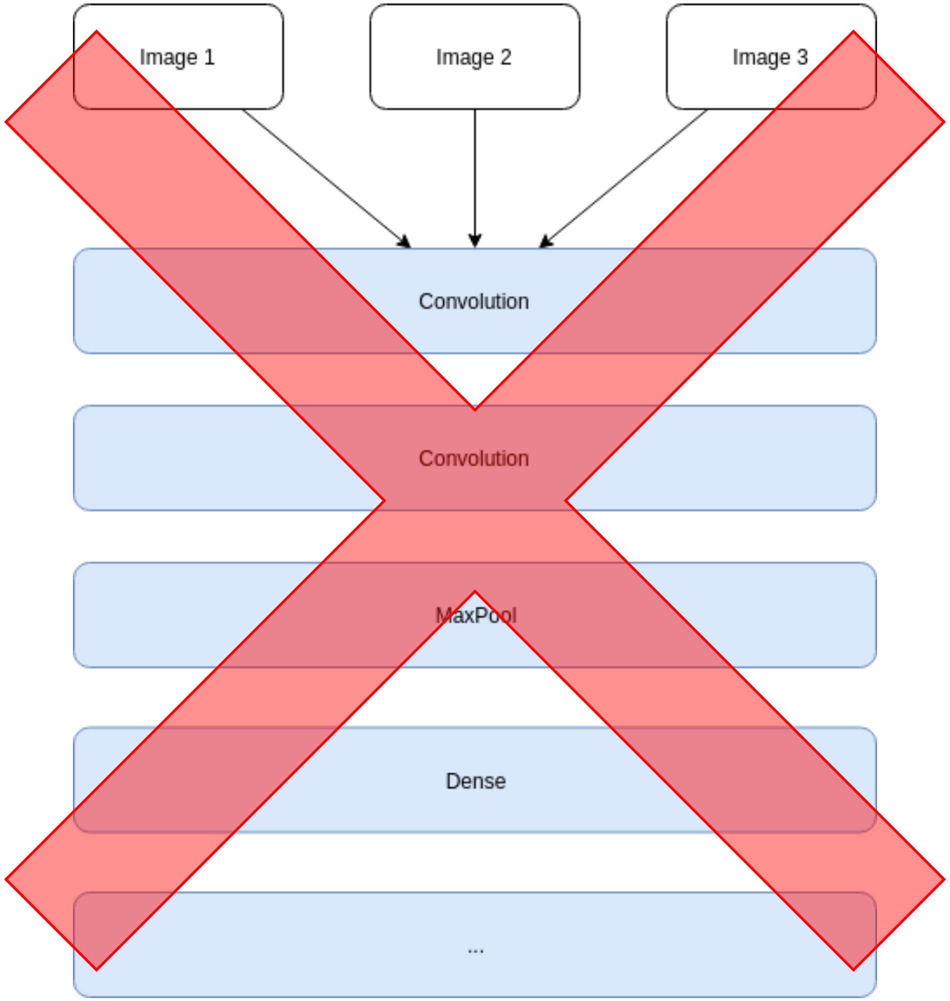
- Modified with Time Distributed
- Modify your input to have one X with shape (10000, 2, 250, 250, 3)
  You can use X = np.stack((X1,X2), 1)

```
img_size = 250
channels = 3

input = tf.keras.layers.Input(shape=(img_size, img_size, channels))
input_sequence1 = tf.keras.layers.Input(shape=(2,img_size, img_size, channels))

# CNN Block 1
first_input = tf.keras.layers.TimeDistributed(Conv2D(8, kernel_size=3, activation='relu'))(input_sequence1)
first_input = tf.keras.layers.TimeDistributed(MaxPooling2D(pool_size = 2))(first_input)
first_input = tf.keras.layers.TimeDistributed(Conv2D(8, kernel_size=3, activation='relu'))(first_input)
first_input = tf.keras.layers.TimeDistributed(MaxPooling2D(pool_size = 2))(first_input)

# CNN Block 2
first_input = tf.keras.layers.TimeDistributed(Conv2D(16, kernel_size=3, activation='relu'))(first_input)
first_input = tf.keras.layers.TimeDistributed(MaxPooling2D(pool_size = 2))(first_input)
first_input = tf.keras.layers.TimeDistributed(Conv2D(16, kernel_size=3, activation='relu'))(first_input)
first_input = tf.keras.layers.TimeDistributed(MaxPooling2D(pool_size = 2))(first_input)

# Flattening and Reshaping for LSTM
flt = tf.keras.layers.TimeDistributed(Flatten())(first_input)
flt = tf.keras.layers.Reshape((1, -1))(flt)

# LSTM
lstm1 = tf.keras.layers.LSTM(32, activation='relu', return_sequences=True)(flt)

# Output
dense1 = tf.keras.layers.Dense(1, activation='relu')(lstm1)

# Setting input and ouput of model
model = tf.keras.models.Model(inputs=[input_sequence1], outputs=dense1)

model.compile(optimizer="Adam", loss='mse')

# Remember X_train should be something like (samples, number of images per sample, image width, image height, channels) so (10000,2,250,250,3)
history = model.fit(np.asarray(X_train), np.asarray(y_train), ...)
```

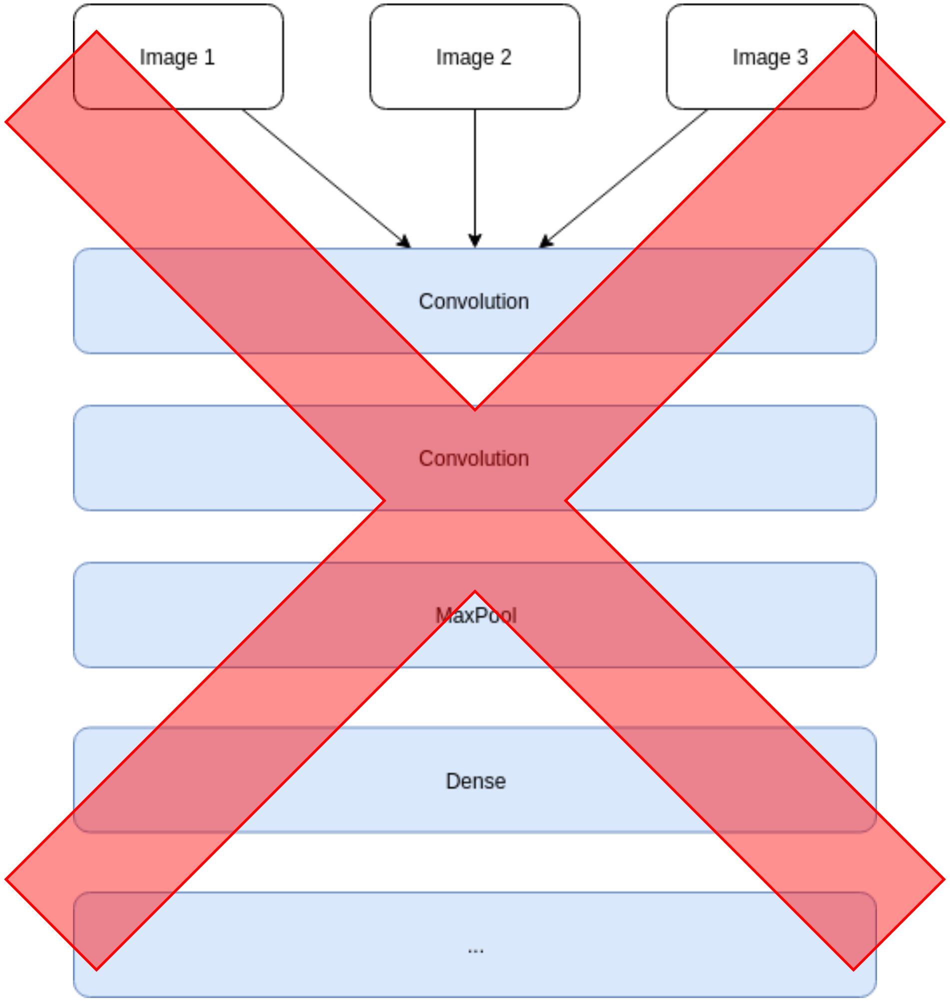
Network Architecture – Time Distributed



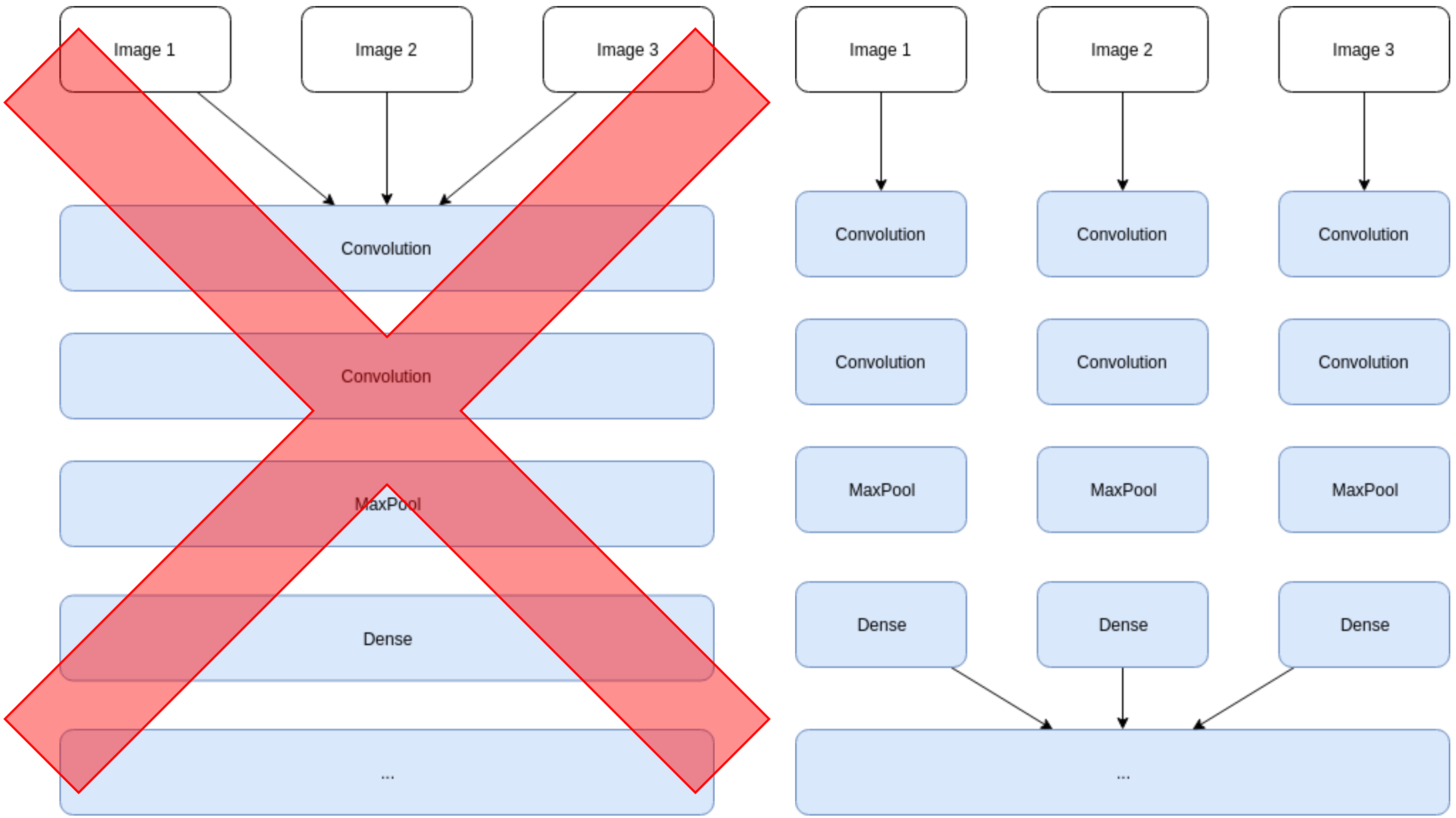
Network Architecture – Time Distributed



Network Architecture – Time Distributed



Network Architecture – Time Distributed



- Requires long training as we need to train several convolution flows (one per input image)

Network Architecture – Time Distributed

- The Time Distributed Layer applies the same instance of the layer to each image.
- So we don't have different sets of weights for this layer.
- The same set of weights are applied to all images.
- By using this layer, we will not be increasing the complexity of the model. Yet, the model has the ability to learn from different images separately with just one layer.