**Project Title:** Deep Learning model to estimate land-surface temperature in Israel

**Student Name**: Itay Barkai

**Advisor**: Dr. Oren Glickman (CS, BIU).

Chapter 1:

The goal of this project is to build a deep-learning model to estimate land surface temperature in Israel at a 1x1 km resolution. The Land Surface Temperature (LST) is the radiative skin temperature of the land surface, as measured in the direction of the remote sensor. Historical LST data is available at a 1x1 km resolution, however we would like to see if we can use this data as a target for a deep learning model to learn to downscale temperature data to a finer resolution.

Available Data (For Israel):

* Historical Hourly LST data for the past 20 years. Data includes:
  + lat, long of observation
  + Date & Hour in day
  + LST temperature
* Topography:
  + lat, long
  + height (above sea level)
* Vegetation index from satellite imaging (NDVI) - at a 1x1 resolution
* Additional information (e.g. 9x9 Air temperature, land use, …) may be also available - to be possibly used in the project

[shilo.shiff@biu.ac.il](mailto:shilo.shiff@biu.ac.il) - will provide the data for this project.

Learning Task:

The goal of the deep learning model is to estimate the 1x1 km resolution land surface temperature given the surrounding topography.

For every such 1x1 km pixel, a 9x9 ‘patch’ surrounding the pixel as in the following illustration:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  | x |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

The overall LST of the 9x9 grid at a given time will be calculated as the average of the 81 LST observations at the same time. Similarly, the average height of the 9x9 grid will be calculated.

The difference between the 9x9 LST and center pixel 1x1 LST will be the target value to predict. Evaluation of the regression problem will be measured via RMSE.

For every such observation, the input to the model will include:

* The Average height of the 9x9 grid
* The average LST of the 9x9 grid
* 81 height diffs (difference between pixel height to the average height for every 1x1 pixel in the grid)
* The day in the the year (0-365, consider transform via sine and cosine)
* The time in the day (0-24, consider transform via sine and cosine)
* lat, long of center pixel
* We may add additional features as we progress (e.g. NDVI

Project work includes finding the optimal feature representation and deep learning neural net architecture (# layers etc, ) as well as analysis of the results.

It is important to test the model on unseen (held out) pixels and dates.

Research Questions:

1. Can you improve RMSE predictions results over the simple baseline of always predicting 0 (temp of 1x1 center pixel identical to the 9x9 average).
2. What is the optimal DL model
3. How the various features impact the prediction

**Chapter 2: Startup:**

Had a Zoom meeting with [shilo.shiff@biu.ac.il](mailto:shilo.shiff@biu.ac.il)

- He helped me understand the project goals and terminology,

- Gave me the LST data for Israel in 2002-2020 (in NetCDF format): <https://zenodo.org/record/4533677#.ZEvTr3ZBwuV>

- Gave me the topography data of Israel as exported from "Google Earth Engine".

- Explained how to work with this format of data in python.

Following that I asked Dr. Oren some clarification questions:

1. More general purpose of the project?

a. Proof of concept for Resolution Downscale

b. Further downscale for LST

c. Downscale air temp

d. Using learned relations of various features for other tasks.

e. General ML research in case of similar records (same place at different times)

2. Pytorch vs tensorflow?

- Up to me.

3. Research common ways to represent "day" and "hour"

- Using cos and sin

4. Requesting access to ML servers for computation and storage that are larger than my home computer.

a. Got a user, and a welcome guide on how to use them.

(Had some back and forth with dsi support to make it work)

**Chapter 3: Learning before and during the project**

To better understand how to code "correctly" and how to use Tensorflow, I've seen freeCodeCamp's 7 hour "TensorFlow 2.0 Complete Course"

- [www.youtube.com/watch?v=tPYj3fFJGjk](http://www.youtube.com/watch?v=tPYj3fFJGjk)

Research some MLOps utils that would help plan the structure and tech stack of the project

- <https://mymlops.com/examples/airflow-mlflow> - this being very close to what I ended up doing.

**Chapter 4: Infrastructure and local environment:**

- I have opened a public repository on my github to better source control the project.

- Multiple folders such as Documents (Project definition, This document), Data and Sources.

- Installed python 3.8.8 Anaconda distribution,

- Ipython 7.22 for easily playing with the data and debugging.

- Decided to use the Tensorflow library (to gain experience in another library)

- Installed it and studied the basics.

Later on finding Git not enough for modern and well documented ML project.

I've research online and also asked Oren about useful MLOps.

Came to conclusion I could use this **stack**:

- "Git" (github) for Code verion control

- "DVC" for Data version control

- "MLFlow" for automated logging of models and their parameters

- (optional) Use TensorBoard for nice visualization of performance.

**Chapter 5: Start Coding the basic modules and play with the data.**

- Figured it is recommended to use the netCDF4 library to work with the NetCDF LST files.

- Got familiar with the library, with the data format.

- Using ipython, and the data's documentation here: <https://zenodo.org/record/4533677#.ZFDyHHZBwuX>

Figured the format of the actual data in our NetCDFs

**Data Documentation:**

**The LST Data:**

Consists of NetCDFs for each year between 2002-2020.

Each one should have 2 samples per day of the LST on each 1X1 block in Israel.

Practically each NetCDF has 5 variables named: ["y", "x", "band", "time", "\_\_xarray\_dataarray\_variable\_\_"]

**"y"** = latitude, **603** points between 29.996 and 34.007

**"x"** = longitude, **409** points between 33.200 and 36.596

**"band"** = 4 values (channels):

NIGHT\_LST = 0

DAY\_LST = 1

DAILY\_LST = 2 (average of 2 previous ones)

QA = 3

QA means if there were clouds that interrupted the actual sample and the data is generated from a previous model: (<https://zenodo.org/record/4533677#.ZFkQRHZBwuX>).  
The values in the QA band represents:

NO\_DATA\_BOTH\_DAY\_NIGHT = 0

NO\_DATA\_DAY = 1

NO\_DATA\_NIGHT = 2

BOTH\_VALID = 3

**"time"** = day in the year for the sample – from 0 to 365 (366 in leap years).

There might be skips in time in days that for some reason did not have a sample.

**"\_\_xarray\_dataarray\_variable\_\_"** = The actual LST value for each combination of the above variables.

Has shape of (364, 4, 603, 409) – (for example in 2020)

\* Notice the time variable has 364 instead of 366 (leap year) because 2 samples are missing.

This is a numpy masked array which means it may have invalid data (like NaN).

For example in a single specific day there were 170k out of 240k valid values.

**####################**

**Coding Documentation:**

- Started coding the "consts.py" module with some file paths formats to easily open the data. And for future Consts.

- Started coding the "netCDFHandler.py" module – exports my class api to the NetCDF file, preferably doing the translation to tensors

- Staerted coding the "datasets.py" module – that exports api for different datasets we might make out of the raw data.

**####################**

**Chapter 6: Data Decisions**

As said, the data might be invalid/missing in some 1x1 spots on the grid.

We need to decide how to treat such data points, while we want to make samples out of a 1x1 spot and its 9x9 surrounding cell. And we have some options.

1. Only pick valid 1x1 spots, ignore their 9x9 cell

2. Only pick valid 1x1 spots, with more than threshold% of valid spots in their 9x9 cell

3. Only pick valid 1x1 spots, where all of their 9x9 are valid spots.

Of course option 1 would not give us any data to learn from, and option 2 with lower threshold% means low possible correlation to the surrounding.  
Also option 3 is actually option 2 with 100% threshold

The Question is what is our wanted threshold%, for that lets try to understand how many datapoints would remain if we use a 100% threshold

We can make the threshold a hyper parameter of the model (of the preprocessing part)