PROJECTS for the Hackathon @ SMR4067

PROJECT 1: Create a DL model (inspired by ClimateNet) for extreme event detection

The goal of this project will be to build a machine learning algorithm to detect extreme weather events (tropical cyclones and atmospheric rivers) from atmospheric data. Models that are capable of accurately detecting such events are crucial for our understanding of how they may evolve under various climate change scenarios. A simplified version of this project was designed as the first Kaggle competition for IFT 3395/6390 - Fundamentals of machine learning in Fall 2021.

This project is inspired by the availability of a benchmark data set for the detection of extreme weather events, ClimateNet. ClimateNet contains time series of climate variables from nearly 900,000 locations (latitude and longitude) around the world, each labelled according to one of three classes:

- Standard background conditions
- Tropical cyclone
- Atmospheric river

The number of data points per class is not balanced, as the *standard background conditions* represent the majority class. This turns the task into a *detection* problem. The data set consists of 16 atmospheric variables such as pressure, temperature and humidity, besides the latitude, longitude and time. The complete data set amounts to almost 30 GB.

Relevant methods

Teams working on this project will likely look into detection algorithms, methods for time series data (such as recurrent neural networks) and for spatial data (such as convolutional networks). Methods developed to deal specifically with climate data are likely to be relevant too.

Minimum goals

A reasonable minimum requirement will be for the teams to reproduce the original results, possibly making use of the available open-sourced code by the authors of the data set, and ideally other follow-up of work tackling this benchmark.

A successful project will additionally extend the available methods or propose alternatives that are able to improve the existing previous work according to some relevant criteria, such as final performance, training speed, data efficiency, etc.

References

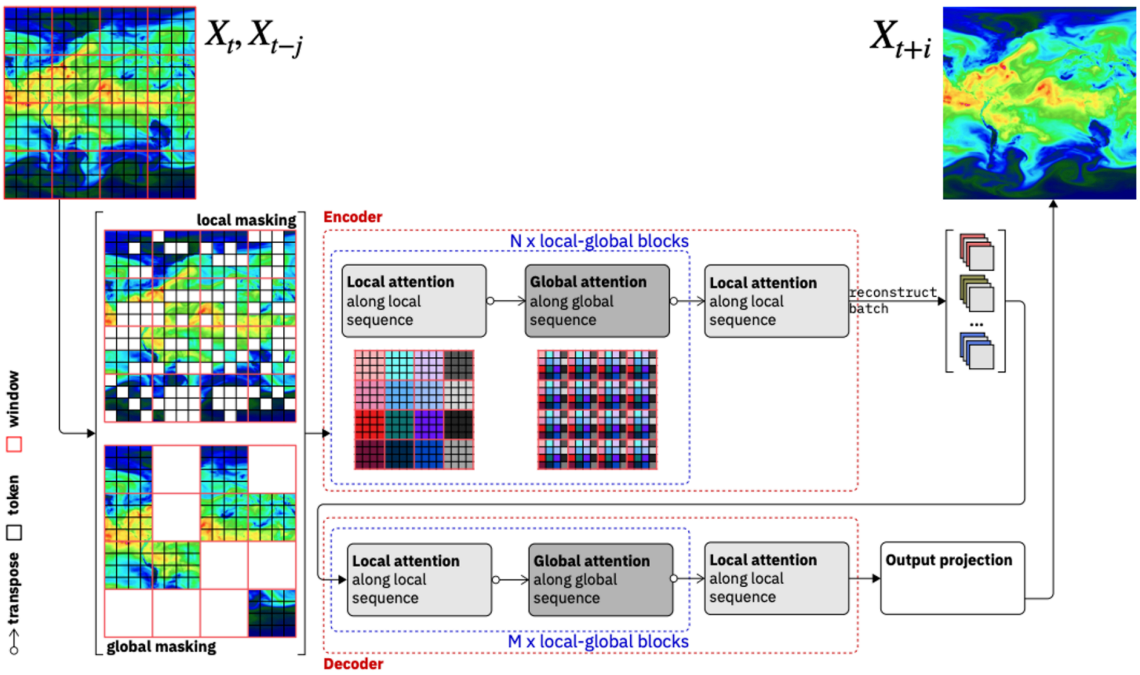
- **ClimateNet data set page**
- **GitHub repository**
- **Paper: Spatio-temporal segmentation and tracking of weather patterns with light-weight Neural Networks**

PROJECT 2: Benchmarking PrihtviWxC model on WeatherBench2 (or -X)

The goal of this project will be to benchmark the Foundational Model PrithviWxC on one of the datasets available through WeatherBench2 (or Weather Bench-X).

The model

PrithviWxC consists of a Masked Autoencoder architecture which is constructed to have local and global attention through local and global masking. The model follows the flavour of Multi-axis ViT (MaxViT) that leverages multi-axial attention without the constraint of having a rectangular input grid. To this end, during the pre-training phase, the main constraint on the model is to specify the windows and tokens therein, such that input batches have the dimensions (batch, windows, tokens, features). Attention is alternated to act across tokens and windows. This can be implemented by alternatively transposing the token and window dimension and rolling the one into the batch dimension such that attention works across the other. Additionally, after the initial training step this also provides an opportunity to add a transformer architecture on top of the learnt masked encoding structures. In its final configuration, PrithviWxC consists of 25 encoder and 5 decoder blocks.



The inputs to PrithviWxC come from the The Modern-Era Retrospective Analysis for Research and Applications Version 2 (MERRA-2) dataset and static inputs of elevation, land cover, ice cover, lake cover are provided. Additionally, encoding of the lead time δt and initialization time $\delta\tau$ are learned. PrithviWxC is trained in two phases: (1) using a 5% drop path, a 50% masking ratio and alternating “local” and “global” masking with randomly selected forecast lead times (among 0, 6, 12 and 24 hours ahead) as well as delta between inputs (-3, -6, -9, -12); and (2) reducing the the masking ratio to 0% and adding a Swin-shift to the encoder. The loss function used looks at information added on top of climatology (see here equation 2).

PROJECT 3: Benchmarking GraphCast model on WeatherBench2 (-X)

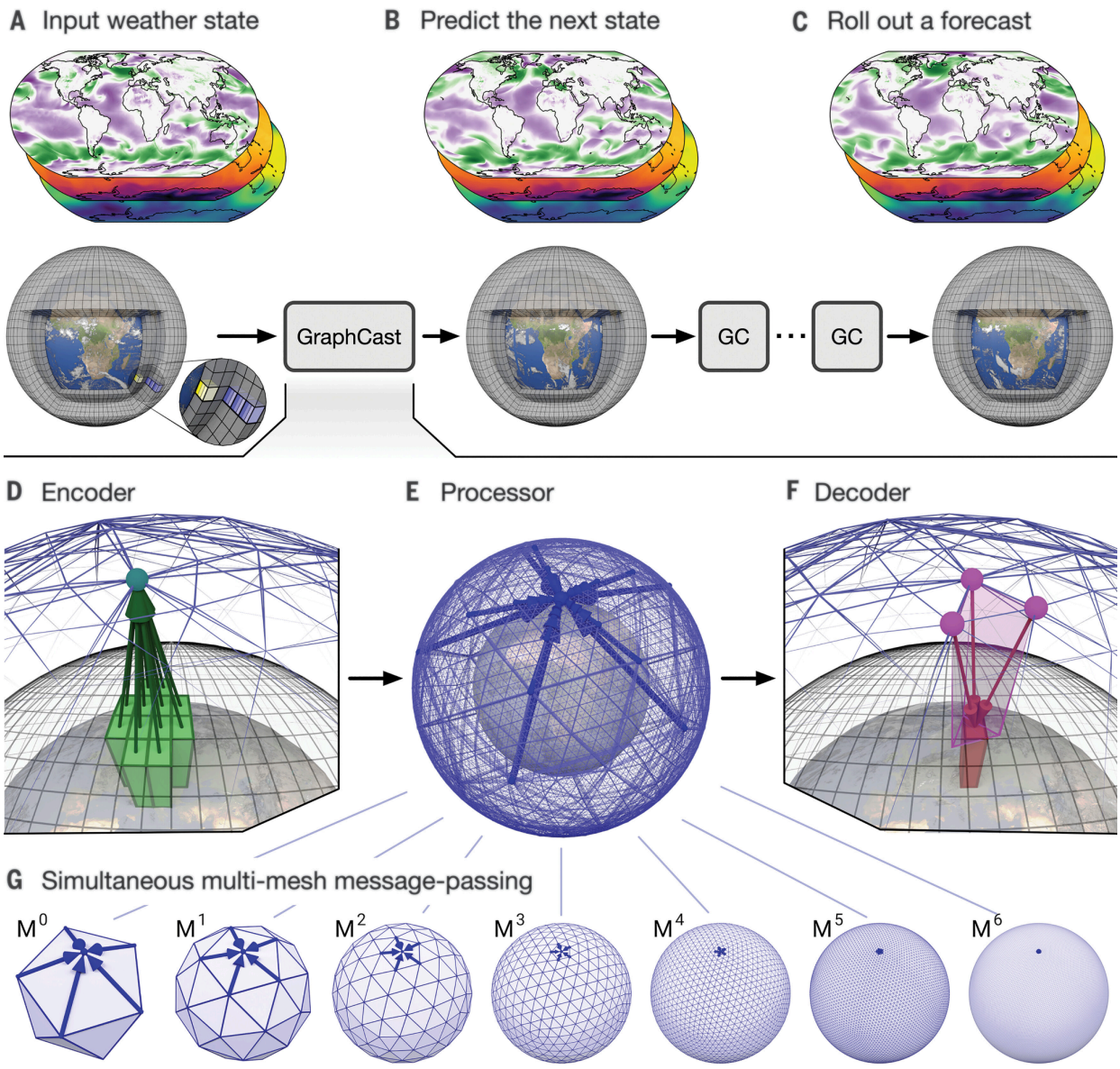
The goal of this project will be to benchmark the Foundational Model Graphcast on one of the datasets available through WeatherBench2 (or Weather Bench-X).

The model

GraphCast is implemented as a neural network architecture, based on GNNs in an “encoder-processor-decoder” configuration.

The encoder (Fig.D) uses a single GNN layer to map variables (normalized to zero-mean unit variance) represented as node attributes on the input grid to learned node attributes on an internal “multimesh” representation. The multimesh (Fig.G) is a graph that is spatially homogeneous, with high spatial resolution over the globe. It is defined by refining a regular icosahedron (12 nodes, 20 faces, 30 edges) iteratively six times, where each refinement divides

each triangle into four smaller ones (leading to four times more faces and edges), and reprojecting the nodes onto the sphere. The multimesh contains the 40,962 nodes from the highest-resolution mesh (which is roughly 1/25 the number of latitude-longitude grid points at 0.25°) and the union of all the edges created in the intermediate graphs, forming a flat hierarchy of edges with varying lengths. The processor (Fig.E) uses 16 unshared GNN layers to perform learned message-passing on the multimesh, enabling efficient local and long-range information propagation with few message-passing steps. The decoder (Fig. 1F) maps the final processor layer's learned features from the multimesh representation back to the latitude-longitude grid. It uses a single GNN layer and predicts the output as a residual update to the most recent input state (with output normalization to achieve unit variance on the target residual).
You can look at a schematic of the model below:



Github repos providing different model version (in Jax or Pytorch):

- Original Google repo
- nvidia-PhysicsNemo

The datasets

WeatherBench2 (initially presented in the paper by) consists of:

- Publicly available, cloud-optimized ground truth and baseline datasets.
- Open-source evaluation code. See this quick-start to explore the basic functionality or the API docs for more detail.
- A website displaying up-to-date scores of many of the state-of-the-art data-driven and physical approaches.

If you would like your model to be part of WeatherBench, check out this guide.

WeatherBench-X is the successor of WeatherBench2.
The core design principles behind WeatherBench-X are:

- Modularity: Data loaders, Interpolations, Metrics and the Aggregation can be defined through interoperable classes.
- Xarray: All internal logic is based on xarray DataArrays.
- Scalability: Each operation can be split into small chunks allowing scalable evaluation

To get started using WeatherBench-X, check out the quickstart notebook.

Relevant methods

Teams working on this project will likely look into GNN-based Foundation Model, ML-based weather forecast and standard analysis for benchmarking this kind of models on standardized datasets.

Minimum goals

A reasonable minimum requirement will be for the teams to run the evaluation python script reported in the WeatherBench2 (or WeatherBench-X) repo and compare their results to the one reported on the WeatherBench2 website for the graphcast_hres_init_vs_era5 run.

OPTIONAL: Since in the original Google repo it is also provided the Gencast model code you can try to run the benchmark for this model also

References

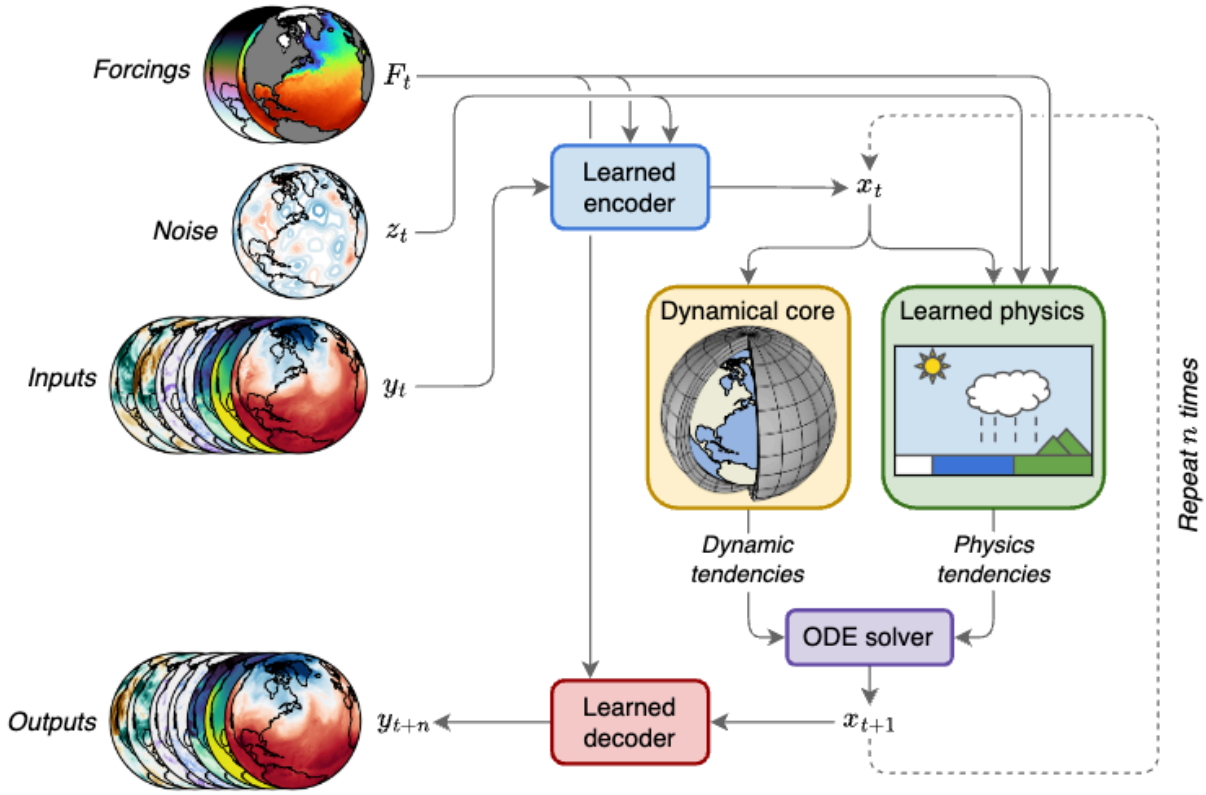
- WeatherBench2 paper
- Graphcast paper
- GenCast paper

PROJECT 4: NeuralGCM Ensemble forecast

The goal of this project will be to create an ensemble forecast with the NeuralGCM model, studying how the model predictions are impacted from the ensemble size.
NeuralGCM is a GCM that combines a differentiable solver for atmospheric dynamics with machine-learning components.

The model

You can look at a schematic representation of the model below:



The two key components of NeuralGCM are a differentiable dynamical core for solving the discretized governing dynamical equations and a learned physics module that parametrizes physical processes with a neural network.

The dynamical core simulates large-scale fluid motion and thermodynamics under the influence of gravity and the Coriolis force. The learned physics module predicts the effect of unresolved processes, such as cloud formation, radiative transport, precipitation and subgrid-scale dynamics, on the simulated fields using a neural network.
In Kochlov et al. (2020) the authors trained the NeuralGCM model at horizontal resolutions with grid spacing of 2.8°, 1.4° and 0.7°

Relevant methods

Teams working on this project will likely look into ODE solvers,AutoEncoder DL models and Physics-Informed training for DL models. Methods developed to deal specifically with ensemble climate forecasts are likely to be relevant too. In particular to evaluate the uncertainty and the bias of Ensemble Forecast you can use metrics for probabilistic scoring (we report below a table taken from the WeatherBench website, which summarizes possible metrics).

Probabilistic metrics

Short name	Long name	Description
CRPS	Continuous Ranked Probability Score	The CRPS measures how well the forecasted distribution (e.g. the ensemble) matched the ground-truth. To achieve good CRPS values, the forecasts have to be reliable, i.e. the forecasted uncertainty has to match the actual uncertainty, and sharp, i.e. a smaller uncertainty is better.
Ensemble Mean RMSE	Ensemble Mean Root Mean Squared Error	The RMSE of the ensemble mean. Also often called "skill".
Spread	Spread	The standard deviation of the ensemble.
Spread/skill ratio	Spread/skill ratio	In a reliable forecast the Ensemble Mean RMSE should match the spread. Smaller values indicate that the ensemble is underdispersive (too confident), larger values indicate that the ensemble is overdispersive (not confident enough). Note that this is a necessary but not sufficient metric for reliably forecasts.

Minimum goals

A reasonable minimum requirement will be for the teams to generate an ensemble forecast for at least two variables (you can choose temperature..), making use of the available open-sourced code by the authors of the , and ideally testing different ensemble sizes and

References

- Paper: Neural General Circulation Model for Weather and Climate
- Ensemble Verification Metrics
- **GitHub repo**

PROJECT 5: Deterministic model with random perturbation using Radiative-Convective Equilibrium on a limited periodic equatorial domain.

The radiative-convective equilibrium (RCE) is a concept for the radiative balance of the atmosphere. It describes the balance between the net radiative longwave cooling and the heating due to convection and surface fluxes.[1] The main difference to the pure radiative equilibrium is that the lapse rate in the troposphere is adjusted to a more realistic one.

In climate models, the concept is used to simulate the globally averaged thermal structure of the atmosphere and offers the opportunity to analyse the sensitivity of this structure to CO_2. On Earth, the tropical atmosphere is on many scales close to RCE. Therefore, the concept has also been used for studying tropical circulation and different aspects of moist convection.

The proposed experiment will be carried out by running multiple experiments using a modified version of the ICTP Regional Climate Model on the leonardo cluster. Each proposed experiment will use a different value for the sea surface temperature (295K, 300K, 305K), and if possible a different level of random noise addition to break the symmetry of the system and permit reasonable time evolution and if again possible different domain sizes to take into account the possible self aggregation of the randomly scattered convective events.

Relevant methods

Teams working on this project will be provided the model binary of the RegCM model compiled for the leonardo platform and the Fortran input namelist to submit the physical time integration process. The NetCDF output files must be analyzed using the python xarray library by the team members to evaluate the model results. The directory to copy out is:

```
rsync -a –progress /leonardo/pub/userexternal/ggiulian/public/smr-4067/rcemip
.
cd rcemip
sbatch model.job

squeue -u $USER

[ modify profile.in : comments inside ]
[ modify rcemip.in : comments inside ]
```

Run again to reach goals.

Minimum goals

A reasonable minimum requirement will be for the teams to plot the model output results after processing the data using cdo or directly the xarray methods and compare two different model outputs using two different sea surface temperatures.

Extension would be to run two different simulations for each of the above proposed three SST levels by increasing the noise level, and compare the model produced results.

References

- **RegCM model**
- **RCEMIP experiment**
- **Possible scripts to adapt for analysis**

PROJECT 6: Run a global simulation using the ICTP Speedy Model

HOWTO:

- **Run perturbation run:**

```
GGDIR=/leonardo/pub/userexternal/ggiulian/public/smr-4067
tar zxvf ${GGDIR}/speedy_ver42.2.tar.gz
cd speedy_ver42.2/run
source ../modules
ksh run_exp.s 501 300
```

Answer NO **[n]** to all the questions. Check that the job is successfully submitted:

```
squeue -u $USER
```

- **Run the control run:**

```
ksh run_exp.s 502 300
```

Answer YES **[y]** to the first question. An editor window pops up, and you should change in the edited file the the line reading:

`ISSTAN = 1` into: `ISSTAN = 0`

Answer NO **[n]** to all other questions. Check that the job is successfully submitted:

```
squeue -u $USER
```

- **When all the jobs are completed, you can transform all the output files into NetCDF format with:**

```
cd ../output/exp_501
../../ctl2nc *.ctl
cd ../exp_502
../../ctl2nc *.ctl
```

- **Analyze the results**

More in depth documentation on Speedy model can be found in the pdf file in the documentation directory.

GENERAL PRACTICAL DETAILS

To run the ML pipeline that you will develop for any of the ML-based projects (1-4) you can use the Booster partition of the Leonardo cluster hosted at CINECA (in Bologna), while the groups working on projects 5 and 6 will use the CPU-only partition (**LEONARDO Data Centric General Purpose (DCGP) partition**). More info on the two Leonardo partitions available here: https://wiki.u-gov.it/confluence/display/SCAIUS/LEONARDO+User+Guide

The instructions to access Leonardo, and run the Jupyter notebooks on the clusters are available in the official school Github repo:

https://github.com/Sera91/SMR4067-ICTP/tree/main