ECMWF Summer of Weather Code 2020

Challenge #21: Exploring or machine/deep learning techniques to detect and track tropical cyclones

Stream: 2 (Machine Learning and Artificial Intelligence)
Mentors: Linus Magnusson, Pedro Maciel

Proposal by: Ashwin Samudre
GitHub: @kiryteo

Contact Information:

Name: Ashwin Samudre
Phone: +33749237864
Email(s): capnoi8@gmail.com,
ashwin.samudre@embl.de
Twitter: @samudre_ashwin
Personal page: https://kiryteo.github.io/

Project Summary

Cyclones are the complex events characterized by strong winds surrounding a low pressure area. These are among the most harmful events on the Earth and accurate prediction of them is essential to make reliable warnings. This prediction is possible using the track of the cyclone and its intensity, which can be extracted from the forecasts. The events are manually detected and stored in the BestTrack database (IBTrACS) [1] using the satellite imagery data. Intensity classification is usually performed using Dvorak technique [2] which mainly focuses on statistical relationships between different environmental parameters and the intensity. ADT (Advanced Dvorak Technique) [3] is an automated and improved method based on Dvorak technique. These methods are mainly based on human experience. Current numerical weather forecast models predict a 4-dimensional volume including the spatial dimensions with time [4] and the 3-dimensional output can be used to simulate satellite images [5]. The availability of data for events like tropical cyclones is very large and thus using data-oriented techniques like machine learning and deep learning algorithms can be highly useful. These algorithms have shown remarkable performance for pattern recognition tasks and here we propose the application of deep learning algorithms to detect and classify the tropical cyclones efficiently.

Project goals

- Create a deep learning based model to recognize and classify tropical cyclones based on their intensities.
- Utilize 2 types of data for the task - a) Satellite imaging data b) BestTrack database information for tropical cyclones.
- Build the model for static (per satellite image) detection and classification and extend it to perform for dynamic (continuous real-time) detection and classification while maintaining robustness.

Implementation details

Tools and libraries required:

Python 3, PyTorch/ Keras, AWS/ WEkEO cloud services for data storage, model training and testing. Google Colaboratory (alternative for experiments), Binder.
Data - BestTrack database, ECMWF Satellite Image data (simulated), real-time forecast data (will be provided or assisted to acquire by mentors), netCDF4, cfgrib/ pygrib modules.

This project is divided into 2 stages:

1. Detection and classification of tropical cyclones using 2 data sources - 1] Satellite Images 2] Parameters like Wind, Pressure values from the BestTrack database.
2. Use the trained model from stage 1 to automatically detect tropical cyclones in the forecast data.

Stage 1

Object detection task

The task to detect cyclone features from the satellite images can be termed as an object detection problem. Object detection is defined as the task to locate the presence of objects with a bounding box and predict the classes of located objects in the image [6]. Object detection algorithms work based on the idea of defining the region of interest (ROI) along with its label and training the model to efficiently predict this ROI and its corresponding class.

Data handling and annotation

In the task of detecting the features for a tropical cyclone, we'll utilize the information regarding the centre of the cyclone based on minimum pressure from the ECMWF Analysis data (sample file [7]) to annotate the images. Here, we propose to use a bounding box with k*k degrees grid around the centre location, the value of k is considered 20 based on [8] or 25 based on [9] but can be adjusted based on need. The analysis data also provides the code of the cyclone, which acts as the label of the object (bounding box).
The annotations format: [Image_name], [xmin], [ymin], [xmax], [ymax], [label], in case of multiple labels for the same image (presence of multiple cyclones in the same image), the [Image_name] column will be the same and other values will differ accordingly.

The data format (NetCDF, Grib) can be handled using standard modules like netCDF4 [12] and cfgrib [13] or pygrib [14] modules. An open source version to handle NetCDF files is available for reference [15].

In [8], authors have preprocessed the satellite images in order to obtain the individual tropical cyclone image samples. These cropped images act as the input data to the Convolutional Neural Network (VGG-19 [9] model) which classifies them into different intensity classes (Image classification problem).

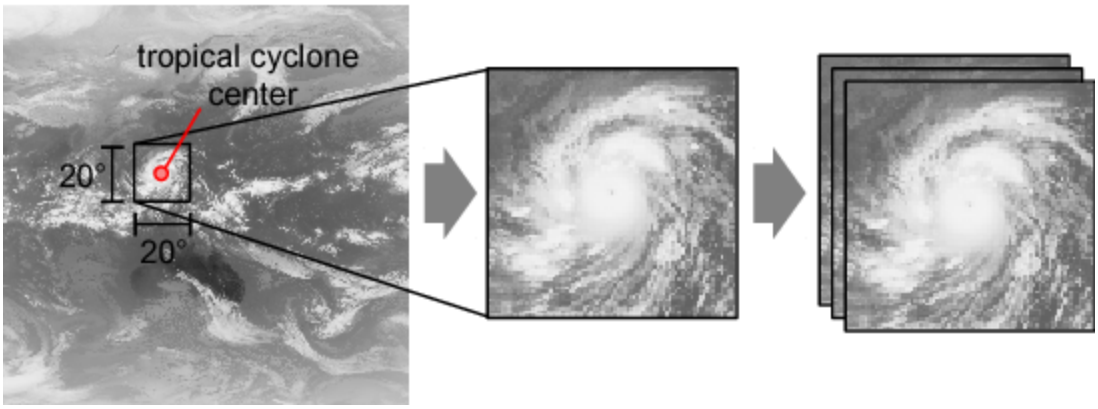
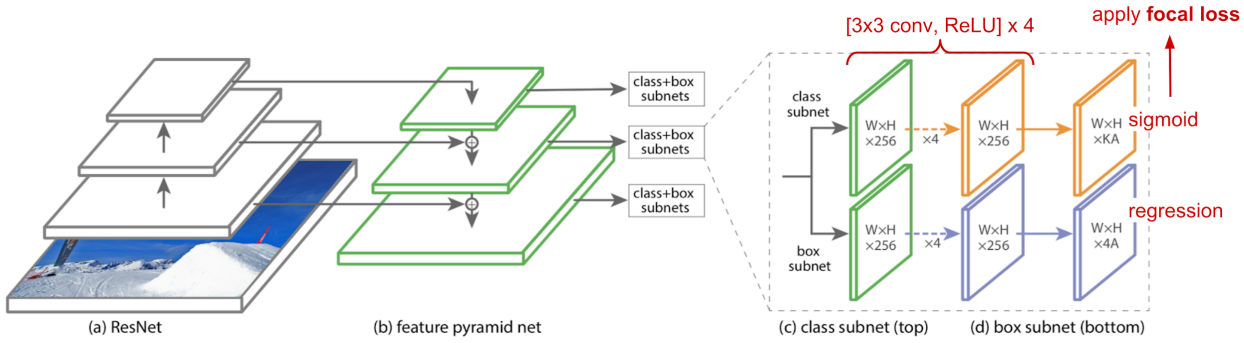


Figure 1. Preprocessing step in [8], Image source: [https://www.semanticscholar.org/paper/A-Convolutional-Neural-Network-Approach-for-Cyclone-Combinido-Mendoza/8bd178f4d32a1108565fb62dde562332a19f6684/figure/0]

In this project, we focus on training the deep learning model to detect and classify the cyclone intensity in an integrated manner (object recognition + image classification) and thus avoiding the preprocessing task.

Neural Network architecture

Deep learning algorithms perform better for computer vision tasks and thus we'll use neural network architecture for our case. The possible network architecture to be used for the task is RetinaNet [10]. RetinaNet is one of the better performing one-stage object detection models and has worked well with dense and small scale objects. The 4 important elements of the network are a) Resnet - for feature maps calculation, b) Feature pyramid network - Upsampling of spatially coarser feature maps from higher pyramid levels and lateral connections to merge the top-down, bottom-up layers with same spatial size, c) Class subnet - To predict probability of presence of an object at each spatial location for each anchor box and object class. d) Box (regression) subnet - regression of the offset for bounding boxes from anchor boxes for each ground-truth object.
Focal Loss - Based on Cross-Entropy loss and reduces the contribution from simple examples and increases the importance of prediction for hard examples adaptively.

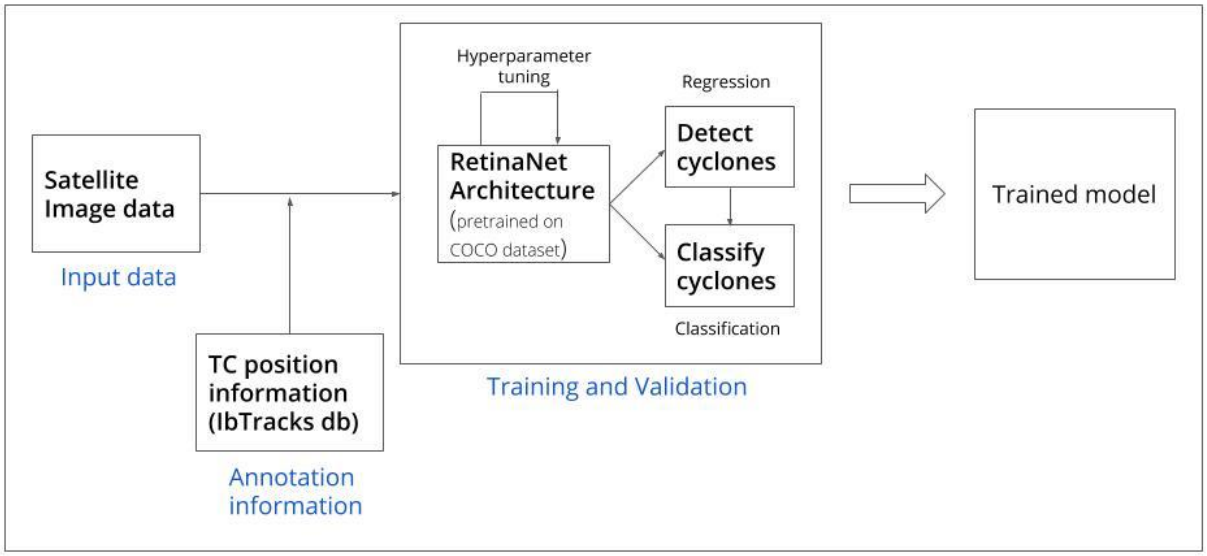


RetinaNet architecture

Source: [https://arxiv.org/abs/1612.03144]

Instead of training the architecture from scratch, we'll use a standard open source object detection dataset (COCO [17]) for pre-training the model. This is mainly to set the initial weights instead of random initialization, this helps in faster training and convergence of the model. The satellite image dataset being grayscale, there is also possibility to convert the pre-training dataset samples to grayscale and train the model (experimental idea).

The complete pipeline for stage 1 can be observed in the figure below.

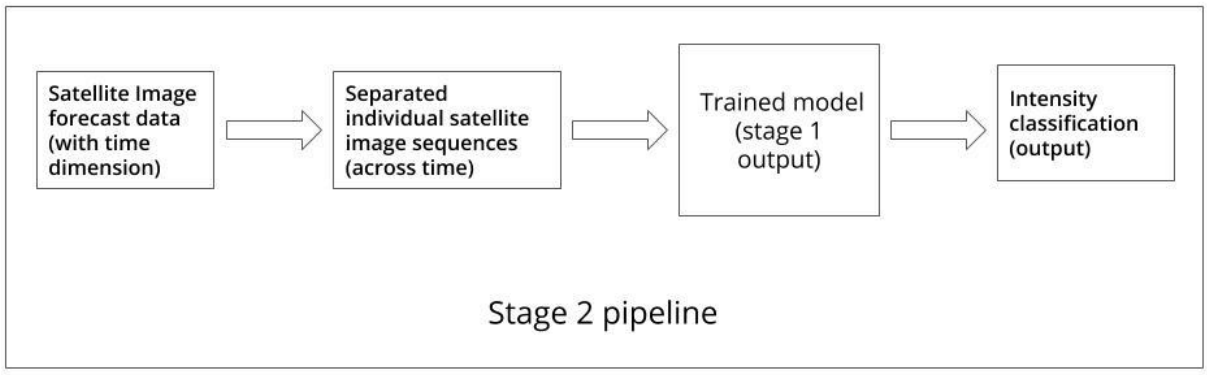


Stage 1 pipeline

Stage 2

The trained system will be applied for the forecast data. Forecast data includes the time parameter and thus the idea would be to separate the data into individual frames based on timestamp and provide these as input to the trained model from stage 1.

Stage 2 flow can be observed in the figure below.



Stage 2 pipeline

Project Timeline

Pre coding period	● Sync with the mentors to discuss the key elements and objectives of the project. ● Refine project proposal for final implementation. ● Create a GitHub repository for the project with initial files.
May 4 - May 15	● AWS/ WEkEO Cloud setup for deep learning experiments. ● Write scripts for data acquisition - satellite images + IbTracks database, handling NetCDF/ grib format.
May 16 - May 28	● Write scripts for creation of annotation file (.csv) using the information from IbTracks data. ● Write the Dataloader based on acquired data for RetinaNet
May 29 - Jun 22	● Implement RetinaNet architecture. ○ Implement ResNet module (Resnet-18/34/50, etc.). ○ Implement FPN module.
Milestone Jun 23 - Jun 28	**Satellite Image Data Acquisition complete and Neural network architecture ready for experiments (training and validation)** Add tests for scripts, resolve bugs.
Jun 29 - Jul 6	● Acquire a standard object detection dataset for pre-training the model (COCO dataset). ● Train the RetinaNet on the COCO dataset and save the weights.
Jul 7 - Jul 19	● Load pre-trained weights and train the model on our task dataset. ● Hyperparameter tuning and performance check of the model. ● Save the model as an HDF5 file.
Milestone Jul 20 - Jul 25	**Stage 1 complete, trained model ready for testing on forecast data** Add tests for scripts, resolve bugs.
Jul 26 - Aug 10	● Acquire forecast data for testing purposes.* ● Write processing scripts for separation of forecasting data volume into individual frames. ● Write scripts for testing the model on forecast data.
Milestone Aug 11 - Aug 16	**Stage 2 complete, check performance on forecast data** Add tests for scripts, resolve bugs.
Aug 17 - Aug 31	● Write jupyter notebooks utilizing the scripts from stage 1 and stage 2. ● Documentation. ● Release the notebooks on Binder platform for simplified reproducibility. ● Prepare for the presentation. ● Buffer for unpredictable delay.

* This step can be performed during initial data acquisition.

Project outcomes release

The trained model will be made available as an HDF5 file for testing on forecast data. Scripts to handle and process data (dataloader), train the model, hyperparameter setting and testing the model will be available in the project GitHub repository and instructions will be provided in a readme file. This way the model can be further used with a new set of data. Training and testing scripts can also be provided as Jupyter notebooks for simplified usage. These notebooks will be released on Binder [16] for better reproducibility of the results.

About me

I am currently a visiting fellow at EMBL Heidelberg, working on deep learning for bioimage analysis. Previously, I was a research engineer at CNRS CINAM lab, Marseille, France. Apart from regularly working in an open source environment, I have a strong experience in open source programs, having successfully completed Google Summer of Code 2018 with CERN-HSF (project) and Season of KDE 2018 with Kdenlive (project).

References

[1] https://journals.ametsoc.org/doi/10.1175/2009BAMS2755.1
[2] https://en.wikipedia.org/wiki/Dvorak_technique
[3] http://tropic.ssec.wisc.edu/misc/adt/info.html
[4] https://docs.google.com/document/d/1PRYXkUbRj3BLtQ00mVuIR9nDkgUAaXFtLlYp3DKkEBc/edit
[5] https://www.ecmwf.int/en/newsletter/152/news/fresh-look-tropical-cyclone-intensity-estimates
[6] https://arxiv.org/abs/1409.0575
[7] https://github.com/esowc/challenges_2020/files/4431910/tc_an_sample.csv.txt
[8]https://www.semanticscholar.org/paper/A-Convolutional-Neural-Network-Approach-for-Cyclone-Combinido-Mendoza/8bd178f4d32a1108565fb62dde562332a19f6684
[9] https://hal.archives-ouvertes.fr/hal-01851001
[10] https://arxiv.org/abs/1708.02002
[11] https://arxiv.org/abs/1409.1556
[12] https://unidata.github.io/netcdf4-python/netCDF4/index.html
[13] https://github.com/ecmwf/cfgrib
[14] https://jswhit.github.io/pygrib/docs/
[15] https://github.com/sophiegif/FusionCNN_hurricanes/blob/master/DataProcessing/ModuleStormReader.py
[16] https://mybinder.org/
[17] http://cocodataset.org/#home