storm_forcast documentation

Release 1.0

Katrina

May 27, 2022

THE DAY AFTER TOMORROW

Real-time tropical storm forecasting model with interactive jupyter notebook.

|111|

1.1 About

Hurricanes would kill up to 1,000 people and produce over $50 billion in damage in a single occurrence, and have killed far over 160,000 people in recent history. Humanitarian response activities during a tropical cyclone rely on reliable risk approximation models that can assist anticipate optimal emergency strategy options.

FEMA (Federal Emergency Management Agency) in the United States has announced an open competition to strengthen its emergency processes in the event of hurricanes, which is open to teams of ML specialists who are able to solve the challenge of forecasting the the evolution of tropical storms in real time. Team Katrina participated the competition and implementing custom networks to solve the challenge.

1.2 Data

- NASA Satellite images of tropical storms
- 494 storms around the Atlantic and East Pacific Oceans (precise locations undisclosed)
- Each with varied number of time samples (4 - 648, avg 142)
- Labelled by id, ocean (1 or 2) and wind speed
- Data source (To download it you must create an account with Radient MLHub)

1.3 Installation Guide

Clone the local repository by running:

```
git clone https://github.com/ese-msc-2021/acds-day-after-tomorrow-katrina.git
```

Then navigate to the root of the repository that you cloned and install all the required packages by running:

```
pip install -r requirements.txt
pip install -e .
```

1.4 User instructions

Click the `surprise_storm.ipynb` file, then you are ready to go.

All the interactive function is written inside the notebook, every function will trigger corresponding packaged utility scripts. The dataset used in `train.py` is provided by NASA. The dataset used in `surprise_storm.ipynb` is provied by Imperial College London ESE department. You can use the downloaded dataset by setting hyperparameter`download` as True.

1.5 Documentation

Documentation can be found inside the the `docs` folder in the `storm_forcast.pdf` which contains installation instructions and `storm_forcast` function APIs.

1.6 References

1. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.; Wong, W.; Woo, W. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting.
2. Graves A. Generating Sequences With Recurrent Neural Networks 2022.
3. J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation.
4. R. Panda, "Video Frame Prediction using ConvLSTM Network in PyTorch", Medium, 2022. [Online]. Available: https://sladewinter.medium.com/video-frame-prediction-using-convlstm-network-in-pytorch-b5210a6ce582. [Accessed: 26- May- 2022]

`storm_forcast` Module

class `storm_forcast.`**`ConvLSTMCell`** (*in_channels*, *out_channels*, *kernel_size*, *padding*, *activation*, *frame_size*)

A convolution unit in ConvLSTM algorithm as proposed by Shi et al.

Parameters

- **`in_channels`** (*`int`*) – Number of channels of input tensor.
- **`out_channels`** (*`int`*) – Number of output feature maps
- **`kernel_size`** (*`(int, int)`*) – Size of the convolutional kernel.
- **`padding`** (*`Union[str, _size_2_t]`*) – Padding added to the input tensor
- **`activation`** (*`str`*) – Activation functions type ('tanh' or 'relu')
- **`frame_size`** (*`(int, int)`*) – Size of the input image

Notes
See https://github.com/sladewinter/ConvLSTM

`_is_full_backward_hook`: Optional[bool]

`forward` (*X*, *H_prev*, *C_prev*)

Defines the computation performed at every call.
Should be overridden by all subclasses.

Note Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

`training`: bool

class `storm_forcast.`**`ConvLSTM`** (*in_channels*, *out_channels*, *kernel_size*, *padding*, *activation*, *frame_size*)

a single ConvLSTM layer in our network.

Parameters

- **`in_channels`** (*`int`*) – Number of channels of input tensor.
- **`out_channels`** (*`int`*) – Number of output feature maps
- **`kernel_size`** (*`(int, int)`*) – Size of the convolutional kernel.
- **`padding`** (*`Union[str, _size_2_t]`*) – Padding added to the input tensor
- **`activation`** (*`str`*) – Activation functions type ('tanh' or 'relu')
- **`frame_size`** (*`(int, int)`*) – Size of the input image

Notes
See https://github.com/sladewinter/ConvLSTM

`_is_full_backward_hook`: Optional[bool]

`forward` (*X*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

`training`: bool

class `storm_forcast.`**`Seq2Seq`** (*num_channels*, *num_kernels*, *kernel_size*, *padding*, *activation*, *frame_size*, *num_layers*)

A model that generates sequence from sequence

We finish off the network by stacking up a few ConvLSTM layers, followed by BatchNorm3d layers, and the finally followed by a Conv2d layer, which can take the hidden state at the last time step of the last layer and predict a frame. Finally, we pass this through a Sigmoid activation to get pixel values between 0 and 1.

Parameters
- **`num_channels`** (*`int`*) – Number of channels of input image.
- **`num_kernels`** (*`int`*) – Number of feature map output of convolutional layers
- **`kernel_size`** (*`(int, int)`*) – Size of the convolutional kernel.
- **`padding`** (*`Union[str, _size_2_t]`*) – Padding added to the input tensor
- **`activation`** (*`str`*) – Activation functions type ('tanh' or 'relu')
- **`frame_size`** (*`(int, int)`*) – Size of the input image
- **`num_layers`** (*`int`*) – Number of internal hidden layers

Notes

See https://github.com/sladewinter/ConvLSTM

`_is_full_backward_hook`: Optional[bool]

`forward` (*X*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

`training`: bool

class `storm_forcast.`**`StormDataset`** (*root_dir*, *storm_list*, *test=False*, *transform=None*, *seq_size=10*, *download=False*, *surprise=False*)

A dataset class that stores storm data

Collection of data download, loading, pre-processing functions

Parameters
- **`root_dir`** (*`str`*) – The path to the data set (can be a relative path)
- **`storm_list`** (*`list`*) – Storm ID to load
- **`test`** (*`bool`*) – Whether to read the test set. Default is False
- **`transform`** (*`class`*) – The transformation to be made to the image

- **seq_size** (*int*) – The length of each sequence. Default is 10
- **download** (*bool*) – Whether to download and decompress the dataset before loading. Default is False
- **surprise** (*bool*) – Compatible for Surprise Storm, whether to read special types of folders. Default is False

_download ( )
Download the raw data set from the Radiant MLHub and unzip it.

_get_structure (*test=False*)

Parameters **test** (*bool*) – Whether to load the test data set

_load_data ( )
Read the file structure from the folder, filter out the storm ID you want, and read the images that will be associated. Re-structure the data according to seq_size.
e.g. If we have 10 data [1,2,3,4,5,6,7,8,9,10], seq_size is 6, the re-structured data will be [[1,2,3,4,5,6], [2,3,4,5,6,7], [3,4,5,6,7,8], [4,5,6,7,8,9], [5,6,7,8,9,10]

storm_forcast.**set_seed** (*seed*)
Use this to set ALL the random seeds to a fixed value and take out any randomness from cuda kernels

Parameters **seed** (*int*) – Random seed to set

storm_forcast.**set_device** ( )

Parameters
- **exists** (*If a Nvidia GPU*) –
- **cuda** (*set the computing device to*) –

storm_forcast.**make_storm_dataloader** (*config*, *test*)
Split data set (training set, validation set, test set) and load it in dataloader.

Parameters
- **config** (*class*) – hyperparameters to training. Contains *root_dir*, *storm_list*, *surprise*, *seq_size*, *download* Etc.
- **test** (*bool*) – Whether to load the test set

storm_forcast.**set_up** (*config*)
Prepare the parameters for wandb

storm_forcast.**train** (*model*, *optimizer*, *criterion*, *ssim_loss*, *data_loader*)
Perform a single epoch training on the model

Parameters
- **model** – The model to train
- **optimizer** – The optimizer to use
- **criterion** – Define the method for calculating loss
- **ssim_loss** – Define the method for calculating SSIM Loss
- **data_loader** – A Dataloader containing training data

storm_forcast.**validate** (*model*, *criterion*, *ssim_loss*, *data_loader*)
Perform a single epoch training on the model

Parameters
- **model** – The model to train
- **criterion** – Define the method for calculating loss
- **ssim_loss** – Define the method for calculating SSIM Loss

- **`data_loader`** – A Dataloader containing training data

`storm_forcast.`**`train_model`** (*hyperparameters*)

According to the input of the super-parameter training model, and save the model. The run progress metrics will be displayed on https://wandb.ai/katrina-2022