Online Data Filtration in a Climate Model with Deep Learning

Daniel Galea [1] and Bryan Lawrence [2]

[1]Department of Computer Science School of Mathematical, Physical and Computer Science University of Reading, Whiteknights, Reading
[2]National Centre of Atmospheric Science, Department of Meteorology, Department of Computer Science, University of Reading, UK

Correspondence: Daniel Galea (galea.daniel18@gmail.com)

Abstract.

Machine learning, especially deep learning, has been experiencing a period of increased interest of late. It has also become popular in the meteorological community. However, integrating a deep learning model into a climate model is not trivial. Here, we detail how a deep learning model was integrated into the UK Met Office's Unified Model for the application of an online filtering method for tropical cyclones. We also address how adaptable the method is to data from different sources and horizontal resolutions. We show that the method can be used "as is" for data coming from different simulated climates. For data originating from different simulations and reanalysis, retraining of the DL model is needed if going from higher to lower resolution data, but is found to perform well otherwise.

Climate change is being investigated by the use of General Circulation Models (GCMs). These simulate the state of the Earth for many simulated years, usually producing outputs for analysis every six simulated hours. Therefore, these GCMs produce large volumes of data. Also, these simulations are being executed at increasingly fine resolutions producing even more data.

This coarse global-level data is further used in Regional Climate Models (RCMs) to better understand how a changing climate will affect a region. RCMs simulate the Earth system for only a region of the Earth. As such, RCMs need to have boundary conditions for the simulation to be possible. These are usually obtained from the coarser GCM outputs.

In this study, we present an online data filtering method to be run during a GCM simulation such that data from a region is outputted to disk only if the presence of a Tropical Cyclone (TC) is detected in the region using a deep learning model, thus reducing the data volume outputted to disk. This data is then expected to be used as boundary conditions for RCMs and further analysis. In this study, this method is applied to the Met Office's Unified Model (UM).

The method utilises a deep learning model, termed TCDetect, which was trained to detect the presence of TCs in a region of the Earth from meteorological simulation data. Details about the training process can be found in Galea et al. (2022). The performance of the model has also been verified against a state-of-the-art detection and tracking algorithm and compared to observational data in Galea and Lawrence (2022).

The two detection algorithms and observational dataset used in the analysis of results are presented in Section 1. An overview of the existing UM simulation is given in Section 2. The process by which the deep learning model, trained in Python, was included in the UM is shown in Section 3.

TCDetect inference was inserted into a model and used in a range of simulations. These different simulations, at different horizontal resolutions and for different climates constitute different sources. Section 4 discusses the ability for the method to be applied on data from different simulations and reanalysis and the expected reduction in data. Finally, the method's computational performance is detailed in Section 5.

1 Detection Algorithms and Observations

In this study, we use three different sources of data – IBTrACS, TRACK and TCDetect.

The International Best Track Archive for Climate Stewardship (IBTrACS) provides the best source of ground truth. Initially developed by the National Oceanic and Atmospheric Administration (NOAA), it combines all the best-track data for TCs from all the official Tropical Cyclone Warning Centers, the WMO Regional Specialized Meteorological Centers (RSMCs), and other sources.

TRACK is a state-of-the-art automatic detection and tracking system for different types of atmospheric disturbances with considerable use since inception in Hodges (1995), Hodges (1996) and Hodges (1999). Here, the TC tracking component is used as a gold-standard comparator against which to compare the results from the deep learning model.

We use TCDetect and TRACK applied to reanalysis data and compare them to each other and the IBTrACS dataset. Reanalysis data provides the best possible synthesized observations of meteorological variables; we choose to use the ERA-Interim product (Dee et al. (2011)). ERA-Interim utilises version CY31r2 of the European Centre for Medium-Range Weather Forecasts (ECMWF) numerical weather prediction system, the Integrated Forecasting System (IFS), together with assimilation of observations from 1979 through to 2019. The comparison is limited to the 25 months between the 1st of August 2017 until the end of August 2019 as earlier data is used in training the deep learning algorithm.

ERA-Interim data is produced at a spatial resolution of 79km, a temporal resolution of 6 hours and has 60 vertical levels up to 0.1hPa. Of the many parameters produced, only the mean-sea level pressure (MSLP), 10-metre wind speed and relative vorticity at 850hPa, 700hPa and 600hPa are used in this study.

1.1 IBTrACS

The IBTrACS dataset has information about reported storms, such as the storm centre in latitude and longitude, maximum surface wind speed, minimum sea level pressure and category.

While IBTrACS is the best available observational dataset, some inhomogeneity exists between each source as the contributing centres have differing observing systems and parametric approaches. Such observing systems can be limited in time and space, leading to the omission of systems not detected or an incomplete record of their evolution, particularly if they had limited or no human impact, or they were out of range of detection systems such as airborne missions.

1.2 TRACK

TRACK has four different stages: data preparation; segmentation; feature point detection and tracking. In the first step, TRACK treats the data so that features of interest are easier to detect. This is done with the help of spectral filtering to only keep features which have spatial scales in the range of the features of interest. With regards to tropical cyclones, the features present in wavenumbers 5 to 63 are kept in the vertical average of vorticity between the heights of 850hPa and 600hPa.

During the segmentation stage, each point in each timestep of any data used is classified as a background or an object point, depending on whether the value for the vertical average of vorticity at 850hPa, 700hPa and 600hPa is above or below the threshold of 5×10^{-6} s^{-1}. The object points are then collected into objects.

Feature point detection then allocates a feature point to each object, representing its centre. This feature point could be selected as the centroid of the object, a local extrema or using some other technique, depending on the type of data used.

Finally, the tracking stage uses the feature points generated to minimise a constrained cost function to get the smoothest possible tracks.

The complete TRACK algorithm finds a range of cyclones, some of which may be TCs. The tracks produced can be processed to identify only TC tracks. Bengtsson et al. (2007) summarise the necessary processing criteria:

- a lifetime of at least 2 days
- the initial point in the track must be in between the latitudes of 20°S and 20°N if over land or 30°S and 30°N if over an ocean
- a maximum in T63 vertically-averaged relative vorticity intensity at 850hPa over 5×10^{-6} s^{-1}
- a warm core check: a T63 vorticity maxima for each atmosphere level up to 250hPa and that the difference between the maxima at 850hPa and 250hPa is above a 5×10^{-6} s^{-1}
- the last two conditions holding for the last n timesteps, where n is a user-defined value

We refer to the set of tracks which conform to these criteria as the "truncated-TRACK" dataset or T-TRACK.

1.3 The TCDetect Deep Learning Model

The TCDetect deep learning TC detection scheme was described in Galea et al. (2022). It uses a deep learning scheme trained on ERA-Interim data, sectioned off into regions as shown in Figure 1, and utilises the mean sea-level pressure (MSLP), 10-metre wind speed, and vorticity at 850hPa, 700hPa and 600hPa fields; all coarsened to a sixteenth of ERA-Interim's native spatial resolution, resulting in an input resolution of approximately 320km.

These data were passed through a convolutional base connected to a fully-connected dense classifier trained to detect TCs labelled using IBTrACS. The system outputted a classifier value ranging between 0 and 1; a TC is inferred to be present if the value is greater than 0.5, and absent if less than or equal to 0.5.

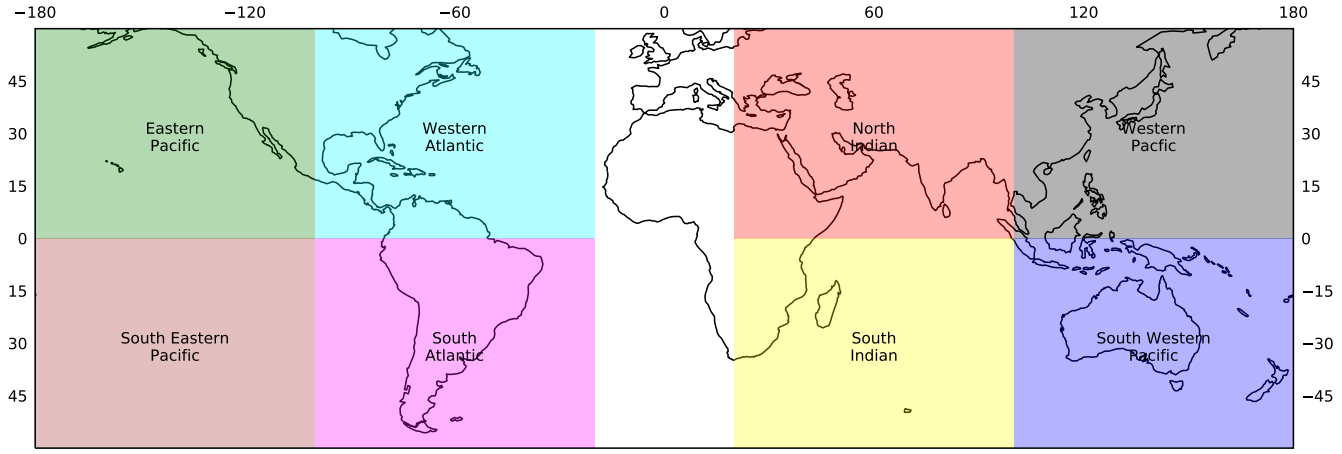


Figure 1. Figure showing how each timestep in the ERA-Interim dataset was split into 8 equal parts to create the training dataset for TCDetect.

For the identification of TCs and using 0.5 as the boundary, when trained on ERA-Interim, the TCDetect algorithm obtained a recall rate of 92% with a precision rate of 36%. In practice, this means that the while most of the actual TCs were detected, many of the TCs identified were technically false negatives (i.e. not storms of strength 1 or greater on the Saffir-Simpson scale). However, as discussed in Galea et al. (2022) and Galea and Lawrence (2022), most of these were actually meteorologically significant.

The recall rate and precision were calculated in terms of the application of the technique to ERA-Interim data, but the labels came from IBTrACS. It is reasonable then to ask "to what extent does the ability of ERA-Interim to reproduce the original storm strength and timing impact on these results"? We address this question by applying both T-TRACK and TCDetect to ERA-Interim, and comparing the results with the IBTrACS "ground truth-labels".

The TC centre is not given by TCDetect, so a way to extract it was needed. For this, the Gradient Class Activation Map technique (Grad-CAM, Selvaraju et al. (2017)) was used: for a given input, the output of the deep learning model is passed back through the model and together with gradient maximisation, produces a heatmap of the input areas used in a selected layer en route to the output. For TC location, we selected the first convolutional layer, and assumed that the TC central position in latitude and longitude is co-located with the maximum activation.

Because the heatmaps used for Grad-CAM were generated from the coarsened (320km resolution) data, the resulting TC centres were coarsely quantized and only poor quality comparisons were possible. To mitigate this effect, the Grad-CAM centres ("interim centres") were then passed through an additional refinement step to generate more accurate locations. A box with sides of 10 degrees in latitude and longitude was centred on the interim centres, and the original full resolution ERA-interim vorticity values at 850hPa, 700hPa and 600hPa were obtained and vertically averaged. The TC centre was assumed to be located at the position of the maximum in the absolute value of the averaged vorticity.

These TC centres were then used to make up TC tracks. Given that only one TC centre could be produced per region at any one timestep, a track was first defined as having TC centres which were present in consecutive timesteps in the same region. However, this produced many short ($<$ 2 days) tracks. To try and fix this, tracks for a single region which had at most 2 days (8 timesteps) of no TC being detected and a separation distance of 20 degrees (geodesic) between the final TC centre from one track and the initial TC centre of the next track were joined to make up one track. This process was carried out until no more tracks could be joined. The separation distance criterion might intuitively seem to be too wide, but as will be shown below, TCDetect had some trouble with locating TC centres, so some buffer was built into this criterion.

2 The Met Office Unified Model

TCDetect, was used during a climate simulation using the GA7.0 UM11 AMIP (Atmospheric Model Intercomparison Project) version of the UK Met Office's Unified Model (UM).

The original workflow of the UM is shown in Figure 2. It starts by initialising various variables and arrays which are required to store any calculated prognostic and diagnostic variables in memory. It then loads the starting conditions. The number of timesteps required is calculated and a loop is used to process the required timesteps.

A timestep is computed by executing various physics schemes with potential data writing interspersed between different schemes. First, local variables used in the timestep are initialised. Then a physics scheme is run and any required output variables are written. Another physics scheme is then executed and any further required prognostic variables are written. The diffusion scheme is run next with potentially more data writing of prognostic variables following. The dynamics solver is subsequently executed and any further required variables are written. Penultimately, the aerosol modelling scheme is run and is followed by the writing out of any needed variables. Finally, the incremental analysis unit (IAU) is performed. The IAU is a method by which a variable is changed across several timesteps. Changing a field at each timestep introduces noise to gravity waves, so the IAU introduces the changes across a series of timesteps to help lessen these. Also, any end of timestep diagnostics are calculated and any final required variables are written.

After all of the required timesteps are computed, the UM finishes the simulation by performing some cleanup routines before exiting.

For this study, the UM was used to produce two different simulations, each at different resolutions. The first was a standard AMIP simulation of 20.5 years at a horizontal resolution of N96 ($\approx$135km). It was run on 8 nodes of the ARCHER supercomputer, with each node having 24 processing elements. The domain was decomposed into 12 East-West processes and 8 North-South processes with 2 OpenMP threads, where OpenMP is a library which enables shared memory parallelisation. The second was a 3 year simulation at a horizontal resolution of N512 ($\approx$25 km). It was run on 104 nodes of the ARCHER2 supercomputer. While each node has 128 processing elements, the nodes were depopulated to 50 processes per node to avoid out-of-memory errors produced by the UM. The domain was decomposed into 36 East-West processes and 36 North-South processes with 4 OpenMP threads.

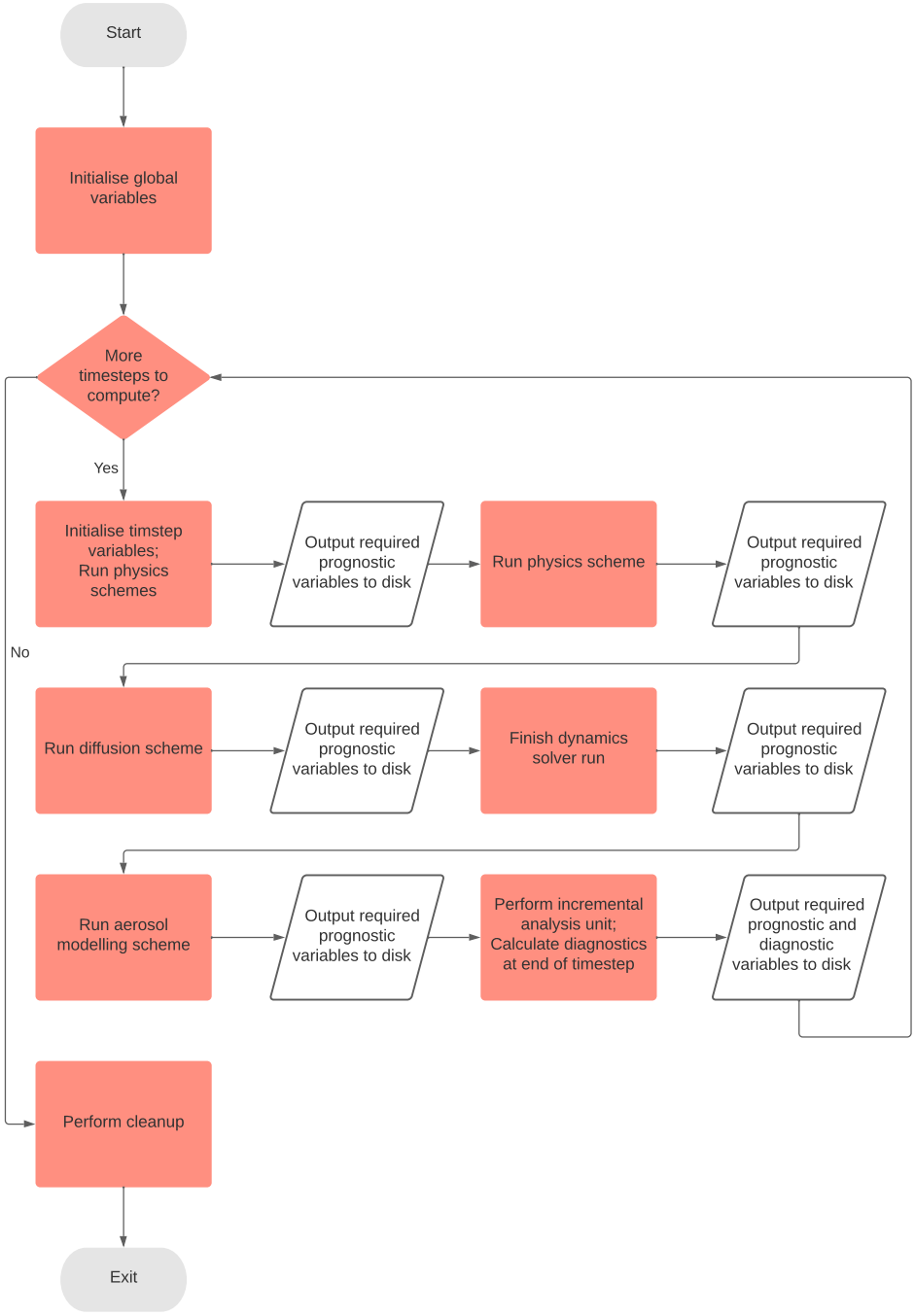


Figure 2. The original workflow of the UK Met Office's Unified Model to carry out a full climate simulation run. Objects in white are data saving steps in the UM.

Data from these runs is used to quantify how much data would not be saved to disk had the method been implemented fully. These datasets will also help understand how the method performs with one data source in the training dataset and another one for the testing dataset and also how the method performs when using data having different native horizontal resolutions.

The computational performance of the method is also calculated in shorter, seven-simulated-days runs at both resolutions.

3 Implementation

In the previous section, Figure 2 showed the original workflow of the UM. Figure 3 shows the amended workflow that uses TCDetect in the timesteps previous to those which write out analysis data. New processes are given in green.

A C++ header-only package, frugally-deep (Hermann (2020)), is utilised as an intermediate layer to be able to use the deep learning model, trained in Python, as an inference engine in the FORTRAN-based UM. An explanation on how this package works is contained in Appendix A. A FORTRAN module utilising this C++ package was written in which subroutines to load and use TCDetect as well as perform any data manipulation needed were created. The SPHEREPACK package (Adams and Swarztrauber (1999)) was also used to perform the calculations needed for spherical filtering of data.

The deep learning model, saved as an HDF5 file in Python and converted to a `frugally-deep` compatible JSON file, is loaded by the process controlling the simulation before any timesteps are processed. The spherical coefficients needed for spherical filtering at the resolution of ERA-Interim are also calculated and stored by the same process. The spherical coefficients are calculated once at the initialisation stage and are used in each timestep to calculate the spherical harmonics of the data. The required flags on which the data-writing decision are determined are also initialised and set to false.

The first set of timesteps before that which is meant to write out analysis data is calculated. At this point, the deep learning model is utilised to check whether it detects a tropical cyclone in any of the regions in the timestep just calculated.

The required data – the 10-metre wind speed, mean sea-level pressure (MSLP) and vorticity at 850hPa, 700hPa and 600hPa fields – are gathered from their host processing elements onto that controlling the simulation. All others are forced to wait until the method is run fully.

The collected data is then resized to ERA-Interim resolution using simple linear interpolation if necessary. Spherical filtering is then applied such that wavenumbers 5 to 106 are kept for the MSLP and 10-metre wind fields and wavenumbers 1 to 63 are kept for all vorticity fields.

At this point, the global image is split up into the eight regions used in the training of TCDetect, shown in Figure 1, and passed onto the deep learning model for inference making. The regions are passed to the deep learning model serially and the inferences are collected. If TCDetect infers the presence of a TC in any of the regions, the required flags for data-writing are set to true so that data from any of the required regions in the next timestep is written out to disk. At this point, all processors are released to calculate the next timestep. This continues until all timesteps are calculated and the climate model exits.

It should be noted that the method is aimed at reducing the size of certain data outputs. Any climate model produces multiple types of data which are outputted at various different intervals. The UM outputs three different types of data: dumps, means and analysis data. Dumps are when a model checkpoints its execution by writing out all of its data to disk, so that if anything

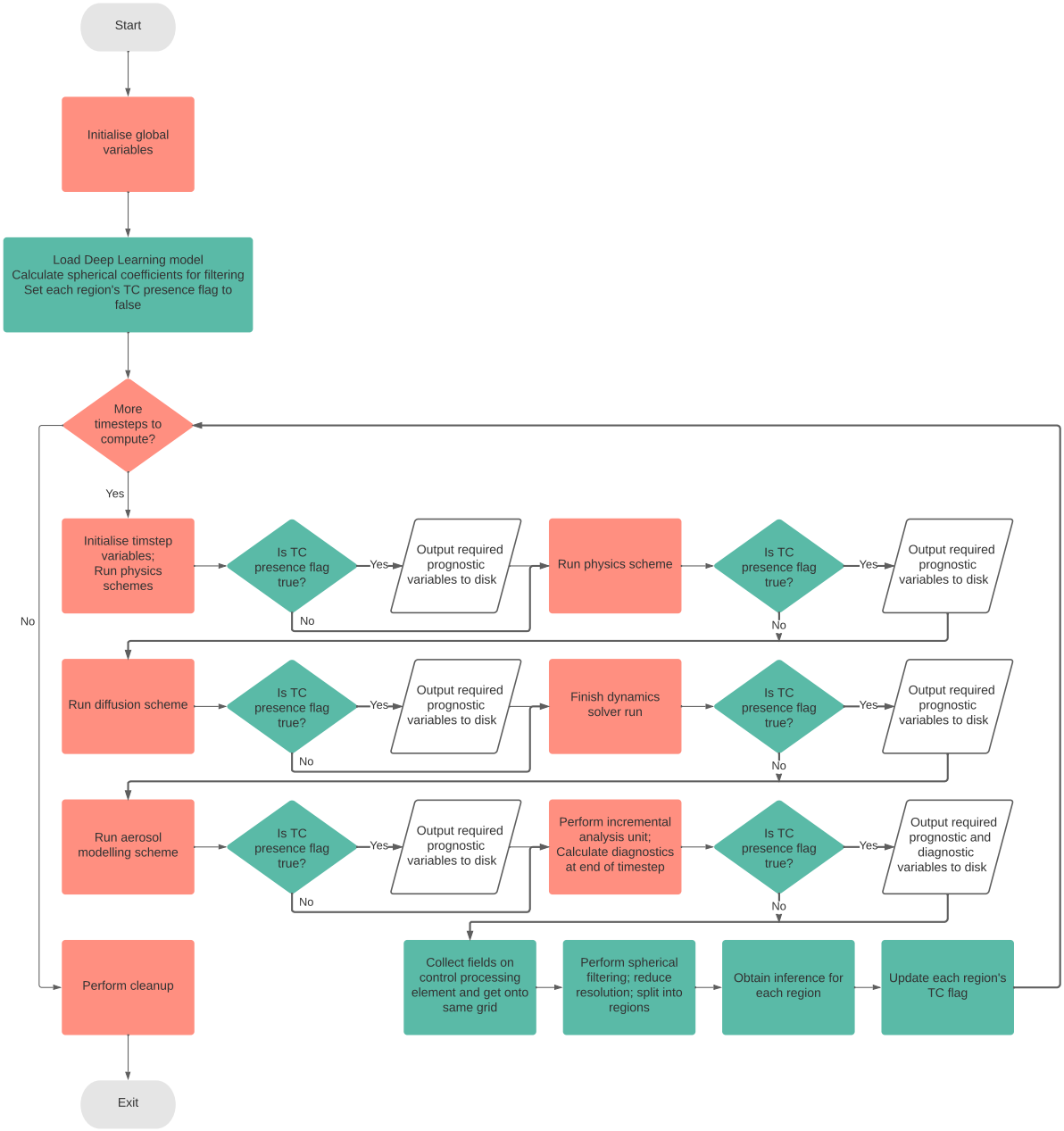


Figure 3. The UM workflow for a full climate simulation run when using the deep learning model inference to decide whether to write out data to disk. Red objects show process present in the original workflow shown in Figure 2. White Objects are data saving routines. Green object show the extra processes needed to use the deep learning model in the UM.

happens in the rest of its execution, the model can be restarted at these checkpoints instead of having to start from the beginning. The next type of data is mean data. This is when climatological means for the simulation are outputted to disk. These are used when studying trends of simulated phenomena at a coarse temporal resolution, usually monthly or seasonally. The final type of data is analysis data. This data is when the state of the simulated system is outputted for further analysis after the simulation has finished executing. Most of this data is produced for every few simulated hours and is useful when studying how a certain simulated event evolves over time. Our method is specifically aimed at this last type of data. In most cases, analysis data is outputted every six simulated hours and thus, this filtering method is run at the same temporal frequency. Therefore, in the N96 and N512 simulations used here, this method is run every 18 and 36 timesteps respectively.

(For the purposes of the following sections, the result of the presence of a TC is output to the run logs and no changes to the data writing functions were made. This was so that the data could be collected and used in further investigation.)

4 Results and Discussion

This section will present the results and discuss the adaptability of the method across different data sources. It will also discuss the effect of changing labelling techniques as well as the the amount of data reduction achieved when applying the method described above.

Six different data sources are used: the original data from ERA-Interim with labels obtained from IBTrACS, the same data but with labels obtained from T-TRACK and data from the N96 and N512 runs of the UM detailed in Section in 2. The final two datasets that were used were obtained from the CMIP6 project (Eyring et al. (2016)). The first, termed Hist1950, uses the model output from ensemble member r1i1p1f1 of a run of Natural Environment Research Council (NERC) HadGEM3-GC31-HH model from the hist-1950 set of simulations (Met Office Hadley Centre (2020b)). This is a coupled atmosphere-ocean model using historical temperature forcings. The second, termed Future, uses the model output from ensemble member r1i1p1f1 of a run of the same model from the highres-future set of experiments (Met Office Hadley Centre (2020a)), where the modelled climate is forced to be close to the CMIP5 RCP8.5 scenario.

Our ERA-Interim testing period is 25 months, while it is 36 months and 12 months respectively for the UM N96 and N512 simulations. The Hist1950 dataset is 13 years long and the Future dataset is 6 years long. These represent around 20% of the available data, the other 80% of which was used to train the different variants of TCDetect. Table 1 summarises the different datasets used in this section and includes their horizontal resolutions and length in simulated years.

As the deep learning model underpins the whole method, we will be discussing performance of the method in terms of the deep learning model's recall rate.

We should note that TCDetect and the method as a whole will not obtain a perfect recall rate. As expected, no detection algorithm can detect all of the TCs present. For example, TRACK detects 97% of observed TCs. From the Venn diagram of Figure 4a, showing the number of events reported by IBTrACS and detected by T-TRACK and TCDetect applied to ERA-Interim data, T-TRACK correctly classified 87% of the regions shown in Figure 1 in which IBTrACS observed a TC. Also, as discussed in Galea and Lawrence (2022), other detection methods were shown to need to have their thresholds tuned to the

Dataset	Labelling Method	Horizontal Resolution	Length of Testing Dataset / simulated years
ERA-Interim	IBTrACS	∼79 km	2
ERA-Interim	T-TRACK	∼79 km	2
UM N96	T-TRACK	∼135 km	3
UM N512	T-TRACK	∼25 km	1
Hist1950	T-TRACK	∼25 km	13
Future	T-TRACK	∼25 km	6

Table 1. Datasets used for exploring the adaptability of the method across different data.

datasets used, inferring that even they cannot detect all the TCs present. Also, IBTrACS itself is not perfect as discussed in Section 1.1. With this in mind, the performance of our method is discussed below.

4.1 Method Adaptability

An important aspect of this method being presented it how adaptable and reliable it is to different data sources. The following attempts to understand this.

TCDetect, as a deep learning model, is made up of two parts - its architecture, i.e. how each layer is arranged to make up the final model; and its weights, which are the values used to make an inference. To quantify the ability of the model to be applied across different data sources, we keep the same underlying architecture as TCDetect but retrain the model on each data source to come up with different sets of weights to make up new variants of TCDetect. TCDetect is the original model as trained on ERA-Interim data with labels obtained from IBTrACS. TCDetect-TRACK is the model when trained on the same data but using labels derived from T-TRACK. TCDetect-N96 and TCDetect-N512 refer to models trained on N96 and N512 UM data respectively, with labels acquired from T-TRACK. Finally, TCDetect-CC and TCDetect-FC are models trained on the Hist1950 and Future datasets from the CMIP6 HighResMIP experiment.

The recall rate for each of these models for each of the datasets available is tabulated in Table 2. Table 3 shows the mean and standard deviation of the monthly recall rate for each model as tested on each dataset.

TCDetect obtained a recall rate of 92% on the ERA-Interim dataset which was labelled using IBTrACS. However, in the following work, we will be utilising data from simulations using the UM. For this, IBTrACS is not available. Therefore, the first step was to retrain the model using T-TRACK labels.

The version of TCDetect trained with labelling obtained from T-TRACK is now termed TCDetect-TRACK. As seen from Table 2, this obtained a recall rate of 85% when it is tested on data with IBTrACS labels, compared to the 87% obtained from T-TRACK, and 59% when tested on data with T-TRACK labels. T-TRACK is known to contain some systems which are not TCs, so it is not surprising that the performance of the model decreased. This could be due to the different types of systems present in the labelling system that may be forcing the deep learning model to try to learn two different patterns. We attempt to see if this is the case by splitting the cases in the test dataset by the highest category of any TCs present as given by IBTrACS, with

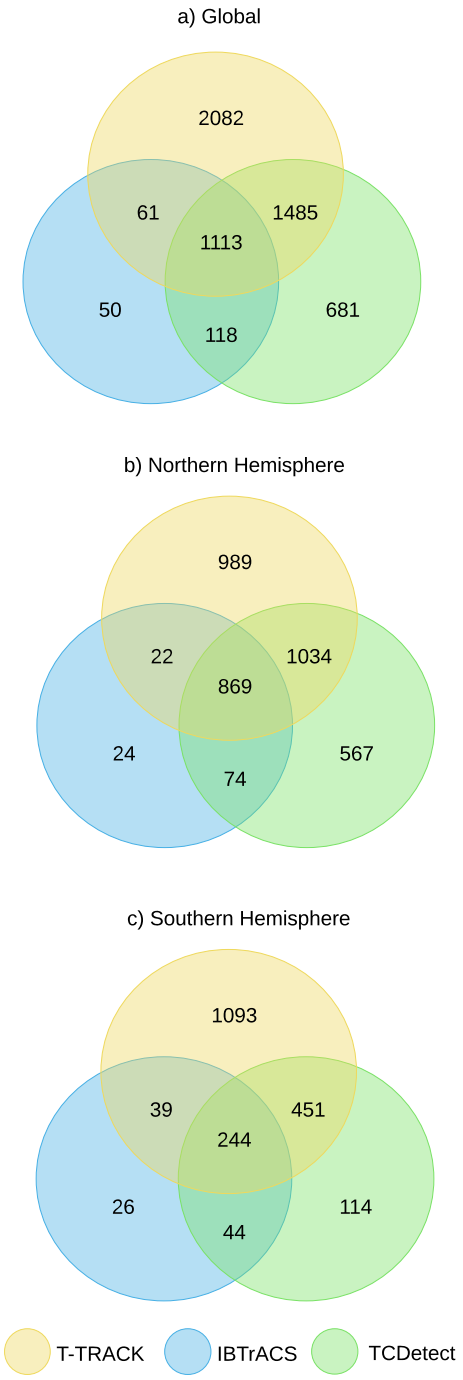


Figure 4. Events reported by observations (IBTrACS) and detected by T-TRACK and TCDetect applied to ERA-Interim data for (a) the whole globe, (b) the Northern Hemisphere and (c) the Southern Hemisphere.

		Detection Method					
Data Source	**Label Source**	**TCDetect**	**TCDetect -TRACK**	**TCDetect -N96**	**TCDetect -N512**	**TCDetect -CC**	**TCDetect -FC**
ERA-Interim	IBTrACS	92%	85%	97%	91%	93%	89%
ERA-Interim	T-TRACK	59%	62%	76%	70%	73%	62%
UM N96	T-TRACK	34%	41%	78%	50%	62%	44%
UM N512	T-TRACK	58%	58%	71%	72%	77%	60%
Hist1950	T-TRACK	60%	58%	73%	73%	77%	64%
Future	T-TRACK	62%	62%	74%	74%	79%	66%

Table 2. Recall rates when each variant of TCDetect was tested on combinations of different data and labelling sources.

		Detection Method					
Data Source	**Label Source**	**TCDetect**	**TCDetect -TRACK**	**TCDetect -N96**	**TCDetect -N512**	**TCDetect -CC**	**TCDetect -FC**
ERA-Interim	IBTrACS	$\bar{x}: 88\%$ $\sigma: 13\%$	$\bar{x}: 84\%$ $\sigma: 17\%$	$\bar{x}: 95\%$ $\sigma: 9\%$	$\bar{x}: 87\%$ $\sigma: 19\%$	$\bar{x}: 92\%$ $\sigma: 18\%$	$\bar{x}: 87\%$ $\sigma: 16\%$
ERA-Interim	T-TRACK	$\bar{x}: 51\%$ $\sigma: 24\%$	$\bar{x}: 59\%$ $\sigma: 16\%$	$\bar{x}: 71\%$ $\sigma: 17\%$	$\bar{x}: 64\%$ $\sigma: 19\%$	$\bar{x}: 69\%$ $\sigma: 16\%$	$\bar{x}: 56\%$ $\sigma: 20\%$
UM N96	T-TRACK	$\bar{x}: 33\%$ $\sigma: 16\%$	$\bar{x}: 38\%$ $\sigma: 15\%$	$\bar{x}: 76\%$ $\sigma: 13\%$	$\bar{x}: 49\%$ $\sigma: 14\%$	$\bar{x}: 59\%$ $\sigma: 15\%$	$\bar{x}: 42\%$ $\sigma: 15\%$
UM N512	T-TRACK	$\bar{x}: 58\%$ $\sigma: 14\%$	$\bar{x}: 59\%$ $\sigma: 13\%$	$\bar{x}: 71\%$ $\sigma: 13\%$	$\bar{x}: 72\%$ $\sigma: 11\%$	$\bar{x}: 77\%$ $\sigma: 9\%$	$\bar{x}: 60\%$ $\sigma: 13\%$
Hist1950	T-TRACK	$\bar{x}: 57\%$ $\sigma: 13\%$	$\bar{x}: 72\%$ $\sigma: 11\%$	$\bar{x}: 72\%$ $\sigma: 13\%$	$\bar{x}: 77\%$ $\sigma: 10\%$	$\bar{x}: 63\%$ $\sigma: 14\%$	$\bar{x}: 60\%$ $\sigma: 17\%$
Future	T-TRACK	$\bar{x}: 60\%$ $\sigma: 17\%$	$\bar{x}: 60\%$ $\sigma: 15\%$	$\bar{x}: 73\%$ $\sigma: 14\%$	$\bar{x}: 73\%$ $\sigma: 14\%$	$\bar{x}: 78\%$ $\sigma: 12\%$	$\bar{x}: 65\%$ $\sigma: 16\%$

Table 3. Monthly mean and standard deviation recall rates when each variant of TCDetect was tested on each different data source.

	TCDetect		TCDetect-TRACK	
Class	**Positive Inference**	**Negative Inference**	**Positive Inference**	**Negative Inference**
No Meteorological System	506	19253	1393	18366
Unknown	2	30	5	27
Post-tropical systems	18	47	20	45
Disturbances	165	337	111	391
Subtropical systems	32	51	40	43
Tropical Depressions	348	625	320	653
Tropical Storms	1095	501	1049	547
Category 1 TCs	426	58	395	89
Category 2 TCs	281	26	252	55
Category 3 TCs	243	15	224	34
Category 4 TCs	212	12	208	16
Category 5 TCs	69	0	68	1

Table 4. Split of cases by storm type (rows) as given by IBTrACS given a positive inference (second/fourth columns) or a negative inference (third/fifth columns) by TCDetect and TCDetect-TRACK. For example, of the 19759 cases which had no meteorological system, TCDetect classified 506 as having a TC present (i.e. false positives). Also of the 484 cases in which a Category 1 TC was the strongest system present, 426 were classified as having a TC (i.e. true positives).

results shown in Table 4. When compared to TCDetect's results, TCDetect-TRACK produces more than double the true false positives (positive inference; no meteorological system) than TCDetect, while its recall of category-strength storms decreases.

Now that we have a model trained with T-TRACK labelling, we can use it to determine how adaptable the method is to data from different sources.

Starting with the original ERA-Interim data, the original model, TCDetect, obtained a recall rate of 92% while TCDetect-TRACK obtained a recall rate of 85%. On the other hand, most of those models trained on UM data obtained recall rates close to or better than TCDetect, with the notable exception being TCDetect-FC, the model trained on a future climate. These numbers start to indicate that differences are to be expected in the model's performance when applying it to data from different sources and different climates.

When considering ERA-Interim data with T-TRACK labels, a clear stratification can be noted. Those models trained on UM data, with the same exception as before in TCDetect-FC, obtain a higher recall rate than those trained on ERA-Interim data, although all recall rates are more 20% lower in each case. This is, in fact seen when testing with the rest of the available datasets. This points to the possibility that models trained on UM data continue to perform well when used on ERA-Interim data, but not vice-versa.

These values are reflected in the mean and standard deviation of the monthly recall rate. The standard deviations reported when the models are tested on UM data are smaller than when tested on ERA-Interim data. This could show that the models

perform more consistently across all timesteps, although some caution needs to be taken due to the smaller testing dataset in the latter case. It should also be noted that the standard deviation values are still large, showing that the differences in the mean monthly recall may not be representative, however Figure 5, which shows the monthly recall rate from the TCDetect, TCDetect-TRACK, TCDetect-N96 and TCDetect-N512 models across the UM N96 testing dataset, shows that these differences are representative as TCDetect (green line) and TCDetect-TRACK (red line) are consistently the worst performing of all the models considered.

Furthermore, the model trained on the dataset considered obtains the best or second best recall rate. This is expected as from the discussion in Galea and Lawrence (2022), classical algorithms performed best when they were tuned to the dataset being considered. There is no reason to expect a deep learning algorithm to differ from this behaviour. Once again, these findings are consistent with the numbers reported for the mean daily recall.

It was also noted that all models except one, TCDetect-N96, performed better on UM N512 data than UM N96 data. For TCDetect-N96, performance on N512 data was worse. These findings could point to the model being applicable to higher resolutions, albeit with a slight drop in performance, but not vice-versa. To be able to go from low resolution to a higher resolution, a simple retraining of the model would capture some of the original model's performance back, but not all of it.

The different variants of TCDetect were tested on the two datasets intended to infer the adaptability of the method to different climates, Hist1950 and Future. The recall rates across the two datasets as well as the mean monthly recall rates and their standard deviations were very close. This shows that the method is found to work in future climates if an acceptable recall rate is obtained on data from the current climate, at least for detecting TCs.

Finally, we touch on the reliability of the method, i.e. the ability of the method to produce similar performance across the dataset. To check this, we show the monthly recall rate from the TCDetect, TCDetect-TRACK, TCDetect-N96 and TCDetect-N512 models across the UM N96 testing dataset (Figure 5). This shows a high variability with one clear drop in performance across all models around May 2006. This occurs when there are the fewest TCs, and most of which come from the South Atlantic and South Eastern Pacific regions (Figure 6). These are the two regions with which TCDetect struggles the most due to a lack of observed TCs from IBTrACS in the region. From Table 3, this variability is highest when testing on data from ERA-Interim with labels from IBTrACS. This decreases slightly when using T-TRACK labels and decreases further when using UM data.

4.2 The Effect of Different Labelling

One aspect which might be important to understand is that of the effect of changing the labelling source. In the previous section, we noted that IBTrACS, the labelling source of TCDetect, is not available for UM simulations. Hence, the labelling source was switched to T-TRACK. While the change was done due to necessity, it is important to investigate how that effected the performance of the deep learning model.

To do this, the analysis of Galea and Lawrence (2022) using TCDetect was repeated with TCDetect-TRACK and compared to that previously done.

Figure 5. Monthly ecall rate from the TCDetect, TCDetect-TRACK, TCDetect-N96, TCDetect-N512, TCDetect-CC and TCDetect-FC models across the UM N96 testing dataset.

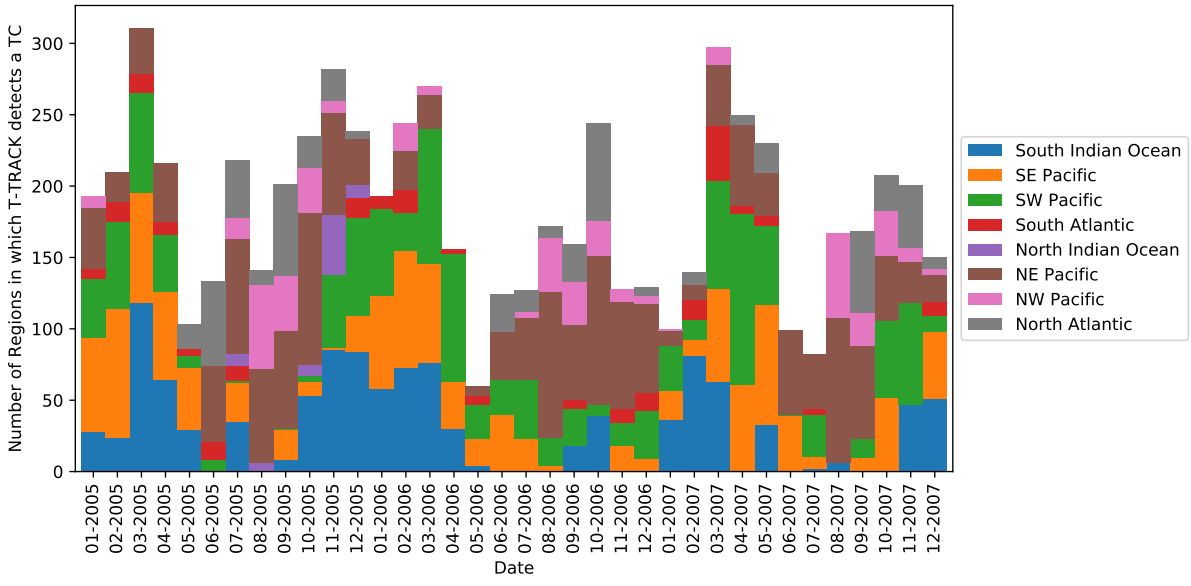


Figure 6. Monthly number of regions having a TC detected by T-TRACK across the UM N96 testing dataset.

Firstly, Figure 7 shows how these two models compare to each other in terms of the regions in which a TC was detected by the deep learning models, T-TRACK and observed in IBTrACS. Figure 7i repeats Figure 4 for ease of comparison.

It could be noted that TCDetect-TRACK detects the presence of a TC in more regions than TCDetect. This is expected as T-TRACK detects more TCs than IBTrACS, so the model trained with T-TRACK labels is expected to mirror the labelling more closely. However, the proportion of the regions in which a TC has only been detected by the deep learning model grows (20% by TCDetect vs 28% by TCDetect-TRACK). This increase points to a larger number of false positives produced, as shown in Table 4.

Despite the increase in the total number of regions in which TCDetect-TRACK detected a TC, the number in the Northern Hemisphere decrease slightly while the number in the Southern Hemisphere increase considerably. The latter aligns to the knowledge that IBTrACS underestimates the TC count in the Southern Hemisphere. This is mirrored in the number of matches between the deep learning model and T-TRACK in both hemispheres.

TCDetect-TRACK detected more of the TCs also detected by T-TRACK than TCDetect did (intersection of yellow and green areas). This is expected due to the labelling used when training TCDetect-TRACK as this model is expected to perform closer to the new labelling. However, fewer regions in which IBTrACS observed a TC were detected by TCDetect-TRACK (intersection of blue and green regions). This, and the knowledge that T-TRACK produces tracks which are too long, i.e. containing systems which are not hurricane-strength TCs, could be pointing to a possibility that the labelling is making the model learn different patterns for a positive inference during training.

This is more evident in Figures 10 and 11, which show the composite TCs of all cases by Venn diagram region (7i) when using TCDetect-TRACK. When compared to the similar plots when using TCDetect in Figures 8 and 9, it is seen that only the row showing the composite TC of the cases detected only by TCDetect-TRACK changes considerably. This composite is now much more similar to the expected structure of a TC, albeit one which is not at hurricane-strength due to the width of the MSLP field and lack of a clear center in the wind speed field. This points to TCDetect-TRACK detecting systems which are valid meteorological systems but do not have the strength of a TC.

Given that T-TRACK produces overly long tracks, we would expect TCDetect-TRACK to produce TC centres at seemingly wrong positions as we would expect more TC centres to reach the extra-tropics. Figure 12 shows just this. There are more TC centres in the wrong place, i.e. over land, and more points at the extra-tropics, especially in the Northern Hemisphere. Also, more TC centres are generated over the Southern Hemisphere, as expected from the increase in regions in which TCDetect-TRACK detected a TC in the Southern Hemisphere. The latter behaviour also mirrors the T-TRACK labelling more closely - expected due to the change in labelling.

One complicating point is that only one TC centre can be produced in each region in which the deep learning model detects the presence of a TC. This is due to how the Grad-CAM technique (Selvaraju et al. (2017)) was used in this study. It was assumed that the the activation point which produced the highest value from Grad-CAM was to be assigned as a TC centre. This was as having Grad-CAM produce multiple TC centres would need a threshold above which any points would be a TC and thus this would introduce subjectivity. However, with the change of labelling, TCDetect-TRACK effectively changes its detection criteria to detect most tropical systems. This change would make the assumption to not necessarily hold as many

(i) TCDetect (ii) TCDetect-TRACK

Figure 7. Events reported by observations (IBTrACS) and detected by T-TRACK and (i) TCDetect and (ii) TCDetect-TRACK, applied to ERA-Interim data for (a) the whole globe, (b) the Northern Hemisphere and (c) the Southern Hemisphere.

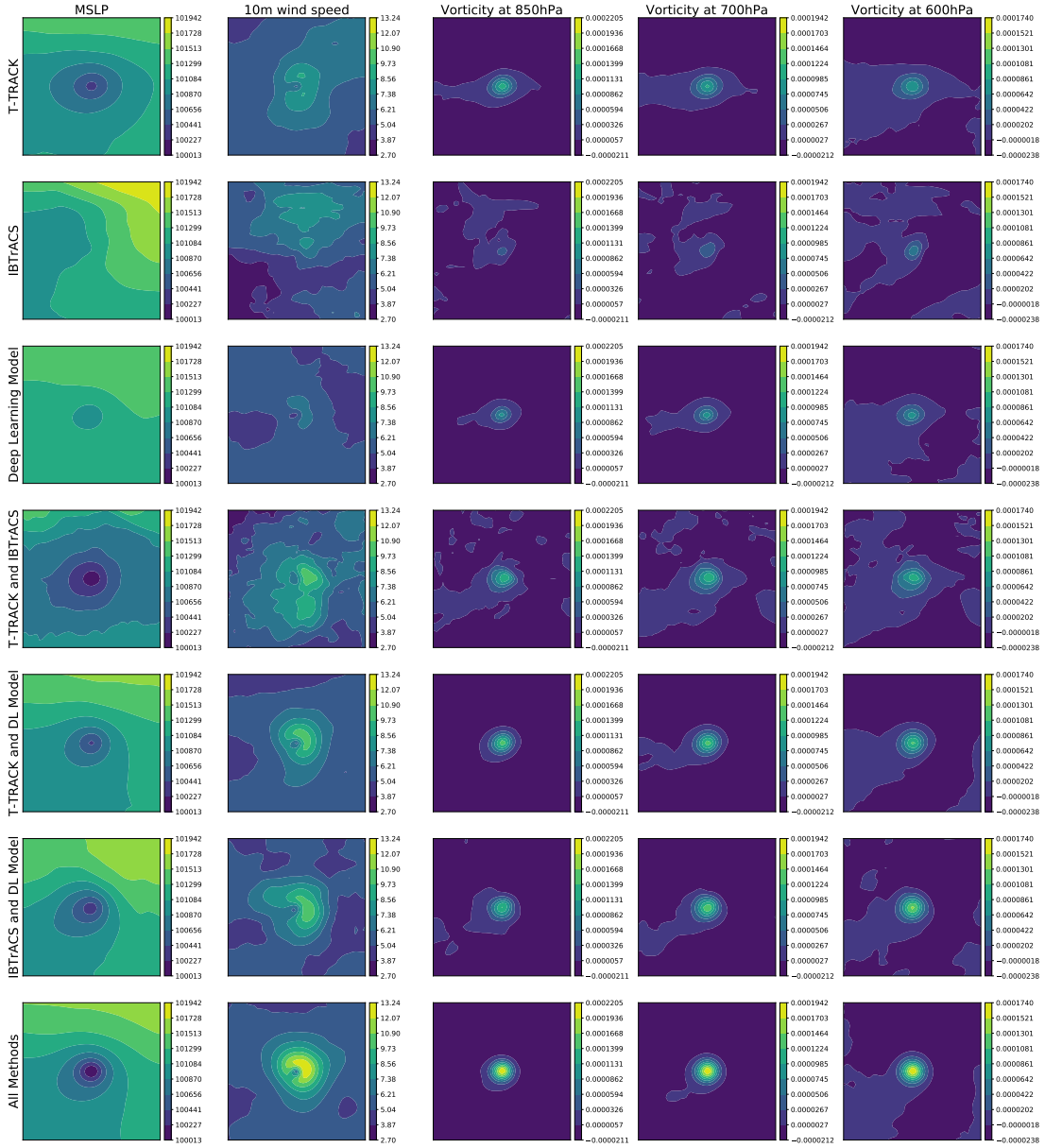


Figure 8. Composite view of Northern Hemisphere events by detection algorithm (T-TRACK or TCDetect) or observations which pick up the TC. Total number of cases used to produce each composite can be obtained from 4. Columns correspond to the variables used: MSLP (first column), 10-metre wind speed (second column), vorticity at 850hPa (third column), vorticity at 700hPa (fourth column) and vorticity at 600hPa (fifth column).

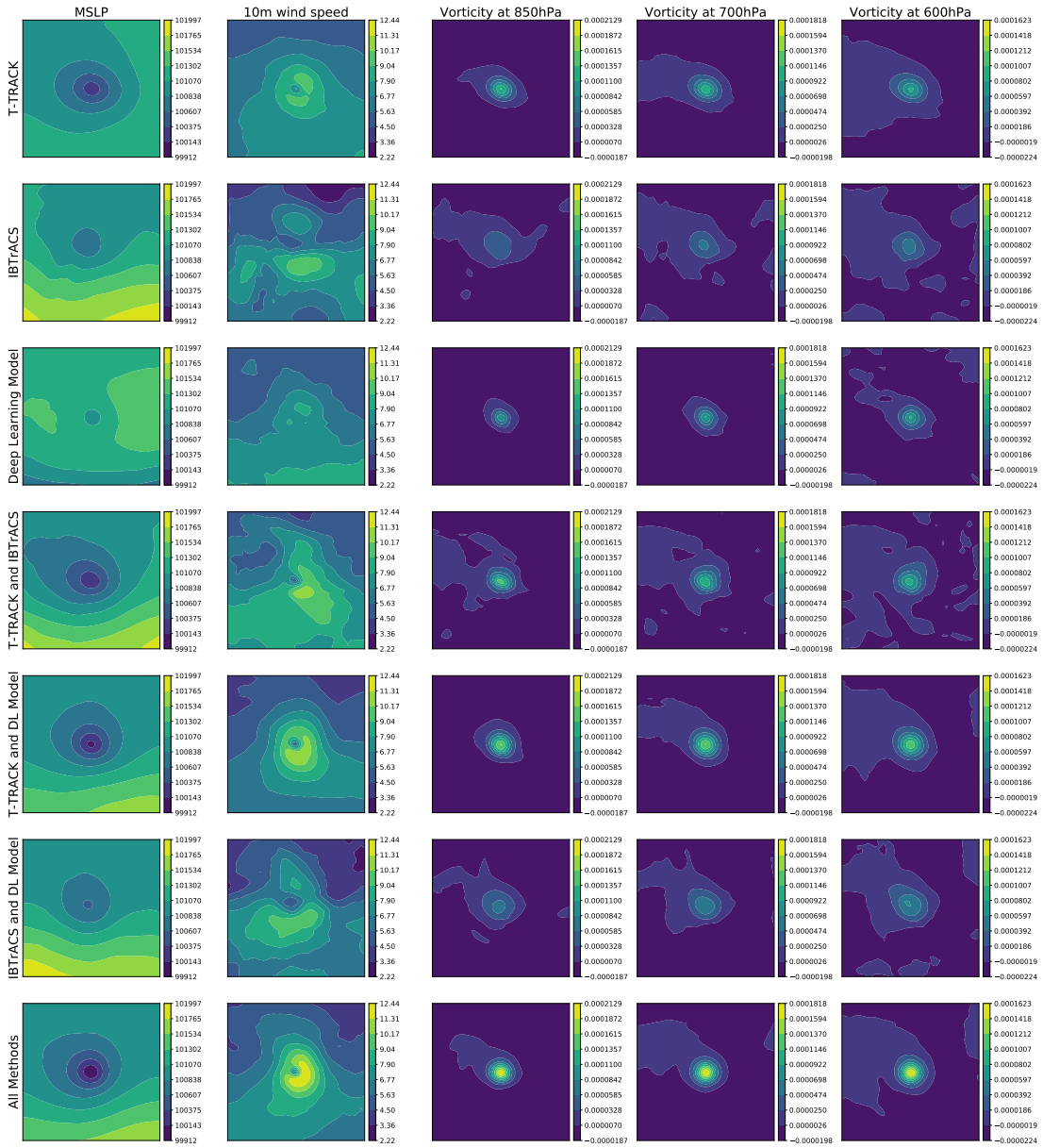


Figure 9. Composite view of the Southern Hemisphere cases (rows and columns as described in Figure 8) - but the sign of vorticity has been reversed for ease of comparison).

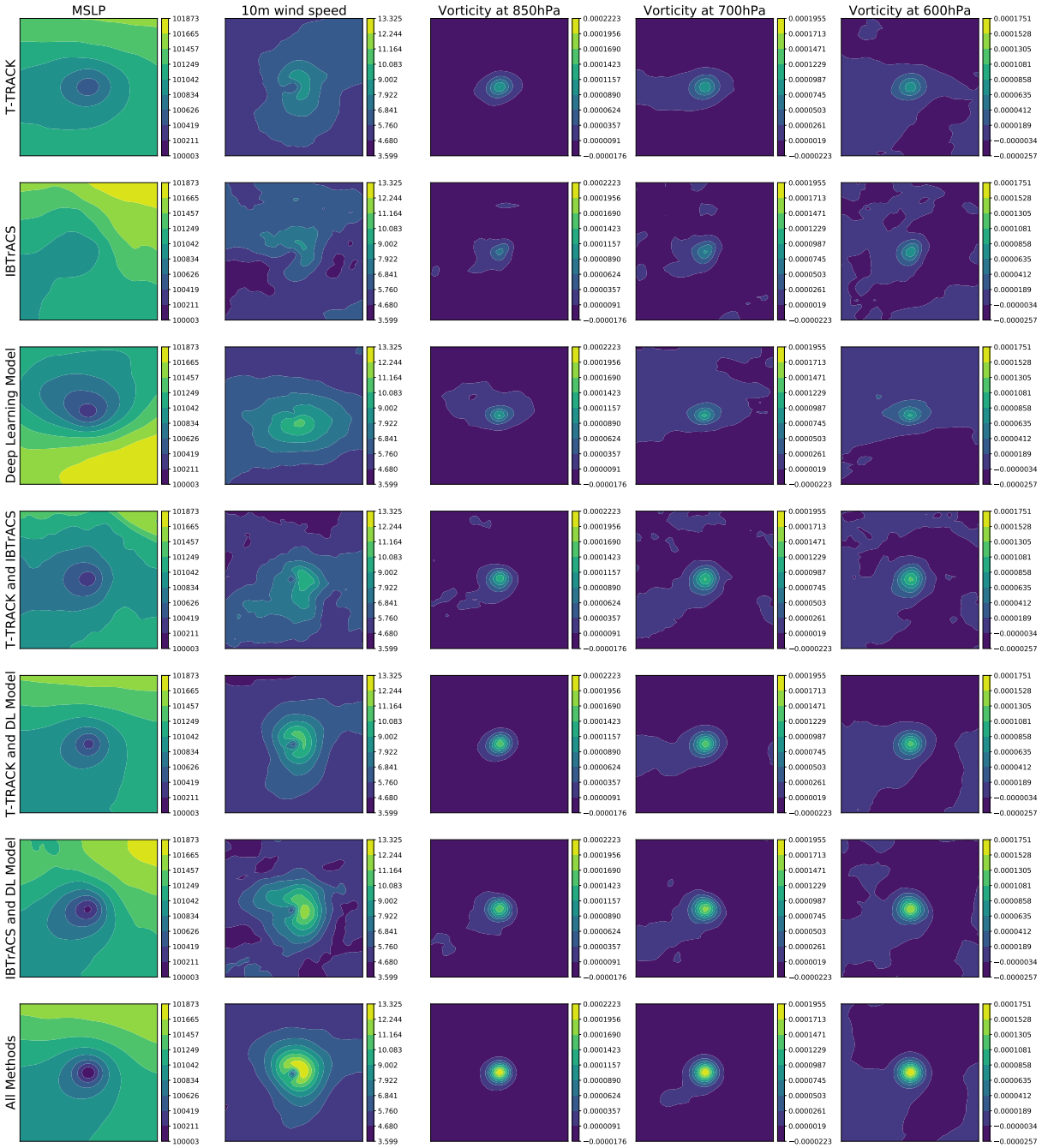


Figure 10. Composite view of Northern Hemisphere events by detection algorithm (T-TRACK or TCDetect-TRACK) or observations which pick up the TC. Total number of cases used to produce each composite can be obtained from 7ii. Columns correspond to the variables used: MSLP (first column), 10-metre wind speed (second column), vorticity at 850hPa (third column), vorticity at 700hPa (fourth column) and vorticity at 600hPa (fifth column).

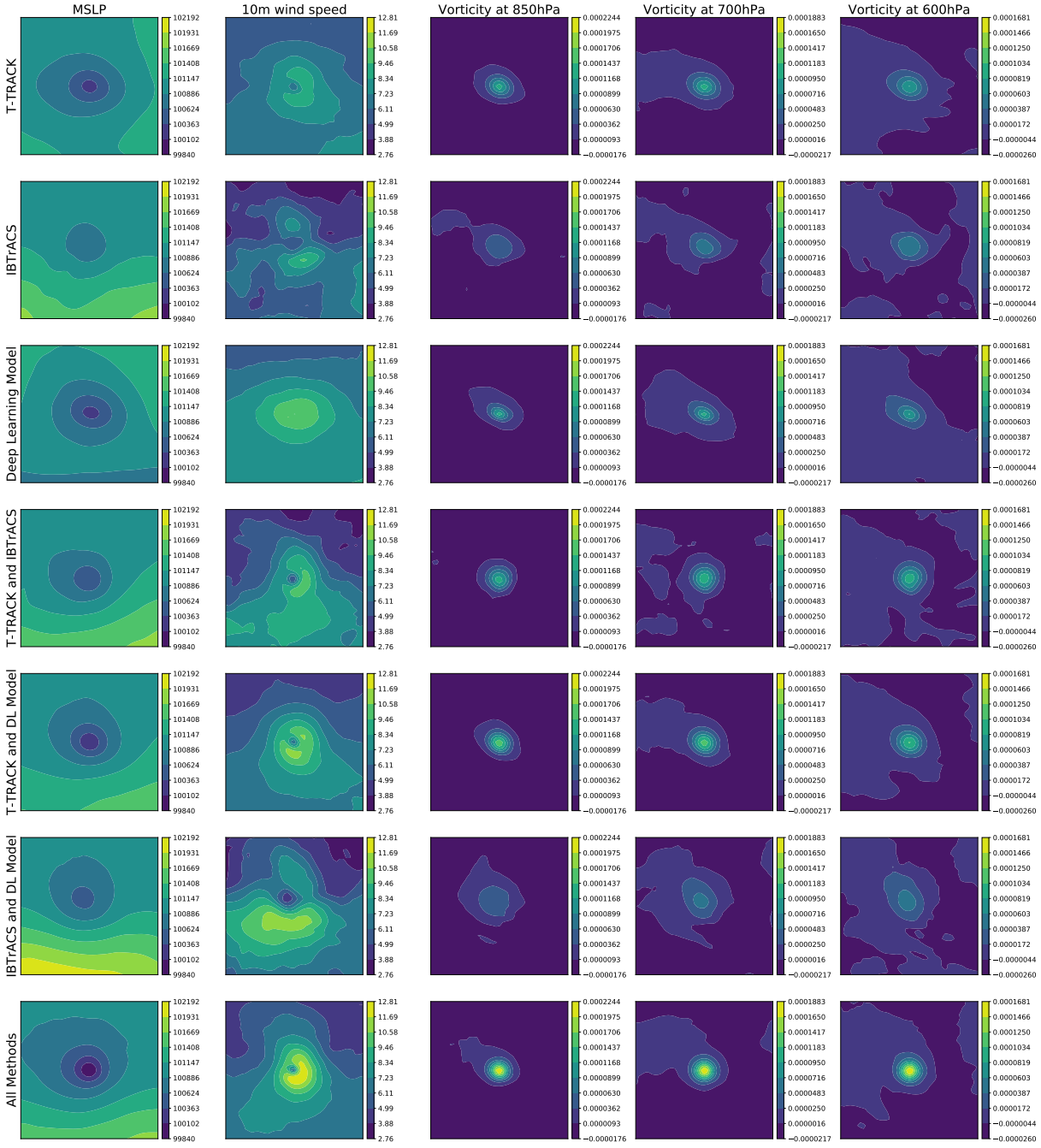


Figure 11. Composite view of the Southern Hemisphere cases (rows and columns as described in Figure 10) - but the sign of vorticity has been reversed for ease of comparison).

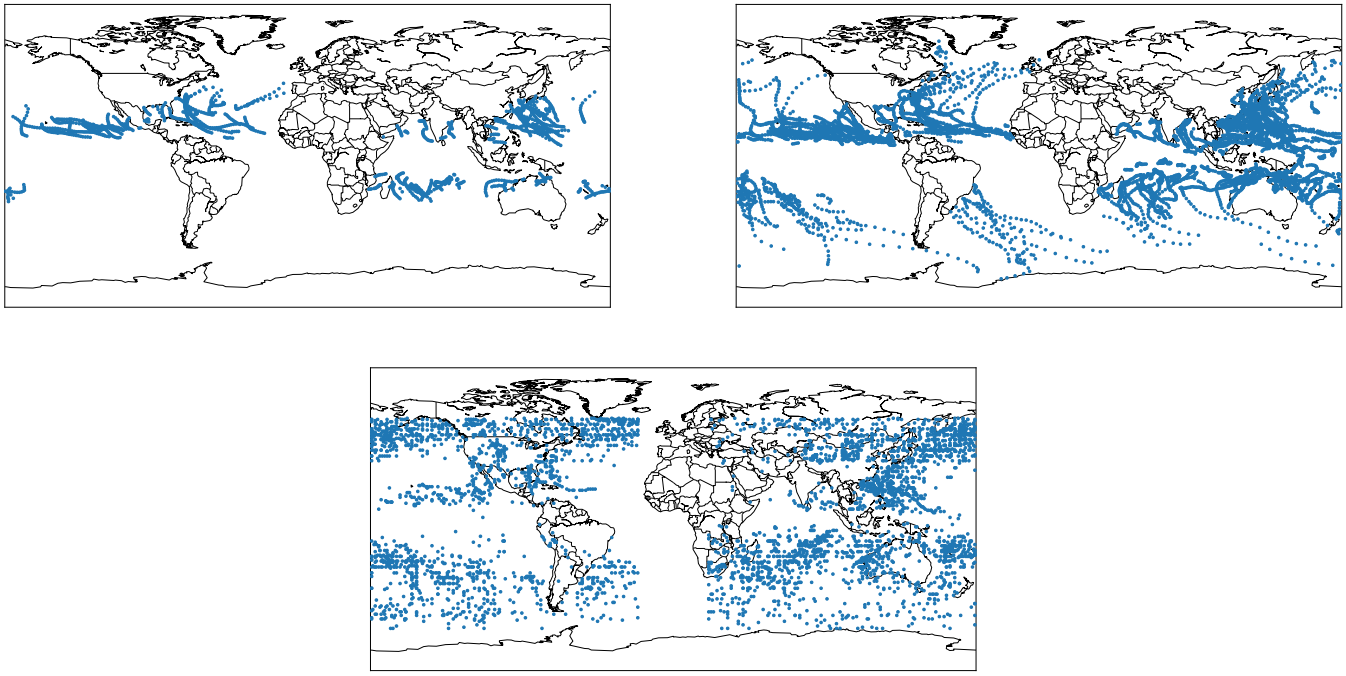


Figure 12. Position of each Tropical Cyclone event center as given by IBTrACS (top-left); T-TRACK (top-right) and the TCDetect-TRACK (bottom).

cases exist where multiple tropical systems are present in the region, with Grad-CAM possibly assigning the highest activation to a region which is still a tropical system, but not a hurricane-strength TC.

This could be seen in the specific example shown in Figure 13. This shows a region in which two tropical systems can easily be identified in the region's data (top row) as areas of low values in the MSLP field and matching areas of high values in the other fields in the middle of the region. The area in the top half of the region is a tropical storm and the one in the bottom half is a Category 5 TC, however, both are present in the T-TRACK dataset. The first panel on the bottom row shows the output of Grad-CAM when using TCDetect with this region. The area of the highest activation (yellow area), and the TC position given by Grad-CAM (red dot) are loosely near the Category 5 TC. However, when doing this with TCDetect-TRACK in the second panel, the area of highest activation and the TC position given by Grad-CAM is nearer the top of the region, much closer to the tropical storm. Thus, a TC centre which is too Northerly is produced. To make sure that TCDetect-TRACK could still detect the Category 5 TC, the top half of the region was set to the mean of the field, effectively blanking out that part of the field, and Grad-CAM now puts the area of highest activation closer and the TC position generated close to the Category 5 TC as expected, shown the the third panel. This shows that TCDetect-TRACK is performing as expected, but the use of Grad-CAM is problematic.

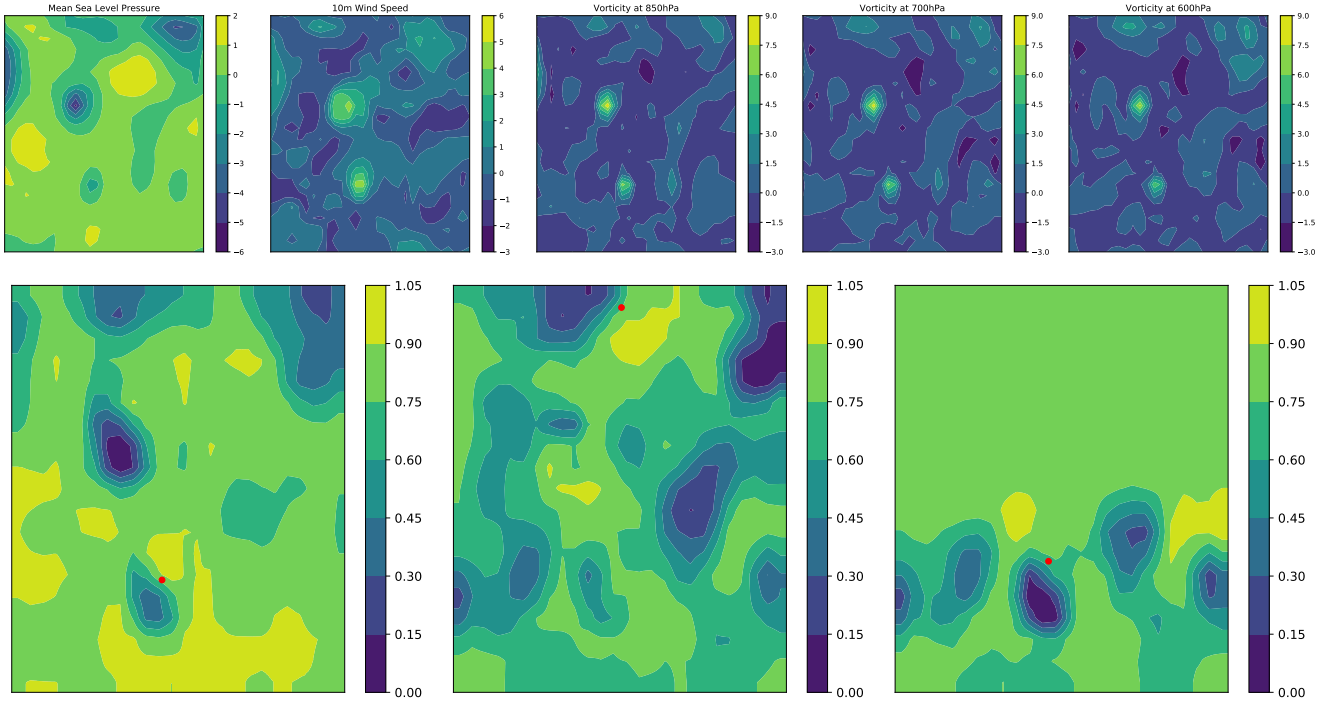


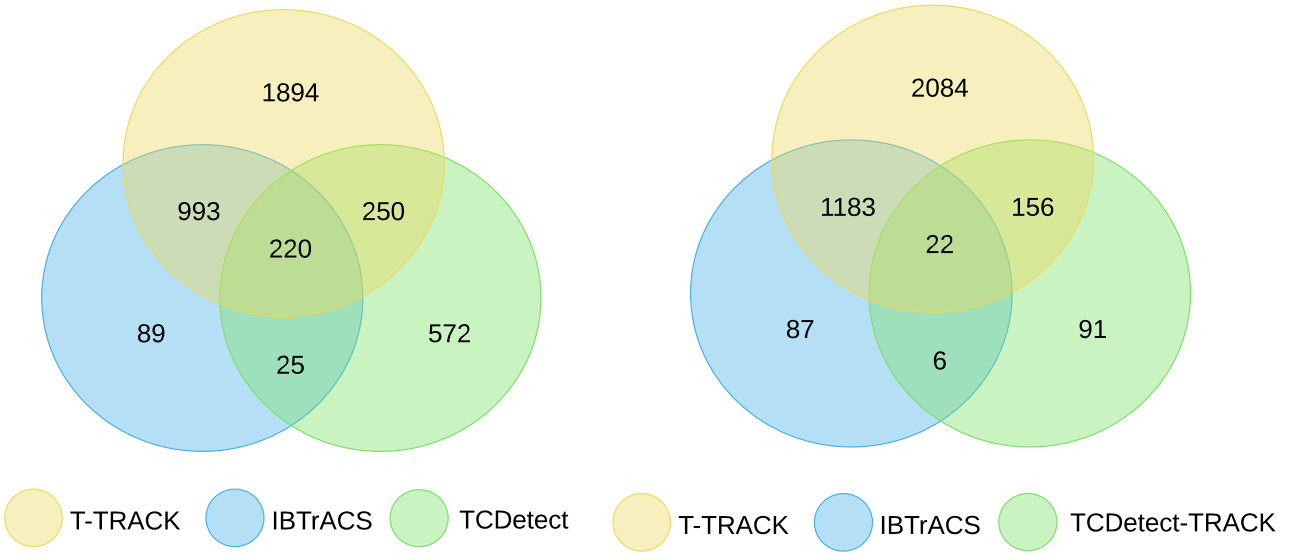
Figure 13. Example of region having a multiple tropical systems present, only one of which is a hurricane-strength TC (top row). Grad-CAM outputs (bottom row) for the example from TCDetect (left panel), TCDetect-TRACK (middle panel) and TCDetect-TRACK with top half of the example taken out (right panel). TC positions as given by Grad-CAM shown by red dots.

These sub-optimal TC positions from TCDetect-TRACK should mean that while it may be detecting a tropical system in a region, matching it with TCs from the other two sources becomes harder. If this is done in the same manner as in Galea and Lawrence (2022) using the constraints similar to those used by Hodges et al. (2017):

- the mean separation distance between all overlapping points between tracks is less than 5^{o} (geodesic)
- the tracks need to overlap for at least 10% of the base track's lifetime
- the track with the least mean separation distance is chosen if multiple matching tracks exist

matching tracks becomes harder, mostly due to the constraint requiring the mean separation distance between the overlapping parts of two tracks to be within 5 degrees (geodesic).

The Venn diagrams of matching TCs after the constraints detailed previously are applied using TCDetect and TCDetect-TRACK are shown in Figures 14i and 14ii. This shows a large reduction in the number of regions detected by TCDetect-TRACK which were matched to a TC from either T-TRACK or IBTrACS or both when compared to what was reported using TCDetect. This reduction is mostly down to the TC centres produced by Grad-CAM being in the wrong place. Despite this, unsurprisingly the largest number of matches is between TCDetect-TRACK and T-TRACK.



(i) TCDetect

(ii) TCDetect-TRACK

Figure 14. Events detected by TRACK, (i) TCDetect and (ii) TCDetect-TRACK and/or reported by IBTrACS which fall on matching tracks, defined by applying constraints similar to those of Hodges et al. (2017).

The analysis above shows that the effect of changing the labels from those derived from IBTrACS to those derived from T-TRACK is not insignificant, mainly due to these labels effectively changing what the models are being trained to do. As such, TCDetect-TRACK is not expected to perform as well as TCDetect, as the architecture was developed with IBTrACS labelling. Despite this, it is still skillful at detecting hurricane strength TCs, as they are a subset of tropical cyclones.

4.3 Region Filtering

The method presented above was initially designed to act as a filtering mechanism to be used during a climate model run. As such, it is important to measure the amount of data filtered.

For this and the following sections, data from the UM runs detailed in Section 2 was processed to get it in a form that can be used to obtain the deep learning model's inference, with the process shown in Figure 15.

Data output from UM runs is formatted as .pp files with timesteps grouped monthly. These are then converted to netCDF files and the required fields - MSLP, 10-metre wind speed, and vorticity at 850hPA, 700hPa and 600hPa - are collected into their own netCDF files. TRACK and T-TRACK are run using these and output detected TCs along with a timestamp in a text file.

The netCDF files are also processed for use by the deep learning model, by resizing each timestep to the resolution used by ERA-Interim (≈79km). After, any spherical filtering is performed and each timestep's data is split into the eight regions shown in Figure 1. The data from each region is saved to disk as a .npz file, with the presence of a TC according to T-TRACK

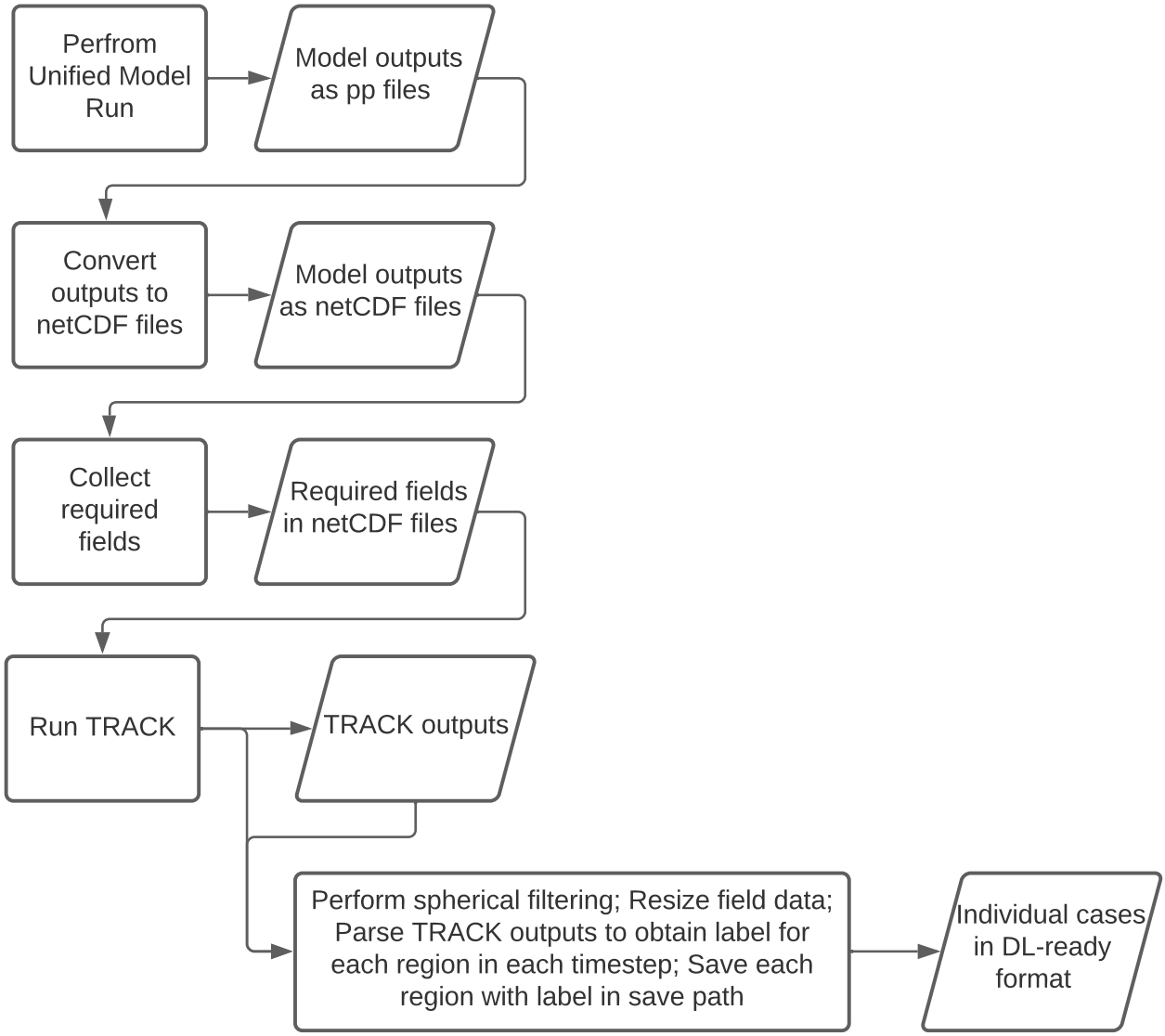


Figure 15. Data preparation workflow for producing training data for DL model from Unified Model outputs.

	IBTrACS	T-TRACK
ERA-Interim	36%	70%
UM N96	N/A	84%
UM N512	N/A	99%
Hist1950	N/A	97%
Future	N/A	98%

Table 5. Percentage of timesteps in which IBTrACS or T-TRACK detect a TC for each dataset.

included in the file's path. The presence of a TC in the region is obtained by processing the TC centres obtained by T-TRACK and checking whether a TC was reported in the timestamp and region being processed.

The method was initially intended to be used to save data from whole timesteps based on whether a TC was detected by TCDetect. However, it was found that a TC was detected in the majority of the timesteps. This would have made the method less useful as the data reduction would have been minimal. T-TRACK also showed this, as seen in Table 5, especially for non-ERA-Interim data. As such, the method was changed to only output data from regions in which a TC was detected, so that regional climate modelling could use this data as boundary conditions for further analysis.

As shown in Figure 16, TCDetect overestimates the number of TCs but is able to capture intra-annual variability when detecting TCs. We aim to use this behaviour to formulate the filtering method to only save high resolution data from regions which have a TC present.

Table 6 shows the number of instances of regions having a TC according to IBTrACS, T-TRACK and TCDetect for ERA-Interim data from the 1st July 2017 to 31st August 2019. As expected, both T-TRACK and TCDetect overestimate these instances with respect to IBTrACS. The latter observed a TC in 6% of all of these regions (3044 timesteps multipled by eight regions), while T-TRACK and TCDetect detected a TC in 19% and 14% of such regions. This means that if the filtering method was used there would be a reduction of 86% of the original data size.

A similar exercise was performed on data from the N96 simulation run of the UM, with results shown in Table 7. As IBTrACS data is not available for this dataset, it was not included. Also, to remove the effect of the deep learning model being tested on data which has a different horizontal resolution than that it is being tested on, the architecture of TCDetect was retrained on UM N96 data to obtain TCDetect-N96. When testing with this model, the results show that T-TRACK detected a TC in 5799 (17%) of 34560 regions, while TCDetect-N96 detected a TC in 9749 (28%) of the regions. The latter would represent a 72% reduction in the data saved to disk.

Finally, the exercise was repeated on data from the N512 simulation run of the UM. Retraining of the original TCDetect architecture on this data was done, with TCDetect-N512 being obtained. Results of the number of regions in which a TC was detected by both T-TRACK and TCDetect-N512 are shown in Table 8. The results show that T-TRACK detected a TC in 2633 (23%) of 11480 regions, while TCDetect detected a TC in 3097 (27%) of the regions. The latter would represent a 73% reduction in the data saved to disk.

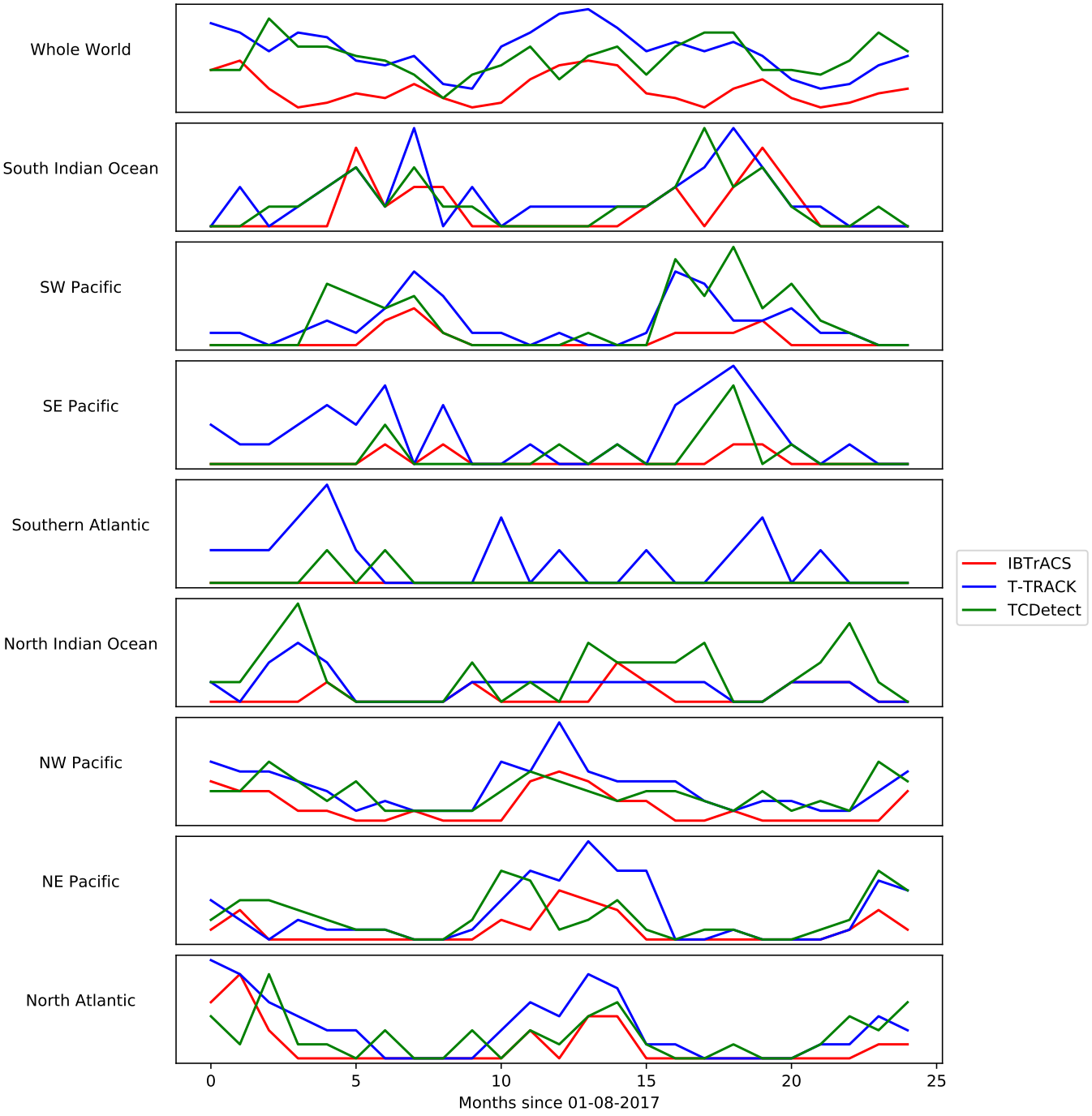


Figure 16. TC frequency, i.e. number of TC tracks present in a month, as given by IBTrACS, T-TRACK and TCDetect for each of the regions used by TCDetect as shown in Figure 1.

	IBTrACS	T-TRACK	TCDetect
North Indian	72 (2%)	277 (9%)	239 (8%)
North Western Pacific	400 (13%)	1222 (40%)	933 (31%)
North Eastern Pacific	267 (9%)	712 (23%)	875 (29%)
North Atlantic	250 (8%)	703 (23%)	497 (16%)
South Indian	214 (7%)	646 (21%)	406 (13%)
South Western Pacific	113 (4%)	740 (24%)	399 (13%)
South Eastern Pacific	26 (1%)	322 (11%)	35 (1%)
South Atlantic	0 (0%)	119 (4%)	13 (0%)

Table 6. Number of instances of regions having a TC according to IBTrACS (first column), T-TRACK (second column) and TCDetect (third column) for ERA-Interim data from the 1st July 2017 to 31st August 2019. Each region has a total of 3044 timesteps and the percentage in brackets shows the percentage of the total number of timesteps which have a TC present.

	T-TRACK	TCDetect-N96
North Indian	74 (2%)	105 (2%)
North Western Pacific	1674 (39%)	2617 (61%)
North Eastern Pacific	442 (11%)	689 (16%)
North Atlantic	559 (13%)	999 (23%)
South Indian	913 (21%)	1596 (37%)
South Western Pacific	874 (20%)	1436 (33%)
South Eastern Pacific	1096 (25%)	1945 (45%)
South Atlantic	167 (4%)	362 (8%)

Table 7. Number of instances of regions having a TC according to T-TRACK (first column) and TCDetect (second column) for three simulated years of test data from the N96 simulation of the UM. Each region has a total of 4320 instances and the percentage in brackets shows the percentage of the total number of instances which have a TC present.

	T-TRACK	TCDetect-N512
North Indian	157 (11%)	197 (14%)
North Western Pacific	956 (67%)	920 (64%)
North Eastern Pacific	524 (37%)	714 (50%)
North Atlantic	431 (30%)	683 (48%)
South Indian	234 (16%)	552 (38%)
South Western Pacific	154 (11%)	374 (26%)
South Eastern Pacific	149 (10%)	370 (26%)
South Atlantic	28 (2%)	97 (7%)

Table 8. Number of instances of regions having a TC according to T-TRACK (first column) and TCDetect (second column) for the simulated year of test data from the N512 simulation of the UM. Each region has a total of 1435 instances and the percentage in brackets shows the percentage of the total number of instances which have a TC present.

Therefore, the method as presented here would help to filter out the majority of data when data containing only TCs is required. For high resolution (N512) data, this is a 73% reduction. Given the large size of such high resolution data, this reduction represents a significant reduction in data.

5 Computational Performance

One important aspect of the method presented is that should be as lightweight as possible, i.e. it should not slow down the execution of the simulation excessively. Two seven-simulated-days runs at both N96 and N512 resolutions were carried out where the mean CPU time of the method as a whole and various parts of the method were timed, with results tabulated in Table 9. The model used a timestep length of 20 simulated minutes for the run at a resolution of N96, giving a total of 504 timesteps. The model run at a resolution of N512 used a timestep length of 10 simulated minutes, giving a total of 1008 timesteps.

The results obtained are put in a way that would show that the method would not be computationally viable if applied for every timestep, but would be so if applied to analysis data written out very number of timesteps, usually every six simulated hours.

In the N96 simulation, the simulation for a whole timestep took, on average, 6.15 seconds to compute, with 4.96 seconds used for the filtering method. This shows that around 80% of the total timestep duration is taken up by the filtering method being presented. Similarly, in the N512 simulation, a whole timestep simulation takes, on average, 15.64 seconds, with 10.22 seconds, or around 65%, dedicated to the filtering method. In both cases, the filtering method is slowing down the simulation considerably.

Table 9 also shows a breakdown of execution times of various sections of the method. Only one section of the code, that interpolating the MSLP field from an Arakawa C-grid to an Arakawa B-grid, is majorly affected by the change in resolution. In

Function	Times Applied	Mean CPU time (seconds) N96	Mean CPU time (seconds) N512
Collect data on central pe	1	4.64×10^{-4}	2.99×10^{-3}
Interplate MSLP field to B-grid	1	3.15×10^{-4}	0.35
Resizing field to ERA-Interim resolution	5	9.1×10^{-3}	3.67×10^{-2}
Calculate vorticity	3	1.16×10^{-3}	2.28×10^{-3}
Perform spherical filtering on a field	5	0.68	1.36
Perform standardisation on a field	5	4.13×10^{-5}	6.64×10^{-5}
Prepare a region's data in DL format	8	1.63×10^{-5}	3.66×10^{-5}
Obtain DL inference for a region	8	0.19	0.37
Full data filtration method		4.96	10.22
Full Timestep		6.15	15.64

Table 9. Timings for UM runs including the filtering method at horizontal resolutions of N96 and N512.

the N96 resolution simulation, this took 3.15×10^{-4} seconds to complete, while it took 0.35 seconds to complete, an increase of more than 1000%, in the N512 resolution simulation. This is due to the larger amount of points that need to be interpolated.

Two sections of code, those that perform spherical filtering of a field and that obtains the inference from the DL model take up the majority of the time in both simulations. The former takes 0.68 seconds and 1.36 seconds in the N96 and N512 resolution simulations respectively. The latter takes 0.19 seconds and 0.37 seconds in the N96 and N512 resolution simulations respectively.

The code that performs the spherical filtering is called on each of the five fields while that which obtains the inference from the deep learning model is called on each of the eight regions shown in Figure 1. Currently, these are done serially, hence a large chunk of the time required for the whole timestep, is taken up by these two computationally expensive sections of code. On the other hand, had these been done in parallel and assuming ideal speedup, the filtering method would take 0.91 seconds for the N96 resolution simulation and 2.19 seconds for the N512 resolution simulation. These represent a reduction of around 80% for the time required to execute the filtering method. Also, had the method been executed in such a way that an inference from each method is obtained at the same time, i.e. utilising many CPUs to run in parallel, it would take around 43% of a timestep's execution time (neglecting any time needed to scatter the data to each required CPU, but this should be at the same order of collecting the data on the central CPU, which is insignificant) in the N96 resolution simulation and around 29% of a timestep's execution in the N512 resolution simulation, resulting in a much more lightweight filtering method.

All of this would mean that the method being presented is not lightweight if applied at each model timestep as it slows the execution of the simulation down considerably. However, it might be more acceptable at even higher resolutions than currently tested. The cost of the method has been shown to decrease from 43% of the total timestep execution time at N96 resolution to 29% at N512. Hence, we would expect this to decrease further, mainly due to a timestep taking longer to compute at higher

resolutions. Also, the importance of the method would increase with a higher resolution simulation as more data would be produced.

However, when applying this method to analysis data which is only outputted every certain amount of timesteps, the computation cost decreases. Assuming that our method is to applied every six simulated hours and in the N96 simulation, the timestep used by the model is that of 20 simulated minues, our method would be used every 18 timesteps, so it would add 4.96 seconds of computational time for every 21.42 seconds of runtime, representing a 23% slowdown in computation. The N512 simulation uses a timestep of 10 simulated minutes, so our method would add 10.22 seconds per 195 seconds of runtime, slowing down the computation by around 5%.

These show that the method if applied in this way does not slow down the computation of the simulation significantly, especially for those run at higher resolution.

6 Summary

TCDetect a deep learning model developed by Galea et al. (2022) to detect the presence of a Tropical Cyclone in meteorological data, was incorporated into a fully-fledged method that could be used by the Met Office's Unified Model (UM) to decide whether to save analysis data to disk depending on the inferred presence of a Tropical Cyclone in a certain region.

The amended UM algorithm to include this method was detailed and shown to require minimal changes to the original UM. This included how a C++ intermediate layer was used to load the deep learning model trained in Python to be used by the FORTRAN-based UM.

It was shown that both TCDetect and T-TRACK, a state of the art detection and tracking algorithm, detected a TC in most analysis timesteps. This would have made the reduction in data negligible. Hence, the method was changed to only save data from a certain region of the globe when a TC was detected in that region. This data would serve as boundary conditions for regional climate modelling which is performed to study the adaptation of the Earth system to a changing climate. The method as now intended reduces the data saved to disk considerably.

The adaptability of the method to operate on data with different sources was also discussed. Multiple sources of data were used for this but having only IBTrACS labelling available for one, the change of labelling from IBTrACS to T-TRACK was shown to be significant. Given that T-TRACK still contains various types of tropical systems, not just hurricane-force systems, the performance of a variant of TCDetect trained with this labelling was shown to be less than that of TCDetect, possibly due to the change of labelling effectively changing what the deep learning model is being asked to detect. Despite this, it was shown that a deep learning model trained on UM data could be applied on ERA-Interim data, but not vice-versa. It was also shown that a model trained on the data that reflected the testing data performed best or second best. It was also found that a model can be applied to data of a higher resolution, albeit with a slight drop in performance, but not vice-versa. It was finally shown that a model trained on data from the current climate an be applied to data from a future climate, at least when TCs are considered.

Finally, the computational expense of the method was discussed. It was shown that the method applied to the output of analysis data, which only happens after every period of simulated time, slows down the computation of the simulation by 23%

in a N96 simulation using the UM, and only by 5% in a N512 simulation. Given that this method is more likely to be applied to higher resolution data, i.e. to the N512 simulation, the computational cost of the method is sufficient for it to be considered a lightweight method.

Appendix A: The `frugally-deep` package

As explained in the Chapter 5, the package `frugally-deep` is used to be able to use the deep learning model trained in Python in the FORTRAN-based UM. In this appendix, we will attempt to show the inner workings for this package and how it was used in the UM implementation discussed in Chapter 5.

The package is a header-only library written in C++ except for a script which saves the trained model to disk which is written in Python. The aim of this package is to use Tensorflow trained deep learning models in C or C++ code, without having to configure Tensorflow on the system being used. As such, `frugally-deep` can only produce inferences using the trained model.

First, an important Python script is included in the package which converts and saves the trained model to disk in a readable format for `frugally-deep`. It requires a trained model that has been saved to an HDF5 file via Tensorflow's `model.save()` routine. This model is loaded by the converter and the converted model is saved as a as a JavaScript Object Notation (JSON) file which has six entries. The first is the architecture, i.e. the layer types and other hyperparameters, of the model in JSON format. The second entry is the image data format, i.e. whether the data that will be put into the model will have the channels index as the first or the last dimension of any array used. The next two entries are the expected input and output array shapes. The penultimate entry contains all of the model's weights in ASCII format and the final entry has the model's hash stored.

In C++ code, `frugally-deep` loads the model via its own `model.load_model()` routine. This uses the json dependency to read in the json file and constructs the model by loading the weights and architecture into vectors. This C++ code is called from the UM's FORTRAN code and returns a pointer to the loaded deep learning model to be stored and used in the FORTRAN code.

After this, the data needed to produce the inference is constructed. The preprocessed data is obtained from the UM and is given as a three dimensional array. This array is passed onto a C++ routine and used to construct `frugally-deep` tensors. These tensors are then passed onto the constructed model for an inference to be made via `frugally-deep`'s `model.predict()` which return another `frugally-deep` tensor. The inference is extracted from this tensor and returned to the UM's FORTRAN code, where it is handled and processed to get the needed decision on whether to save data to disk or not.

Author contributions. TEXT

Competing interests. TEXT

Acknowledgements. This work was funded by Natural Environment Research Council (NERC) as part of the UK Government Department for Business, Energy and Industrial Strategy (BEIS) National Productivity Investment Fund (NPIF), grant number NE/R008868/1. Special thanks must be given to Dr. Simon Wilson an Dr. Jeff Cole from the National Centre for Atmospheric Science (NCAS), UK for their help in including the deep learning model in the Met Office's Unified Model.

References

Adams, J. C. and Swarztrauber, P. N.: SPHEREPACK 3.0: A Model Development Facility, Monthly Weather Review, 127, 1872 – 1878, https://doi.org/10.1175/1520-0493(1999)127<1872:SAMDF>2.0.CO;2, 1999.

Bengtsson, L., Hodges, K. I., and Esch, M.: Tropical cyclones in a T159 resolution global climate model: comparison with observations and re-analyses, Tellus A: Dynamic Meteorology and Oceanography, 59, 396–416, https://doi.org/10.1111/j.1600-0870.2007.00236.x, 2007.

Dee, D. P., Uppala, S. M., Simmons, A. J., Berrisford, P., Poli, P., Kobayashi, S., Andrae, U., Balmaseda, M. A., Balsamo, G., Bauer, P., Bechtold, P., Beljaars, A. C. M., van de Berg, L., Bidlot, J., Bormann, N., Delsol, C., Dragani, R., Fuentes, M., Geer, A. J., Haimberger, L., Healy, S. B., Hersbach, H., Hólm, E. V., Isaksen, L., Kållberg, P., Köhler, M., Matricardi, M., McNally, A. P., Monge-Sanz, B. M., Morcrette, J.-J., Park, B.-K., Peubey, C., de Rosnay, P., Tavolato, C., Thépaut, J.-N., and Vitart, F.: The ERA-Interim reanalysis: configuration and performance of the data assimilation system, Q. J. R. Meteorol. Soc., 137, 553–597, https://doi.org/10.1002/qj.828, 2011.

Eyring, V., Bony, S., Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J., and Taylor, K. E.: Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization, Geoscientific Model Development, 9, 1937–1958, https://doi.org/10.5194/gmd-9-1937-2016, 2016.

Galea, D. and Lawrence, B.: paper2 blah blah, this one, submitted, 2022.

Galea, D., Lawrence, B., and Kunkel, J.: paper1 blah blah, this one, submitted, 2022.

Hermann, T.: frugally-deep, https://github.com/Dobiasd/frugally-deep, 2020.

Hodges, K., Cobb, A., and Vidale, P. L.: How Well Are Tropical Cyclones Represented in Reanalysis Datasets?, Journal of Climate, 30, 5243 – 5264, https://doi.org/10.1175/JCLI-D-16-0557.1, 2017.

Hodges, K. I.: Feature Tracking on the Unit Sphere, Monthly Weather Review, 123, 3458–3465, https://doi.org/10.1175/1520-0493(1995)123<3458:FTOTUS>2.0.CO;2, 1995.

Hodges, K. I.: Spherical Nonparametric Estimators Applied to the UGAMP Model Integration for AMIP, Monthly Weather Review, 124, 2914–2932, https://doi.org/10.1175/1520-0493(1996)124<2914:SNEATT>2.0.CO;2, 1996.

Hodges, K. I.: Adaptive Constraints for Feature Tracking, Monthly Weather Review, 127, 1362–1373, https://doi.org/10.1175/1520-0493(1999)127<1362:ACFFT>2.0.CO;2, 1999.

Met Office Hadley Centre: WCRP CMIP6: Met Office Hadley Centre (MOHC) HadGEM3-GC31-HH model output for the "highres-future" experiment, https://catalogue.ceda.ac.uk/uuid/6674daf12f33474e826952c8d27a9f9b, 2020a.

Met Office Hadley Centre: WCRP CMIP6: Met Office Hadley Centre (MOHC) HadGEM3-GC31-HH model output for the "hist-1950" experiment, https://catalogue.ceda.ac.uk/uuid/56fd51e7a5854128bffb82302fb6e513, 2020b.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D.: Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization, in: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 618–626, https://doi.org/10.1109/ICCV.2017.74, 2017.