TCDetect: A new method of Detecting the Presence of Tropical Cyclones using Deep Learning

Daniel Galea,[a] Julian Kunkel,[b] Bryan N. Lawrence,[a,c]

[a] *Department of Computer Science, University of Reading, UK*

[b] *University of Göttingen/GWDG, Germany*

[c] *National Centre of Atmospheric Science, Department of Meteorology, University of Reading, UK*

Corresponding author: Daniel Galea, galea.daniel18@gmail.com

ABSTRACT: Tropical cyclones are severe weather events which have massive human and economic effect, so it is important to be able to understand how their location, frequency and structure might change in future climate. Here, a lightweight Deep Learning model is presented which is intended for detecting the presence or absence of tropical cyclones during the execution of numerical simulations. The model was trained on ERA-Interim reanalysis data from 1979 to 2017 and the training concentrated on delivering the highest possible recall rate (successful detection of cyclones) while rejecting enough data to make a difference in outputs. When tested using data from the two subsequent years, the recall rate was 92% and the precision was 36%. For the desired filtration application, if the desired target included relevant meteorological events, the effective precision was 85%. The recall rate compares favourably with other methods of cyclone identification having the best Area Under Curve for the Precision/Recall (AUC-PR) and using the smallest number of parameters for both training and inference. This model has been developed to investigate lightweight methods of finding tropical cyclones during the simulation so as to avoid saving vast amounts of data for analysis after the simulation is complete. With run-time detection it might be possible to reduce the need for some of the high-frequency high-resolution output which would otherwise be required.

1. Introduction

Tropical Cyclones (TCs) are extreme weather events that can leave devastating effects on human populations; for example, Hurricane Irma impacted the Caribbean Islands and the Southeast USA in September 2017 causing 47 direct deaths, 82 indirect deaths, hundreds of injuries and an estimated monetary damage of around 50 billion USD (Cangialosi et al. 2018). TCs can be detected and tracked both in satellite data and in simulations carried out using numerical weather prediction or climate models.

Climate models can be used to understand how the properties of TCs and other meteorological phenomena might evolve in a changing climate, but such Global Circulation Models (GCMs) produce a large amount of data. A method to filter this data in order to target analysis would be useful; and such a method is presented here, based on a deep learning model that detects the presence of tropical cyclones in simulation output. While tested offline (i.e. after data has been output), the method is intended for eventual deployment online (i.e. while a simulation model is running) so as to preclude the need to output data which does not include TCs (at least for the situation where TCs are the product of interest). The method presented here is lightweight, with relatively short training and inference times, and requires no explicit a-priori thresholds in meteorological variables. It is also shown to perform at least as well as other more standard, and more complex, deep learning models.

Following motivation and a description of previous work, the deep learning model itself is presented, along with a description of the data used for training and validation. The performance of the model is then discussed, both in terms of metrics of success (precision and recall) and in comparison with other deep learning models. A range of techniques were used to attempt to explain the results obtained.

Motivation

Data volumes from climate simulations are huge. The current phase of the climate model intercomparison project (CMIP6, Eyring et al. 2016) comprises hundreds of different model simulations was projected to produce 18 PB of data (Balaji et al. 2018). While it may not reach that volume, it will be close, and that total does not include the data that was produced and analysed in

the production of the archived data. Whether in or out of the CMIP archive, such data are costly to store and maintain and the volume makes analysis difficult.

A method of automatically detecting interesting phenomena in a model before saving the data could have two major benefits:

1. A fast way of finding and tabulating summary data (without writing out the actual data used), or

2. A method for reducing the need to write all the data to disk for subsequent analysis — for example, only data from a specific region and time where an event was present could be saved.

In both cases this would likely have increased scientific productivity in that there would be significant savings in time and data volume saved — leading to more efficient science (efficient in time saved, storage costs, and storage energy consumption).

The possibility of efficiencies arise because although many simulations are carried out to target multiple use-cases, some are carried out to investigate specific phenomena (e.g. when checking the impact of resolution on simulated TCs as in Roberts et al. 2015). In these cases, data relating to other phenomena might not be needed. However, currently in order to be able to retrieve the data for specific phenomena, the simulation will store sufficient data for post-processing analysis at fixed intervals. The first post-processing step then involves the retrieval of only relevant data, the other data is not used.

To select the correct data for analysis, it is important to have confidence in the method used for identifying the feature of interest. There is sometimes a conflict between the abstract notion of the feature of interest (in this case a TC) and the practical implementation of a definition of for a TC — the latter is intimately related to the tool for discovering it. For example, if the practical definition of a TC is the same as the metric for detecting it, of course we have confidence in it - but this definition may miss (or include) things which we would abstractly consider to be TCs (or detect phenomena which we would not consider to be TCs, but fall inside a poorly drawn definition). We show some examples of this later. Many previous techniques for TC identification in numerical data generally conflate the detection method with the definition — with deep learning it is clear that will not be the case, so understanding the distinction is important.

Although our initial interest is in detecting TCs, the conceptual method is expected to be extensible to other important phenomena such as fronts, atmospheric rivers etc.

2. Previous Work

Several methods used to detect TCs have been developed. Most operate by using thresholds set for a few meteorological variables to determine the presence of a tropical cyclone. The use of thresholds leads to two problems: setting such thresholds involves scientific subjectivity, and the combination of method and threshold may not be transferable across different models or data. More recently deep learning has been used, and while deep learning may suffer from aspects of the transferability problem, it should be possible to avoid subjectivity.

a. TC detection using conventional techniques

Conventional techniques usually work by identifying TC centres by applying various thresholds to the available data. TC tracks are then created by arranging the identified TC centres according to some mathematically-based method. A few examples of such methods follow, with a tubular summary in Table 1.

Kleppek et al. (2008) use multiple thresholds to identify the TC centers. The first is that a local minimum of sea-level pressure (SLP) needs to observed within a neighbourhood of an 8x8 square of grid points. This is assigned as a storm centre. For it to be designated a TC centre, there needs to be a maximum relative vorticity at 850hPa above 5×10^{-5} s^{-1} at the storm centre. The presence of vertical wind shear between 850hPa and 200hPa of at least 10 ms^{-1} is also required, as well as an event lifetime of 36 or more hours. Finally, if the storm centre is over land, the relative vorticity condition has to be fulfilled or the wind speed maximum at 850hPa needs to be inside 250km from the TC centre.

Similarly, Vitart et al. (1997) used the closest minimum of mean sea level pressure (MSLP) to a local maximum of relative vorticity at 850hPa over 3.5×10^{-5} s^{-1} as a storm centre. A warm-core check, similar to Roberts et al. (2015), is performed to classify the storm centre as a TC. This requires that the closest local maximum of the average temperature between 550hPa and 200hPa must be within 2^{o} of the storm centre and the temperature decreases by at least 0.5^{o}C for at least 8^{o} latitude in all directions. Also, the closest maximum thickness between 1000hPa and 200hPa must be within 2^{o} of the storm centre and the thickness must decrease at least 50 metres for at least 8^{o} latitude in all directions.

Camargo and Zebiak (2002) introduce a detection method that uses vorticity at 850hPa, surface wind speed and a vertically integrated temperature anomaly as variables on which to impose basin-dependant thresholds.

A final example is Roberts et al. (2015) who use the method explained by Hodges (1995), Hodges (1996), Hodges (1999) and Bengtsson et al. (2007), where a TC is identified by a maximum of 850hPa relative vorticity, in data which has been spectrally filtered using a T42 filter (i.e. keeps features greater than 250km in scale) and a warm-core check on a T63 grid (to keep features larger than 180km) using the 850hPa, 500hPa, 300hPa and 200hPa levels.

b. TC detection using Deep Learning

A newer crop of algorithms have been developed to detect and track TCs using deep learning methods. These are summarized in Table 2.

Racah et al. (2017) created a method where a deep learning model takes in a snapshot of the world simulated by the CAM5 climate model with 16 different meteorological variables and creates bounding boxes around the detected TCs. The variables used were: total precipitation rate, surface pressure, sea level pressure, reference height humidity, temperature at 200 millibar and 500 millibar, total vertically integrated precipitable water, reference height temperature, radiative surface temperature, meridional and zonal wind speed at 850 millibar and at the lowest available model level, geopotential at 100 millibar and 200 millibar and the lowest model level height. The architecture used was that of an auto-encoder with three smaller networks using the bottleneck layer to draw a box around a suspected TC. Given the size of the inputs and number of kernels used in the convolution layers, the model presented was expected to be time consuming to train. It certainly required supercomputing: an adaptation of this deep learning model was trained using 9622 nodes of 68 cores each with a peak throughput of 15.04PF/s and reached a sustained throughput of 13.27 PF/s, although the total time to train was not reported (Kurth et al. 2017). The accuracy for this model was specified as the percentage of overlap between the predicted box and the box given as the ground truth — an Intersection of Union (IOU) — which was created using the Toolkit for Extreme Climate Analysis (TECA) (Prabhat et al. 2012, 2015). The model had 24.74% of the predicted boxes having at least an overlap of 10% with the ground truth, while 15.53% of the predicted boxes had at least an overlap of 50% with the ground truth.

Author/s	Variable	Threshold
Kleppek et al. (2008)	Sea Level Pressure (SLP)	Local minimum in a neighbourhood of eight grid points
	Relative Vorticity at 850hPa	$> 5\times10^{-5}$ s^{-1}s and positioned at SLP minimum
	Vertical Wind Shear between 850hPa and 200hPa	> 10 ms^{-1}
	Event time	> 36 hours
	SLP Minimum Position	If over land, relative vorticity at 850hPa $> 5\times10^{-5}$ s^{-1}s and positioned at SLP minimum; Otherwise, wind speed maximum at 850hPa needs to be inside 250km from the TC centre
Vitart et al. (1997)	Relative Vorticity at 850hPa	Local maximum $> 3.5\times10^{-5}$ s^{-1}
	Mean Sea Level Pressure (MSLP)	Minimum closest to relative vorticity local maximum – taken as storm centre
	Average Temperature between 550hPa and 200hPa	Closest maximum within 2^o of the storm centre and the temperature decreases by at least 0.5^oC for at least 8^o latitude in all directions
	Maximum Thickness between 1000hPa and 200hPa	Closest maximum within 2^o of the storm centre and the thickness must decrease at least 50 metres for at least 8^o latitude in all directions
Camargo and Zebiak (2002)	Relative Vorticity at 850hPa	> twice the vorticity standard deviation in each basin
	Surface Wind Speed	> the sum of wind speed standard deviation in each basin and the global average oceanic wind speed in a 7x7 box centered around the relative vorticity maximum
	Sea Level Pressure (SLP)	A local minimum is present in a 7x7 box centered around the relative vorticity maximum
	Temperature Anomaly	Anomaly averaged over a 7x7 box centered around the relative vorticity minimum and over the 300, 500, 700 hPa pressure levels > the basin standard deviation
		Anomaly averaged over a 7x7 box centered around the relative vorticity minimum is positive in all three of 300, 500, and 700 hPa pressure levels
		Anomaly averaged over a 7x7 box centered around the relative vorticity minimum is greater at 300hPa than at 850hPa
	Wind Speed	Wind speed averaged over a 7x7 box centered around the relative vorticity minimum is greater at 850hPa than at 300hPa
	Distance Travelled	Storm centre – defined as the relative vorticity minimum – must not have travelled a distance greater than 5.6^o if 6-hourly output or 8.5^o if daily output
	Event Time	At least 1.5 days if 6-hourly output or 2 days if daily output
Roberts et al. (2015)	Relative Vorticity	$> 6\times10^{-5}$ s^{-1} at 850 hPa
		Reduction of at least 6×10^{-5} s^{-1} in vorticity between 850 and 250 hPa at T63 resolution
		Positive vorticity centre at all available levels between 850 and 250 hPa
	Wind Speed	Wind speed averaged over a 7x7 box centered around the relative vorticity minimum is greater at 850hPa than at 300hPa
	Event Time	At least 1.5 days

TABLE 1: Overview of thresholds applied to meteorological variables for detecting and tracking Tropical Cyclones with the conventional techniques given.

Mudigonda et al. (2017) created a deep learning model which used integrated water vapour (IWV) snapshots and image segmentation techniques to classify whether each pixel in an image was a part of a TC or not. It used an adaptation of the Tiramisu model, which applies the DenseNet architecture to semantic segmentation. The labels were created using TECA and Otsu's method (Otsu 1979). It was trained and tested on images which had at least 10% of the pixels which were

not background pixels. An accuracy of 92% was obtained but it was noted that had the model predicted all the pixels as being all background pixels, the accuracy would have been of 98%.

Similarly, Kumler-Bonfanti et al. (2020) use a U-net network to perform image segmentation for TCs using the total precipitable water field from a weather forecast model, the National Center for Environmental Prediction (NCEP) Global Forecast System (GFS). A U-net network is very similar to an auto-encoder network, with the difference that connections between the two branches of the network are used to convey any information missed while creating the dense representation of the inputs at the end of the encoder branch of the network. The inputs used also take in a measure of time as two snapshots are given to the deep learning model, one approximating the state of the atmosphere at the time the inference was initiated and another having the forecasted state of the atmosphere at the next forecast step available, usually three hours after. Labels of areas belonging to a TC were generated by creating a 25x25 pixel box, approximating 300km^2, around a latitude and longitude pair obtained from International Best Track Archive for Climate Stewardship (IBTrACS, Knapp et al. 2010, Knapp et al. 2018) dataset. This model managed to obtain an IOU of 75% but it should be noted that the labelling boxes were of the same size, so it is possible that the value for the IOU is inflated.

Finally, Liu et al. (2016) used an image cropped in such a way that if a TC was present, it was centred in the image. They used 8 different meteorological variables: surface level pressure, meridional and zonal wind speed at 850 millibar and at the lowest available model level, temperature at 200 millibar and 500 millibar and total vertically integrated precipitable water. They then predicted whether the image was one of a TC or not. The model obtained a 99% accuracy with a relatively simple model and a pre-processing cropping step involving significant noise reduction which would have helped obtain good performance.

3. Deep Learning Model

This section presents the data used to train the deep learning model, the model architecture, and summarises the method used to develop it. Full details of the training appear in the appendices.

Authors	Data Used	Ground Truth	Architectures	Results
Racah et al. (2017)	CAM5 Climate Model 1979-2005 3-hourly 25km resolution Image size of 768 x 1158 pixels 16 channels Training Set: Timesteps during 1979 Testing Set: Timesteps during 1984	TECA	Encoder: Conv: 8(layers) x 384(height) x 576(width) @ 64(kernels) Conv: 8 x 192 x 288 @ 128 Conv: 8 x 96 x 144 @ 256 Conv: 8 x 48 x 72 @ 384 Conv: 8 x 24 x 36 @ 512 Conv: 8 x 12 x 18 @ 640 Decoder: Conv: 8 x 12 x 18 @ 640 Conv: 8 x 24 x 36 @ 512 Conv: 8 x 48 x 72 @ 384 Conv: 8 x 96 x 144 @ 256 Conv: 8 x 192 x 288 @ 128 Conv: 8 x 384 x 576 @ 64 Box Locator: Conv: 4 x 12 x 18 @ 4 Class Probabilities: Conv: 4 x 12 x 18 @ 4 Objectiveness Probabilities: Conv: 4 x 12 x 18 @ 2	IOU = 0.1: 25% IOU = 0.5: 16%
Liu et al. (2016)	CAM5.1 Climate Model 1979-2005 3-hourly 0.23^o x 0.31^o res. ERA-Interim Climate Model 1979-2011 3-hourly 0.25^o x 0.25^o res. 20^{th} Century Reanalyses 1908-1948 daily 1^o x 1^o res. NCEP-NCAR Reanalyses 1949-2009 daily 1^o x 1^o res. Images cropped to 32 x 32 pixels	TECA Manual expert labelling	Conv: 5 x 5 @ 8 Pool: 2 x 2 Conv: 5 x 5 @ 16 Pool: 2 x 2 Dense: 50 Dense: 2	99%
Kumler-Bonfanti et al. (2020)	GFS Weather model 720 x 361 pixels 0.5^o res.	IBTrACS	U-net of 6 layers	IOU = 1: 75%
Mudigonda et al. (2017)	CAM5 Climate Model 1996-2015 Images cropped to 96 x 144 pixels	TECA Otsu's method	Adaptation of Tiramisu Model	92%

TABLE 2: Previous Deep Learning models that detect and track Tropical Cyclones

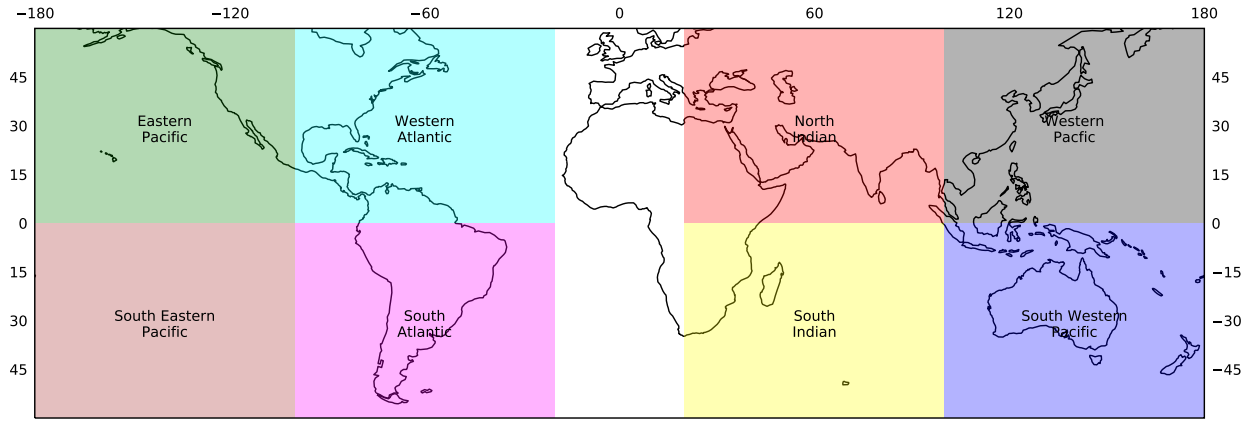


FIG. 1: Timestep split into 8 equal regions

a. Data

The deep learning model, hereafter referred to as TCDetect, was trained, tested and validated on data extracted from the ERA-Interim reanalysis (Dee et al. 2011), with the validation data used for manual hyperparameter tuning as described in Appendix B and the testing set used for producing the final testing statistics and for interpreting the results produced by the trained model.

Five six-hourly fields from the 1st of January 1979 until the 31st of July 2017 were used: mean-sea level pressure (MSLP), 10-metre wind speed, and vorticity at 850hPa, 700hPa, and 600hPa, each at a spatial resolution of $\approx 0.7^{o}$x0.7^{o}. These fields were chosen because they had been used in previous TC detection algorithms and produced the best-performing deep learning model during hyperparameter tuning, as shown in Appendix B. Spherical filtering was performed on each field to reduce some of the smaller scale features. For the MSLP and 10-metre wind speed fields, spherical harmonic filtering is performed to keep wave numbers between 5 and 106. The vorticity fields were spherical harmonic filtered between wave numbers 1 and 63.

Each field was further split into eight regions as shown in Figure 1. These regions are loosely based on those used by the International Best Track Archive for Climate Stewardship (IBTrACS, Knapp et al. 2010, Knapp et al. 2018) dataset. Thus, the entire dataset included 450,944 individual cases, each with dimensions of 86 rows, 114 columns and 5 channels.

Labels for these cases were derived from the IBTrACS dataset. IBTrACS contains temporal information, including category, latitude and longitude of the storm centre, for all major storms across the globe. The labels were set up in such a way that each case was labelled according to

Partition	Years Included	+ve Cases	-ve Cases
Training	1980 - 1981, 1982 - 1985, 1987 - 1990, 1992 - 1995, 1997 - 2000, 2002 - 2005, 2007 - 2010, 2012 - 2015, 2017	17862 (5.00%)	339546 (95.00%)
Validation	1979, 1986, 1991, 1996 2001, 2006, 2011, 2016	4853 (5.19%)	88651 (94.81%)
Testing	2017 - 2019	1342 (5.51%)	23010 (94.49%)

TABLE 3: Dataset Split

the presence or absence of a TC recorded in IBTrACS. At the end of the labelling process, 22,826 (5.06%) positive labels were generated as well as 428,118 (94.94%) negative labels.

This dataset was used for training and validation; it was split by taking data from 1979, 1986, 1991, 1996, 2001, 2006, 2011 and 2016 to make up the validation set and the rest of the data to make up the training set. This method of splitting the available data was chosen so that the possible effects of a changing climate were taken into consideration, so that any hyperparameter tuning performed would not be skewed.

After splitting the available data, the training set had a total of 357,408 cases; 17,862 (5%) with a TC and 339,546 (95%) without. The validation set had a total of 93,504 cases, 4,853 (5.19%) with, and 88,651 (94.81%) without a TC.

Additional data from the 1st of August 2017 until the 31st of August 2019 was used as a testing set. This had a total of 24,352 cases, 1,342 (5.51%) with, and 23,010 (94.49%) without a TC. Table 3 shows how the splits are made and that the split between positive and negative cases is similar across training and testing.

All data was preprocessed to reduce resolution to a sixteenth of the original ERA-Interim resolution by taking the mean value of all data points inside a 4x4 box, reducing the dimensionality of each case to 22 rows, 29 columns and 5 channels. This reduction of resolution was arrived at during hyperparameter tuning. Standardisation for each of the input variable fields was then performed for each channel according to

$$\text{field} = \frac{\text{field} - \mu_{\text{field}}}{\sigma_{\text{field}}} \tag{1}$$

where μ_{field} and σ_{field} are the mean and standard deviations of the values in the field. The resulting standardised fields have a mean of 0 with a standard deviation of 1.

Figure 2 shows an example of the data used, before and after preprocessing, from the time when Hurricane Katrina obtained its maximum strength, the 28$^{\text{th}}$ of August 2005 at 18Z.

b. Architecture

TCDetect uses a convolutional base attached to a fully-connected classifier which outputs a value between 0 and 1, with any values larger than 0.5 signifying that the model detects a TC and any values smaller or equal to 0.5 meaning that the model does not detect a TC. A more detailed explanation of the model architecture can be found in Appendix A, while a graphical view is shown in Figure 3.

To arrive at the model architecture, manual hyperparameter tuning was used to determine which changes to the architecture performed well (see Appendix B).

Various metrics could have been used to determine how the architecture should be changed to produce a better-performing model. Accuracy, defined by

$$\text{accuracy} = \frac{TP+TN}{TP+FP+TN+FP} \times 100,$$

where TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives and FP is the number of false positives, was considered, but was not suitable due to the large class imbalance present in our training set. Due to large class imbalance, the model would learn to infer "no TC" for all cases. This leads to TN being a high number, producing a high accuracy, but a model with no skill.

Given that obtaining the highest possible number of TCs is important for the use of the developed model, recall, defined by

$$\text{recall} = \frac{TP}{TP+FN},$$

could have been used, as a high value would indicate that the number of false negatives, i.e. not detected TC cases, is small. However, this would not have considered the need of detecting non-TC

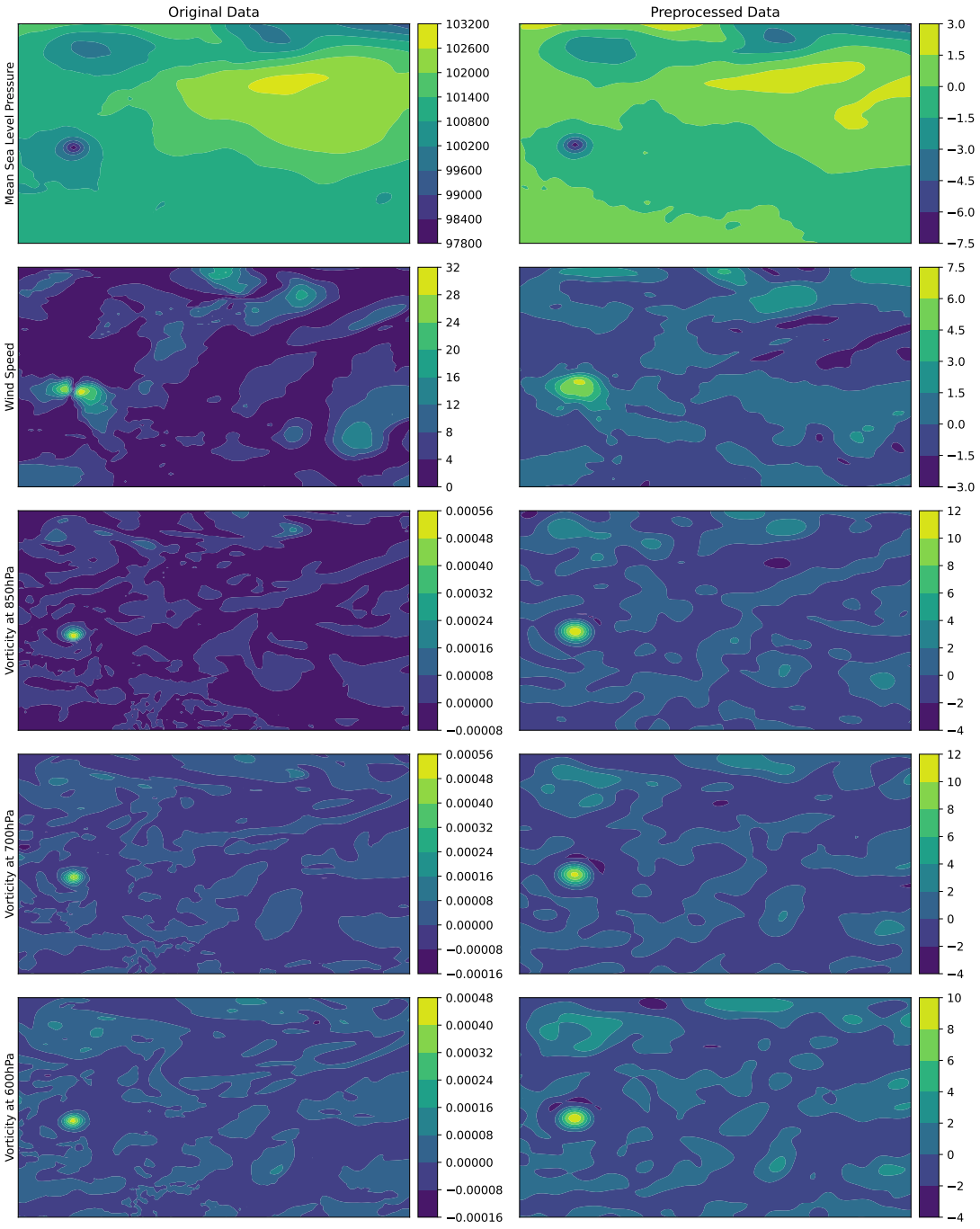


FIG. 2: An example of the data that was used to train TCDetect. The data is from the re-analysis at the point where Hurricane Katrina obtained its maximum strength (28th of August 2005 at 18Z). The left column shows the original data from ERA-Interim and the right column shows how this data is transformed after preprocessing. The rows are (in reverse order of height above the surface): Mean Sea Level Pressure (MSLP), 10-metre wind speed, and the vorticity at 850hPa, 700hPa, and 600hPa.

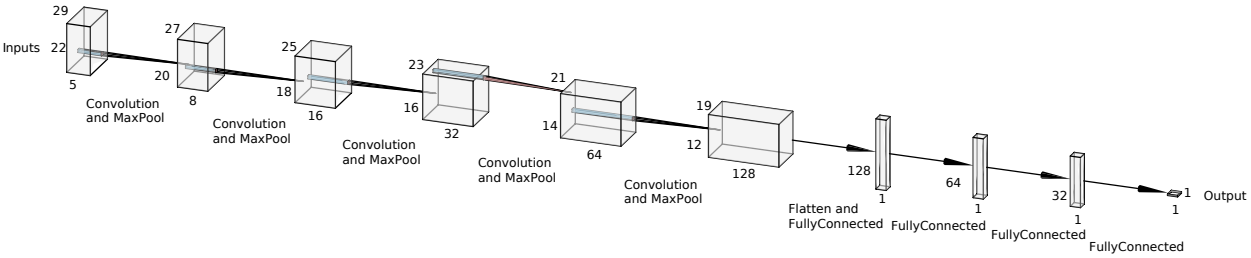


FIG. 3: Visual representation of the architecture of TCDetect. The inputs, having 22 rows, 29 columns and 5 fields, are passed through a 2x2 convolution window whose weights are learnt producing 8 feature maps, but losing one row and column. The resulting feature maps are passed through a MaxPool window, of size 2x2, which takes the maximum value in its window. This further reduces the feature maps' size by a row and a column, to 20 rows and 27 columns. This is repeated for 3 more times, with each step producing more feature maps. The final ones of these are combined and reshaped into one long array. This array is used as the input to a fully-connected layer of 128 nodes, each of which output a single value. Hence, 128 values are now obtained and are used as inputs to the next layer. This is repeated for three more times, ending with the final value.

cases as such. For this, precision, defined by

$$\text{precision} = \frac{TP}{TP+FP},$$

could be used as a high value would indicate that the number of false positives, i.e. non-TC cases inferred as TC cases, is small.

To get the right balance between the two functions of the model, the Area-under-Curve for the Precision-Recall curve (AUC-PR) was used. This gives a single value which takes into account the two important functions of the model. This value can still be slightly obscure as the same value could be produced for high precision and low recall rates, high recall and low precision rate or average recall and precision rates. However, as the model was being developed primarily to identify data for further post-processing, false negatives would be a bigger problem than false positives, so improvements caused by different values for hyperparameters in recall were favoured over those in precision if AUC-PR varied only marginally or the balance between recall and precision needed to be addressed as a change was assessed.

The training of the deep learning model was done using a NVIDIA Volta 100 GPU on a node having 32GB of RAM and 32 CPU cores. Some initial tests were performed on a cloud instance provided by Oracle, while the main model development was performed on the JASMIN platform (Lawrence et al. 2012). The software packages used were Python 3.6.8 and Tensorflow v2.20

		TCDetect	
		Yes	**No**
Labelled	**Yes**	1231	111
	No	2166	20844

TABLE 4: Confusion matrix resulting from inference on the testing dataset.

(Abadi et al. 2015). The training set was traversed 21 times for the model to converge to a solution with a total time to train of 12 minutes. Although the time taken to train the final model was relatively short, much time was taken up in progressing through the various optimisations detailed in Appendix B, mainly due to the use of 10-fold cross-validation.

4. Results

The resulting deep learning model, TCDetect, was evaluated using the test set described above. The inferences obtained were also investigated to understand how the model generates its results. We present these results in this section.

a. Model Statistics

The model, after training, correctly classified 1231 (91.73%) of the 1,342 cases as having a TC and 20844 (90.59%) of the 23,010 without. It misclassified 111 (8.27%) cases in which a TC was present and 2166 (9.41%) cases in which a TC was not present. These are summarised in the confusion matrix in Table 4.

These results conform to an accuracy of 90.65%, a recall rate of 91.73% and a precision rate of 36.24%. The use of AUC-PR as a metric to maximise when performing hyperparameter tuning resulted in a high recall rate and a sufficiently high precision rate for this model to be used as a data filtration technique.

The outcome could have been further varied by changing the value which is the boundary between a positive and a negative prediction, currently 0.5. Figure 4 shows the AUC-PR curve for the model with the values at each point signifying the boundary at which the corresponding recall and precision rates are obtained.

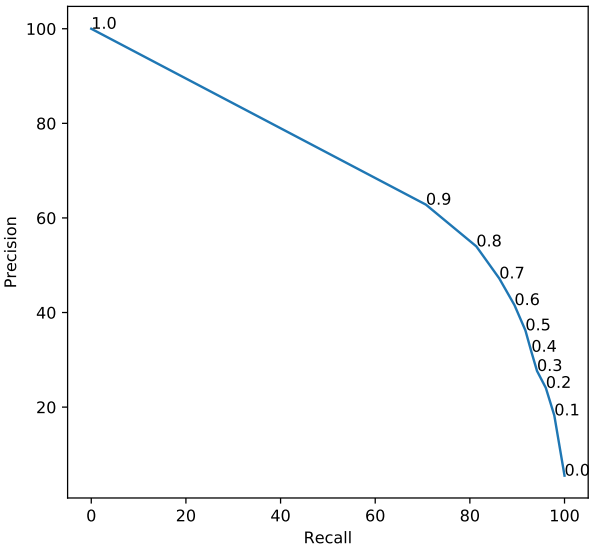


FIG. 4: Precision-Recall curve for final trained model evaluated on the testing dataset.

b. Comparison with Standard Models

There are many existing deep learning standard models, so it is reasonable to ask "Would any of those do better than the model developed here?".

To test this, the convolutional base, i.e. the part of the network which learns spatial patterns, from a variety of standard model architectures were used in conjunction with the classifier developed here and compared. The convolutional bases of the following standard architectures were used: DenseNet121 (Huang et al. 2016), DenseNet169 (Huang et al. 2016), DenseNet201 (Huang et al. 2016), InceptionResNetV2 (Szegedy et al. 2017), InceptionV3 (Szegedy et al. 2016), MobileNet (Howard et al. 2017), MobileNetV2 (Sandler et al. 2018), ResNet101 (He et al. 2016a), ResNet101V2 (He et al. 2016b), ResNet152 (He et al. 2016a), ResNet152V2 (He et al. 2016b), ResNet50 (He et al. 2016a), ResNet50V2 (He et al. 2016b), VGG16 (Simonyan and Zisserman 2014), VGG19 (Simonyan and Zisserman 2014) and Xception (Chollet 2017). The weights of all of these bases were obtained by training on the ImageNet dataset (Deng et al. 2009). These were then frozen and the weights of the classifier were retrained on data from all regions with the presented model's learning rate, momentum value and L2 normalisation factor.

Given that these convolutional bases required inputs of at least 75 pixels by 75 pixels with 3 channels, some changes to the inputs were required. Firstly, as an input with only 3 channels is required for the most of the above architectures, the fields retained were those of vorticity at

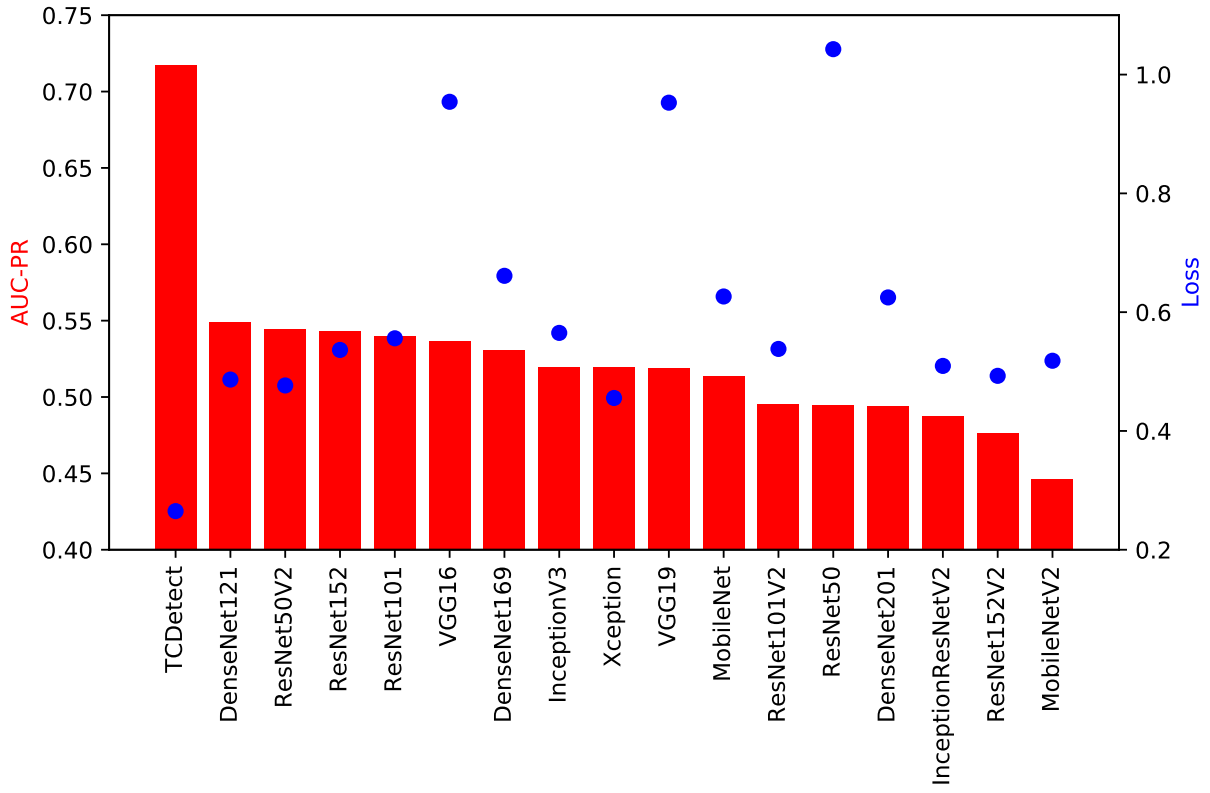


FIG. 5: Test AUC-PR (bars) and test loss (points) for standard convolutional bases, pre-trained on the ImageNet database, attached to the fully-connected classifier developed in the presented model. The classifier was re-trained for each convolutional base with data from all regions.

850hPa, vorticity at 600hPa and MSLP. This choice was made as these three fields were deemed the most important for the model being presented by tests detailed in Section 5a. Secondly, the input size was extended five fold from 22x29 pixels to 110x145 pixels by interpolating any intermediate values.

Of the standard architectures tested, none managed to obtain better AUC-PR or loss values on the test set (Figure 5). Table 5 compares the complexity of some of these more standard models and their performance metrics with TCDetect. All of the more standard models had far higher complexity in terms of the number of parameters used than TDDetect, and the latter also outperforms the others in terms of AUC-PR, precision rate and loss. While TCDetect recall is not outperforming all of the other models, it is in the top third of the list.

Convolutional Base	Total Parameters	AUC-PR	Recall	Precision	Loss
TCDetect	**3,789,977**	**0.7173**	92%	**36%**	**0.2650**
DenseNet121	8,620,865	0.5409	83%	25%	0.4865
DenseNet169	15,209,281	0.5307	93%	13%	0.6612
DenseNet201	21,821,601	0.4940	88%	20%	0.6248
InceptionResNet v2	55,526,881	0.4874	83%	20%	0.5095
Inception v3	23,386,145	0.5184	90%	18%	0.5651
MobileNet	4,812,225	0.5139	88%	17%	0.6264
MobileNet v2	5,545,281	0.4461	86%	18%	0.5182
ResNet101	47,911,553	0.5397	89%	17%	0.5560
ResNet101 v2	47,879,937	0.4955	88%	21%	0.5381
ResNet152	63,624,321	0.5430	84%	23%	0.5364
ResNet152 v2	63,585,025	0.4765	83%	27%	0.4928
ResNet50	28,841,089	0.4949	95%	11%	1.0428
ResNet50 v2	28,818,177	0.5447	89%	19%	0.4766
VGG16	15,511,617	0.5369	**97%**	11%	0.9542
VGG19	20,821,313	0.5187	95%	11%	0.9527

TABLE 5: Comparison of total parameters used and performance metrics for model being presented in this paper and similar models using more standard convolutional bases.

5. Model Explainability

Deep learning outcomes can be inscrutable and arise from unexpected aspects of the inputs and so in order to trust the inferences, it is helpful to try and explain aspects of the outcomes in terms of the inputs and the process. In this section four aspects are investigated: feature importance, to determine which inputs influence the inferences most; feature strength, how strength of the TC influences the results; how results are influenced by regional location; and how the size of the training dataset used affects the model's performance.

In the discussion which follows, it is important to remember that the data used as input to TCDetect is ERA-Interim, which is itself an inaccurate representation of reality, so there will inevitably an element of discrepancy which can be explained by the data used as input, and not the deep learning tools themselves.

a. Input Feature Importance

It is possible to quantify the relative impact of each input field importance to the learned output. Two methods are employed for this: the Breimann method (Breiman 2001) and the Lakshmanan method (Lakshmanan et al. 2015).

The Briemann method involves permuting the data from one field across all the test cases and then re-testing the model with this modified dataset. A decrease in the model's performance is expected, with the most important field obtaining the largest decrease in performance.

The Lakshmanan method involves several steps: First, to permute the data as in the previous method for one field. Once the field with the most importance, i.e., the field which produces the largest decline in performance, is found, it is kept permuted, while the others fields are permuted individually. The next most important field is now found repeating the algorithm on the remaining fields. This process keeps on going until all the fields are permuted.

Both methods were performed 30 times each and an average was taken to make sure of consistent and robust results: Figure 6 shows the results. The most important field was found to be that of vorticity at 850 hPa with the Breiman method showing a decrease in AUC-PR from 0.7173 to 0.0936. Then, the Breiman method shows that rest of the ranking for the most important field is as follows: MSLP, vorticity at 600hPa, vorticity at 700hPa and 10-metre wind speed. The Lakshmanan method shows a slightly different ranking, with MSLP demoted from being the second-most important field to the fourth most-important field. Not much should be read into this slight change as the difference in AUC-PR values is minimal.

Together these suggest the most important field is that of vorticity at 850hPa and the second-most is that of vorticity at 600hPa. If this is because it is matching areas of high vorticity at 850hPa and 600hPa then this would be consistent with a physical interpretation that it is checking for deep convection.

b. Performance by Strength of Tropical Cyclone

A manual exploration of instances incorrectly classified by the deep learning model being presented here indicated that stronger tropical cyclones are picked up better. To provide quantitative support for this conclusion the recall rate was examined, stratified by the tropical cyclone category

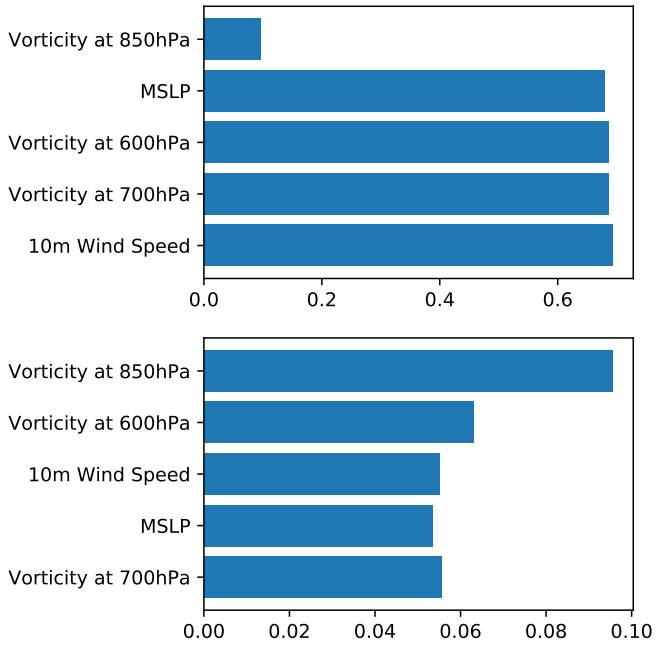


FIG. 6: Feature Importance using the Breiman (top) and Lakshamanan (bottom) methods for model trained and tested on data from all regions.

Category	**1**	**2**	**3**	**4**	**5**
Recall	88%	92%	94%	95%	100%

TABLE 6: Recall based on TC category when using validation data

on the input labels. This allowed us to examine the proportion of positively labelled cases being correctly classified as a function of labelled cyclone strength.

Table 6 shows very high recall for all Saffir-Simpsom scale (Kelman (2013)) categories, as recorded in the IBTrACS database. The model has a recall rate of 88% for tropical cyclones of Category 1 (the weakest class of TC) and a perfect recall rate for Category 5 (the strongest). As expected, an increasing trend of recall against category can be seen as higher category cyclones are easier to detect.

A possible reason for this trend of increasing recall with strength is that the deep learning model developed used data at a sixteenth of ERA-Interim's original resolution. This means that the model was using data with a resolution of around 2.8 degrees, or around 280km. While this value was chosen during the manual hyperparameter search as detailed in the Appendix, and optimises the overall results, it is likely to impact the representation of the weakest features the most.

IBTrACS Label	# cases
No meteorological system	506
Unknown	2
Post-tropical systems	18
Disturbances	165
Subtropical systems	32
Tropical Depressions	348
Tropical Storms	1095
Category 1 TCs	426
Category 2 TCs	281
Category 3 TCs	243
Category 4 TCs	212
Category 5 TCs	69

TABLE 7: The IBTrACS classification seen for the 3397 cases where TCDetect inferred the presence of a TC.

The hypothesis as to why the hyperparameter search selected such averaging is that the coarsening of the data filtered out some of the noise present in the data while still preserving the structure of any system present.

The nature of the cases labelled by TCDetect as TCs was also investigated and compared to expectations Table 7. It was found that only 506 out of the 3397 cases that obtained a positive inference from the model were cases that had no meteorological system present. This suggests that the deep learning model is picking up the required pattern needed, but is mislabelling weaker features as TCs, despite the averaging discussed earlier.

c. Impact of Location

Having investigated the impact of the strength of the features of interest and which fields influence the results the most, the next question is to what extent the results are dependent on regionality.

During development, due to time and computational constraints the manual hyperparameter tuning was carried out on only the Western Atlantic and Western Pacific (WAWP) regions (see Appendix B). When doing this, two linked assumptions were made: that any change made to the architecture which caused an improvement in performance would result in a similar improvement when the architecture was trained and tested on data from all regions; and that a model trained on data from the WAWP regions would generalise well when tested on data from all regions of the world.

Step	Model trained and tested on WAWP	Model trained on WAWP tested on Whole World	Whole World Model
Choice of Data	0.5830	0.0660	0.4928
Early Stopping	0.7111	0.0815	0.5915
Normalisation	0.7469	0.1772	0.6790
Resolution	0.7908	0.3690	0.6794
Dataset Balancing	0.7721	0.2905	0.6856
Loss and Optimiser	0.7849	0.3946	0.6733
Learning Rate Momentum	0.7980	0.6149	0.6646
Data Augmentation	0.8038	0.4457	0.6901
Data Augmentation Rate	0.8035	0.6377	0.6759
Dropout Position Dropout Rate	0.8091	0.6076	0.6832
L2 Norm Position L2 Norm Rate	0.8128	0.5331	0.6955
Batch Size	0.8176	0.6315	0.6756

TABLE 8: Results when using validation data

Model	**Training Region**	**Evaluation Region**	**AUC-PR**
WAWP	WAWP	WAWP	0.7884
WAWP	WAWP	Global	0.6491
Global	Global	Global	0.7173

TABLE 9: Changes in AUC-PR with different training and testing regions.

The first and third columns of Table 8 show that the first assumption holds, although it can be seen that the magnitude of the improvements between the two models can vary. Also, as shown in Table 9, the architecture has similar performance when trained and tested only on data from the WAWP regions and when trained and tested on data from all regions.

However, the second assumption was found to not hold. The first and second columns of Table 8 show that a model trained on data from the WAWP regions decreased in performance considerably when tested on data from all regions. This is mirrored when using the final models, as shown in Table 9.

A possible reason for this is that the background meteorological state in the WAWP regions differs from that of the whole world. Figure 7 shows the mean state from the WAWP regions in the left column and the mean state of all the other regions in the right column, with each row corresponding

Latitude	0°N - 60°N	0°N - 60°N	0°N - 60°N	0°N - 60°N	60°S - 0°S	60°S - 0°S	60°S - 0°S	60°S - 0°S
Longitude	20°E - 100°E	100°E - 180°E	180°E - 260°E	260°E - 340°E	20°E - 100°E	100°E - 180°E	180°E - 260°E	260°E - 340°E
Number of positive cases	72	400	267	250	214	113	26	0
Recall obtained by model trained on WAWP data	56.94%	90.75%	80.15%	90.80%	53.74%	58.41%	30.77%	N/A
Number of Positively Labelled cases correctly classified by model trained on WAWP data	41	363	214	227	115	66	8	N/A
Recall obtained by model trained on whole world data	88.89%	93.00%	99.25%	96.80%	80.37%	92.92%	42.31%	N/A
Number of Positively Labelled cases correctly classified by model trained on whole world data	64	372	265	242	172	105	11	N/A

TABLE 10: Evolution of accuracy during model development by basin (see text for explanation of rows).

to each variable used, i.e. MSLP, 10-metre wind speed, vorticity at 850hPa, vorticity at 700hPa and vorticity at 600hPa. The individual regions all differ significantly, which yields to the noise seen in the average. These differences could be confounding the results when the model is trained on TCs seen only in one region.

To further understand how the model trained on WAWP data differs from that trained on data from all regions, the results have been split by basin as shown in Table 10 to quantify any differences.

As expected, the model trained on WAWP data performs best on the Western Atlantic and Western Pacific regions, with a recall of 90.80% and 90.75% respectively. It also performs well in the Eastern Pacific region with a recall of 80.15%. However, all other regions do not surpass the recall rate of 60%.

On the other hand, when the model trained on data from all regions is used, all recall rates improve, some significantly. The Western Atlantic and Western Pacific regions improve their recall rates by 2.25% and 6% to 93% and 96.80% respectively. The most improved region is the one bounded by 100°E-180°E in the Southern Hemisphere, with its recall rate increasing by more than half from 58.41% to 92.92%. All but one region obtained a recall rate of at least 80%, with many surpassing a recall rate of 90%. The region that did not do well was that bounded by 180°E-260°E in the Southern Hemisphere, which obtained a recall rate of 42.31%. A possible reason for this to not perform as well as the other regions is that a smaller number of cases with TCs are available for this region, with only 26 in the test set.

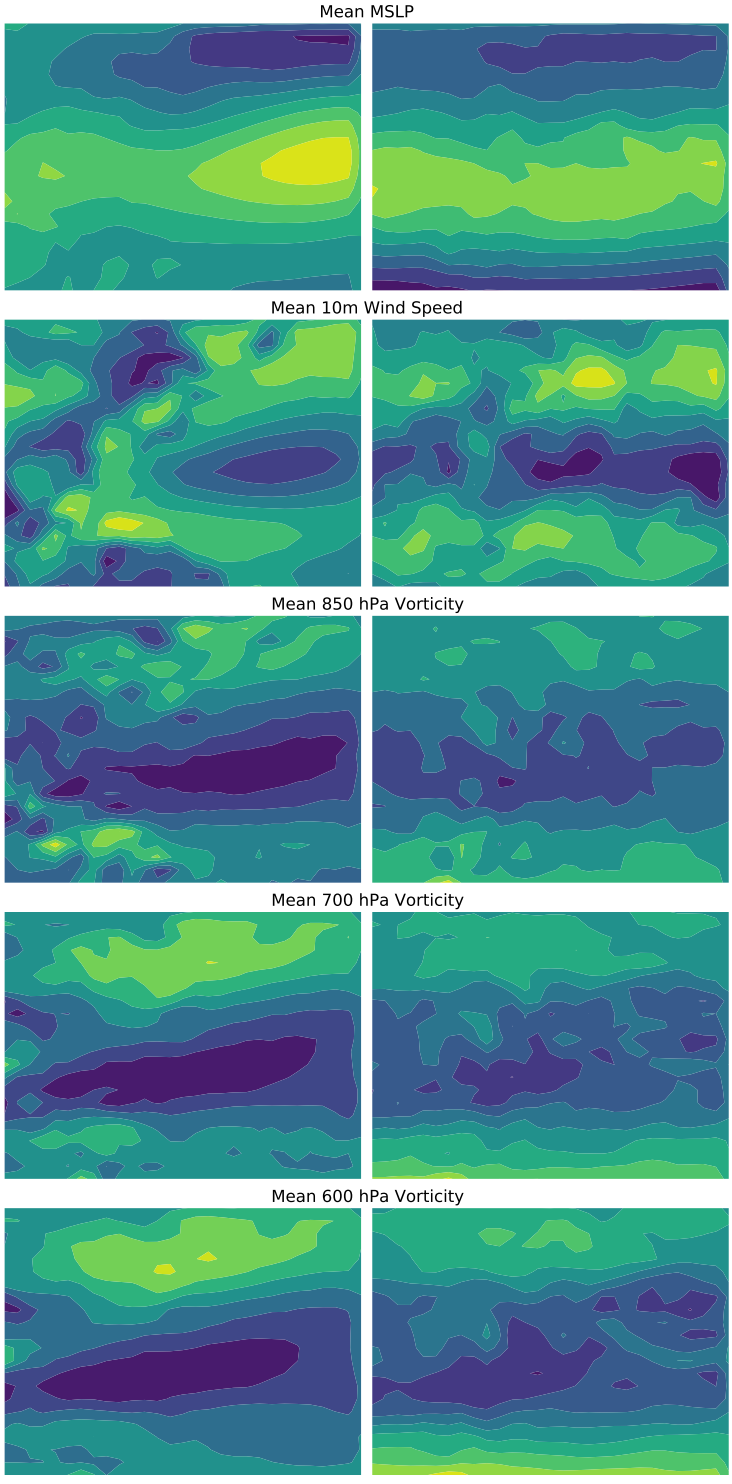


FIG. 7: Mean Case for data originating only from the Western Atlantic and Western Pacific regions (first column) and for data originating from all regions (second column). Rows show MSLP (1st row), 10-metre wind speed (2nd row), vorticity at 850hPa (3rd row), vorticity at 700hPa (4th row) and vorticity at 600hPa (5th row).

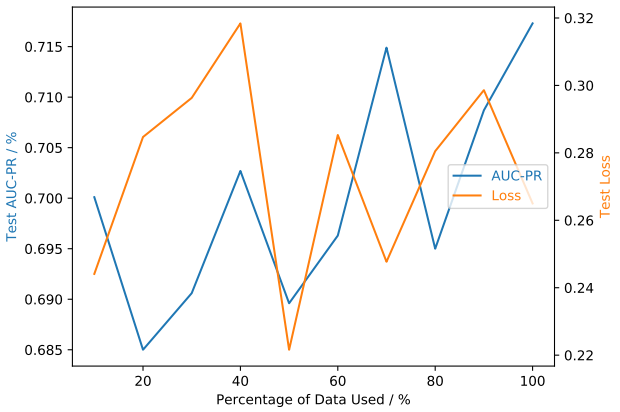


FIG. 8: Test AUC-PR and Loss for model trained and tested on data from regions around the world.

d. Size of Training Dataset

Finally, though not strictly relevant for explaining the inferences, it is interesting to address the impact of the size of the training dataset on the results. To do this, the final architecture was trained using varying amounts of the training dataset from 10% to 1000%. When training using the entire global dataset (all regions) it is seen that despite the fluctuations (Figure 8 the AUC-PR is generally increasing, while the testing loss is mostly flat, as more data is added. (The average testing loss essentially measures bad predictions, lower is better, if it is flat while the desired output is increasing, that's a good thing!) This suggests that a larger training dataset would have been beneficial.

6. Summary

A deep learning method to identify the presence or absence of Tropical Cyclones in simulation data is presented. Trained on ERA-Interim data, the deep learning model – TCDetect – obtained an AUC-PR of 0.7173 with an accuracy of 91% on a test set, which was made up of 24352 cases.

The performance of TCDetect is comparable to other standard (and more complex) models in terms of test-loss, and better in terms of the AUC-PR metric which takes in account both precision and recall. It is relatively cheap to train and run the inference, although there is a pre-processing stage which involves relatively expensive spherical harmonic smoothing.

The architecture of the model was developed using manual hyperparameter tuning on the Western Pacific and Western Atlantic basins, two of the eight regions used for labelling the presence or absence of tropical cyclones from the IBTrACS dataset. While the model trained on those regions did not generalise well to all regions, when that model architecture is used with the global training set it does better. An analysis of the impact of the size of the influence of the size of the training dataset suggests that even better results might be obtained with more training data.

TCDetect performs best with the strongest events, with a 100% recall rate for Category 5 TCs. The most important inputs were found to be vorticity at 850 and 650 hPa, suggesting that the key physical characteristic of the data which the deep learning has identified is the presence of strong deep convection. While recall increases with storm strength, there are also many tropical storms mis-identified as TCs by TCDetect. While affecting performance measured using deep learning metrics, the inclusion of such storms is not a problem from an application point of view – tropical storms are important for those studying dynamics and can also wreak significant damage in their own right.

While the training data was obtained from ERA-Interim, the ground truth used was IBTrACS, which introduces an element of uncertainty in interpreting the results – is an incorrect label (presence/absence) a consequence of the presence or absence of the TC in the ERA-Interim data or an issue with the deep learning? It is known that reanalysis data cannot resolve the full strength of storms, and so will likely undercount TCs, and hence depress the possible accuracy rates. We discuss the impact of such uncertainties in a companion paper (Galea and Lawrence 2021).

Future work includes attempting to improve TCDetect to better handle TCs of a low category potentially via ideas imported from other standard techniques or using different meteorological fields, as well as implementing an inference step using a version of the model in a full General Circulation Model to evaluate the pros and cons of avoiding data output.

Acknowledgments. 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 under the SCENARIO Doctoral Training Partnership hosted by the University of Reading. Additional support was provided by Mr Jeff Adie from NVIDIA, and Oracle Corporation.

Data availability statement. The data and code to produce the dataset, deep learning model/s and subsequent results is available to access at Galea et al. (2022). Also included is the IBTrACS version 4 dataset, obtained from Knapp et al. 2018.

APPENDIX A

Model Architecture

Table A1 gives the details of the architecture that makes up TCDetect: An input of dimensions 22 rows by 29 columns by 5 fields goes through five convolutional blocks, each made up of a convolutional layer of 8, 16, 32, 64 and 128 kernels respectively, with weights initialized using the Glorot Uniform method (Glorot and Bengio 2010) with ReLU activation functions, each with strides of 1 and a kernel size of 2x2; a dropout layer with a dropout rate of 10%; and a maximum pooling layer with strides equal to 1. The resulting kernels are flattened and passed through three fully-connected blocks, each made up of a dense layer of 128, 64 and 32 hidden nodes respectively, with L2 normalisation with a normalisation factor of 0.005, weights initialized by the Glorot Uniform method and a dropout layer with a dropout rate of 10%. TCDetect finishes off with another fully-connected layer of one node, this time using the sigmoid activation function with weights initialized by the Glorot Uniform method, as well as L2 normalisation with a normalisation factor of 0.005 which outputs a prediction. The optimizer used was the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.01 and momentum of 0.8 with the loss function being that of binary cross-entropy.

APPENDIX B

Hyperparameter Tuning

Layer (type)	Layer (specification)	Output Shape	Number of parameters
Input		22, 29, 5	
Conv2D	Glorot Uniform Weight Initialisation; ReLU Activation Function 8 Kernels; Size = 2x2; Strides - (1, 1)	21, 28, 8	168
Dropout	Dropout Rate - 0.1	21, 28, 8	
MaxPooling2D	Strides - (1, 1)	20, 27, 8	
Conv2D	Glorot Uniform Weight Initialisation; ReLU Activation Function 16 Kernels; Size = 2x2; Strides - (1, 1)	19, 26, 16	528
Dropout	Dropout Rate - 0.1	19, 26, 16	
MaxPooling2D	Strides - (1, 1)	18, 25, 16	
Conv2D	Glorot Uniform Weight Initialisation; ReLU Activation Function 32 Kernels; Size = 2x2; Strides - (1, 1)	17, 24, 32	2080
Dropout	Dropout Rate - 0.1	17, 24, 32	
MaxPooling2D	Strides - (1, 1)	16, 23, 32	
Conv2D	Glorot Uniform Weight Initialisation; ReLU Activation Function 64 Kernels; Size = 2x2; Strides - (1, 1)	15, 22, 64	8256
Dropout	Dropout Rate - 0.1	15, 22, 64	
MaxPooling2D	Strides - (1, 1)	14, 21, 64	
Conv2D	Glorot Uniform Weight Initialisation; ReLU Activation Function 128 Kernels; Size = 2x2; Strides - (1, 1)	13, 20, 128	32896
Dropout	Dropout Rate - 0.1	13, 20, 128	
MaxPooling2D	Strides - (1, 1)	12, 19, 28	
Flatten		29184	
Dense	Glorot Uniform Weight Initialisation; ReLU Activation Function 128 nodes; L2 Norm; Factor = 0.005	128	3735680
Dropout	Dropout Rate - 0.1	128	
Dense	Glorot Uniform Weight Initialisation; ReLU Activation Function 64 nodes; L2 Norm; Factor = 0.005	64	8256
Dropout	Dropout Rate - 0.1	64	
Dense	Glorot Uniform Weight Initialisation; ReLU Activation Function 32 nodes; L2 Norm; Factor = 0.005	32	2080
Dropout	Dropout Rate - 0.1	32	
Dense	Glorot Uniform Weight Initialisation; Sigmoid Activation Function 1 node; L2 Norm; Factor = 0.005	1	33

TABLE A1: The architecture of TCDetect

TCDetect was developed on data from the Western Atlantic and Western Pacific regions. The developments used the training set as described in Section a to perform 10-fold cross-validation. Each fold was then evaluated using the validation set. The metric used to assess whether a change improved the model performance was the Area-under-Curve for the Precision-Recall curve (AUC-

Step	Choice	K-fold CV Score Mean AUC-PR
Choice of Data	Filtered Data 5 Fields	0.5309
Early Stopping	Patience = 10	0.6788
Normalisation	Standarisation	0.7404
Resolution	Sixteenth	0.7842
Dataset Balancing	Undersampling with Replacement	0.7839
Loss and Optimiser	Binary Cross-Entropy Momentum	0.7890
Learning Rate Momentum	LR = 0.01 Momentum = 0.8	0.7891
Data Augmentation	Roll in x direction Rotation by random angle Flip Left-Right	0.7988
Data Augmentation Rate	0.6	0.8018
Dropout Position Dropout Rate	Conv Base & Classifier Rate = 0.1	0.8104
L2 Norm Position L2 Norm Rate	Classifier Rate = 0.005	0.8128
Batch Size	8	0.8135

TABLE B1: K-Fold Cross Validation Results

PR). Given the major class imbalance in the dataset used and that as the model is intended to be used as a filtering technique, this metric is used due to it weighting both precision and recall equally. Given this model was being developed to identify data for further post-processing, false negatives are a bigger problem than false positives, and so improvements in recall were favoured over those in precision if AUC-PR varied only marginally as a change was implemented.

Development and optimisation using this value proceeded as described below, with the final models being described and evaluated using the testing dataset in Sections 3b and a respectively. Table B1 shows a summary of the steps taken during hyperparameter tuning.

The initial architecture that was used as the starting point for developing TCDetect consisted of an input of dimensions 84 rows by 110 columns by 2 channels which passed through five convolutional blocks, each made up of a convolutional layer of 8, 16, 32, 64 and 128 kernels respectively, with weights initialized using the Glorot Uniform method with ReLU activation functions, each with

strides of 1 and a kernel size of 2x2; and a MaxPooling2D layer with strides equal to 1. The resulting kernels are flattened and passed through three fully-connected blocks, each made up of a dense layer of 128, 64 and 32 hidden nodes respectively, with weights initialized by the Glorot Uniform method. The model finishes off with another fully-connected layer of one node, this time using the sigmoid activation function with weights initialized by the Glorot Uniform method which was used to output a prediction. The optimizer used was the Stochastic Gradient Descent (SGD) optimizer with the default learning rate of 0.01 with the loss function being binary cross-entropy. Finally, a batch size of 32 cases was initially used.

a. Choice of Data

The first optimisation made was to choose the number and type of meteorological fields to supply to the model for it to make its predictions. Four possible configurations were tested:

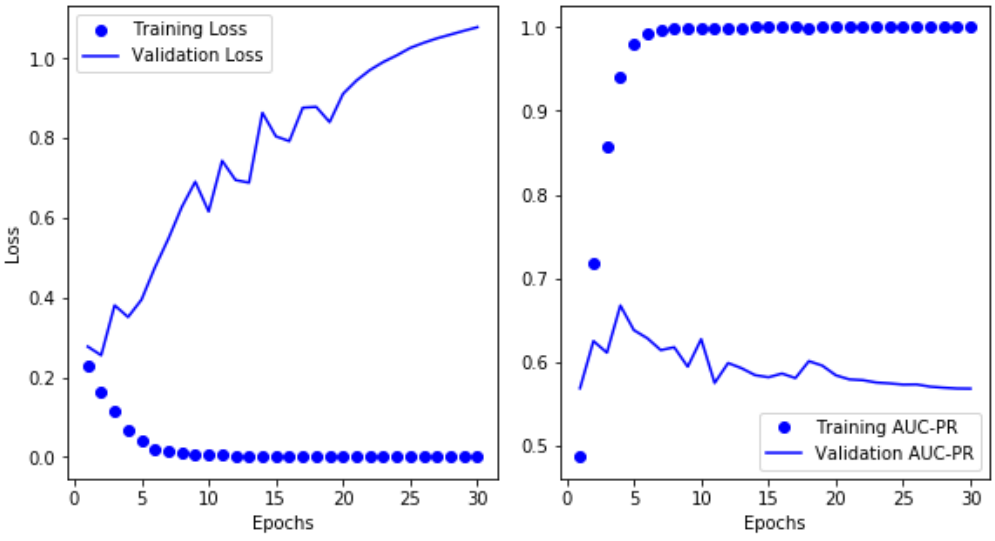
- MSLP and 10-metre wind speed
- MSLP, 10-metre wind speed and vorticity at 850hPa, 700hPa and 600hPa
- MSLP and 10-metre wind speed with spherical harmonic filtering between wave numbers 5 and 106
- MSLP, 10-metre wind speed with spherical harmonic filtering between wave numbers 5 and 106 and vorticity at 850hPa, 700hPa and 600hPa with spherical harmonic filtering between wave numbers 1 and 63

The last option provided the best mean AUC-PR, that of 0.5309.

b. Early Stopping

Next, it was noted that the model was overfitting as Figure B1a shows. Except for the first two epochs, the training loss gets smaller while the validation loss gets larger with an increasing number of epochs. Figure B1b shows similar behaviour with AUC-PR.

To overcome this issue, model training was stopped earlier by stopping training when the training and validation AUC-PR start to diverge. A number of epochs of patience, i.e. the number of epochs to wait until stopping to make sure that training was not stopped too early, were trialled to get the



(a) Loss for model trained and tested on data from the Western Atlantic and Western Pacific regions before applying Early Stopping.

(b) AUC-PR model trained and tested on data from the Western Atlantic and Western Pacific regions before applying Early Stopping.

FIG. B1: Evolution of training and validation loss and AUC-PR across the training process.

best possible performance. Patience values trailled were of 2, 5, 10 and 20 epochs. That of 10 epochs obtained the best mean AUC-PR of 0.67 88.

c. Normalisation

A few methods for normalisation were trialled, namely of normalising values to lie in the range of 0 to 1 or -1 to 1, standardising value to have a mean of 0 and a standard deviation of 1 and a combination of normalisation and standardisation. The method of standardisation produced the best model performance with a mean AUC-PR of 0.7404.

d. Resolution

Resolution of the data used was next checked. The resolution used up to the current stage was that of the original ERA-Interim dataset, but resolutions of 1.4^{o}x1.4^{o}, 2.1^{o}x2.1^{o}, 2.8^{o}x2.8^{o} and 3.5^{o}x3.5^{o} were tested. The resolution of 2.8^{o}x2.8^{o}, which was obtained by taking the mean of every four pixels of the image in both the x and y directions, produced the best mean AUC-PR of 0.7842.

e. Dataset Balancing

One problem that was known when starting hyper-parameter optimisation was that the dataset was heavily dominated by negatively labelled cases. In fact, the training dataset having data from the WAWP regions had 89.46% of the cases negatively labelled, while that having data from all regions had 95% of cases negatively labelled. This split of data would inhibit the model learning the right pattern to maximise its performance. Therefore, six ways of balancing the dataset were investigated.

- Naive Oversampling - Making copies of the positively labelled cases until the dataset is balanced.
- Undersampling without Replacement - Undersample the negatively labelled cases prior to training. Therefore, some data is not used.
- Undersampling with Replacement - Undersample the negatively labelled cases during training, so they change from epoch to epoch. Possible overfitting on positively labelled cases.
- Weighting the Cases - Weighting the cases so that the negatively labelled cases have less influence on the learning process.
- Adding Bias - Add a bias to the output layer to prevent the model from learning the bias.
- Weighting the Cases and Adding Bias - A combination of the previous two options.

The option that produced the best performance of a mean AUC-PR of 0.7839 was that of undersampling with replacement. It can be noted that the model's performance decreased marginally from the previous step, but this was still selected as recall became much favoured by the model, which is important for the use case in mind.

f. Loss and Optimiser

The model so far used the binary cross-entropy loss function with the Stochastic Gradient Descent (SGD) optimizer. All possible combinations of the mean absolute error, mean standard error and binary cross-entropy loss functions and SGD, RMSprop, SGD with Momentum using a momentum parameter of 0.9, Adam, Adagrad, Adamax and Nadam optimizers were examined.

The combination which obtained the best mean AUC-PR of 0.7890 was binary cross-entropy loss with the SGD optimiser with Momentum using a momentum parameter of 0.9.

g. Learning Rate and Momentum

A grid search for the best learning rate and momentum parameters was performed. The values for the learning rate included were those of 0.0001, 0.0005, 0.001, 0.005, 0.01 and 0.05 while those used for the momentum parameter were in the range of 0.1 to 1 with a step of 0.1. The combination which produced the best performing model was that having a learning rate of 0.01 and a momentum of 0.8

h. Data Augmentation Methods

Several techniques including random rolls, rotations, adding random noise, flipping the input data along either the x or y directions and random cropping were evaluated. The augmentation rate was set at 50%. The options which obtained a comparative or better mean AUC-PR were rolling the picture along the x-direction, flipping the picture left to right and rotating the image by a random amount. These were all included in the model and the combined methods produced a mean AUC-PR of 0.7988.

i. Data Augmentation Rate

The best data augmentation rate was also varied from 0.1 to 1 in steps of 0.1 to find the best possible rate. The best performing model with a mean AUC-PR of 0.8018 was that with an augmentation rate of 60%.

j. Dropout Position and Rate

Dropout was investigated next. It was trialled in three places, namely the convolutional base only, the fully-connected classifier only and throughout the model with dropout rates varying from 10% to 100% in steps of 10%. The model with the best AUC-PR, that of 0.8104, was that employing dropout with a rate of 10% throughout the model.

k. L2 Normalisation Position and Factor

L2 normalisation was also investigated. As with the previous optimisation, it was trailled in the same three places. The normalisation factors checked were 0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5. The model that produced the best performance with a mean AUC-PR of 0.8128 was that having L2 normalisation in the classifier only with a rate of 0.005.

l. Batch Size

This final optimisation tested was of varying the batch size. Batch sizes of 8, 16, 64, 128, 256, 512, 1024 and 2048 were tested with the first option producing the best performing model with a mean AUC-PR of 0.8135.

m. Others

Other optimisations tested which did not produce a model with an improved performance included batch normalisation, varying the number of hidden layers and nodes and using different weight initialisation methods and activation functions.

References

Abadi, M., and Coauthors, 2015: TensorFlow: Large-scale machine learning on heterogeneous systems. URL http://tensorflow.org/, software available from tensorflow.org.

Balaji, V., and Coauthors, 2018: Requirements for a global data infrastructure in support of cmip6. *Geosci. Model Dev.*, **11 (9)**, 3659–3680, https://doi.org/10.5194/gmd-11-3659-2018.

Bengtsson, L., K. I. Hodges, and M. Esch, 2007: Tropical cyclones in a t159 resolution global climate model: comparison with observations and re-analyses. *Tellus A: Dynamic Meteorology and Oceanography*, **59 (4)**, 396–416, https://doi.org/10.1111/j.1600-0870.2007.00236.x.

Breiman, L., 2001: Random forests. *Mach. Learn.*, **45 (1)**, 5–32, https://doi.org/10.1023/A:1010933404324.

Camargo, S. J., and S. E. Zebiak, 2002: Improving the detection and tracking of tropical cyclones in atmospheric general circulation models. *Weather Forecast.*, **17 (6)**, 1152–1162, https://doi.org/10.1175/1520-0434(2002)017<1152:ITDATO>2.0.CO;2.

Cangialosi, J. P., A. S. Latto, and R. Berg, 2018: Hurricane Irma (AL112017). Tech. rep., National Oceanic and Atmospheric Administration (NOAA), 111 pp. https://www.scirp.org/(S(351jmbntvnsjt1aadkposzje))/reference/ReferencesPapers.aspx?ReferenceID=2635550, as accessed on 23/03/2021.

Chollet, F., 2017: Xception: Deep learning with depthwise separable convolutions. *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, Institute of Electrical and Electronics Engineers Inc., Vol. 2017-January, 1800–1807, https://doi.org/10.1109/CVPR.2017.195, 1610.02357.

Dee, D. P., and Coauthors, 2011: The ERA-Interim reanalysis: configuration and performance of the data assimilation system. *Q. J. R. Meteorol. Soc.*, **137 (656)**, 553–597, https://doi.org/10.1002/qj.828.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, 2009: Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*, IEEE, 248–255.

Eyring, V., S. Bony, G. A. Meehl, C. A. Senior, B. Stevens, R. J. Stouffer, and K. E. Taylor, 2016: Overview of the coupled model intercomparison project phase 6 (cmip6) experimental design and organization. *Geoscientific Model Development*, **9 (5)**, 1937–1958, https://doi.org/10.5194/gmd-9-1937-2016.

Galea, D., J. Kunkel, and B. N. Lawrence, 2022: TCDetect: A new method of Detecting the Presence of Tropical Cyclones using Deep Learning. Zenodo, URL https://doi.org/10.5281/zenodo.6595071.

Galea, D., and B. N. Lawrence, 2021: Investigating differences between tropical cyclone detection systems. *Artificial Intelligence for the Earth Systems*.

Glorot, X., and Y. Bengio, 2010: Understanding the difficulty of training deep feedforward neural networks. *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*.

He, K., X. Zhang, S. Ren, and J. Sun, 2016a: Deep residual learning for image recognition. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, IEEE Computer Society, Vol. 2016-December, 770–778, https://doi.org/10.1109/CVPR.2016.90, 1512.03385.

He, K., X. Zhang, S. Ren, and J. Sun, 2016b: Identity mappings in deep residual networks. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, Springer Verlag, Vol. 9908 LNCS, 630–645, https://doi.org/10.1007/978-3-319-46493-0_38, 1603.05027.

Hodges, K. I., 1995: Feature tracking on the unit sphere. *Monthly Weather Review*, **123 (12)**, 3458–3465, https://doi.org/10.1175/1520-0493(1995)123<3458:FTOTUS>2.0.CO;2.

Hodges, K. I., 1996: Spherical nonparametric estimators applied to the ugamp model integration for amip. *Monthly Weather Review*, **124 (12)**, 2914–2932, https://doi.org/10.1175/1520-0493(1996)124<2914:SNEATT>2.0.CO;2.

Hodges, K. I., 1999: Adaptive constraints for feature tracking. *Monthly Weather Review*, **127 (6)**, 1362–1373, https://doi.org/10.1175/1520-0493(1999)127<1362:ACFFT>2.0.CO;2.

Howard, A. G., M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, 2017: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. URL http://arxiv.org/abs/1704.04861, 1704.04861.

Huang, G., Z. Liu, L. van der Maaten, and K. Q. Weinberger, 2016: Densely Connected Convolutional Networks. *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, **2017-January**, 2261–2269, URL http://arxiv.org/abs/1608.06993, 1608.06993.

Kelman, I., 2013: *Saffir–Simpson Hurricane Intensity Scale*, 882–883. Springer Netherlands, Dordrecht, https://doi.org/10.1007/978-1-4020-4399-4_306.

Kleppek, S., V. Muccione, C. C. Raible, D. N. Bresch, P. Koellner-Heck, and T. F. Stocker, 2008: Tropical cyclones in ERA-40: A detection and tracking method. *Geophys. Res. Lett.*, **35 (10)**, https://doi.org/10.1029/2008GL033880.

Knapp, K. R., H. J. Diamond, J. P. Kossin, M. C. Kruk, and C. J. Schreck, 2018: International Best Track Archive for Climate Stewardship (IBTrACS) Project, Version 4. NOAA National Centers

for Environmental Information, URL https://data.nodc.noaa.gov/cgi-bin/iso?id=gov.noaa.ncdc:C01552.

Knapp, K. R., M. C. Kruk, D. H. Levinson, H. J. Diamond, and C. J. Neumann, 2010: The international best track archive for climate stewardship (IBTrACS). *Bull. Am. Meteorol. Soc.*, **91 (3)**, 363–376, https://doi.org/10.1175/2009BAMS2755.1.

Kumler-Bonfanti, C., J. Stewart, D. Hall, and M. Govett, 2020: Tropical and extratropical cyclone detection using deep learning. *Journal of Applied Meteorology and Climatology*, **59**, 1971–1985, https://doi.org/10.1175/JAMC-D-20-0117.1.

Kurth, T., and Coauthors, 2017: Deep learning at 15pf: Supervised and semi-supervised classification for scientific data. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Association for Computing Machinery, New York, NY, USA, SC '17, https://doi.org/10.1145/3126908.3126916.

Lakshmanan, V., C. Karstens, J. Krause, K. Elmore, A. Ryzhkov, and S. Berkseth, 2015: Which polarimetric variables are important for weather/no-weather discrimination? *J. Atmos. Ocean. Technol.*, **32 (6)**, 1209–1223, https://doi.org/10.1175/JTECH-D-13-00205.1.

Lawrence, B., V. L. Bennett, J. Churchill, M. Juckes, P. Kershaw, P. Oliver, M. Pritchard, and A. Stephens, 2012: The jasmin super-data-cluster. *ArXiv*, **abs/1204.3553**.

Liu, Y., and Coauthors, 2016: Application of deep convolutional neural networks for detecting extreme weather in climate datasets. *ArXiv*, **abs/1605.01156**.

Mudigonda, M., and Coauthors, 2017: Segmenting and Tracking Extreme Climate Events using Neural Networks. *31st Conf. Neural Inf. Process. Syst.*, 1–5, URL https://dl4physicalsciences.github.io/files/nips{_}dlps{_}2017{_}20.pdf.

Otsu, N., 1979: A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, **9 (1)**, 62–66.

Prabhat, S. Byna, V. Vishwanath, E. Dart, M. Wehner, and W. D. Collins, 2015: Teca: Petascale pattern recognition for climate science. *Computer Analysis of Images and Patterns*, G. Azzopardi, and N. Petkov, Eds., Springer International Publishing, Cham, 426–436.

Prabhat, O. Rübel, S. Byna, K. Wu, F. Li, M. Wehner, and W. Bethel, 2012: TECA: A parallel toolkit for extreme climate analysis. *Procedia Comput. Sci.*, Elsevier B.V., Vol. 9, 866–876, https://doi.org/10.1016/j.procs.2012.04.093.

Racah, E., C. Beckham, T. Maharaj, S. Kahou, Prabhat, and C. Pal, 2017: Extremeweather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events. *NIPS*.

Roberts, M. J., and Coauthors, 2015: Tropical Cyclones in the UPSCALE Ensemble of High-Resolution Global Climate Models. *J. Clim.*, **28 (2)**, 574–596, https://doi.org/10.1175/JCLI-D-14-00131.1.

Sandler, M., A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, 2018: MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, IEEE Computer Society, 4510–4520, https://doi.org/10.1109/CVPR.2018.00474, 1801.04381.

Simonyan, K., and A. Zisserman, 2014: Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv 1409.1556*.

Szegedy, C., S. Ioffe, V. Vanhoucke, and A. A. Alemi, 2017: Inception-v4, inception-ResNet and the impact of residual connections on learning. *31st AAAI Conf. Artif. Intell. AAAI 2017*, AAAI press, 4278–4284, URL https://arxiv.org/abs/1602.07261v2, 1602.07261.

Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, 2016: Rethinking the Inception Architecture for Computer Vision. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, IEEE Computer Society, Vol. 2016-December, 2818–2826, https://doi.org/10.1109/CVPR.2016.308, 1512.00567.

Vitart, F., J. L. Anderson, and W. F. Stern, 1997: Simulation of interannual variability of tropical storm frequency in an ensemble of gcm integrations. *Journal of Climate*, **10 (4)**, 745–760, https://doi.org/10.1175/1520-0442(1997)010<0745:SOIVOT>2.0.CO;2.