TCDetect: a new method of detecting the presence of tropical cyclones using deep learning

TCDetect: A New Method of Detecting the Presence of Tropical Cyclones Using Deep Learning

DANIEL GALEA,[a] JULIAN KUNKEL,[b] AND BRYAN N. LAWRENCE[a,c]

[a] *Department of Computer Science, University of Reading, Reading, United Kingdom*
[b] *Gesellschaft für Wissenschaftliche Datenverarbeitung, University of Göttingen, Göttingen, Germany*
[c] *National Centre of Atmospheric Science, Department of Meteorology, University of Reading, Reading, United Kingdom*

(Manuscript received 7 June 2022, in final form 7 January 2023, accepted 6 March 2023)

ABSTRACT: Tropical cyclones are high-impact weather events that have large human and economic effects, so it is important to be able to understand how their location, frequency, and structure might change in a future climate. Here, a lightweight deep learning model is presented that is intended for detecting the presence or absence of tropical cyclones during the execution of numerical simulations for use in an online data reduction method. This will help 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 that would otherwise be required. The model was trained on ERA-Interim reanalysis data from 1979 to 2017, and the training was 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 or probability of detection rate was 92%. The precision rate or success ratio obtained was that of 36%. For the desired data reduction application, if the desired target included all tropical cyclone events, even those that did not obtain hurricane-strength status, the effective precision was 85%. The recall rate and the area under curve for the precision–recall (AUC-PR) compare favorably with other methods of cyclone identification while using the smallest number of parameters for both training and inference.

KEYWORDS: Artificial intelligence; Deep learning; Machine learning; Neural networks; Postprocessing; Tropical cyclones

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 United States in September 2017 causing 47 direct deaths, 82 indirect deaths, hundreds of injuries, and an estimated monetary damage of around USD 50 billion (Cangialosi et al. 2018). 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 large amounts of data. A method to reduce the data volumes in order to target analysis would be useful, and such a method is presented here that is based on a deep learning model that detects the presence of tropical cyclones in simulation output. Simulated time steps are split into eight regions that the deep learning model uses to infer the presence or absence of TCs. The method is intended for eventual deployment online (i.e., while a simulation model is running) to preclude the need to output data that does not include TCs (at least for the situation where TCs are the product of interest). Hence, the ability to not miss TCs is more important than maximum data reduction, as any analysis would be impaired if TC occurrences are missed. The method presented here is lightweight, with relatively short training and inference times when tested offline (i.e., after the meteorological data are output), and requires no explicit a priori thresholds in meteorological variables. It is also shown to perform similarly to other more standard, and more complex, deep learning models.

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 and 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 analyzed 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

Denotes content that is immediately available upon publication as open access.

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

DOI: 10.1175/AIES-D-22-0045.1 e220045

time and data volume saved—leading to more efficient science (efficient in time saved, storage costs, and storage energy consumption).

The possibility of efficiencies arises 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 postprocessing analysis at fixed intervals. The first postprocessing step then involves the retrieval of relevant data only, the other data are 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 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 we would abstractly consider to be TCs (or detect phenomena we would not consider to be TCs but that fall inside a poorly drawn definition). This could be seen in the differentiation between a TC of category 1 on the Saffir–Simpson scale and a tropical storm. Both show similar properties, but the main difference is the strength of the associated 10-m wind speeds, hence two definitions of a TC, one that includes 10-m wind strength and another that does not, would not produce the same outcome. Many previous techniques for TC identification in numerical data generally conflate the detection method with the definition. Deep learning can avoid this conflation as the definition needs to be clearly laid out before any deep learning work is undertaken, 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 and atmospheric rivers.

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, data, or climates effected by climate change. 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 for identifying TC centers usually work by applying various thresholds to the available data. A few examples of such methods follow, with a tabular summary in Table 1.

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

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

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 be observed within a neighborhood of an 8×8 square of grid points. This is assigned as the storm center. For it to be designated a TC center, there needs to be a maximum relative vorticity at 850 hPa above 5×10^{-5} s^{-1} at the storm center. The presence of vertical wind shear between 850 and 200 hPa of at least 10 m s^{-1} is also required, as well as an event lifetime of 36 or more hours. Finally, if the storm center is over land, the relative vorticity condition has to be fulfilled or the wind speed maximum at 850 hPa needs to be inside 250 km from the TC center.

A final example is that of Roberts et al. (2015) who use the method explained by Hodges (1995, 1996, 1999) and Bengtsson et al. (2007), where a TC is identified by a maximum of 850-hPa relative vorticity in data that have been spectrally filtered to keep features greater than 250 km in scale, and a warm-core check similar to that performed by Vitart et al. (1997) using features larger than 180 km using the 850-, 500-, 300-, and 200-hPa levels.

b. TC detection using deep learning

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

Liu et al. (2016) used an image cropped in such a way that if a TC was present, it was centered in the image. They used eight different meteorological variables: surface level pressure, meridional and zonal wind speed at 850 mb (1 mb = 1 hPa) and at the lowest available model level, temperature at 200 and 500 mb, 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, but the preprocessing cropping step involved significant noise reduction that would have helped obtain good performance.

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 created bounding boxes around the detected TCs.

TABLE 1. Overview of thresholds applied to meteorological variables for detecting and tracking tropical cyclones with the conventional techniques given.

Authors	Variable	Threshold
Vitart et al. (1997)	Relative vorticity at 850 hPa	Local maximum more than 3.5×10^{-5} s^{-1}
	MSLP	Minimum closest to relative vorticity local maximum taken as storm center
	Average temperature between 550 and 200 hPa	Closest maximum within 2° of the storm center and the temperature decreases by at least 0.5°C for at least 8° latitude in all directions
	Maximum thickness between 1000 and 200 hPa	Closest maximum within 2° of the storm center and the thickness must decrease at least 50 m for at least 8° latitude in all directions
Camargo and Zebiak (2002)	Relative vorticity at 850 hPa	More than twice the vorticity standard deviation in each basin
	Surface wind speed	More than the sum of wind speed standard deviation in each basin and the global average oceanic wind speed in a 7×7 box centered around the relative vorticity maximum
	SLP	A local minimum is present in a 7×7 box centered around the relative vorticity maximum
	Temperature anomaly	Anomaly averaged over a 7×7 box centered around the relative vorticity minimum and over the 300-, 500-, and 700-hPa pressure levels more than the basin standard deviation
		Anomaly averaged over a 7×7 box centered around the relative vorticity minimum is positive in all three of 300-, 500-, and 700-hPa pressure levels
		Anomaly averaged over a 7×7 box centered around the relative vorticity minimum is greater at 300 hPa than at 850 hPa
	Wind speed	Wind speed averaged over a 7×7 box centered around the relative vorticity minimum is greater at 850 hPa than at 300 hPa
	Distance traveled	Storm center—defined as the relative vorticity maximum—must not have traveled a distance greater than 5.6° if 6-hourly output or 8.5° if daily output
	Event time	At least 1.5 days if 6-hourly output or 2 days if daily output
Kleppek et al. (2008)	SLP	Local minimum in a neighborhood of eight grid points
	Relative vorticity at 850 hPa	More than 5×10^{-5} s^{-1} s and positioned at SLP minimum
	Vertical wind shear between 850 and 200 hPa	More than 10 m s^{-1}
	Event time	More than 36 h
	SLP minimum position	If over land, relative vorticity at 850 hPa more than 5×10^{-5} s^{-1} s and positioned at SLP minimum; otherwise, wind speed maximum at 850 hPa needs to be inside 250 km from the TC center
Roberts et al. (2015)	Relative vorticity	More than 6×10^{-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 center at all available levels between 850 and 250 hPa
	Wind speed	Wind speed averaged over a 7×7 box centered around the relative vorticity minimum is greater at 850 hPa than at 300 hPa
	Event time	At least 1.5 days
	Average temperature between 550 and 200 hPa	Closest maximum within 2° of the storm center and the temperature decreases by at least 0.5°C for at least 8° latitude in all directions
	Maximum thickness between 1000 and 200 hPa	Closest maximum within 2° of the storm center and the thickness must decrease at least 50 m for at least 8° latitude in all directions

The variables used were total precipitation rate, surface pressure, sea level pressure, reference height humidity, temperature at 200 and 500 mb, total vertically integrated precipitable water, reference height temperature, radiative surface temperature, meridional and zonal wind speed at 850 mb and at the lowest available model level, and geopotential at 100 and 200 mb and the lowest model level height. The architecture used was that of an autoencoder 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 by Kurth et al. (2017) using 9622 nodes of 68 cores each with a peak throughput of 15.04 petaflops (PF) s^{-1} and reached a sustained throughput of 13.27 PF s^{-1}, although the total time to train was not reported. The accuracy for this model was specified as the percentage of overlap between the predicted

TABLE 2. Previous deep learning models that detect and track tropical cyclones.

Authors	Data used	Ground truth	Architectures	Results
Liu et al. (2016)	CAM5.1 climate model 1979–2005 3-hourly 0.23° × 0.31° resolution ERA-Interim climate model 1979–2011 3-hourly 0.25° × 0.25° resolution Twentieth-century reanalyses 1908–48 daily 1° × 1° resolution NCEP–NCAR reanalyses 1949–2009 daily 1° × 1° resolution Images cropped to 32 × 32 pixels	TECA Manual expert labeling	Conv: 5 × 5 @ 8 Pool: 2 × 2 Conv: 5 × 5 @ 16 Pool: 2 × 2 Dense: 50 Dense: 2	99%
Racah et al. (2017)	CAM5 climate model 1979–2005 3-hourly 25-km resolution image size of 768 × 1158 pixels 16 channels Training set: Time steps during 1979 Testing set: Time steps during 1984	TECA	Encoder: Conv: 8 (layers) × 384 (height) × 576 (width) @ 64 (kernels) Conv: 8 × 192 × 288 @ 128 Conv: 8 × 96 × 144 @ 256 Conv: 8 × 48 × 72 @ 384 Conv: 8 × 24 × 36 @ 512 Conv: 8 × 12 × 18 @ 640 Decoder: Conv: 8 × 12 × 18 @ 640 Conv: 8 × 24 × 36 @ 512 Conv: 8 × 48 × 72 @ 384 Conv: 8 × 96 × 144 @ 256 Conv: 8 × 192 × 288 @ 128 Conv: 8 × 384 × 576 @ 64 Box locator: Conv: 4 × 12 × 18 @ 4 Class probabilities: Conv: 4 × 12 × 18 @ 4 Objectiveness probabilities: Conv: 4 × 12 × 18 @ 2	IOU = 0.1: 25% IOU = 0.5: 16%
Mudigonda et al. (2017)	CAM5 climate model 1996–2015 Images cropped to 96 × 144 pixels	TECA Otsu's method	Adaptation of Tiramisu model	92%
Kumler-Bonfanti et al. (2020)	GFS weather model 720 × 361 pixels 0.5° resolution	IBTrACS	U-net of 6 layers	IOU = 1: 75%

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). This is a collection of climate analysis algorithms aimed for extreme event detection and tracking. The model had 24.74% of the predicted boxes having an overlap of at least 10% with the ground truth, while 15.53% of the predicted boxes had an overlap of at least 50% with the ground truth.

Mudigonda et al. (2017) created a deep learning model that used integrated water vapor (IWV) snapshots and image segmentation techniques to classify whether each pixel in an image was a part of a TC. 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 in which at least 10% of the pixels were part of a TC and an accuracy of 92% was obtained.

Similarly, Kumler-Bonfanti et al. (2020) used a U-net network to perform image segmentation for TCs using the total precipitable water field from a weather forecast model: the National Centers for Environmental Prediction (NCEP) Global Forecast System (GFS). A U-net network is very similar to an autoencoder network, with the difference being that connections between the two branches of the network are used to convey

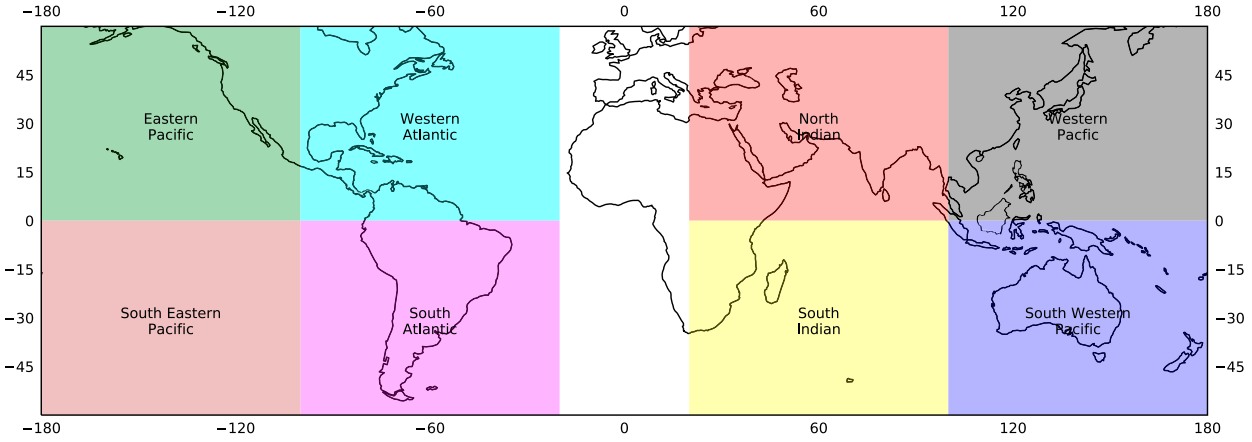

FIG. 1. Time step split into eight equal regions.

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 three hours in the future. Labels of areas belonging to a TC were generated by creating a 25 × 25 pixel box, approximating 300 km^2, around a latitude and longitude pair obtained from the International Best Track Archive for Climate Stewardship (IBTrACS; Knapp et al. 2010, 2018) dataset. This model managed to obtain an IOU of 75% but it should be noted that the labeling boxes, taken as the ground truth, were of the same size, so it is possible that the value for the IOU is inflated.

3. Deep learning model

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

a. Data

The deep learning model TCDetect uses meteorological data from a region to detect the presence of a TC. An output higher than 0.5 would signify that a TC was detected while a value of 0.5 or lower will imply the absence of a TC. TCDetect was trained, tested, and validated on data extracted from the ERA-Interim reanalysis dataset (Dee et al. 2011), with the validation data used for manual hyperparameter tuning as described in appendix B. The testing set was used for producing the final testing statistics and for interpreting the results produced by the trained model.

Five 6-hourly fields from 1 January 1979 until 31 July 2017 were used: MSLP, 10-m wind speed, and relative vorticity at 850, 700, and 600 hPa, each at a spatial resolution of ≈0.7° × 0.7°. 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. Also, the results obtained from TCDetect are compared to those obtained from a non–machine learning (ML) algorithm in a companion paper (Galea and Lawrence 2023, manuscript submitted to *Artif. Intell. Earth Syst.*); hence, the choice of inputs makes the comparison more faithful. Spherical filtering was performed on each field to reduce some of the smaller scale features. For the MSLP and 10-m wind speed fields, spherical harmonic filtering is performed to keep wavenumbers between 5 and 106. The vorticity fields were spherical harmonic filtered between wavenumbers 1 and 63. These field and spherical filtering options were chosen to match those used in the TC tracking algorithm developed in Hodges (1995, 1996, 1999).

Each field was further split into eight regions as shown in Fig. 1. These regions were loosely based on those used by the IBTrACS dataset. The regions were collected into a single dataset that TCDetect was then trained on. Thus, the entire dataset included 450 944 individual regions (14 092 days × 4 time steps per day × 8 regions per time step), each with an 86 × 114 gridpoint domain with 5 channels.

Labels for these cases were derived from the IBTrACS dataset. IBTrACS contains temporal information, including category, latitude, and longitude of the storm center, for all major storms across the globe. The labels were set up in such a way that each case was labeled according to the presence or absence of a TC recorded in IBTrACS. At the end of the labeling 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, 4853 (5.19%) with, and 88 651 (94.81%) without a TC.

Additional data from 1 August 2017 until 31 August 2019 were used as a testing set. This had a total of 24 352 cases,

TABLE 3. Dataset split.

Partition	Years included	Positive cases	Negative cases
Training	1980–81, 1982–85, 1987–90, 1992–95, 1997–2000, 2002–05, 2007–10, 2012–15, 2017	17 862 (5.00%)	339 546 (95.00%)
Validation	1979, 1986, 1991, 1996, 2001, 2006, 2011, 2016	4853 (5.19%)	88 651 (94.81%)
Testing	2017–19	1342 (5.51%)	23 010 (94.49%)

1342 (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 were preprocessed to reduce resolution to 1/16 of the original ERA-Interim resolution by taking the mean value of all data points inside a 4×4 box, reducing the dimensionality of each case to 22×29 gridpoint domain and 5 channels. This reduction of resolution was arrived at during hyperparameter tuning. For each channel (variable), standardization was employed on each individual field 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 that field for that case. The resulting standardized 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, 1800 UTC 28 August 2005.

b. Architecture

TCDetect uses a convolutional base attached to a fully connected classifier that outputs a value between 0 and 1, with any values larger than 0.5 signifying that the model detects a TC. A more detailed explanation of the model architecture can be found in appendix A, while a graphical view is shown in Fig. 3.

To arrive at the model architecture presented, stepwise 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

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

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. Had accuracy been the chosen metric, 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

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

could have been used, as a high value would indicate that the number of false negatives, that is, not detected TC cases, is small. However, this would not have considered the need to detect non-TC cases as such. For this, precision, defined by

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

could be used as a high value would indicate that the number of false positives, that is, 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 that 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 favored 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 32 GB 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 Joint Analysis System Meeting Infrastructure Needs (JASMIN) platform (Lawrence et al. 2012). The software packages used were Python 3.6.8 and Tensorflow, version 2.20 (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 min. Although the time taken to train the final model was relatively short, much more time was taken up in progressing through the various optimizations detailed in appendix B, mainly due to the use of tenfold 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

TCDetect correctly classified 1231 (91.73%) of the 1342 cases as having a TC and 20 844 (90.59%) of the 23 010

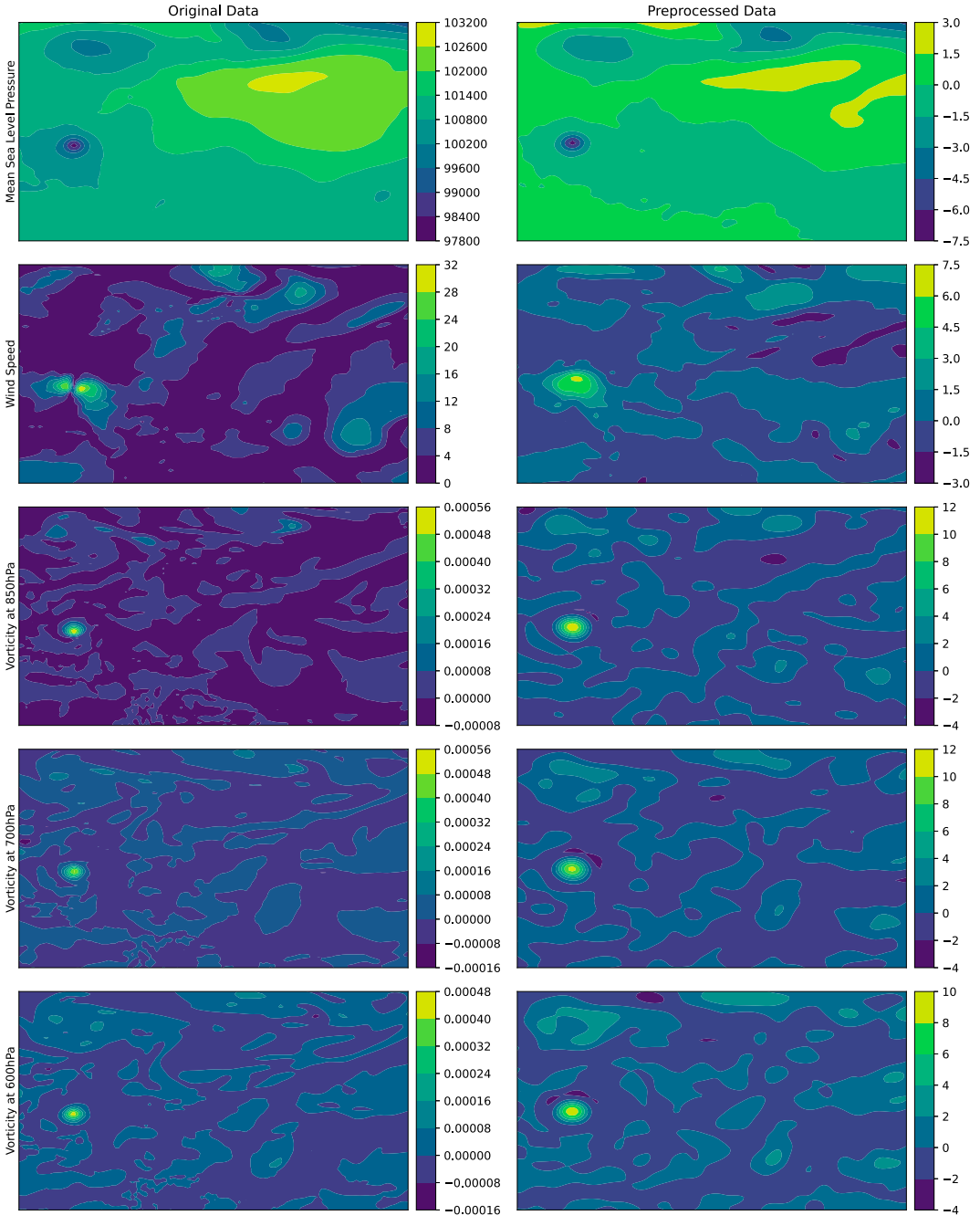

FIG. 2. An example of the data that were used to train TCDetect. The data are from the reanalysis at the point where Hurricane Katrina obtained its maximum strength (1800 UTC 28 Aug 2005). (left) The original data from ERA-Interim and (right) how these data are transformed after preprocessing. The rows are as follows (in reverse order of height above the surface): MSLP, 10-m wind speed, and the vorticity at 850, 700, and 600 hPa.

without. It misclassified 111 (8.27%) TC cases and 2166 (9.41%) non-TC cases. These are summarized in the confusion matrix in Table 4.

These results result in an accuracy of 90.65%, a recall rate of 91.73% and a precision rate of 36.24%. Hyperparameter tuning was performed to maximize AUC-PR. This resulted in a high recall rate and a sufficiently high precision rate for this model to be used in a data reduction method.

The outcome could have been further varied by changing the value that 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

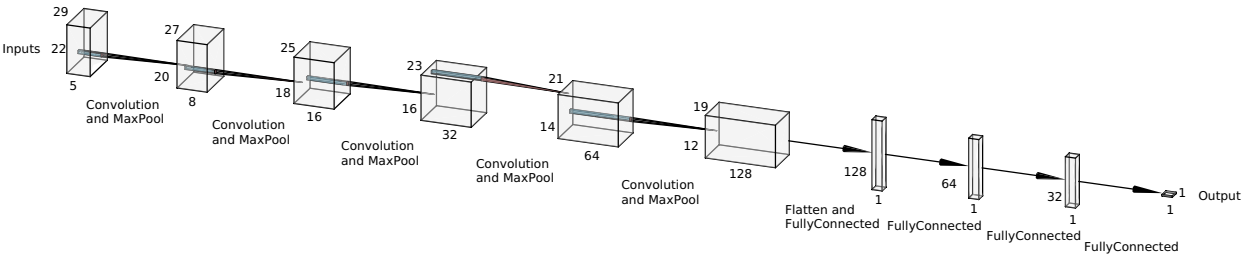

FIG. 3. Visual representation of the architecture of TCDetect. The inputs, having 22 rows, 29 columns, and 5 fields, are passed through a 2×2 convolution window whose weights are learnt producing 8 feature maps, but losing one row and column. The resulting feature maps are passed through a MaxPool window, of size 2×2, 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 three more times, with each step producing more feature maps. The final ones 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 three more times, ending with the final value.

the boundary at which the corresponding recall and precision rates are obtained.

b. *Comparison with standard models*

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

To test this, the convolutional base, that is, the part of the network that 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, DenseNet169, and DenseNet201 (Huang et al. 2017); InceptionResNetV2 (Szegedy et al. 2017); InceptionV3 (Szegedy et al. 2016); MobileNet (Howard et al. 2017); ResNet101, ResNet101V2, ResNet152, ResNet50, and ResNet50V2 (He et al. 2016a,b); VGG16 and 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 normalization factor.

Given that these convolutional bases required inputs of at least 75 pixels $\times$ 75 pixels with 3 channels, some changes to the inputs were required. First, as an input with only 3 channels is required for most of the above architectures, the fields retained were those of relative vorticity at 850 hPa, relative vorticity at 600 hPa, 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. Second, the input size was extended fivefold from 22×29 to 110×145 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 (Fig. 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 TCDetect, and the latter also outperforms the others in terms of AUC-PR, precision rate and loss. While the recall rate obtained by TCDetect does not outperform all of the other models, it is in the top third of the list.

This shows that standard image processing models that are pretrained on a benchmark dataset did not perform very well when applied to meteorological data. This is expected as the latter has different properties than, for example, images of animals, and the standard models had not been trained with any meteorological data. Therefore, the previous analysis was

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

		TCDetect	
		Yes	No
Labeled	Yes	1231	111
	No	2166	20 844

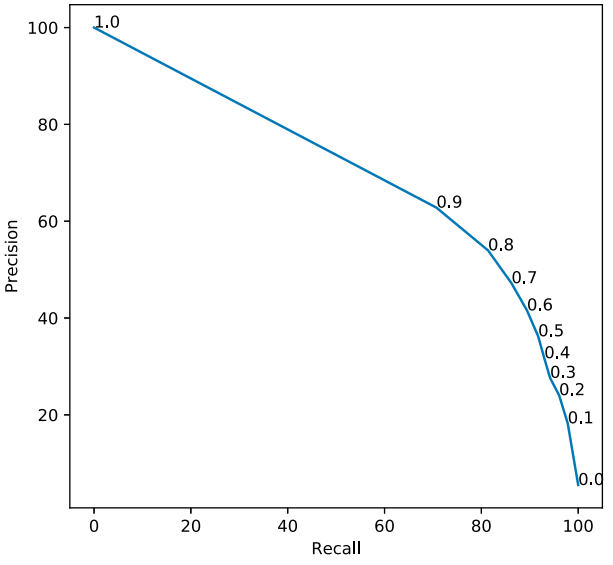

FIG. 4. Precision–recall curve for final trained model evaluated on the testing dataset.

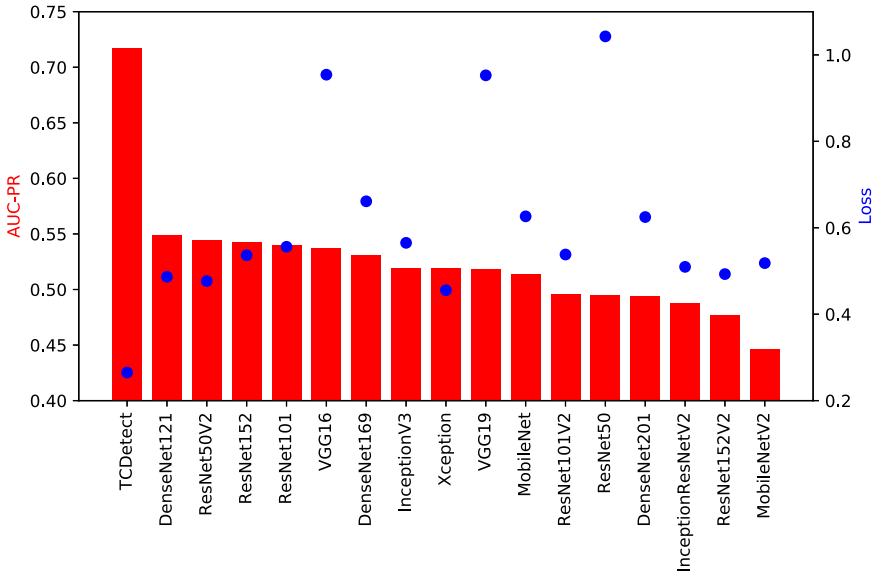

FIG. 5. Test AUC-PR (bars) and test loss (points) for standard convolutional bases, pretrained on the ImageNet database, attached to the fully connected classifier developed in the presented model. The classifier was retrained for each convolutional base with data from all regions.

redone—this time with the whole network retrained, not just the fully connected classifier. When this is performed, a different result is obtained. As can be seen in Fig. 6 and tabulated in Table 6, the AUC-PR and loss values for all models are much closer. Similarly, the recall and precision rates are very close across the different models. Encouragingly, TCDetect is only outperformed by 1% for recall rate but the maximum precision obtained is 44% by ResNet50, 8% higher than TCDetect. This shows that standard models pretrained on standard image processing benchmark datasets are not optimized for meteorological data. This analysis also shows that TCDetect is comparable is standard deep learning architectures when these are trained on the same meteorological dataset, despite having less complexity in terms of the number of parameters used in the network.

c. *Comparison with threshold-based method*

It is reasonable to ask what, if any, value deep learning adds beyond simple threshold-based methods. To address this, and recognizing that most existing threshold-based methods usually use the meteorological parameter of relative vorticity at 850 hPa as one of the criteria when detecting and tracking TCs, we use this variable for a simple comparison.

As we discussed in section 2, the value of a cutoff for such threshold is subjective and can be data dependent, and so we investigate a range of possible thresholds. A case is classified

TABLE 5. Comparison of total parameters used and performance metrics for TCDetect and similar networks using more standard convolutional bases.

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

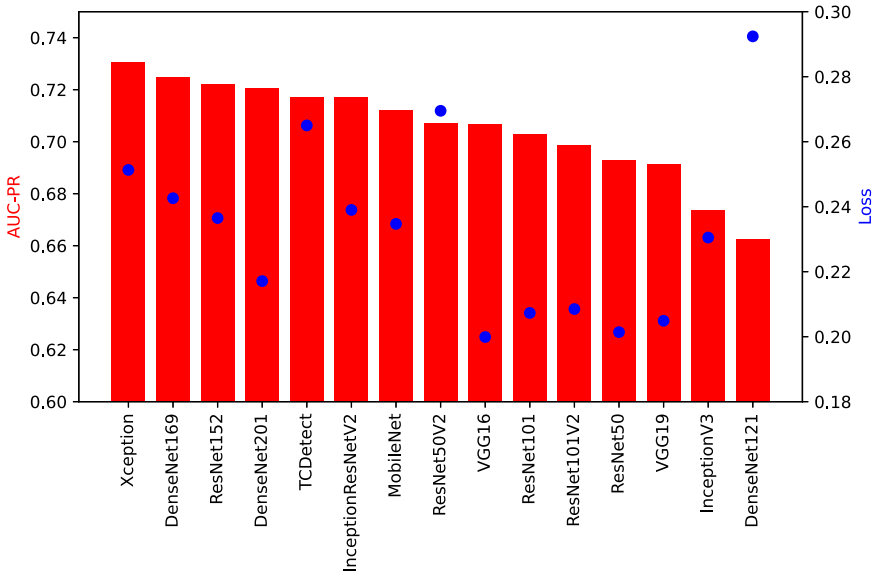

FIG. 6. Test AUC-PR (bars) and test loss (points) for standard convolutional bases attached to the fully connected classifier developed in the presented network. The whole network was retrained for each convolutional base with data from all regions.

as having a TC present if the threshold is exceeded by any value in that area at that time. The results of this exercise are tabulated in Table 7 and similar results were observed for the training and validation datasets (not shown).

Clearly this is a fast method, and it is capable of getting either good accuracy or good recall, but it struggles with precision (as do all techniques). Given for our use case we want the best possible recall, the precision rate of 6% at high recall values is far below that which can be achieved with deep learning (36%).

One might then ask, what about more sophisticated threshold and tracking methods, and what is it about the meteorology that drives these results? We address these issues in a companion paper (Galea et al. 2023, manuscript submitted to *Artif. Intell. Earth Syst.*), which compares TCDetect with a more sophisticated method, and investigates the interaction of the meteorology with the detection technique.

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 the strength of a 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.

TABLE 6. Comparison of total parameters used and performance metrics for TCDetect and similar networks using more standard convolutional bases.

Convolutional base	Total parameters	AUC-PR	Recall	Precision	Loss
TCDetect	3 789 977	0.7173	92%	36%	0.2650
DenseNet121	8 620 865	0.6626	93%	38%	0.2924
DenseNet169	15 209 281	0.7249	93%	35%	0.2426
DenseNet201	21 821 601	0.7204	91%	36%	0.2171
InceptionResNet v2	55 526 881	0.7170	92%	41%	0.2390
Inception v3	23 386 145	0.6735	93%	43%	0.2305
MobileNet	4 812 225	0.7121	91%	39%	0.2347
ResNet101	47 911 553	0.7030	91%	38%	0.2073
ResNet101 v2	47 879 937	0.6988	93%	42%	0.2085
ResNet152	63 624 321	0.7220	90%	36%	0.2365
ResNet50	28 841 089	0.6927	93%	44%	0.2014
ResNet50 v2	28 818 177	0.7072	89%	33%	0.2695
VGG16	15 511 617	0.7067	92%	39%	0.1999
VGG19	20 821 313	0.6912	91%	37%	0.2049
Xception	26 060 329	0.7306	90%	35%	0.2513

TABLE 7. Classification metrics when using a threshold on the field of relative vorticity at 850 hPa.

Cutoff (10^{-4} s^{-1})	Accuracy (%)	Recall (%)	Precision (%)	F1 (%)
1	6	100	6	10
2	24	88	6	11
3	69	50	9	15
4	88	23	14	17
5	93	8	19	11
6	94	3	30	6
7	94	1	30	2
8	94	0	8	1
9	94	0	0	0
10	94	0	0	0

In the discussion that 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. It does not give a perfect reconstruction of history due to physical processes that are not able to be simulated, so there will inevitably be an element of discrepancy that can be explained by the data used as input, and not the deep learning tools themselves. Also, as a reanalysis dataset, ERA-Interim is a compromise between observations and modeling. This is most notable in the strength of 10-m wind speeds, which do not match reality (since they represent values over a larger spatial scale than the observations). This introduces noise in the data that may inhibit TCDetect from obtaining better performance. Finally, not all TCs that occurred are part of IBTrACS and strength classification might not be correct due to a lack of observational data.

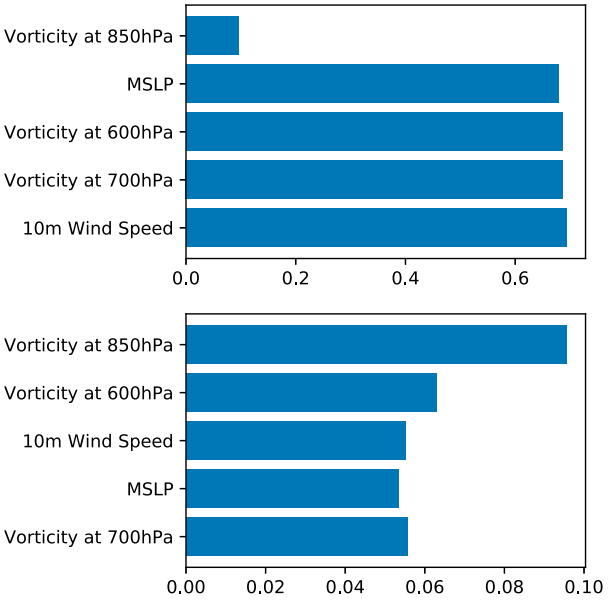

FIG. 7. Feature importance using the (top) Breiman and (bottom) Lakshamanan methods for model trained and tested on data from all regions.

TABLE 8. Number of cases having a TC and associated recall rate stratified by TC category when using test data.

Category	1	2	3	4	5
No.	484	307	258	224	69
Recall	88%	92%	94%	95%	100%

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 Breiman method (Breiman 2001) and the Lakshmanan method (Lakshmanan et al. 2015).

The Brieman method, sometimes called the single-pass permutation test, involves permuting the data from one field across all the test cases and then retesting 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, sometimes called the multipass permutation test, involves several steps: First, to permute the data as in the previous method for one field. Once the field with the most importance, that is, the field that produces the largest decline in performance, is found, it is kept permuted, while the other fields are permuted individually. The next most important field is now found by repeating the algorithm on the remaining fields. This process is repeated until all the fields are permuted.

Both methods were performed 30 times each using the testing dataset and an average was taken to make sure of consistent and robust results: Fig. 7 shows the results. The most important field was found to be that of relative vorticity at 850 hPa, similar to what Roberts et al. (2015) found, with the Breiman method showing a decrease in AUC-PR from 0.7173 to 0.0936. Then, the Breiman method shows the rest of the ranking for the most important field as follows: MSLP, relative vorticity at 600 hPa, relative vorticity at 700 hPa and 10-m 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.

TABLE 9. The IBTrACS classification seen for the 3397 cases where TCDetect inferred the presence of a TC.

IBTrACS label	No. cases
No meteorological system	506
Unknown	2
Posttropical 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 10. Results when using validation data.

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
Normalization	0.7469	0.1772	0.6790
Resolution	0.7908	0.3690	0.6794
Dataset balancing	0.7721	0.2905	0.6856
Loss and optimizer	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

Together these suggest the most important field is that of relative vorticity at 850 hPa and the second-most is that of relative vorticity at 600 hPa. If this is because it is matching areas of high relative vorticity at 850 and 600 hPa, then this would be consistent with a physical interpretation that it is checking for horizontal flow deformations associated with 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 detected with more success. To provide quantitative support for this conclusion, the recall rate stratified by the tropical cyclone category on the input labels was examined. This allowed us to examine the proportion of positively labeled cases being correctly classified as a function of labeled cyclone strength.

Table 8 shows very high recall for all Saffir–Simpson 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 1/16 of ERA-Interim's original resolution. This means that the model was using data with a resolution of around 2.8°, or around 280 km. While this value was chosen during the manual hyperparameter search as detailed in appendix B, and optimizes the overall results, it is likely to impact the representation of the weakest features the most.

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 labeled by TCDetect as TCs was also investigated and compared to expectations in Table 9. 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 has learnt the required pattern but is mislabeling 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: any change made to the architecture that caused an improvement in performance would result in a similar improvement when the architecture was trained and tested on data from all regions; a model trained on data from the WAWP regions would generalize well when tested on data from all regions of the world.

The first and third columns of Table 10 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 11, 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 not to hold. The first two columns of Table 10 show that a model trained on WAWP data decreased in performance considerably when tested on data from all regions. This is mirrored when using the final models, as shown in Table 11.

A possible reason for this is that the background meteorological state in the WAWP regions differs from that of the whole world. Figure 8 shows the mean state from the WAWP regions in the left column and the mean state of all regions in

TABLE 11. Changes in AUC-PR with different training and testing regions.

Model	Training region	Evaluation region	AUC-PR
WAWP	WAWP	WAWP	0.7884
WAWP	WAWP	Global	0.6491
Global	Global	Global	0.7173

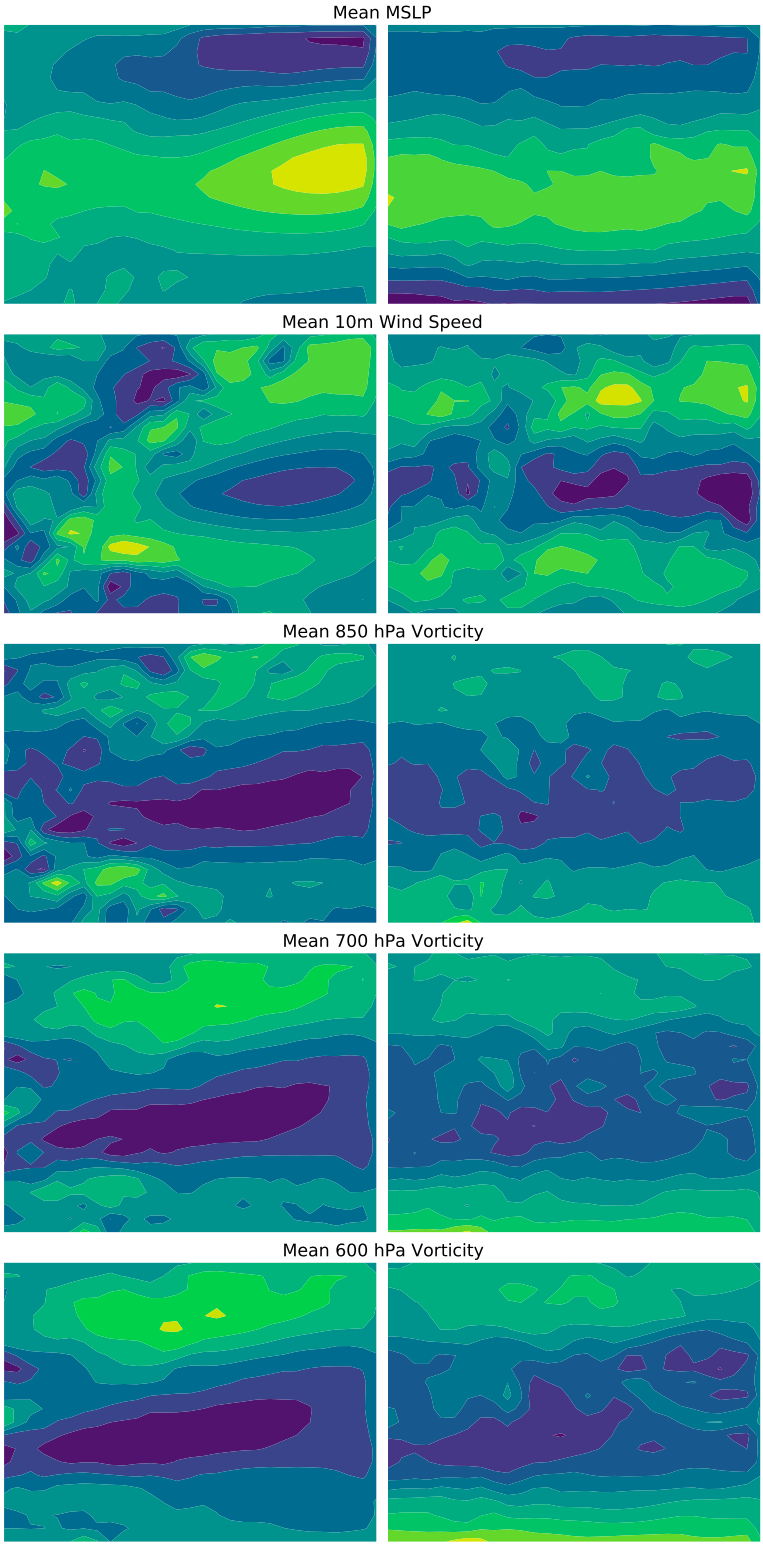

FIG. 8. Mean case (left) for data originating only from the western Atlantic and western Pacific regions and (right) for data originating from all regions. (first row) MSLP, (second row) 10-m wind speed, (third row) vorticity at 850 hPa, (fourth row) vorticity at 700 hPa, and (fifth row) vorticity at 600 hPa.

TABLE 12. Evolution of accuracy during model development by basin (see text for explanation of rows).

Latitude Longitude	0°–60°N 20°–100°E	0°–60°N 100°E–180°	0°–60°N 180°–260°E	0°–60°N 260°–340°E	60°–0°S 20°–100°E	60°S–0° 100°E–180°	60°S–0° 180°–260°E	60°S–0° 260°–340°E
No. 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%	—
No. of positively labeled cases correctly classified by model trained on WAWP data	41	363	214	227	115	66	8	—
Recall obtained by model trained on whole world data	88.89%	93.00%	99.25%	96.80%	80.37%	92.92%	42.31%	—
No. of positively labeled cases correctly classified by model trained on whole world data	64	372	265	242	172	105	11	—

the right column, with each row corresponding to each variable used, that is, MSLP, 10-m wind speed and relative vorticity at 850, 700, and 600 hPa. 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 and shown in Table 12.

As expected, the model trained on WAWP data performs best in 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° 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°–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.

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

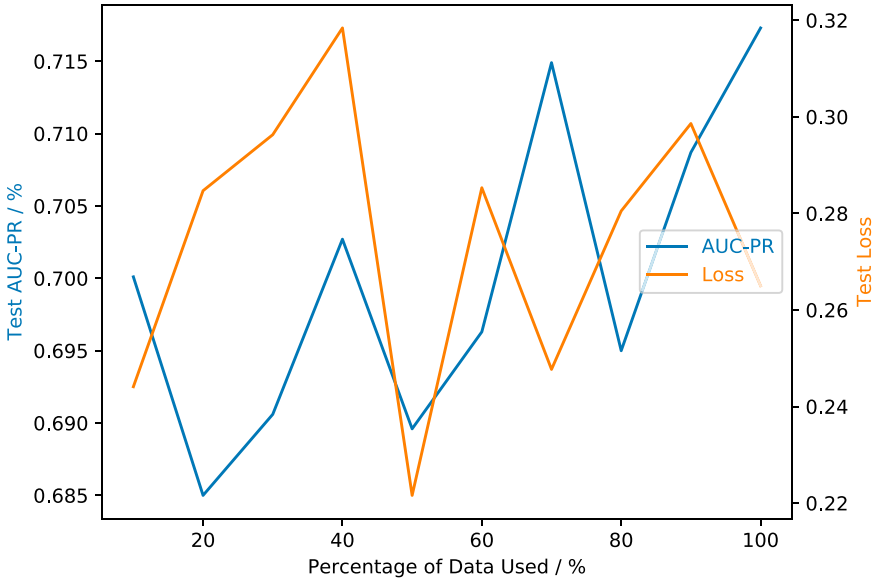

FIG. 9. Test AUC-PR and loss for model trained and tested on data from regions around the world.

was trained using varying amounts of the training dataset from 10% to 100%. When training using the entire global dataset (all regions) it is seen that despite the fluctuations, (Fig. 9) the AUC-PR is generally increasing, while the testing loss is mostly flat, as more data are added. (The average testing loss essentially measures bad predictions: lower is better; if it is flat while the desired output is increasing, that is 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 a recall rate of 93% and a precision rate of 36% on a test set, which was made up of 24 352 cases. As TCDetect is intended to be used during a GCM simulation, not missing TCs (i.e., recall) was deemed more important than obtaining maximum data reduction (i.e., precision) as a lack of recall would impair any future analysis of data output.

The performance of TCDetect is comparable to other standard (and more complex) models in terms of test loss and the AUC-PR metric that takes into account both precision and recall. It is relatively cheap to train and run the inference, although there is a preprocessing stage that 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 labeling the presence or absence of tropical cyclones from the IBTrACS dataset. While the model trained on those regions did not generalize 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 relative vorticity at 850 and 650 hPa, suggesting that the key physical characteristic of the data the deep learning has identified is the presence of strong deep convection. While recall increases with storm strength, there are also many tropical storms misidentified 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 were 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 or IBTrACS 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 et al. 2023, manuscript submitted to *Artif. Intell. Earth Syst.*).

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 U.K. Government Department for Business, Energy and Industrial Strategy (BEIS) National Productivity Investment Fund (NPIF), Grant 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 × 29 columns × 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 2 × 2; 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 normalization with a normalization factor of 0.005, weights initialized by the Glorot Uniform method and a dropout layer with a dropout rate of 10%. TCDetect ends 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 normalization with a normalization factor of 0.005 that 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

TCDetect was developed on data from the western Atlantic and western Pacific regions. The developments used the training set as described in section 3a to perform tenfold cross validation. Each fold was selected such that consecutive time steps, which are bound to be similar, are mostly kept in singular folds to reduce data leakage across

TABLE A1. The architecture of TCDetect.

Layer (type)	Layer (specification)	Output shape	No. of parameters
Input		22, 29, 5	
Conv2D	Glorot uniform weight initialization; ReLU activation function 8 kernels; size = 2 × 2; 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 initialization; ReLU activation function 16 kernels; size = 2 × 2; 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 initialization; ReLU activation function 32 kernels; size = 2 × 2; 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 initialization; ReLU activation function 64 kernels; size = 2 × 2; 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 initialization; ReLU activation function 128 kernels; size = 2 × 2; strides—(1, 1)	13, 20, 128	32 896
Dropout	Dropout rate—0.1	13, 20, 128	
MaxPooling2D	Strides—(1, 1)	12, 19, 28	
Flatten		29 184	
Dense	Glorot uniform weight initialization; ReLU activation function 128 nodes; L2 norm; factor = 0.005	128	37 35 680
Dropout	Dropout rate—0.1	128	
Dense	Glorot uniform weight initialization; ReLU activation function 64 nodes; L2 norm; factor = 0.005	64	8256
Dropout	Dropout rate—0.1	64	
Dense	Glorot uniform weight initialization; ReLU activation function 32 nodes; L2 norm; factor = 0.005	32	2080
Dropout	Dropout rate—0.1	32	
Dense	Glorot uniform weight initialization; sigmoid activation function 1 node; L2 norm; factor = 0.005	1	33

folds. Each fold was then evaluated using the validation set. The metric used to assess whether a change improved the model performance was the AUC-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 because it weights both precision and recall equally. Also, due to the metric chosen, it was ensured that each fold used in the cross-validation process had a similar ratio of positive cases to negative cases that represented that of the full training set. Finally, as this model was being developed to identify data for further postprocessing, false negatives are a bigger problem than false positives, so improvements in recall were favored over those in precision if AUC-PR varied only marginally as a change was implemented.

Development and optimization using this value proceeded as described below, with the final models being described and evaluated using the testing dataset in sections 3b and 4, 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 × 110 columns × 2 channels that passed through 5 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 2 × 2; 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 that was used to output a prediction. The optimizer used was the 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 optimization made was to choose the number and type of meteorological fields to supply as inputs to the model. Four possible configurations were tested:

- MSLP and 10-m wind speed

TABLE B1. *K*-fold cross-validation results.

Step	Choice	*K*-fold CV score mean AUC-PR
Choice of data	Filtered data, 5 fields	0.5309
Early stopping	Patience = 10	0.6788
Feature scaling	Standardization	0.7404
Resolution	1/16	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

- MSLP, 10-m wind speed and relative vorticity at 850, 700 and 600 hPa
- MSLP and 10-m wind speed with spherical harmonic filtering between wavenumbers 5 and 106
- MSLP, 10-m wind speed with spherical harmonic filtering between wavenumbers 5 and 106 and relative vorticity at 850, 700, and 600 hPa with spherical harmonic filtering between wavenumbers 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 Fig. 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 behavior to AUC-PR.

To overcome this issue, model training was stopped when the training and validation AUC-PR started to diverge. A number of epochs of patience, that is, the number of epochs to wait until stopping to make sure that training was not stopped too early, were trialed to get the best possible performance. Patience values trialed were of 2, 5, 10, and 20 epochs. That of 10 epochs obtained the best mean AUC-PR of 0.6788.

c. Feature scaling

A few methods for normalization were trialed, namely normalizing values to lie in the range from 0 to 1 or from −1 to 1, standardizing value to have a mean of 0 and a standard deviation of 1 and a combination of normalization and standardization. The method of standardization produced the best model performance with a mean AUC-PR of 0.7404.

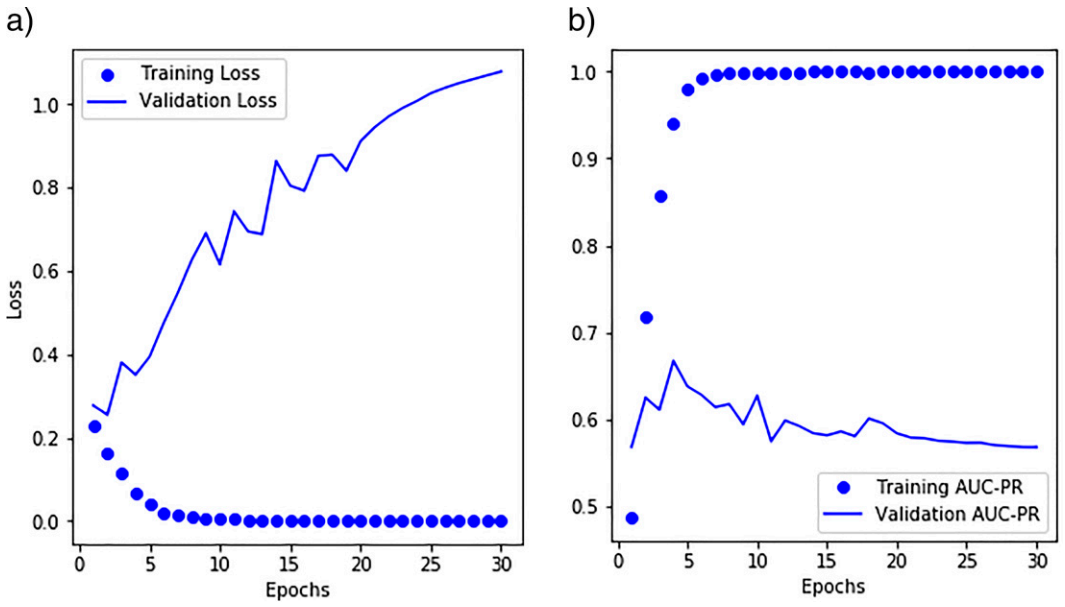

FIG. B1. Evolution of training and validation loss and AUC-PR across the training process. (a) Loss for model trained and tested on data from the western Atlantic and western Pacific regions before applying early stopping. (b) AUC-PR model trained and tested on data from the western Atlantic and western Pacific regions before applying early stopping.

d. Resolution

Resolution of the data used was checked next. The resolution used up to the current stage was that of the original ERA-Interim dataset, but resolutions of 1.4° × 1.4°, 2.1° × 2.1°, 2.8° × 2.8°, and 3.5° × 3.5° were tested. The resolution of 2.8° × 2.8°, 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 optimization was that the dataset was heavily dominated by negatively labeled cases. In fact, the training dataset having data from the WAWP regions had 89.46% of the cases negatively labeled, while that having data from all regions had 95% of cases negatively labeled. This split of data would inhibit the model learning the right pattern to maximize its performance. Therefore, six ways of balancing the dataset were investigated:

- Naive oversampling—Making copies of the positively labeled cases until the dataset is balanced.
- Undersampling without replacement—Undersample the negatively labeled cases prior to training, therefore, some data are not used.
- Undersampling with replacement—Undersample the negatively labeled cases during training, so they change from epoch to epoch; possible overfitting on positively labeled cases.
- Weighting the cases—Weighting the cases so that the negatively labeled 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.

Undersampling with replacement produced the best performance with a mean AUC-PR of 0.7839. It can be noted that the model's performance decreased marginally from the previous step, but this was still selected as recall became much more favored by the model, which is important for the data reduction method in mind.

f. Loss and optimizer

The model so far used the binary cross-entropy loss function with the SGD optimizer. All possible combinations of the mean absolute error, mean standard error, and binary cross-entropy loss functions and SGD, SGD with momentum using a momentum parameter of 0.9 (Qian 1999), RMSProp,[A1] Adam (Kingma and Ba 2014), Adagrad (Duchi et al. 2011), Adamax (Kingma and Ba 2014), and Nadam (Dozat 2016) optimizers were examined.

Binary cross-entropy loss with the SGD optimizer with momentum using a momentum parameter of 0.9 obtained the best mean AUC-PR of 0.7890.

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 0.1–1 with a step of 0.1. The combination that 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 to 50%. The options that 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 10% to 100% in steps of 10% 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 trialed 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 normalization position and factor

L2 normalization was also investigated. It was trialed in the same three places as in the previous optimization. The normalization factors checked were 0.000 01, 0.000 05, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, and 0.5. The model that produced the best performance with a mean AUC-PR of 0.8128 was that having L2 normalization in the classifier only with a rate of 0.005.

l. Batch size

The final optimization tested was of that for 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 optimizations tested that did not produce a model with an improved performance included batch normalization, varying the number of hidden layers and nodes, and

[A1] Presented in http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

using different weight initialization methods and activation functions.

REFERENCES

Abadi, M., and Coauthors, 2015: TensorFlow: Large-scale machine learning on heterogeneous systems. arXiv, 1603.04467v2, https://doi.org/10.48550/arXiv.1603.04467.

Balaji, V., and Coauthors, 2018: Requirements for a global data infrastructure in support of CMIP6. *Geosci. Model Dev.*, **11**, 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*, **59A**, 396–416, https://doi.org/10.1111/j.1600-0870.2007.00236.x.

Breiman, L., 2001: Random forests. *Mach. Learn.*, **45**, 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. *Wea. Forecasting*, **17**, 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: National Hurricane Center tropical cyclone report Hurricane Irma. NOAA Tech. Rep. AL112017, 111 pp., https://www.nhc.noaa.gov/data/tcr/AL112017_Irma.pdf.

Chollet, F., 2017: Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conf. Computer Vision Pattern Recognition*, Honolulu, HI, IEEE, 1800–1807, https://doi.org/10.1109/CVPR.2017.195.

Dee, D. P., and Coauthors, 2011: The ERA-Interim reanalysis: Configuration and performance of the data assimilation system. *Quart. J. Roy. Meteor. Soc.*, **137**, 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 Conf. on Computer Vision and Pattern Recognition*, Miami, FL, IEEE, 248–255, https://doi.org/10.1109/CVPR.2009.5206848.

Dozat, T., 2016: Incorporating Nesterov momentum into Adam. *Int. Conf. on Learning Representations 2016*, San Juan, Puerto Rico, ICLR, 43, https://openreview.net/pdf/OM0jvwB8jIp57ZJjtNEZ.pdf.

Duchi, J., E. Hazan, and Y. Singer, 2011: Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, **12**, 2121–2159.

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. *Geosci. Model Dev.*, **9**, 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, accessed 30 May 2022, https://doi.org/10.5281/zenodo.6595071.

Glorot, X., and Y. Bengio, 2010: Understanding the difficulty of training deep feedforward neural networks. *Proc. Int. Conf. on Artificial Intelligence and Statistics*, Sardinia, Italy, PMLR, 249–256, http://proceedings.mlr.press/v9/glorot10a.

He, K., X. Zhang, S. Ren, and J. Sun, 2016a: Deep residual learning for image recognition. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, IEEE, 770–778, https://doi.org/10.1109/CVPR.2016.90.

——, ——, ——, and ——, 2016b: Identity mappings in deep residual networks. *Computer Vision—ECCV 2016*, B. Leibe et al., Eds., Lecture Notes in Computer Science, Vol. 9908, Springer, 630–645, https://doi.org/10.1007/978-3-319-46493-0_38.

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

——, 1996: Spherical nonparametric estimators applied to the UGAMP model integration for AMIP. *Mon. Wea. Rev.*, **124**, 2914–2932, https://doi.org/10.1175/1520-0493(1996)124<2914:SNEATT>2.0.CO;2.

——, 1999: Adaptive constraints for feature tracking. *Mon. Wea. Rev.*, **127**, 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. arXiv, 1704.04861v1, https://doi.org/10.48550/arXiv.1704.04861.

Huang, G., Z. Liu, L. van der Maaten, and K. Q. Weinberger, 2017: Densely connected convolutional networks. *IEEE Conf. on Computer Vision Pattern Recognition*, Honolulu, HI, IEEE, 2261–2269, https://doi.org/10.1109/CVPR.2017.243.

Kelman, I., 2013: Saffir–Simpson hurricane intensity scale. *Encyclopedia of Natural Hazards*, P. T. Bobrowsky, Ed., Springer, 882–883, https://doi.org/10.1007/978-1-4020-4399-4_306.

Kingma, D. P., and J. Ba, 2014: Adam: A method for stochastic optimization. arXiv, 1412.6980v9, https://doi.org/10.48550/arXiv.1412.6980.

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**, L10705, https://doi.org/10.1029/2008GL033880.

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. Amer. Meteor. Soc.*, **91**, 363–376, https://doi.org/10.1175/2009BAMS2755.1.

——, 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, accessed 30 May 2022, https://data.nodc.noaa.gov/cgi-bin/iso?id=gov.noaa.ncdc:C01552.

Kumler-Bonfanti, C., J. Stewart, D. Hall, and M. Govett, 2020: Tropical and extratropical cyclone detection using deep learning. *J. Appl. Meteor. Climatol.*, **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. *Proc. Int. Conf. for High Performance Computing, Networking, Storage and Analysis*, Denver, CO, Association for Computing Machinery, 1–11, 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. Oceanic Technol.*, **32**, 1209–1223, https://doi.org/10.1175/JTECH-D-13-00205.1.

Lawrence, B. N., V. Bennett, J. Churchill, M. Juckes, P. Kershaw, P. Oliver, M. Pritchard, and A. Stephens, 2012: The Jasmin super-data-cluster. arXiv, abs/1204.3553v1, https://doi.org/10.48550/arXiv.1204.3553.

Liu, Y., and Coauthors, 2016: Application of deep convolutional neural networks for detecting extreme weather in climate

datasets. arXiv, 1605.01156v1, https://doi.org/10.48550/arXiv.1605.01156.

Mudigonda, M., and Coauthors, 2017: Segmenting and tracking extreme climate events using neural networks. *31st Conf. Neural Information Processing System*, Long Beach, CA, Association for Computing Machinery, 1–5, https://dl4physicalsciences.github.io/files/nips_dlps_2017_20.pdf.

Otsu, N., 1979: A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man Cybern.*, **9**, 62–66, https://doi.org/10.1109/TSMC.1979.4310076.

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.*, **9**, 866–876, https://doi.org/10.1016/j.procs.2012.04.093.

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, 426–436.

Qian, N., 1999: On the momentum term in gradient descent learning algorithms. *Neural Networks*, **12**, 145–151, https://doi.org/10.1016/S0893-6080(98)00116-6.

Racah, E., C. Beckham, T. Maharaj, S. E. Kahou, Prabhat, and C. Pal, 2017: ExtremeWeather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events. arXiv, 1612.02095v2, https://doi.org/10.48550/arXiv.1612.02095.

Roberts, M. J., and Coauthors, 2015: Tropical cyclones in the UPSCALE ensemble of high-resolution global climate models. *J. Climate*, **28**, 574–596, https://doi.org/10.1175/JCLI-D-14-00131.1.

Simonyan, K., and A. Zisserman, 2014: Very deep convolutional networks for large-scale image recognition. arXiv, 1409.1556v6, https://doi.org/10.48550/arXiv.1409.1556.

Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, 2016: Rethinking the inception architecture for computer vision. *Proc. IEEE Conf. on Computer Vision Pattern Recognition*, Las Vegas, NV, IEEE, 2818–2826, https://doi.org/10.1109/CVPR.2016.308.

——, S. Ioffe, V. Vanhoucke, and A. Alemi, 2017: Inception-v4, inception-ResNet and the impact of residual connections on learning. *31st AAAI Conf. Artificial Intelligence*, Mountain View, CA, AAAI, 4278–4284, https://doi.org/10.1609/aaai.v31i1.11231.

Vitart, F., J. L. Anderson, and W. F. Stern, 1997: Simulation of interannual variability of tropical storm frequency in an ensemble of GCM integrations. *J. Climate*, **10**, 745–760, https://doi.org/10.1175/1520-0442(1997)010%3c0745:SOIVOT%3e2.0.CO;2.