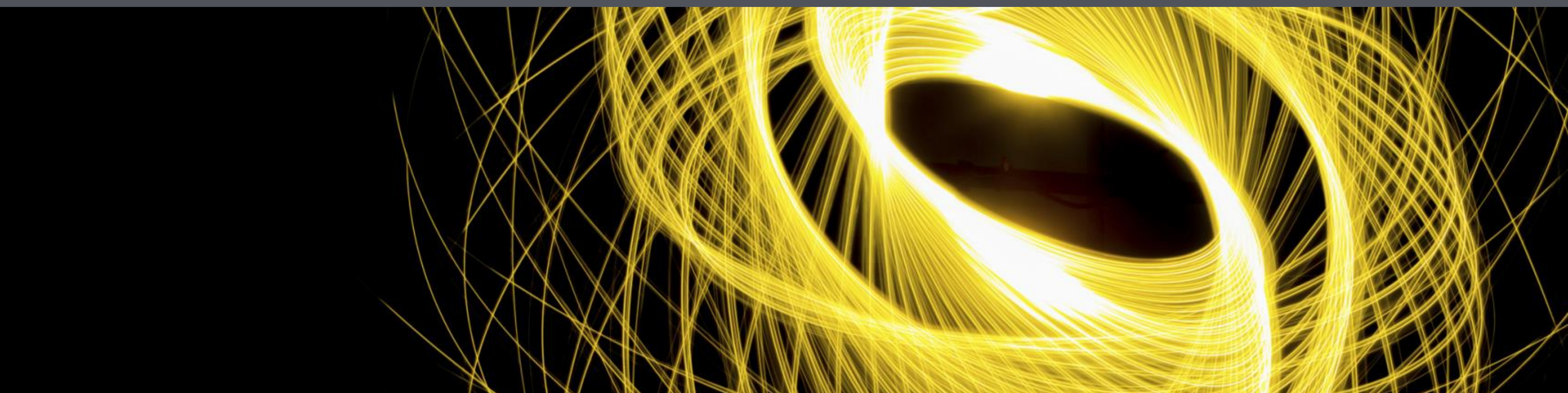
Department of Meteorology

University of Reading

Meteorological Data Filtering for Tropical Cyclones using Deep Learning



Daniel Galea, Bryan Lawrence

Scenario DOCTORAL TRAINING PARTNERSHIP

UKRI Natural Environment Research Council

Copyright University of Reading

Outline

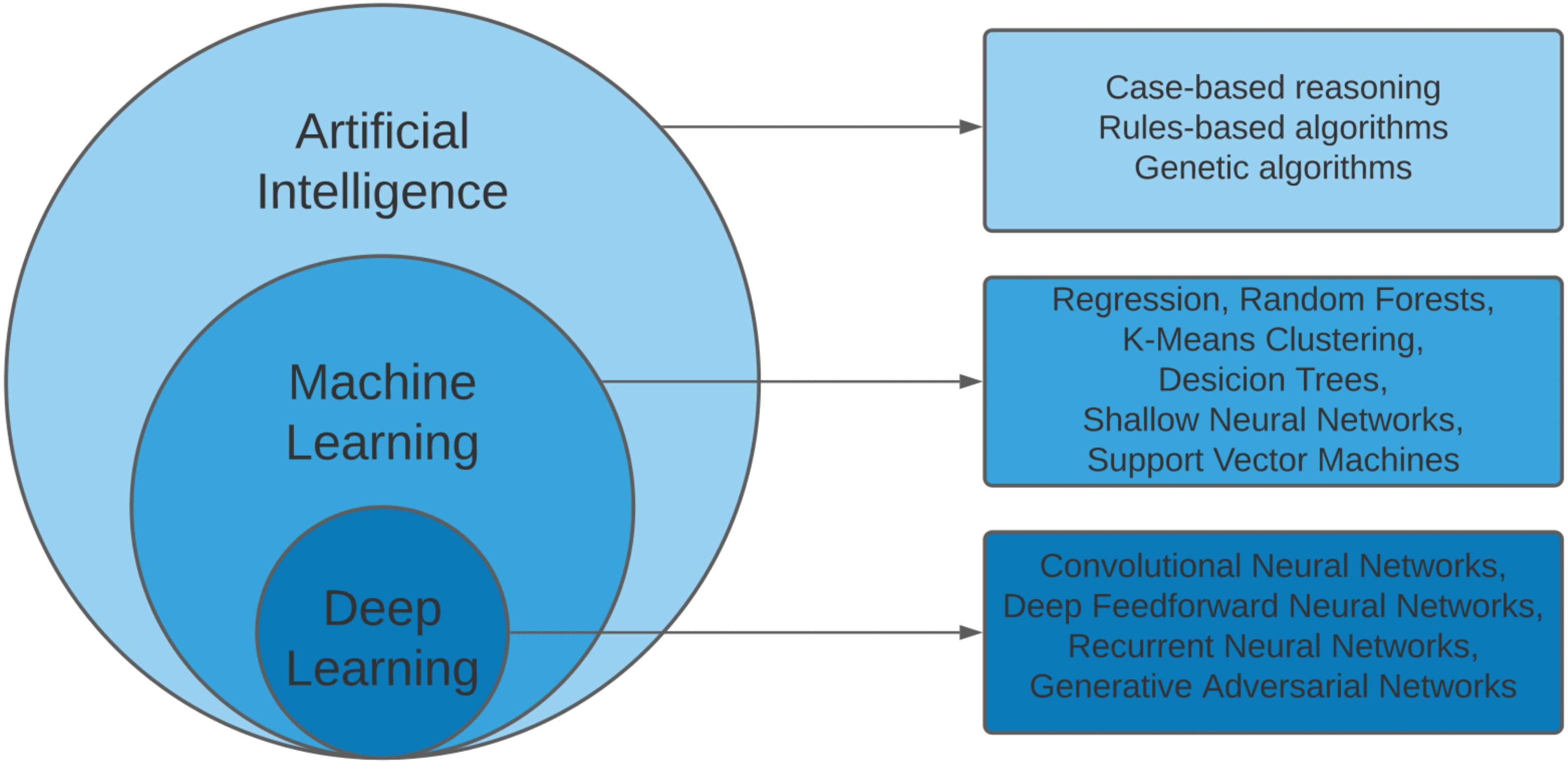
- Introduction and Aims
- Deep Learning
- Detecting Tropical Cyclones with Deep Learning
- Verification of Deep Learning algorithm
- Implementation of Deep Learning algorithm in a GCM
- Summary and Future Work

Introduction and Aims

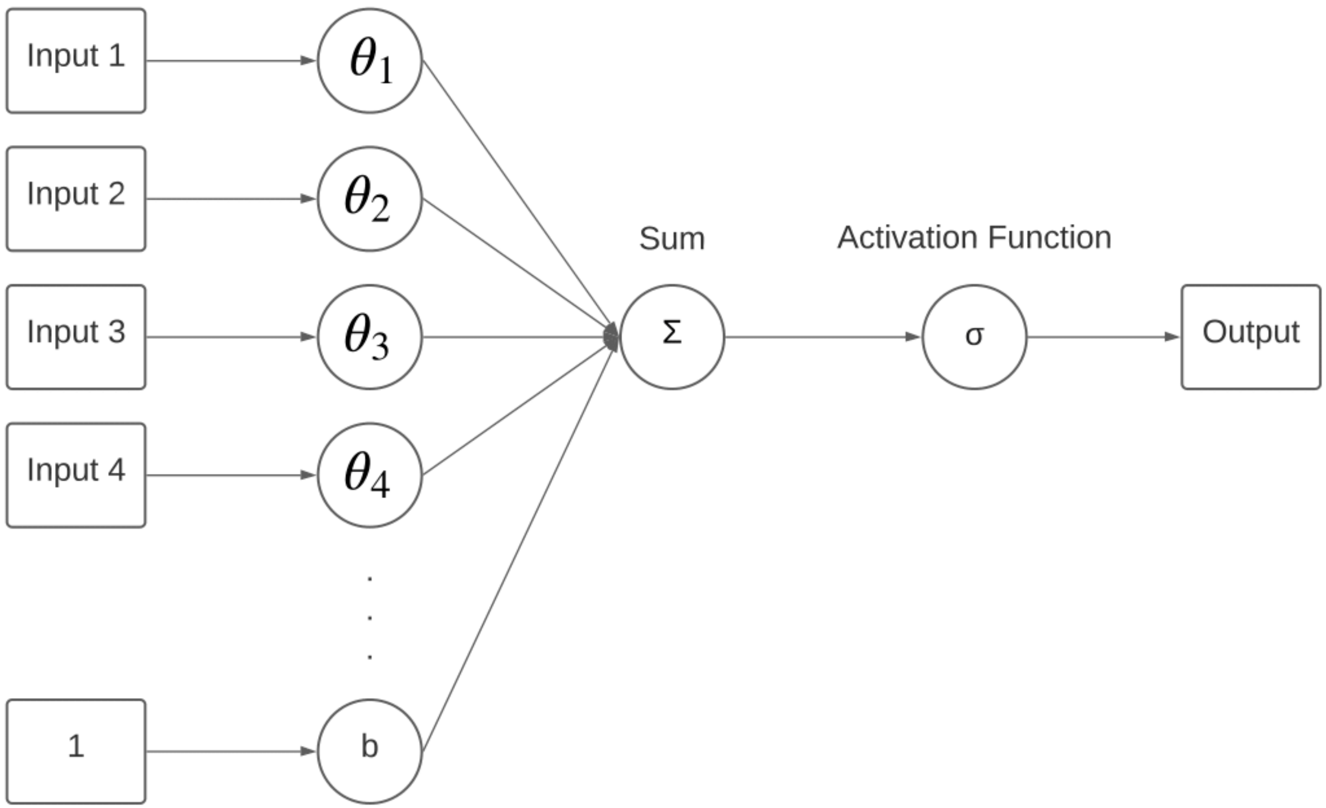
- The effects of a changing climate on multiple meteorological phenomena are being investigated by long General Circulation Model (GCM) simulations
- Each simulation produces large amounts of data which can be inefficient to store and analyse
- We created a method that scans GCM analysis data for the presence of Tropical Cyclones (TCs) and only outputs it to disk if a TC is detected

Deep Learning

What is Deep Learning?



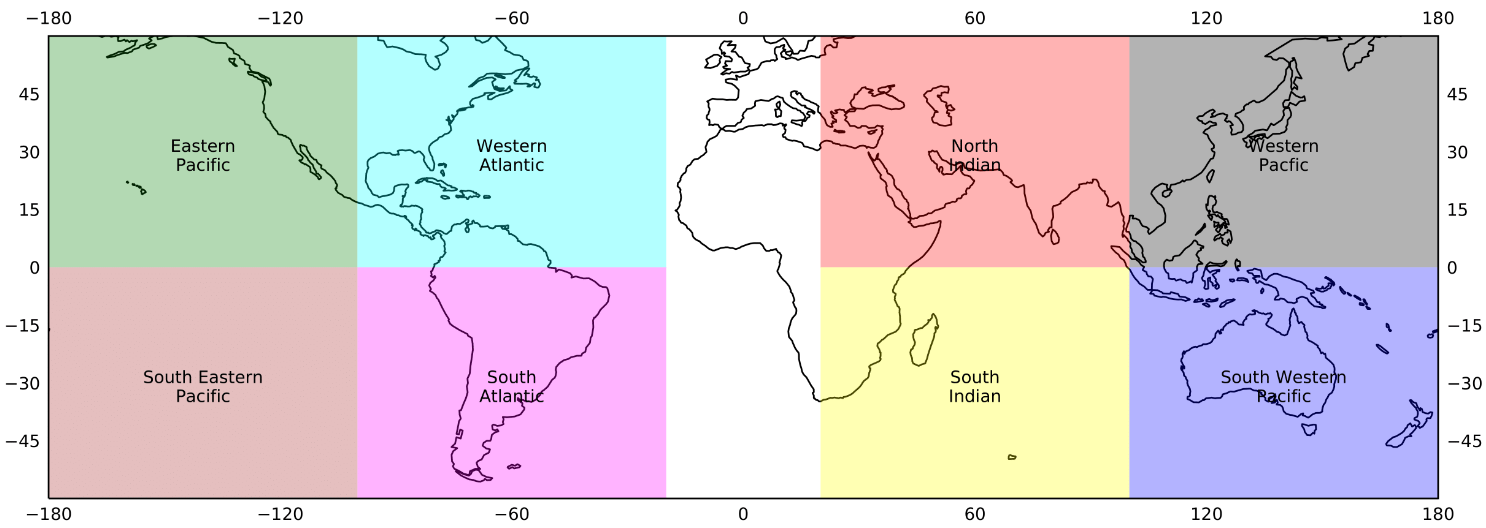
Neural Networks

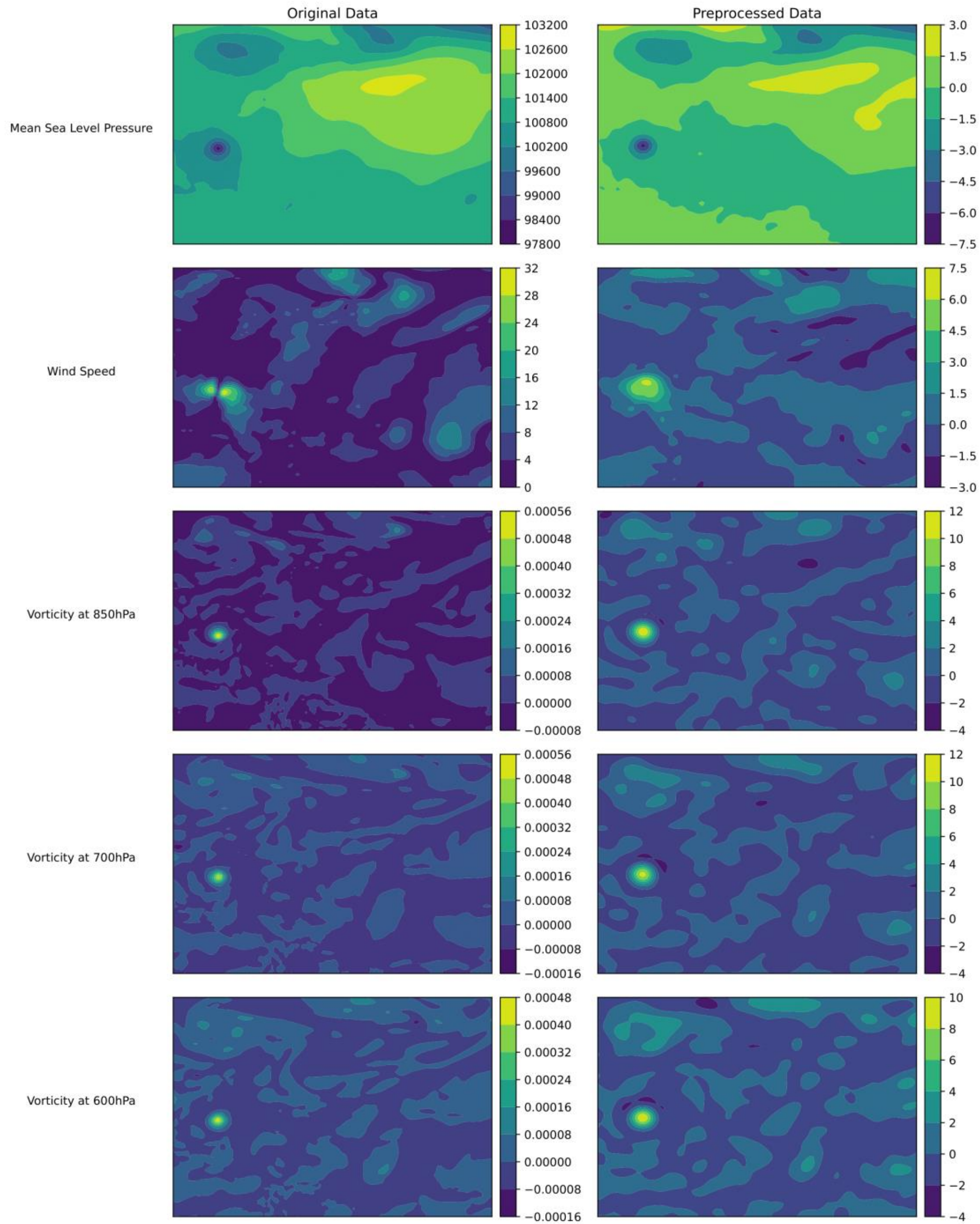


Detecting TCs with Deep Learning

Data

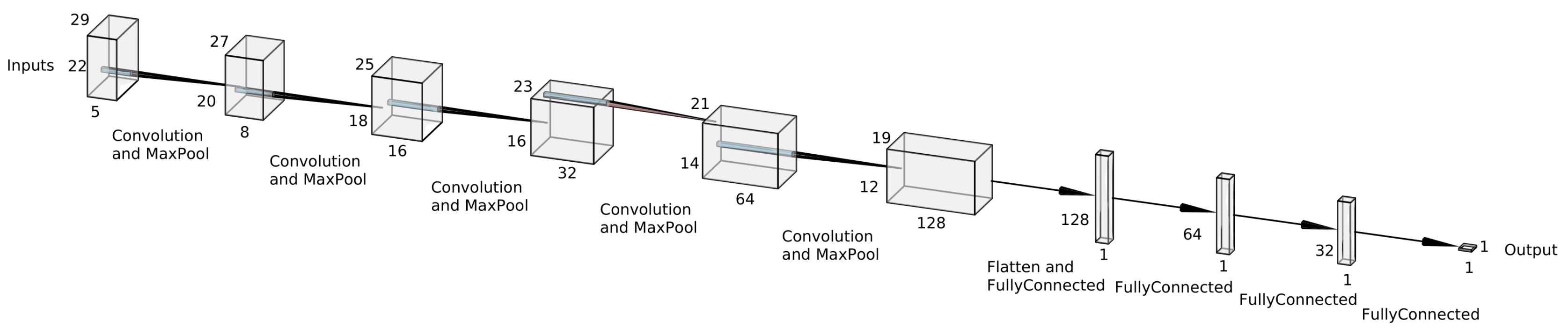
- ERA-Interim reanalyses dataset; each timestep split into 8 regions
- Fields used: 10m wind speed; MSLP; Vorticity at 850hPa, 700hPa, 600hPa at a resolution of 2.8º
- Labels obtained from the IBTrACS database
- Training Set: January 1979 – June 2017 (450912 cases)
- Testing Set: July 2017 – August 2019 (24352 cases)




Original Data
Preprocessed Data
Mean Sea Level Pressure
103200 102600 102000 101400 100800 100200 99600 99000 98400 97800
3.0 1.5 0.0 −1.5 −3.0 −4.5 −6.0 −7.5
Wind Speed
32 28 24 20 16 12 8 4 0
7.5 6.0 4.5 3.0 1.5 0.0 −1.5 −3.0
Vorticity at 850hPa
0.00056 0.00048 0.00040 0.00032 0.00024 0.00016 0.00008 0.00000 −0.00008
12 10 8 6 4 2 0 −2 −4
Vorticity at 700hPa
0.00056 0.00048 0.00040 0.00032 0.00024 0.00016 0.00008 0.00000 −0.00008 −0.00016
12 10 8 6 4 2 0 −2 −4
Vorticity at 600hPa
0.00048 0.00040 0.00032 0.00024 0.00016 0.00008 0.00000 −0.00008 −0.00016
10 8 6 4 2 0 −2 −4

Deep Learning Model: Architecture

- CNN-based classifier, termed TCDetect
- Convolutional Base: 5 blocks using 2D convolution, dropout, glorot uniform weight initialisation, ReLU activation function
- Classifier: 4-layer fully-connected network using dropout, glorot uniform weight initialisation, ReLU activation function, L2 normalisation
- Final layer uses a softmax function to output a single value



Deep Learning Model: Optimisations

- Choice of Data – Filtered Fields; 5 fields
- Early Stopping – 10 steps
- Normalisation - Standardisation
- Resolution - Sixteenth
- Dataset Balancing – Undersampling with replacement
- Loss Function – Binary Cross-Entropy
- Optimiser – Momentum
- Learning and Momentum Rates – 0.01 and 0.8
- Data Augmentation – Roll in x-direction; Random rotation; LR flip
- Dropout Position and Rate – Whole model, 10%
- L2 Position and Rate – Classifier; 0.5%
- Batch Size - 8

Results

- An accuracy of 90.65% was obtained when testing on data from July 2017 until August 2019
- 1231 out of 1342 (91.73%) positive cases were correctly classified

		Identified	
		TC Present	TC Not Present
Ground Truth	TC Present	1231	111
	TC Not Present	2166	20844

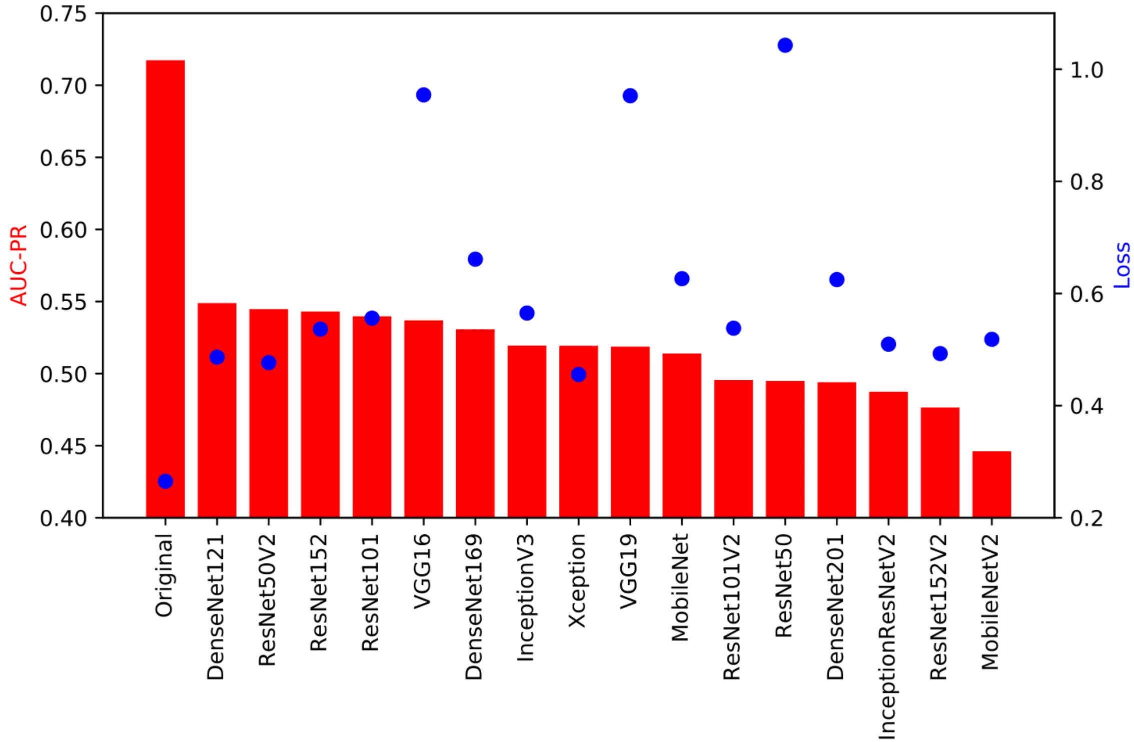
Results: TC Category

Category	Global Model
1	88.02%
2	91.53%
3	94.19%
4	94.64%
5	100.00%

- This upward trend of recall with category shows that TCs of higher categories are being identified better as they have features which are more easily identifiable

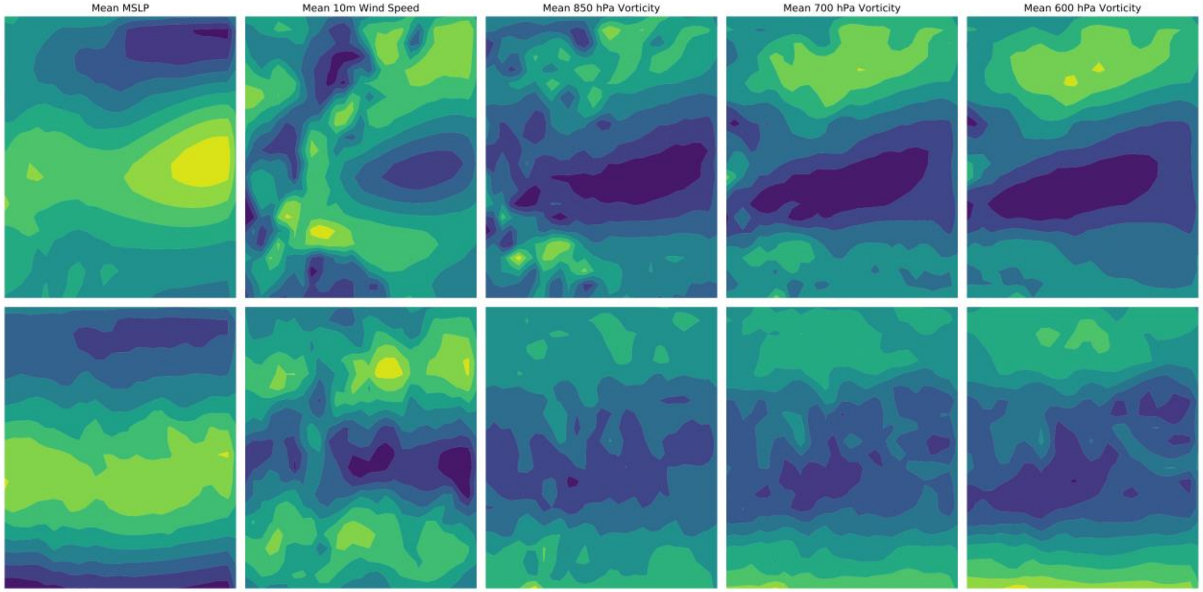
Results: Standard Models

- The model developed was compared with some standard models. It did not obtain the best accuracy, but did get the best loss



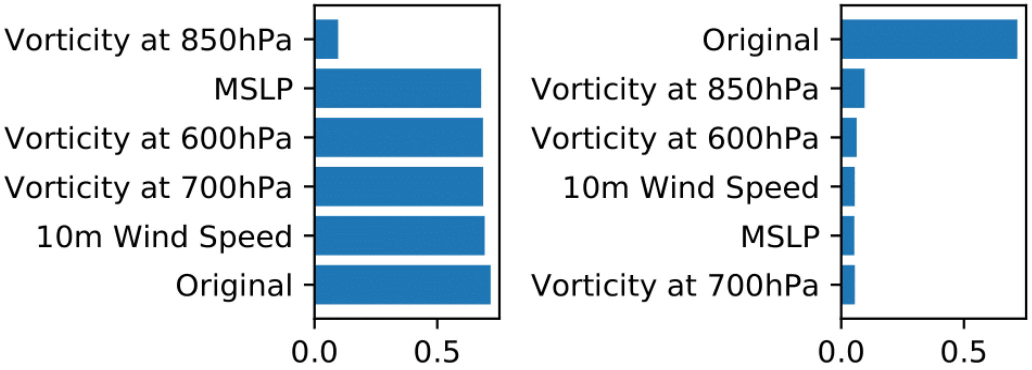
Results: Different Training Areas

Model	Training Region	Testing Region	AUC-PR
WAWP	WAWP	WAWP	0.7884
WAWP	WAWP	Global	0.6491
Global	Global	Global	0.7173

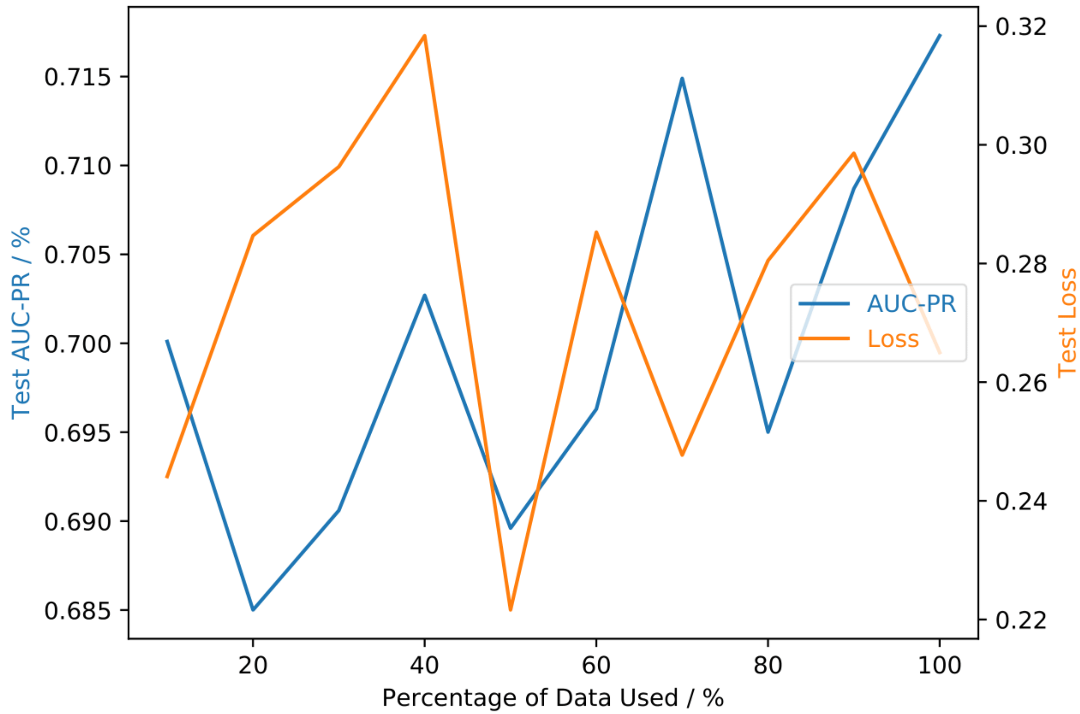


Results: Feature Importance

- Breiman Method:
  - Each field is permuted across all testing cases
  - If performance decreases from original, the field is important for the model
  - The larger the decrease, the more important the field
- Lakshamanan Method:
  - Each field is permuted across all testing cases
  - Most important is kept permuted, while the next important is found
  - Keep on going until all fields are permuted



Results: Dataset Size



Verification

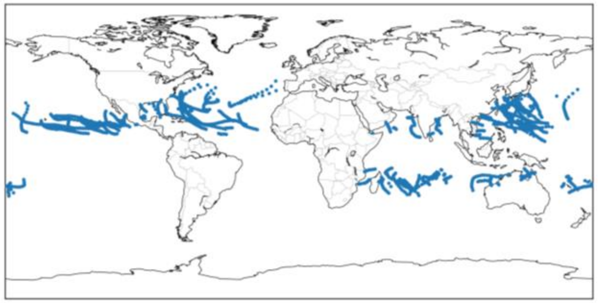
Verification:

- Compare outputs from DL model to an algorithm and observations
- Algorithm: TRACK by Dr Kevin Hodges
  - Uses vorticity and other fields to track all TCs
  - Outputs are tracks of TC centres (lats/lons)
  - We use only hurricane portion of track, T-TRACK
- Observations: IBTrACS
  - Positions of major storms including category on Saffir-Simpson scale

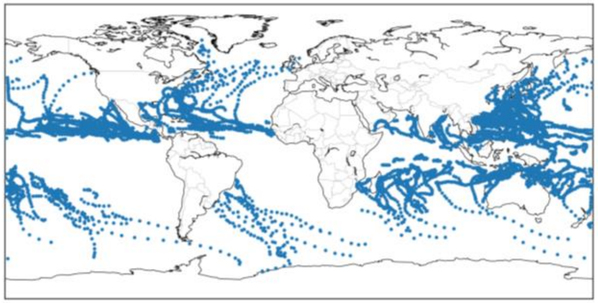
Verification: TC Centres

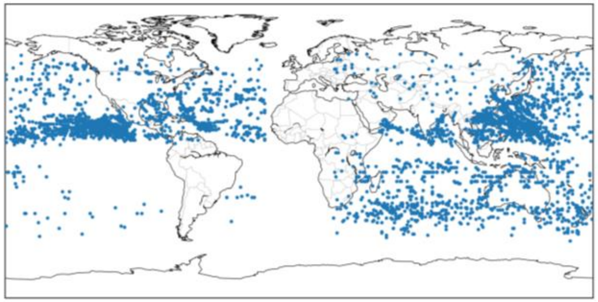
- Compare outputs from DL model to T-TRACK, observations

IBTrACS



T-TRACK

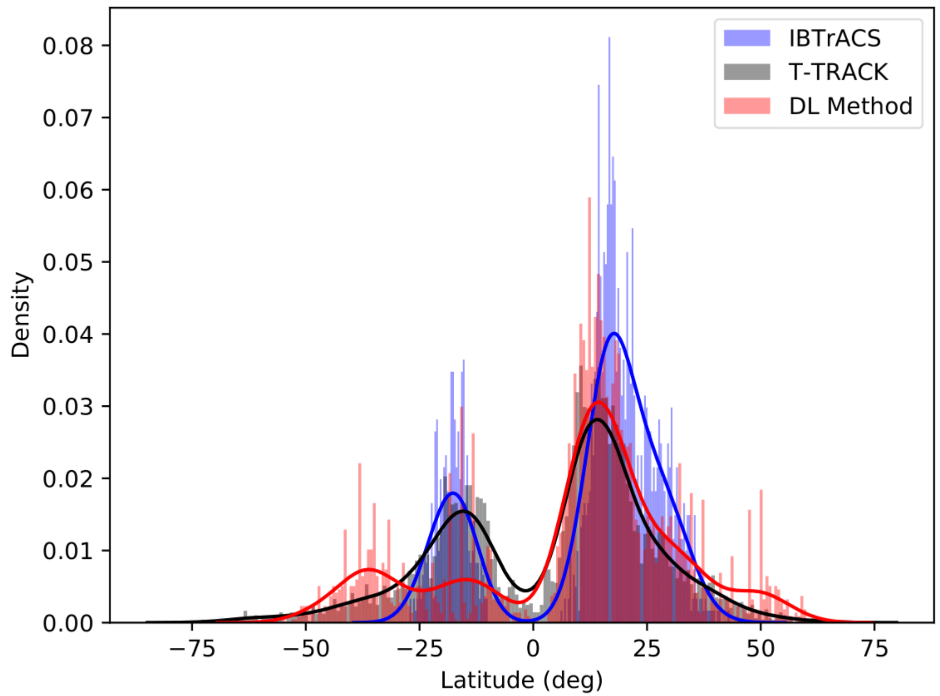




TCDetect

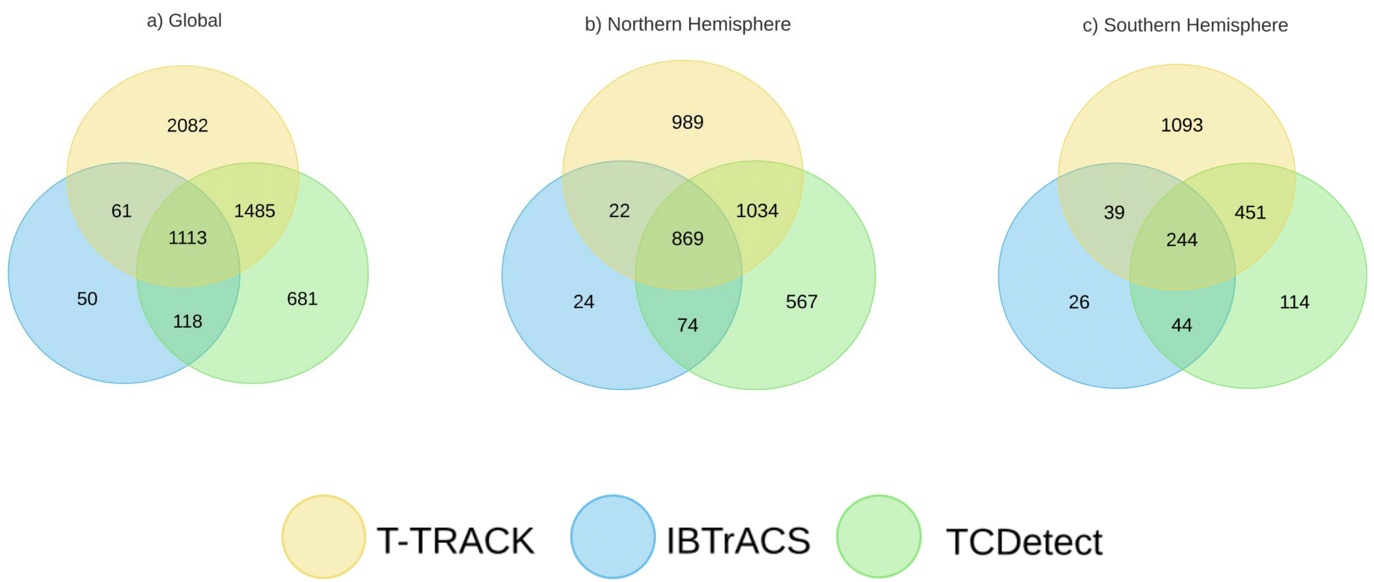
Verification: TC Centres

- Compare outputs from DL model to T-TRACK, observations

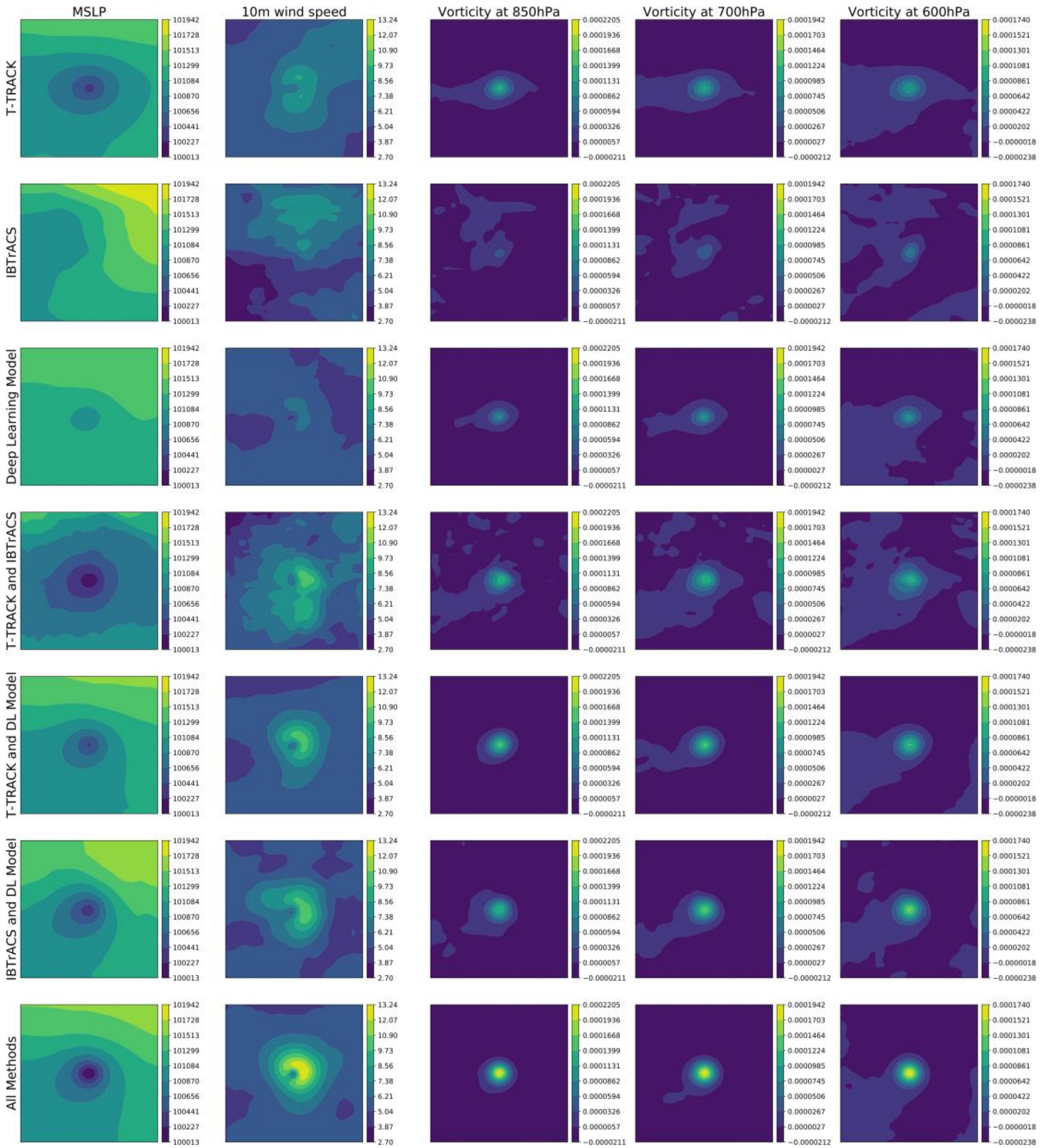


Verification: TC Matches

- Compare outputs from DL model to T-TRACK, observations

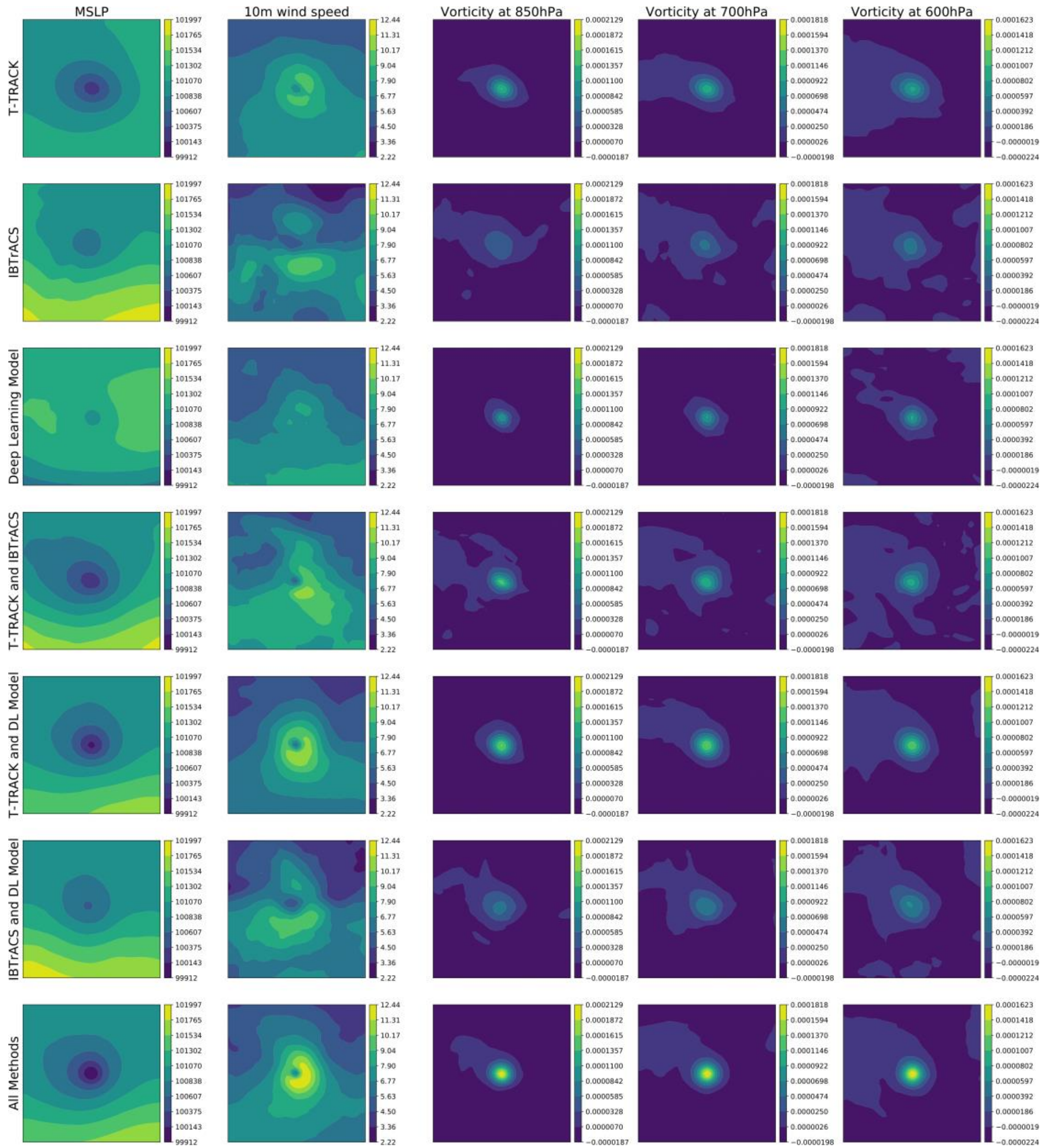


Verification: TC Matches



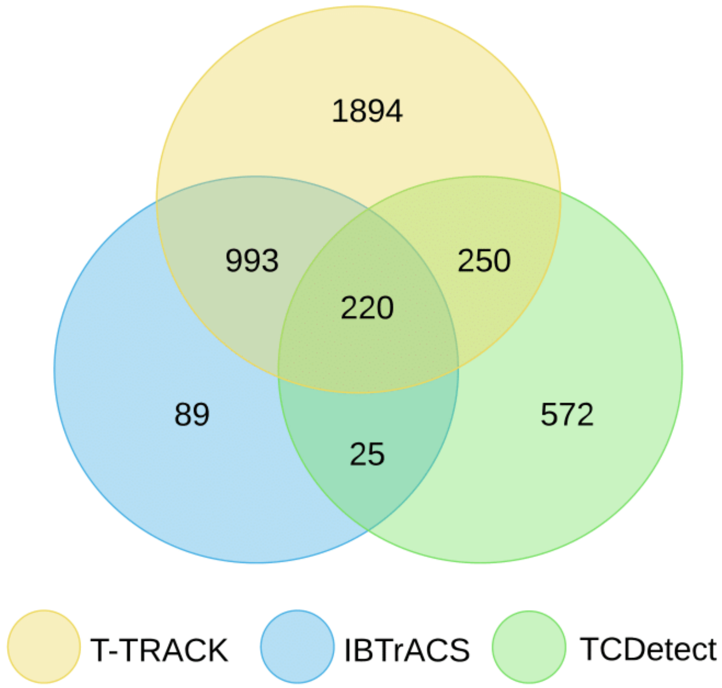
Verification: TC Matches

University of Reading

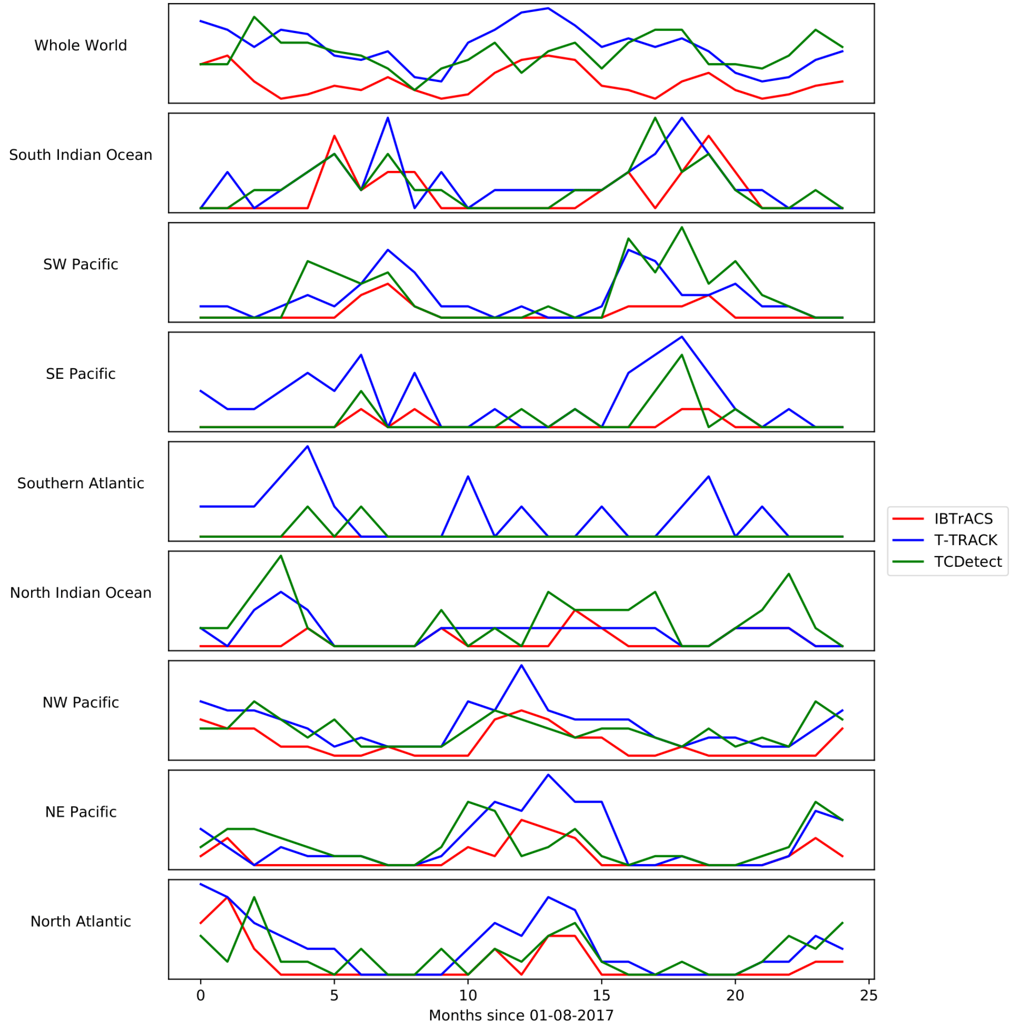


Verification: TC Track Matches

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



Verification: TC Frequencies

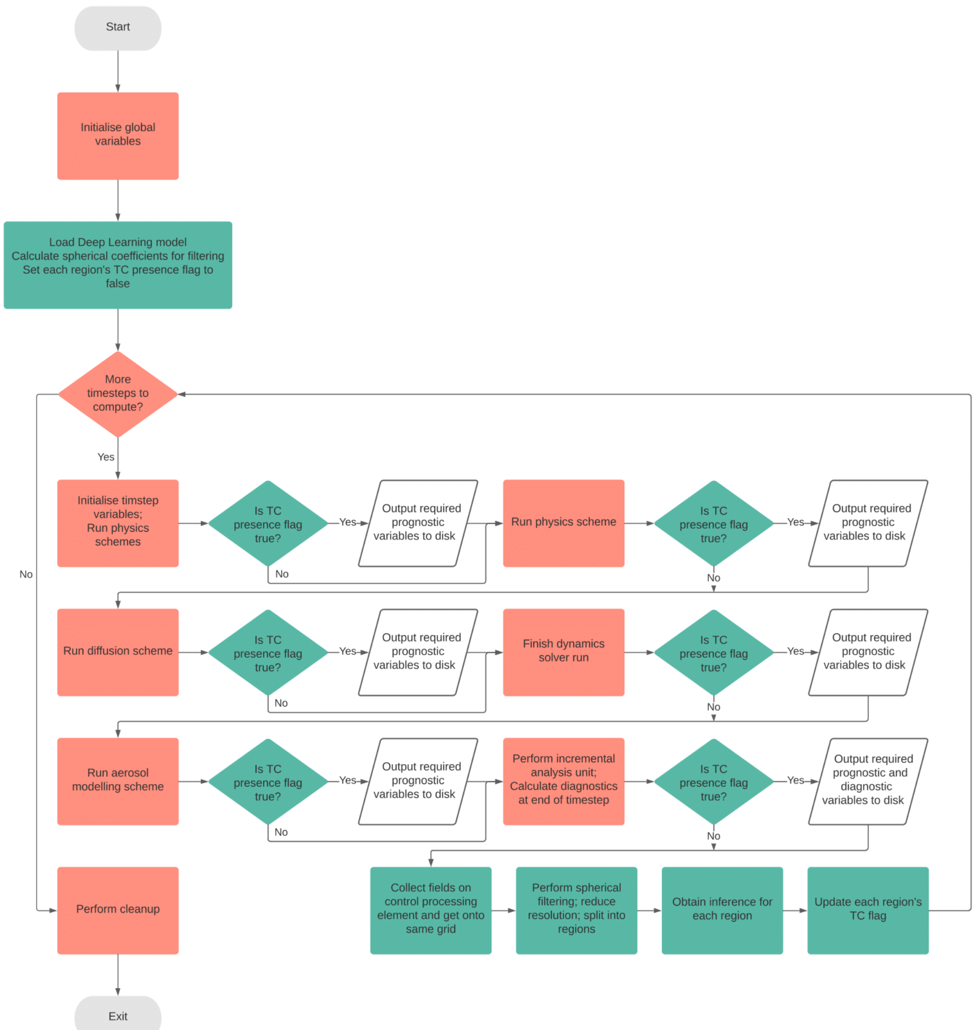


Implementation

Implementation

- TCDetect was trained in Python
- We wanted to have a climate model use it to decide whether to output data to disk
  - We chose the UK Met Office's Unified Model (UM)
  - This is based in FORTRAN
- We needed a way to use the Python-trained TCDetect in the FORTRAN-based UM
  - We use a C++ package, `frugally-deep`, to load and use TCDetect in FORTRAN

Implementation



University of Reading

Results: Overview

- Wanted to check ability of TCDetect to be applied to different data sources, horizontal resolutions and climates
- Utilize 6 different datasets to run the following analysis:

Dataset	Labelling Method	Horizontal Resolution
ERA-Interim	IBTrACS	~79km
ERA-Interim	T-TRACK	~79km
UM N96	T-TRACK	~135km
UM N512	T-TRACK	~25km
Hist1950	T-TRACK	~25km
Future	T-TRACK	~25km

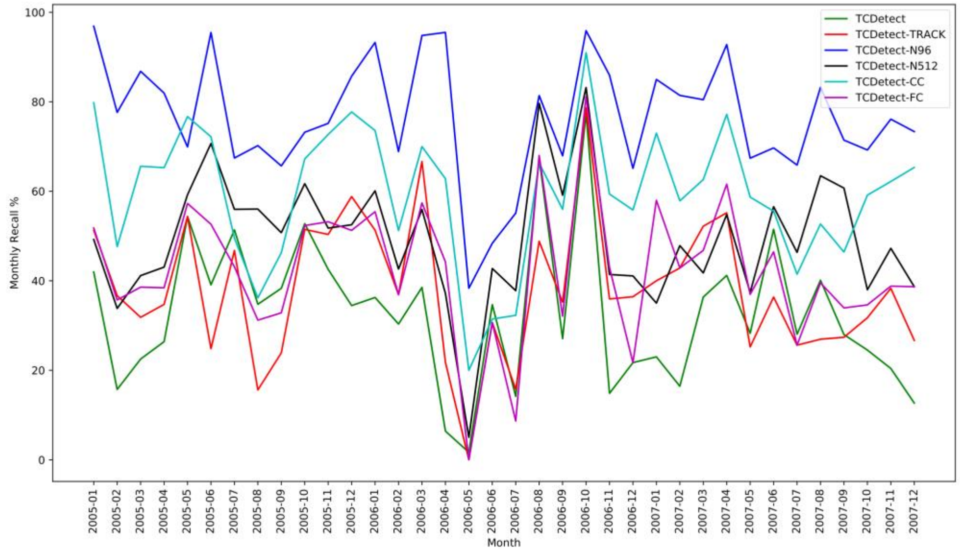
Results

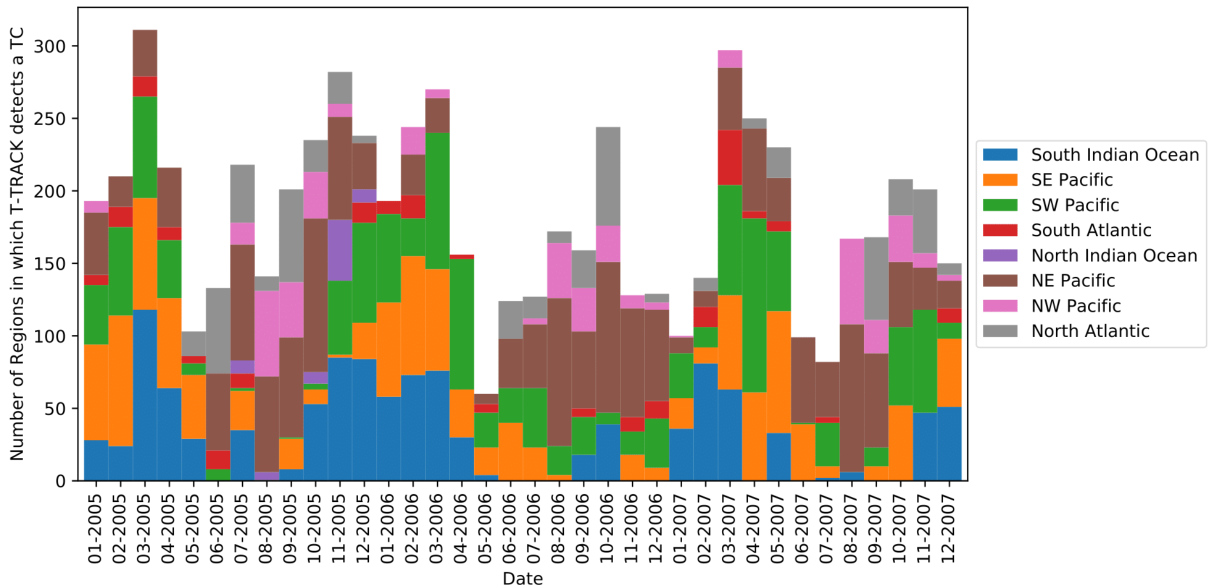
Data Source	Label Source	TCDetect	TCDetect -TRACK	TCDetect -N96	TCDetect -N512	TCDetect -CC	TCDetect -FC
ERAI	IBTrACS	92%	85%	97%	91%	93%	89%
ERAI	T-TRACK	59%	62%	76%	70%	73%	62%
UM N96	T-TRACK	34%	41%	78%	50%	62%	44%
UM N512	T-TRACK	58%	58%	71%	72%	77%	60%
Hist1950	T-TRACK	60%	58%	73%	73%	77%	64%
Future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

Results

- Model trained on the dataset being tested obtained superior recall to most others
- Different sources:
  - Models trained on UM data, except TCDetect-FC, obtain better recall than models trained on ERA-Interim data
- Different resolutions:
  - TCDetect-N96 performed better on N512 data. Not vice-versa
- Different climates:
  - All models are close when testing on the Hist1950 and Future datasets

Results: Variability





Implementation: Data Volume

	ERA-Interim			UM N96		UM N512	
	IBTrACS	T-TRACK	TCDetect	T-TRACK	TCDetect	T-TRACK	TCDetect
NI	72 (2%)	277 (9%)	239 (8%)	74 (2%)	105 (2%)	157 (11%)	197 (14%)
NWP	400 (13%)	1222 (40%)	933 (31%)	1674 (39%)	2617 (61%)	956 (67%)	920 (64%)
NEP	267 (9%)	712 (23%)	875 (29%)	442 (11%)	689 (16%)	524 (37%)	714 (50%)
NA	250 (8%)	703 (23%)	497 (16%)	559 (13%)	999 (23%)	431 (30%)	683 (48%)
SI	214 (7%)	646 (21%)	406 (13%)	913 (31%)	1596 (37%)	234 (16%)	552 (38%)
SWP	113 (4%)	740 (24%)	399 (13%)	874 (20%)	1436 (33%)	154 (11%)	374 (26%)
SEP	26 (1%)	322 (11%)	35 (1%)	1096 (25%)	1945 (45%)	149 (10%)	370 (26%)
SA	0 (0%)	119 (4%)	119 (4%)	167 (4%)	362 (8%)	28 (2%)	97 (7%)

Implementation: Computational Performance

Function	Times Applied	N96 Timings / sec	N512 Timings / sec
Collect data	1	4.64 $\times 10^{-4}$	2.99$\times 10^{-3}$
Interpolate MSLP	1	3.15 $\times 10^{-4}$	0.35
Resize field	5	9.1$\times 10^{-3}$	3.67$\times 10^{-2}$
Calculate Vorticity	3	1.16 $\times 10^{-3}$	2.28$\times 10^{-3}$
Spherical filtering	5	0.68	1.36
Standardisation	5	4.13 $\times 10^{-5}$	6.64$\times 10^{-5}$
Data formatting	8	1.63$\times 10^{-5}$	3.66 $\times 10^{-5}$
DL inference	8	0.19	0.37
Full Method		4.96	10.22
Full Timestep		6.15	15.64
Full Simulation		23%	5%

Summary

Summary

- Data volumes from GCMs is large; we want to reduce it
- We have developed a DL algorithm that infers the presence of a Tropical Cyclone in a set of data
- It achieves a recall rate of 91% over the test dataset
- This was compared to a SOTA non-ML algorithm and was found to be mostly comparable in performance
- The DL algorithm was used in a running GCM (the UM) to decide whether to output data from a region to disk
  - Data volume was reduced by ~70%
  - The DL algorithm does not affect the GCM runtime too much

Possible Future Work

- Better DL model
- Change filtering techniques to be done inside DL model; hopefully producing computationally less expensive method
- Replicate method for other phenomena

Acknowledgements

- This work was supported by NVIDIA and Oracle
- It 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
- UM work was aided by members of the NCAS-CMS team