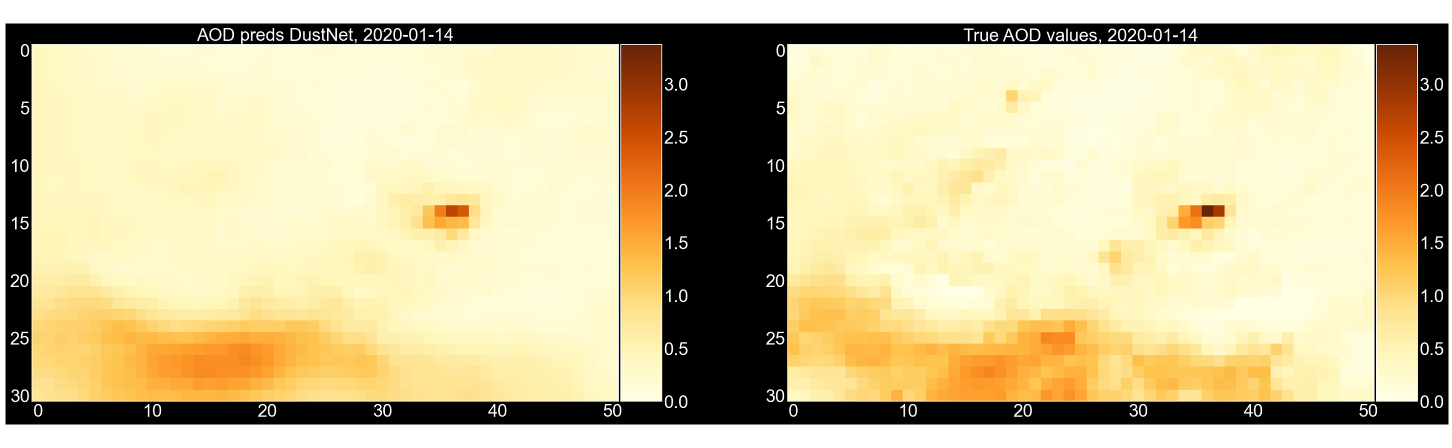
Cite DOI: 10.5281/zenodo.10631954

Zip Files contain:

- **Models input - data and code**
  - Data - structured
  - DustNet model code (example result in Fig. 1)
  - DustNet pre-trained model
  - **Normalised and split - data and code**
    - Data - split into train/validate/test
    - U-NET model code
    - U-NET pre-trained model
    - Conv2D model code
    - Conv2D pre-trained model
- **Models output - data and code**
  - Data - output from model
  - Spatial analysis code (example result in Fig. 2)
  - Temporal analysis code (example result in Fig. 3)
  - Point location analysis code (example result in Fig. 4)
  - GIF of AOD DustNet and MODIS code (example result in Fig. 5)

Fig. 1. Predicted (de-normalised) AOD by DustNet model and corresponding true values for 14th Jan 2020.



Models_input

Processed MODIS AOD data (from Aqua and Terra) and selected ERA5 variables ready for forecasting with Machine Learning. These long-term daily timeseries (2003-2022) are provided as n-dimensional NumPy arrays.

The Python code to handle the data and run the DustNet model is included as Jupyter Notebook 'DustNet_model_code.ipynb'. An example of DustNet model predictions versus true values is illustrated below.

All arrays included in the .zip folder have:

- **Format:** NumPy
- **Resolution:** 1° x 1°
- **Location:** Northern Africa (0°S - 31°N, 20°W - 31°E)
- **Length:** 7305 - corresponds to time in days
- **Longitude:** 51
- **Latitude:** 31
- **Date*:**
  - start: 1st Jan 2003
  - end: 31st Dec 2022

*NOTE that the *datetime* is not included in the features. But, the index number of each array corresponds to the start and end time of timeseries, thus a date index can be added to the model after running predictions.

DATA:

- **1_met_x35.npy**
  Shape(7305, 31, 51, 35)
  Features in the 4th dimension:
  1. -> wind u ground (1000hPa)
  2. -> wind v ground (1000hPa)
  3. -> wind speed ground (1000hPa)
  4. -> wind power ground (1000hPa)
  5. -> wind speed (550hPa)
  6. -> wind speed (750hPa)
  7. -> wind speed (850hPa)
  8. -> wind speed (950hPa)
  9. -> wind power (550hPa)
  10. -> wind power (750hPa)
  11. -> wind power (850hPa)
  12. -> wind power (950hPa)
  13. -> wind u Level 1 (550hPa)
  14. -> wind u Level 2 (750hPa)
  15. -> wind u Level 3 (850hPa)
  16. -> wind u Level 4 (950hPa)
  17. -> wind v Level 1 (550hPa)
  18. -> wind v Level 2 (750hPa)
  19. -> wind v Level 3 (850hPa)
  20. -> wind v Level 4 (950hPa)
  21. -> temperature at Level 5 (1000hPa)
  22. -> temperature at Level 1 (550hPa)
  23. -> temperature at Level 2 (750hPa)
  24. -> temperature at Level 3 (850hPa)
  25. -> temperature at Level 4 (950hPa)
  26. -> relative humidity Level 5 (1000hPa)
  27. -> relative humidity Level 1 (550hPa)
  28. -> relative humidity Level 2 (750hPa)
  29. -> relative humidity Level 3 (850hPa)
  30. -> relative humidity Level 4 (950hPa)
  31. -> vertical velocity Level 5 (1000hPa)
  32. -> vertical velocity Level 1 (550hPa)
  33. -> vertical velocity Level 2 (750hPa)
  34. -> vertical velocity Level 3 (850hPa)
  35. -> vertical velocity Level 4 (950hPa)

- **2_terrain.npy**
  Variable: 'Terrain' (meters above sea level)
  Shape: (7305, 31, 51, 1)

- **3_time_sin_cos.npy**
  Shape(7305, 31, 51, 2)
  Features in the 4th dimension:
  1. sine of timestamps
  2. cosine of timestamps

- **4_AOD_imputed.npy**
  Variable: 'AOD' (imputed)
  Shape(7305, 31, 51, 1)

CODE:

- **DustNet_model_code.ipynb**
  - Use with Models_input_data. The Python (v. 3.10) code to create lagged AOD (x5 days) and lagged meteorological data (x1 day), normalise the data (0-1) and split the data into train/validate/test sets. The DustNet model is defined here (trains in ~7min), and sample images of predicted vs true AOD values are plotted.
  - Code by T.E. Nowak

Normalised_and_split_data

The normalised and split data is called in by UNet_model and Conv2D_model code. The 'x' data contains all atmospheric input features described in 1_met_x35.npy lagged by 1 day, and AOD data (4_AOD_imputed.npy) with 5 days lag. The lagged data was concatenated, normalised and split into training, validation and test sets. The full process of adding lag data, normalising and splitting is included in DustNet_mode_code file. Here are the resulting NumPy files:

- **1_x_train.npy**
  Training set: 70% of data
  Starts: 06th Jan 2003
  Ends: 01st Jan 2017
  Shape: (5110, 31, 51, 43)

- **2_x_valid.npy**
  Validation set: 15% of data
  Starts: 2nd Jan 2017
  Ends: 01st Jan 2020
  Shape: (1095, 31, 51, 43)

- **3_x_test.npy**
  Testing set: 15% of data
  Starts: 2nd Jan 2020
  Ends: 31st Dec 2022
  Shape: (1095, 31, 51, 43)

- **4_y_train.npy**
  Training set: 70% of data
  Starts: 06th Jan 2003
  Ends: 01st Jan 2017
  Shape: (5110, 31, 51, 1)

- **5_y_valid.npy**
  Validation set: 15% of data
  Starts: 2nd Jan 2017
  Ends: 01st Jan 2020
  Shape: (1095, 31, 51, 1)

- **6_y_test.npy**
  Testing set: 15% of data
  Starts: 2nd Jan 2020
  Ends: 31st Dec 2022
  Shape: (1095, 31, 51, 1)

CODE:

- **UNet_model_code.ipynb**
  - Use with Normalised and split data. The Python (v. 3.10) code to run the U-NET model (**trains for ~1hr**)
  - Code by Dr S. Siegert

- **Conv2D_model_code.ipynb**
  - Use with Normalised and split data. The Python (v. 3.10) code to run the Conv2D model (**trains for ~15min**)
  - Code by Dr S. Siegert

Models_output

The output data from DustNet_model_code of de-normalised AOD predictions in NumPy format. The data also contains matching predictions from CAMS and corresponding MODIS data as ground truth. All AOD predictions are for 24-hr ahead and cover **1095 days** from **2nd Jan 2020 to 31st Dec 2022** (inclusive of both days). Jupyter Notebooks with Python (v. 3.8.13) code for statistical analysis of these predictions are also included. See below for full description and examples of plots.

- **1_DustNet_predictions.npy**
  Variable: 'AOD'
  Shape: (1095, 31, 51, 1)

- **2_CAMS_predictions.npy**
  Variable: 'AOD'
  Shape: (1095, 31, 51, 1)

- **3_Persistence_predictions.npy**
  Variable: 'AOD' (imputed)
  Shape: (1095, 31, 51)

- **4_Climatology_predictions.npy**
  Variable: 'AOD'
  Shape: (31, 51)

- **5_MODIS_observations.npy**
  Variable: 'AOD'
  Shape: (1095, 31, 51)

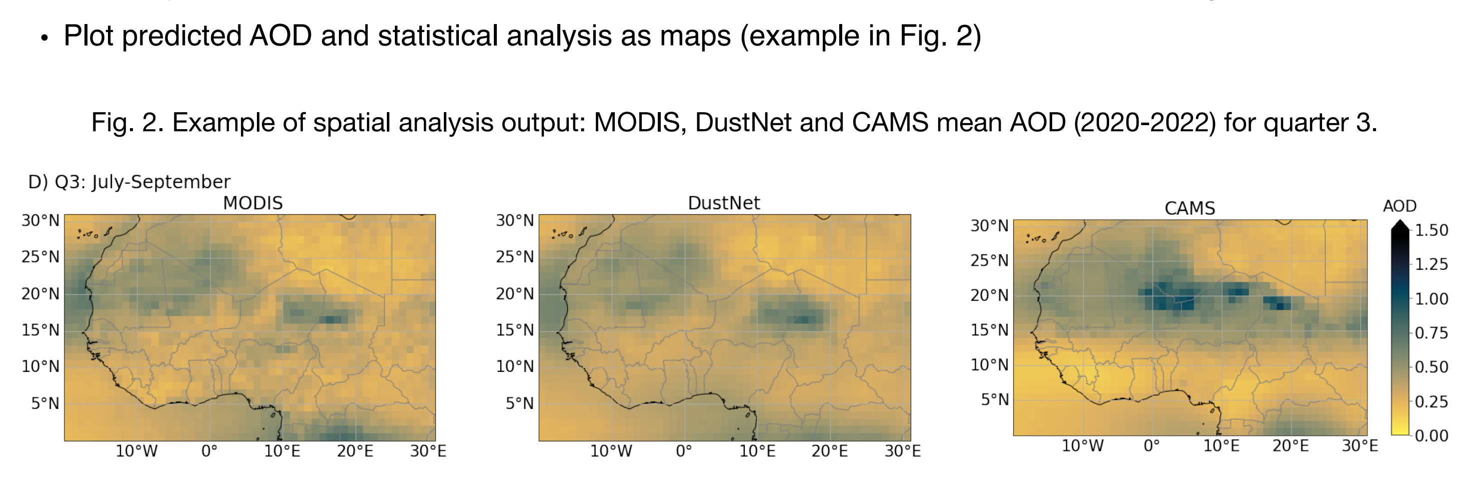
- **Sah_lat.npy**
  Variable: Latitude
  Shape: (31,)

- **Sah_lon.npy**
  Variable: Longitude
  Shape: (51,)

CODE:

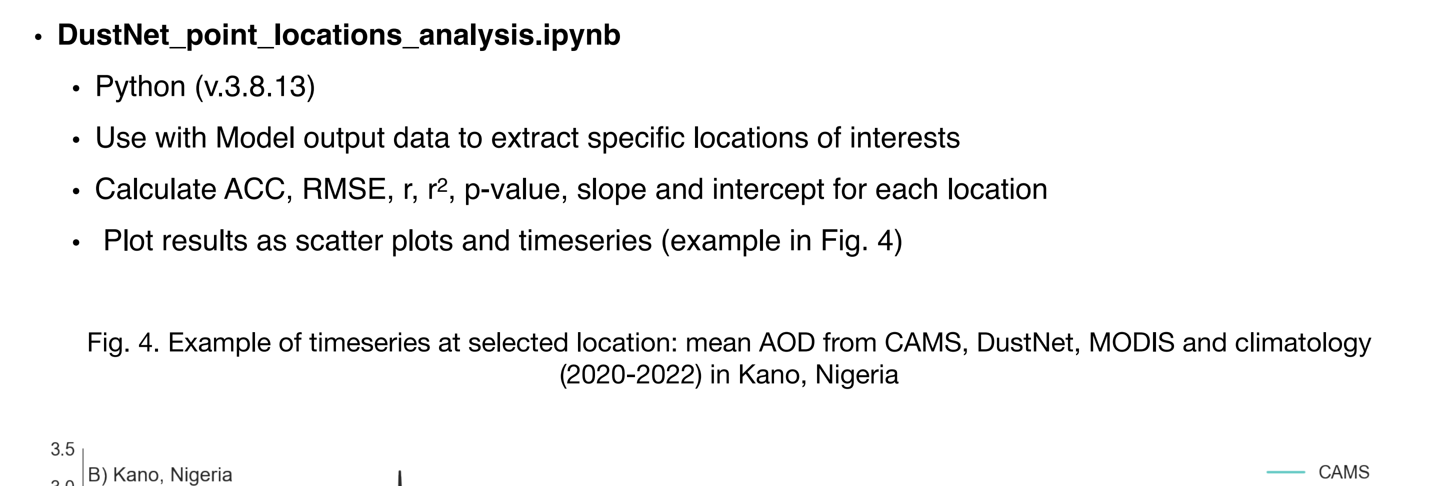
- **DustNet_spatial_analysis.ipynb**
  - Python (v. 3.8.13)
  - Use with Models output data to calculate the mean bias error (MBE), root mean square error (RMSE), and accuracy correlation coefficient (ACC) between predicted and true AOD values for each grid location (spatial)
  - Plot predicted AOD and statistical analysis as maps (example in Fig. 2)

Fig. 2. Example of spatial analysis output: MODIS, DustNet and CAMS mean AOD (2020-2022) for quarter 3.



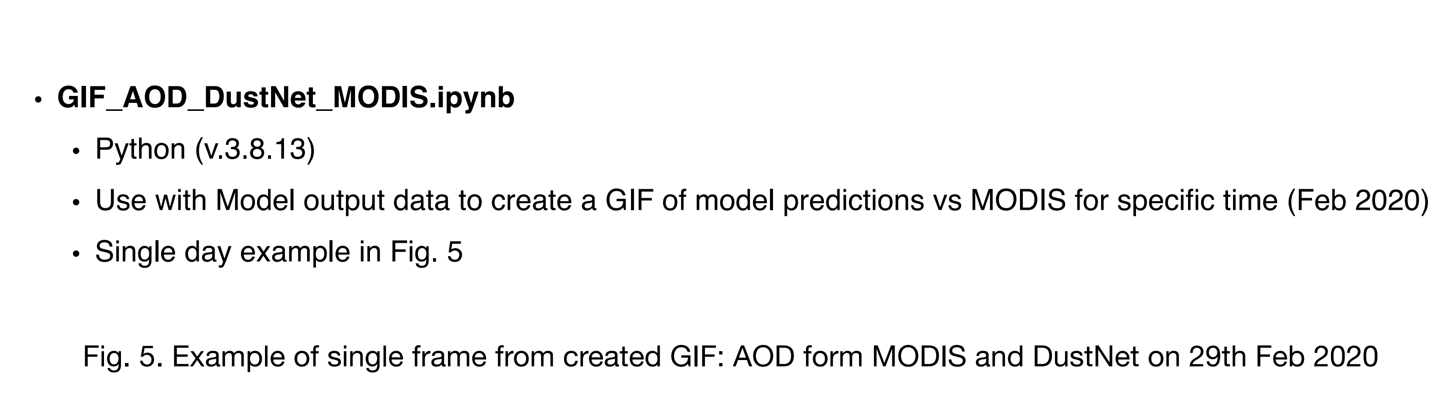
- **DustNet_temporal_analysis.ipynb**
  - Python (v. 3.8.13)
  - Use with Models output data to calculate temporal RMSE, MBE, r, r^2, p-value, slope and intercept
  - Plot scatter, histograms, density plots and timeseries (example in Fig. 3)

Fig. 3. Example of temporal analysis: daily mean AOD (2020-2022) from DustNet and CAMS vs MODIS data



- **DustNet_point_locations_analysis.ipynb**
  - Python (v.3.8.13)
  - Use with Model output data to extract specific locations of interests
  - Calculate ACC, RMSE, r, r^2, p-value, slope and intercept for each location
  - Plot results as scatter plots and timeseries (example in Fig. 4)

Fig. 4. Example of timeseries at selected location: mean AOD from CAMS, DustNet, MODIS and climatology (2020-2022) in Kano, Nigeria



- **GIF_AOD_DustNet_MODIS.ipynb**
  - Python (v.3.8.13)
  - Use with Model output data to create a GIF of model predictions vs MODIS for specific time (Feb 2020)
  - Single day example in Fig. 5

Fig. 5. Example of single frame from created GIF: AOD form MODIS and DustNet on 29th Feb 2020