

# IceNet: A peek into validation commands

Ryan Chan



British  
Antarctic Survey  
NATURAL ENVIRONMENT RESEARCH COUNCIL

The  
Alan Turing  
Institute

POLAR SCIENCE  
FOR PLANET EARTH

# A peek into IceNet validation commands

After using the IceNet library to make forecasts, this is what our forecasts look like:

```
xarray.Dataset
└─ Dimensions: (time: 1826, yc: 432, xc: 432, leadtime: 93)
└─ Coordinates:
  time (time) datetime64[ns] 2014-01-01 ... 2018-12-31
  leadtime (leadtime) int64 1 2 3 4 5 6 7 ... 88 89 90 91 92 93
  forecast_date (time, leadtime) datetime64[ns] ...
  xc (xc) float64 -5.388e+03 -5.362e+03 ... 5.388e+03
  yc (yc) float64 5.388e+03 5.362e+03 ... -5.388e+03
  lat (yc, xc) float32 ...
  lon (yc, xc) float32 ...
└─ Data variables:
  Lambert_Azimuth... () int32 ...
  sic_mean (time, yc, xc, leadtime) float32 ...
  sic_stddev (time, yc, xc, leadtime) float32 ...
└─ Indexes: (4)
└─ Attributes: (36)
```

... and this is what our observed “ground truth” (OSISAF) data looks like:

```
xarray.DataArray 'ice_conc' (time: 93, yc: 432, xc: 432)
└─ Array Chunk
  Bytes 132.42 MiB 132.42 MiB
  Shape (93, 432, 432) (93, 432, 432)
  Dask graph 1 chunks in 3 graph layers
  Data type float64 numpy.ndarray
└─ Coordinates:
  xc (xc) float64 -5.388e+03 -5.362e+03 ... 5.388e+03
  yc (yc) float64 5.388e+03 5.362e+03 ... -5.388e+03
  lon (yc, xc) float32 dask.array<chunksize=(432, 432), meta=np.ndarray>
  time (time) datetime64[ns] 2017-07-02 ... 2017-10-02
  lat (yc, xc) float32 dask.array<chunksize=(432, 432), meta=np.ndarray>
└─ Indexes: (3)
└─ Attributes:
  long_name : fully filtered concentration of sea ice using atmospheric correction of brightness temperatures and open water filters
  valid_max : 10000
  ancillary_variables : total_standard_error_status_flag
  standard_name : sea_ice_area_fraction
  comment : this field is the primary sea ice concentration estimate for this climate data record
  units : %
  grid_mapping : Lambert_Azimuthal_Grid
  valid_min : 0
```

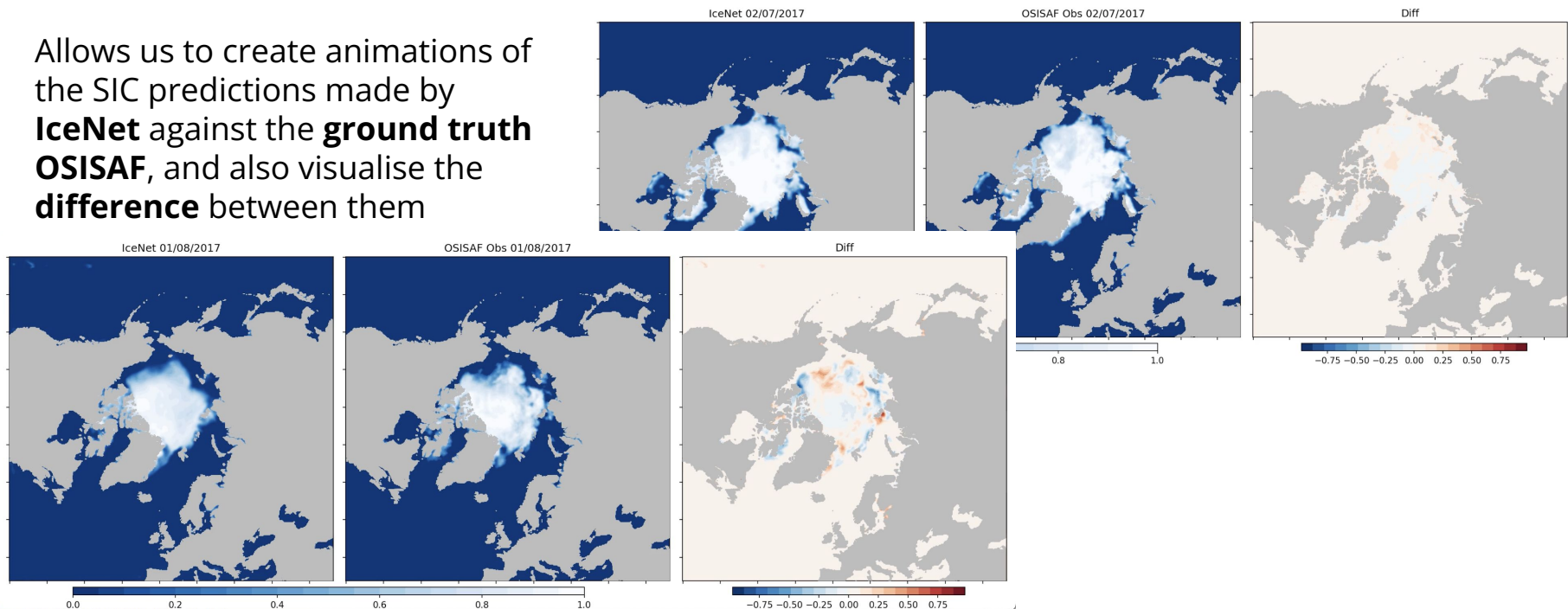
At each grid location, we have a **prediction** of SIC and a “ground truth” OSISAF measure of SIC

**Question:** How can we validate / evaluate our forecasts?

# Visualising/animating sea ice concentration error

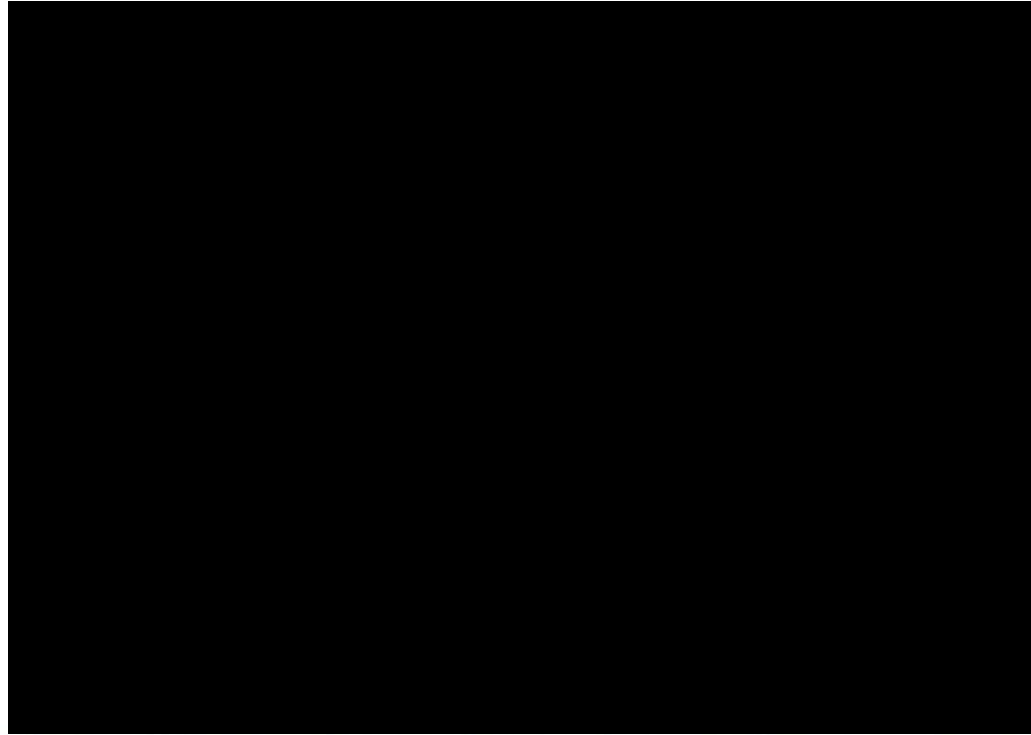
Using `sic_error_video` function (or `icenet_plot_sic_error` CLI)

Allows us to create animations of the SIC predictions made by **IceNet** against the **ground truth OSISAF**, and also visualise the **difference** between them



# Visualising/animating sea ice concentration error

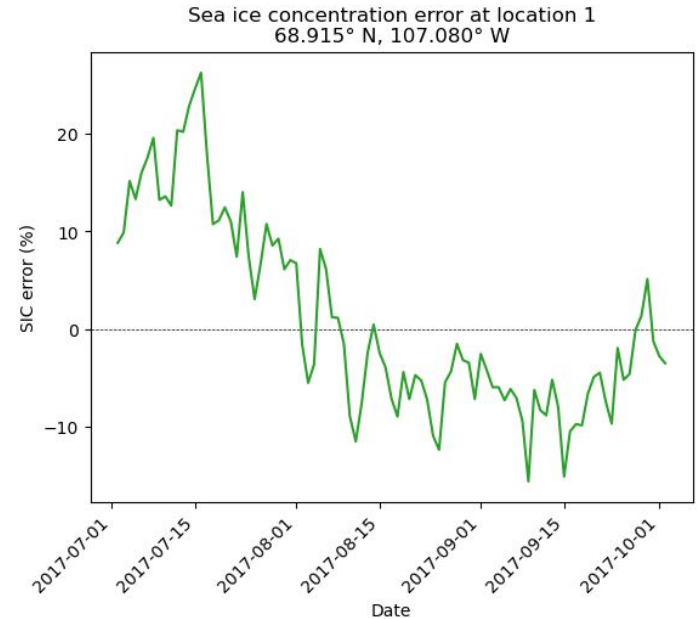
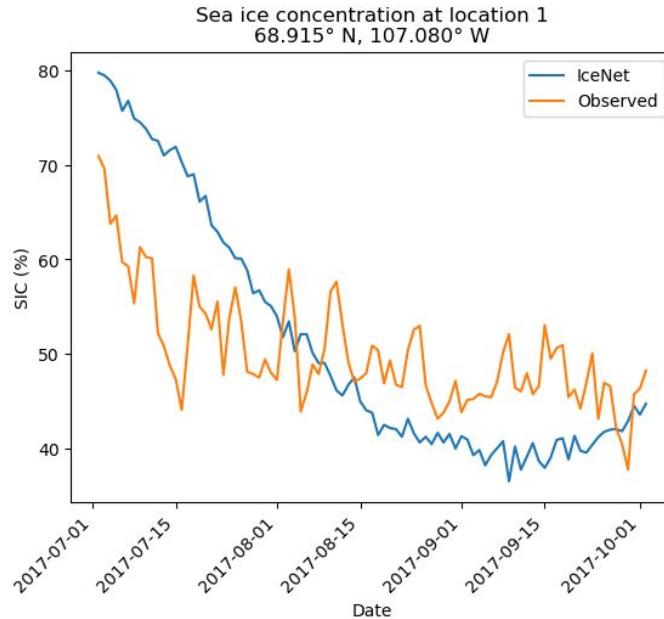
Using `sic_error_video` function (or `icenet_plot_sic_error` CLI)



# Zooming into a particular grid location

Using `sic_error_local_plots` function (or `icenet_plot_sic_error_local` CLI)

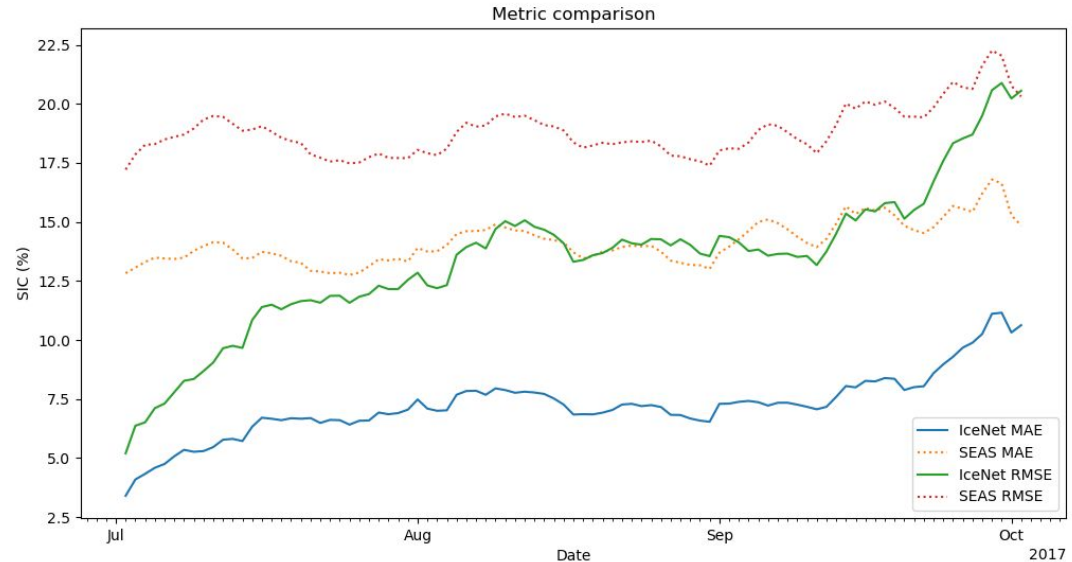
Allows us to zoom in and evaluate the SIC error of the forecast at a **particular grid location / coordinate**



# Summary metrics based on pure sea ice concentration

Using `plot_metrics` function (or `icenet_plot_metrics` CLI)

Allows us to compute various metrics (**MAE**, **MSE**, **RMSE**) to summarise the overall error of a forecast (i.e. aggregate the metric at each grid location)

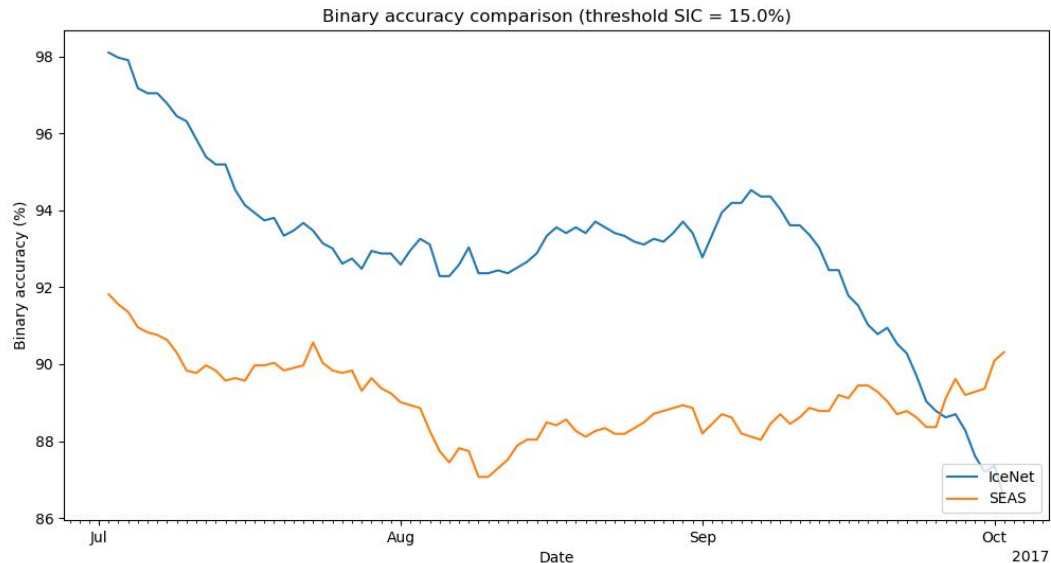




# Binary accuracy metric

Using `plot_binary_accuracy` function (or `icenet_plot_bin_accuracy` CLI)

Assess the performance of the model on the **binary task of predicting ice** (if  $SIC \geq 15\%$ ) and no-ice (if  $SIC < 15\%$ ) for each grid cell

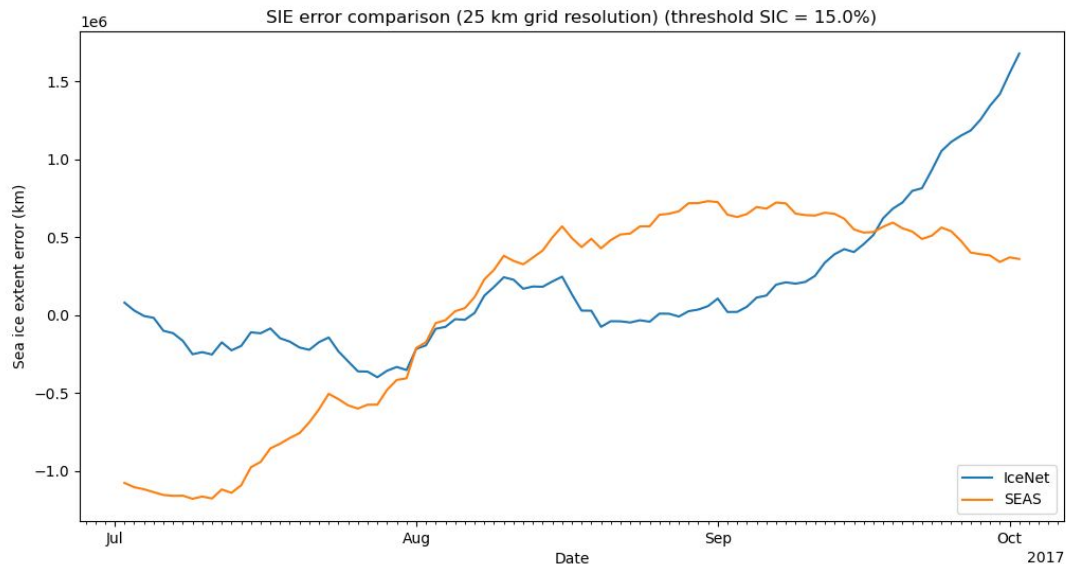


# Sea ice extent (SIE) error

Using `plot_sea_ice_extent_error` function (or `icenet_plot_sie_error` CLI)

**Sea ice extent (SIE)** is the total area of grid cells that has ice ( $SIC > 15\%$ )

We assess the model by looking at the difference in total area of ice between the forecast and OSISAF

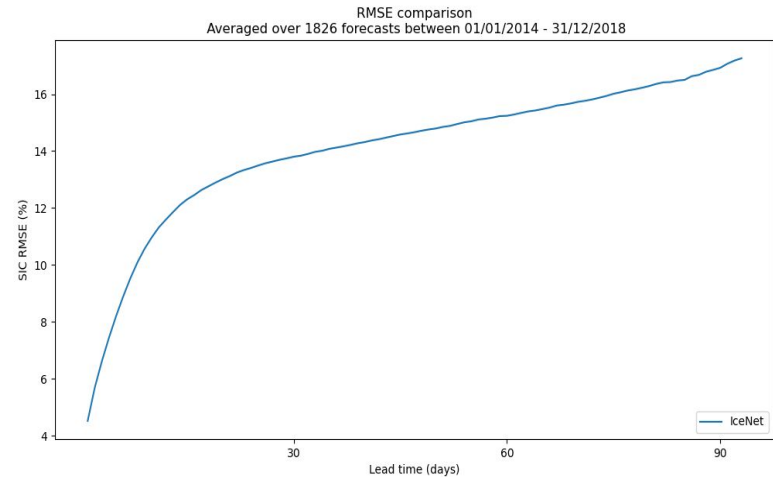
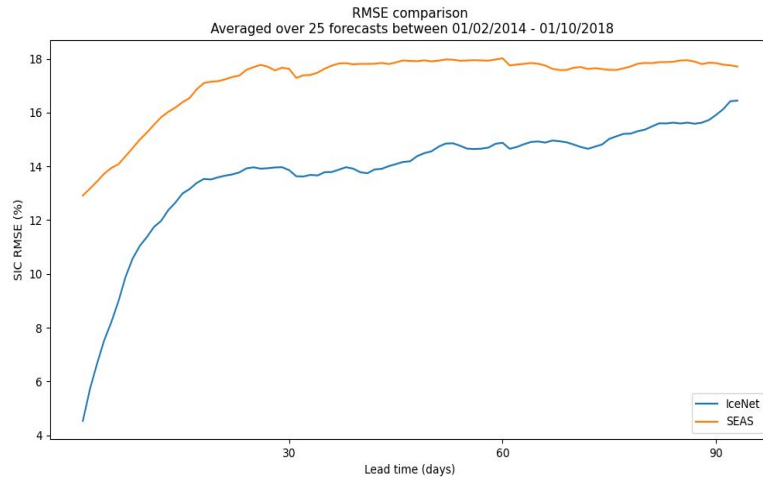




# Leadtime averaged plots (averaging over all forecasts)

Using `plot_metrics_leadtime_avg` function (or `icenet_plot_leadtime_avg` CLI)

Rather than looking at the error of a forecast at each leadtime, we might be interested in **average performance of several forecasts** made between some time period



# Leadtime averaged plots (averaged over month or day)

Using `plot_metrics_leadtime_avg` function (or `icenet_plot_leadtime_avg` CLI)

Rather than averaging over *all* forecasts, we can also average over month and day to get an idea of seasonal IceNet performance

