METEO
IMGW-PIB
meteo.imgw.pl

Increasing number of members in ensemble systems. Results of tests with ALARO and FourCastNet

Bogdan Bochenek
Jadwiga Róg

ALARO-1 Working Days WD
3-5 March 2025, Krakow

Motivation:

1) Does a set of deterministic forecasts make probabilistic forecast?

2) Adding perturbations throughout the forecast better represents the stochastic nature of atmospheric processes?

3) Access to easy-to-use AI global models, then can be run with fast with large number of members, $> 10^3$ elements of ensemble system

4) New cluster in IMGW-PIB that is on test phase now, with 1 PF and 20 000 CPU cores, to test ALARO ensemble system with the same approach

5) AI models are trained on NWP data, why not reverse it and use AI models to test new ideas for NWP?

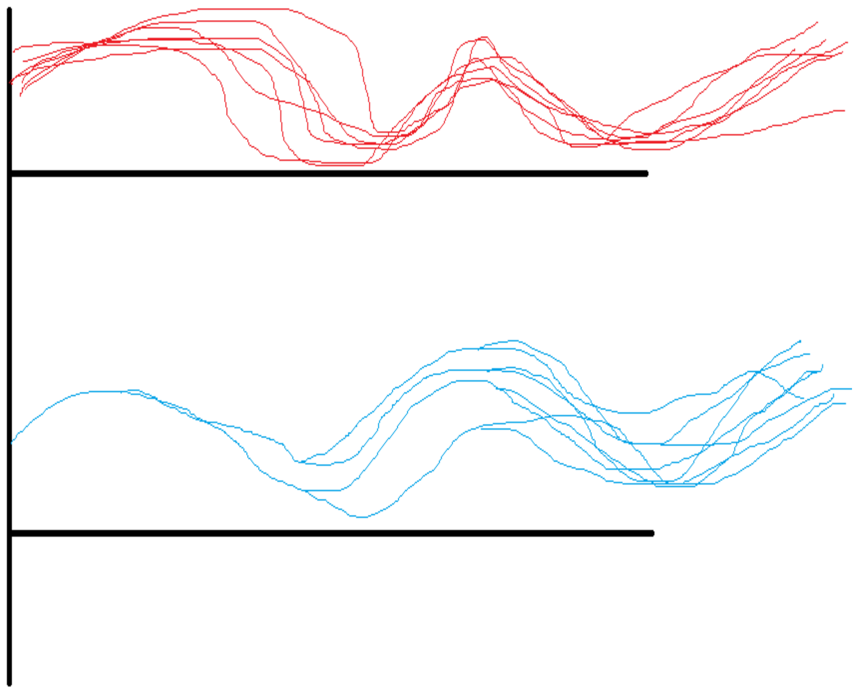
Historical background:

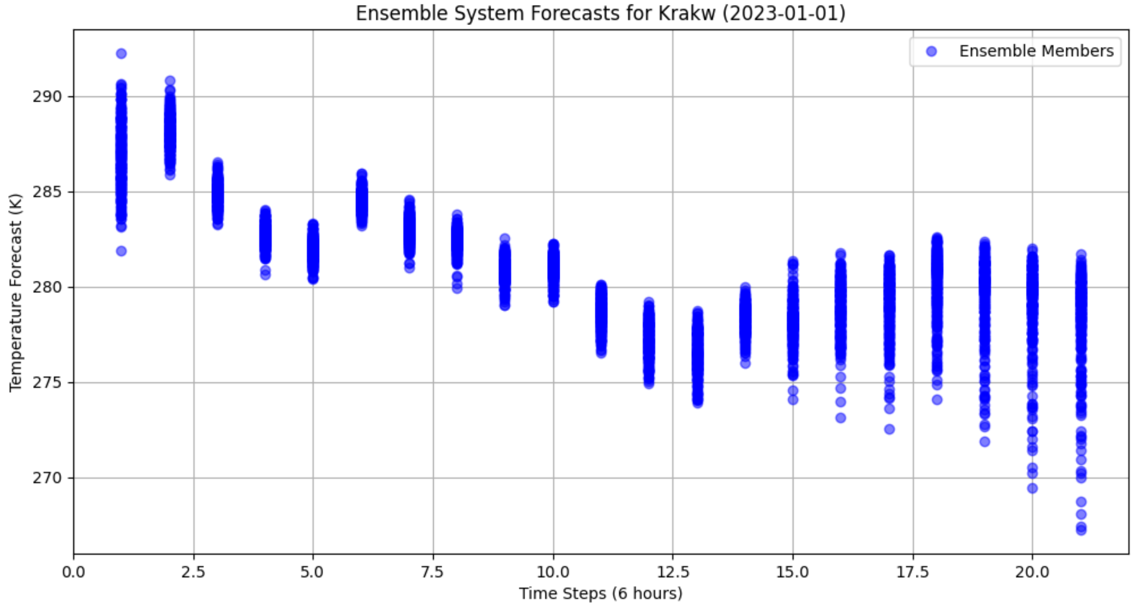
Research in dynamical systems and chaos suggests that introducing *incremental differences* can improve ensemble realism. Methods like the **breeding technique** (Toth & Kalnay, 1993) illustrate this principle. In breeding, a small perturbation is periodically reintroduced and grown over short cycles, producing “bred vectors” that track the fastest-growing error structures. This iterative perturbation mimics how small errors naturally amplify in the atmosphere (2008-2009 **breeding** related reports from M. Belus).

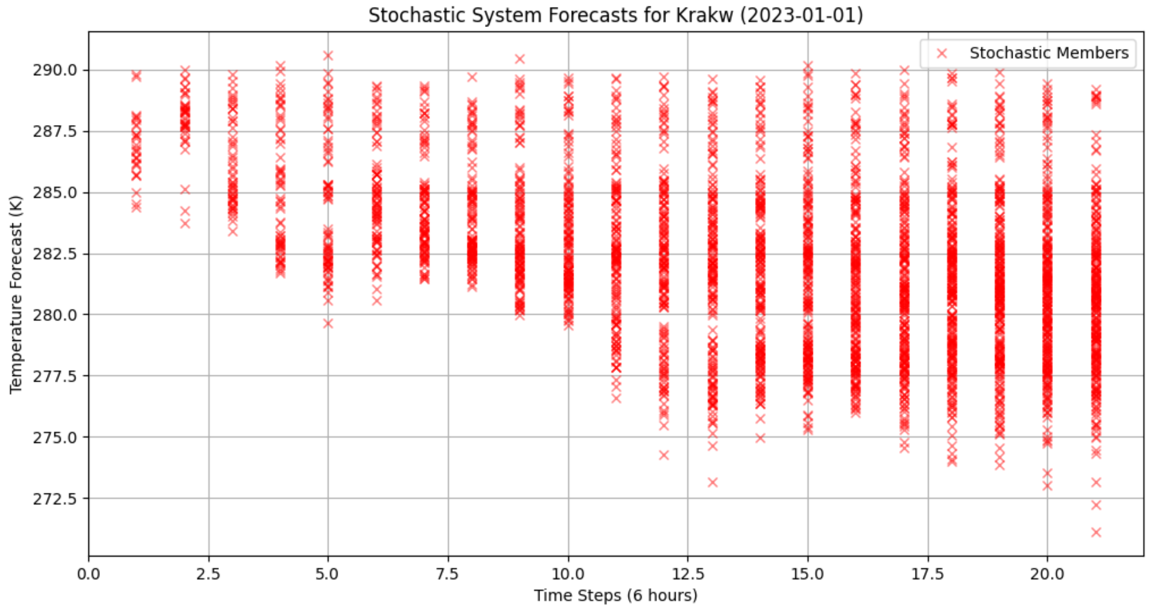
FuXi-ENS (Huang *et al.*, 2022) is a cascaded machine-learning ensemble system for medium-range weather prediction. It uses a Variational Autoencoder to generate perturbations and optimizes the CRPS metric, effectively *adding diversity* to forecasts.

Concept:

Instead of starting all members of ensemble system from the beginning of forecast, start with few members and add new members at later hours of forecast by applying perturbations to model forecast.




Ensemble System Forecasts for Krakw (2023-01-01)
Ensemble Members
Temperature Forecast (K)
290
285
280
275
270
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time Steps (6 hours)


Stochastic System Forecasts for Krakw (2023-01-01)
Stochastic Members
Temperature Forecast (K)
290.0
287.5
285.0
282.5
280.0
277.5
275.0
272.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time Steps (6 hours)

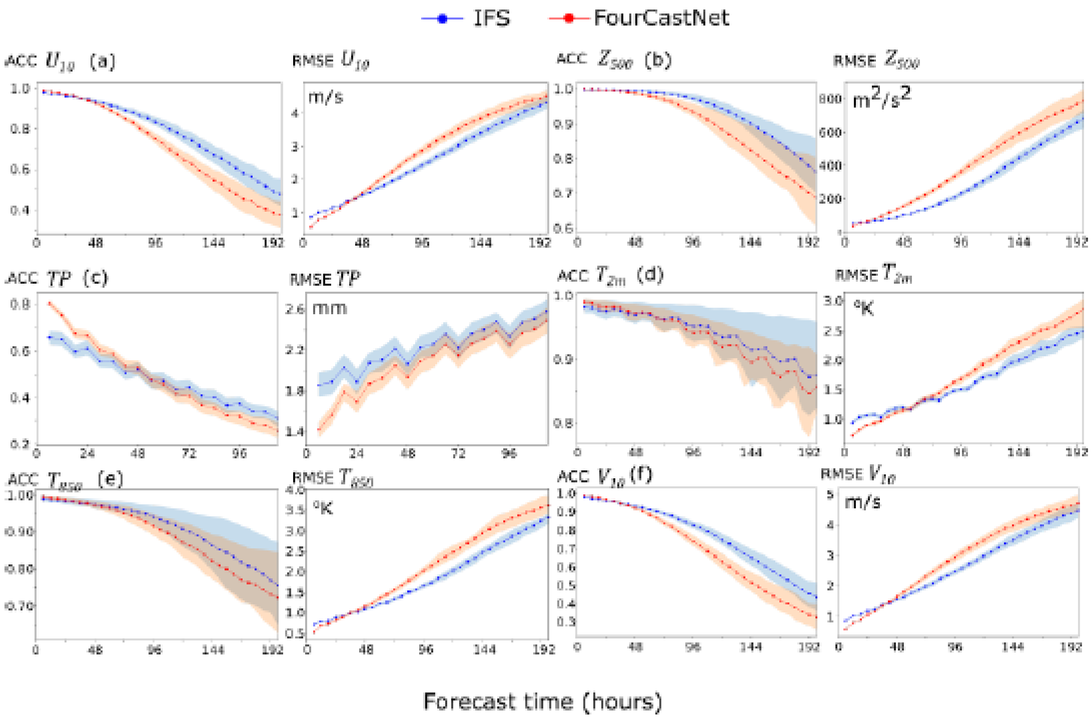
Fourcastnet

- FourCastNet, short for Fourier Forecasting Neural Network, is a global data-driven weather forecasting model that provides accurate short to medium-range global predictions at 0.25 resolution.
- FourCastNet matches the forecasting accuracy of the ECMWF Integrated Forecasting System (IFS), a state-of-the-art Numerical Weather Prediction (NWP) model, at short lead times for large-scale variables, while outperforming IFS for variables with complex fine-scale structure, including precipitation.
- FourCastNet generates a week-long forecast in less than 2 seconds, orders of magnitude faster than IFS.
- The speed of FourCastNet enables the creation of rapid and inexpensive large-ensemble forecasts with thousands of ensemble-members for improving probabilistic forecasting

https://github.com/NVlabs/FourCastNet

Fourcastnet

Figures 13(a-d) show the forecast skill of the FourCastNet model for a few key variables of interest along with the corresponding matched IFS forecast skill. Figure 13 is an extension of Figure 6 in the main text. In Figure 13, we plot the latitude weighted RMSE and latitude-weighted ACC alongside each other for further clarity.

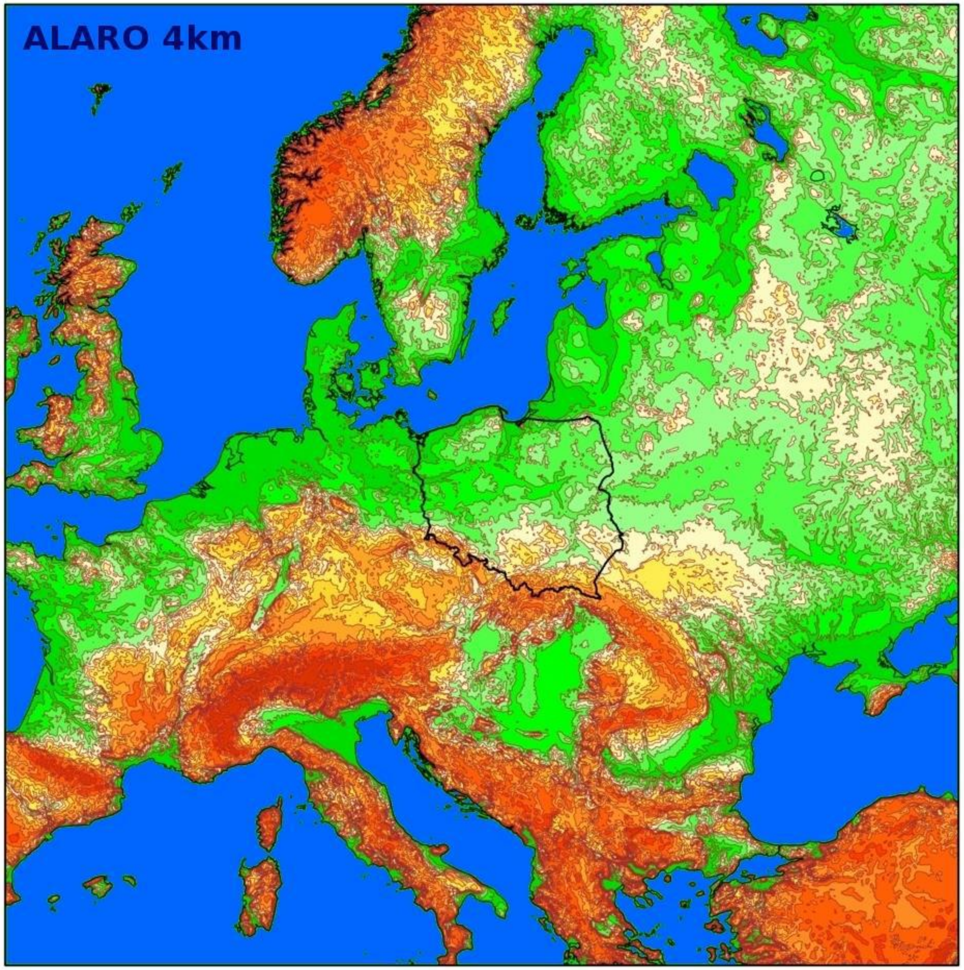


ALARO

ALARO-v1B NH (CY43T2) Operational Domain:

E040 domain:

4.0 km horizontal resolution, 789x789 grid points, 70 vertical model levels on a Lambert projection with 3h coupling frequency and 1h output; Runs 4 times per day (00,06,12 and 18) with 72 hours forecast range; LBC from ARPEGE with 9.4km horizontal resolution; Time step 150s.



Perturbations

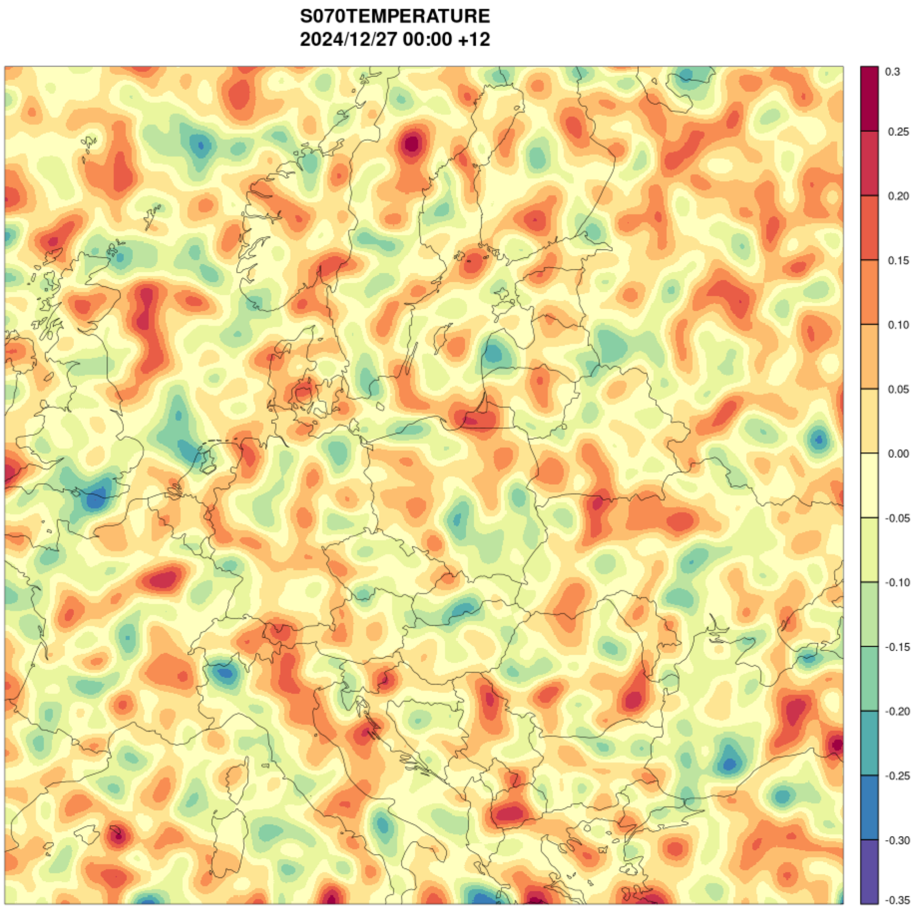
Fourcastnet: 256 members vs. 20 members at 00 + 16 members every 6 hours
1-31 January 2023

ALARO: 30 members vs. 1 member + new members every 12 hours
14-21 February 2025

```
def gaussian_perturb(x, level=0.3, device=0):
    noise = level * torch.randn(x.shape).to(device, dtype=torch.float)
    return (x + noise)
```

```
def physical_perturbation(x, level=0.3, smoothing_sigma=10):
# Generate random Gaussian noise
    noise = np.random.randn(*x.shape)
    # Smooth the noise using a Gaussian filter
    smoothed_noise = gaussian_filter(noise, sigma=smoothing_sigma)
    # Scale the smoothed noise by the field's magnitude
    perturbation = level * smoothed_noise * np.abs(x)
    return x + perturbation
```

Perturbations in ALARO



Epygram in python

levels = [f'{i:03}' for i in range(30, 71)]

fields = ['S{0}TEMPERATURE', 'S{0}HUMI.SPECIFI', 'S{0}WIND.U.PHYS', 'S{0}WIND.V.PHYS']

```
import os
import subprocess
import epygram
import numpy as np
from scipy.ndimage import gaussian_filter

def physical_perturbation(x, level=0.3, smoothing_sigma=10):
# Generate random Gaussian noise
    noise = np.random.randn(*x.shape)
    # Smooth the noise using a Gaussian filter
    smoothed_noise = gaussian_filter(noise, sigma=smoothing_sigma)
    # Scale the smoothed noise by the field's magnitude
    perturbation = level * smoothed_noise * np.abs(x)
    return x + perturbation


# Define levels and fields
levels = [f'{i:03}' for i in range(30, 71)]  # Levels 001 to 070
fields = ['S{0}TEMPERATURE', 'S{0}HUMI.SPECIFI', 'S{0}WIND.U.PHYS', 'S{0}WIND.V.PHYS']

# Generate new filename
original_file = "ICMSHE040INIT"
r = epygram.formats.resource(original_file, "a")
# Loop over levels and fields
for level in levels:
    for field_template in fields:
        field_name = field_template.format(level)
        try:
            # Read the field
            f = r.readfield(field_name)
            # Get spectral geometry
            spectral_geometry = f.spectral_geometry
            # Perform computations
            f.sp2gp()  # Convert to gridpoint space
            f.data = physical_perturbation(f.data)  # Apply physical perturbation
            f.gp2sp(spectral_geometry)  # Convert back to spectral space
            # Write the perturbed field to the resource
            r.writefield(f)
        except Exception as e:
            # Handle any errors (e.g., if a field doesn't exist)
            print(f"Error processing field {field_name} in run {run}: {e}")
# Close the resource
r.close()

print("Perturbations completed!")
```

Run deterministic ALARO forecast RUN 0

Take 12h ICMSH file, pertub it with python script, and start new forecast (12 hours shorter) RUN 0+12
Take 24h ICMSH file, pertub it with python script, and start new forecast (24 hours shorter) RUN 0+24
Take 36h ICMSH file, pertub it with python script, and start new forecast (30 hours shorter) RUN 0+36
...

RUN 0+12

Take 12h ICMSH file, pertub it with python script, and start new forecast (24 hours shorter) RUN 0+12+12
Take 12h ICMSH file, pertub it with python script, and start new forecast (30 hours shorter) RUN 0+12+24
....

RUN 0+12+12

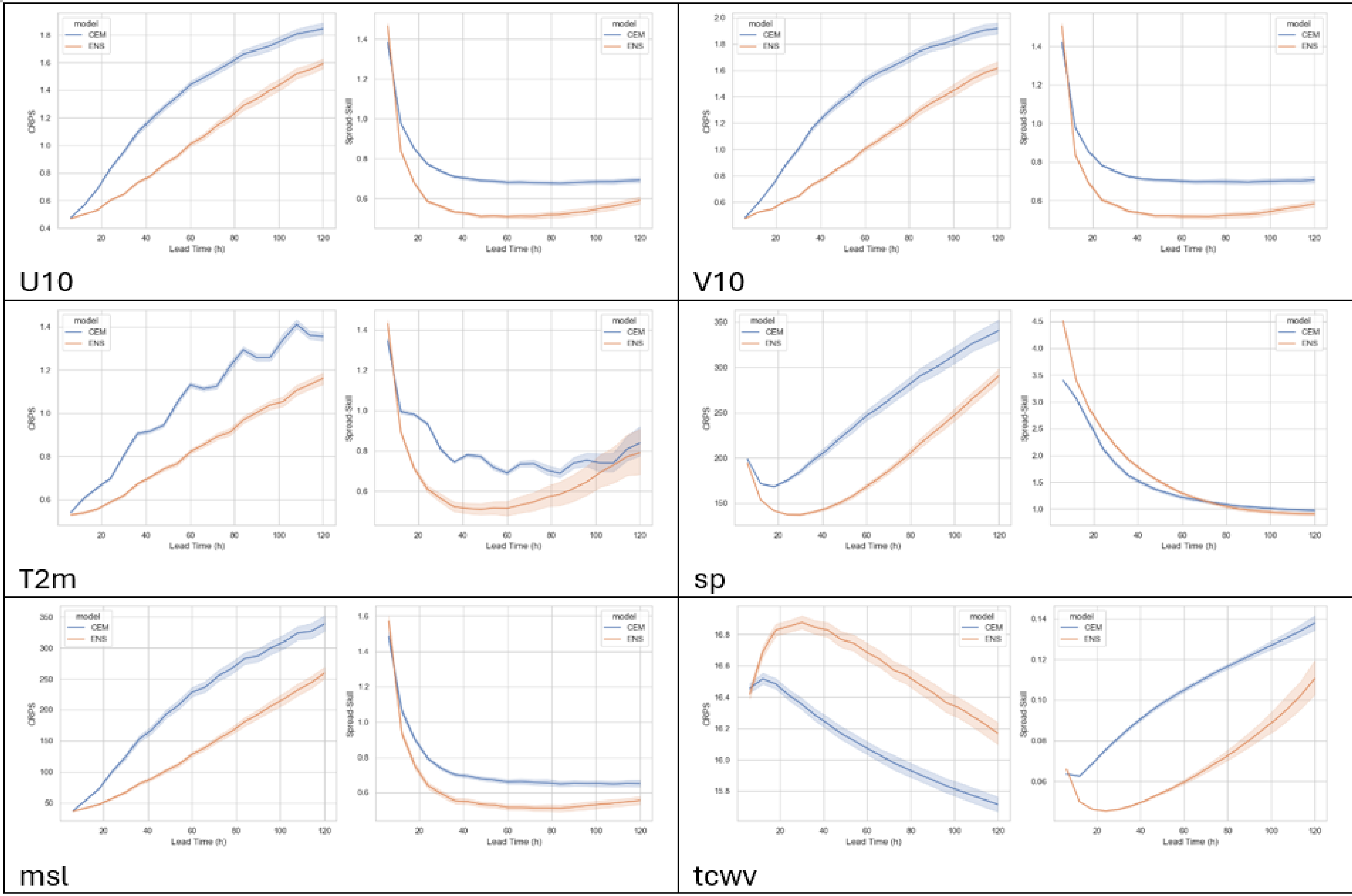
Take 12h ICMSH file, pertub it with python script, and start new forecast (36 hours shorter) RUN 0+12+12+12
...

CEM - Cascading Ensemble Method

Tests of ensemble systems with increasing number of members:

1) Global AI model Fourcastnet (Nvidia) 0.25 resolution, 10 days forecast

Version with increasing number of members (starting with 20 members, and adding 16 members each 6 hours, 256 members at the end of forecast) gives better skill-score and CRPS, when compared with standard system (starting with 256 members). Better results with ½ of computer power and disc usage

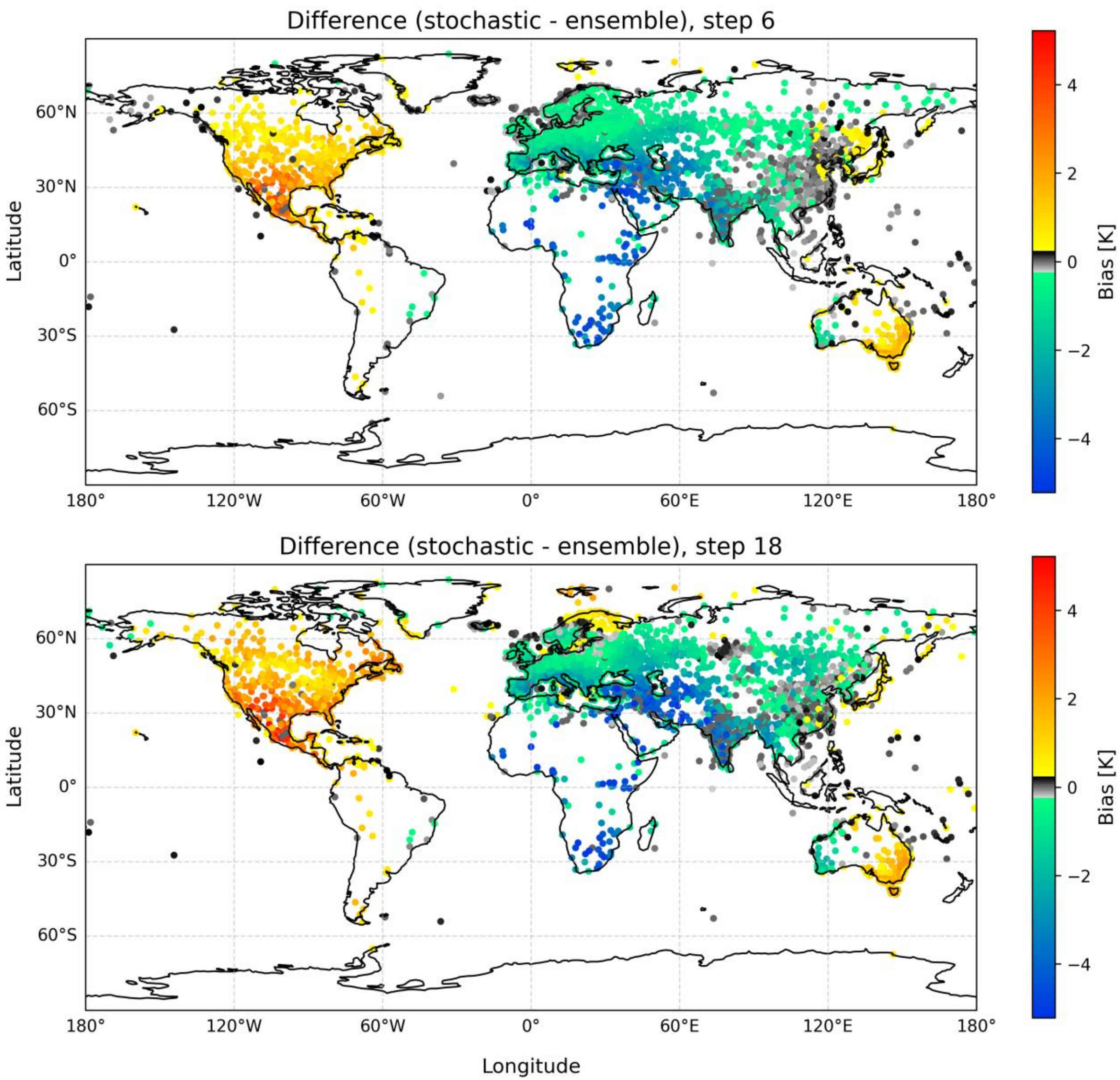


Surface fields

Verification against ERA5

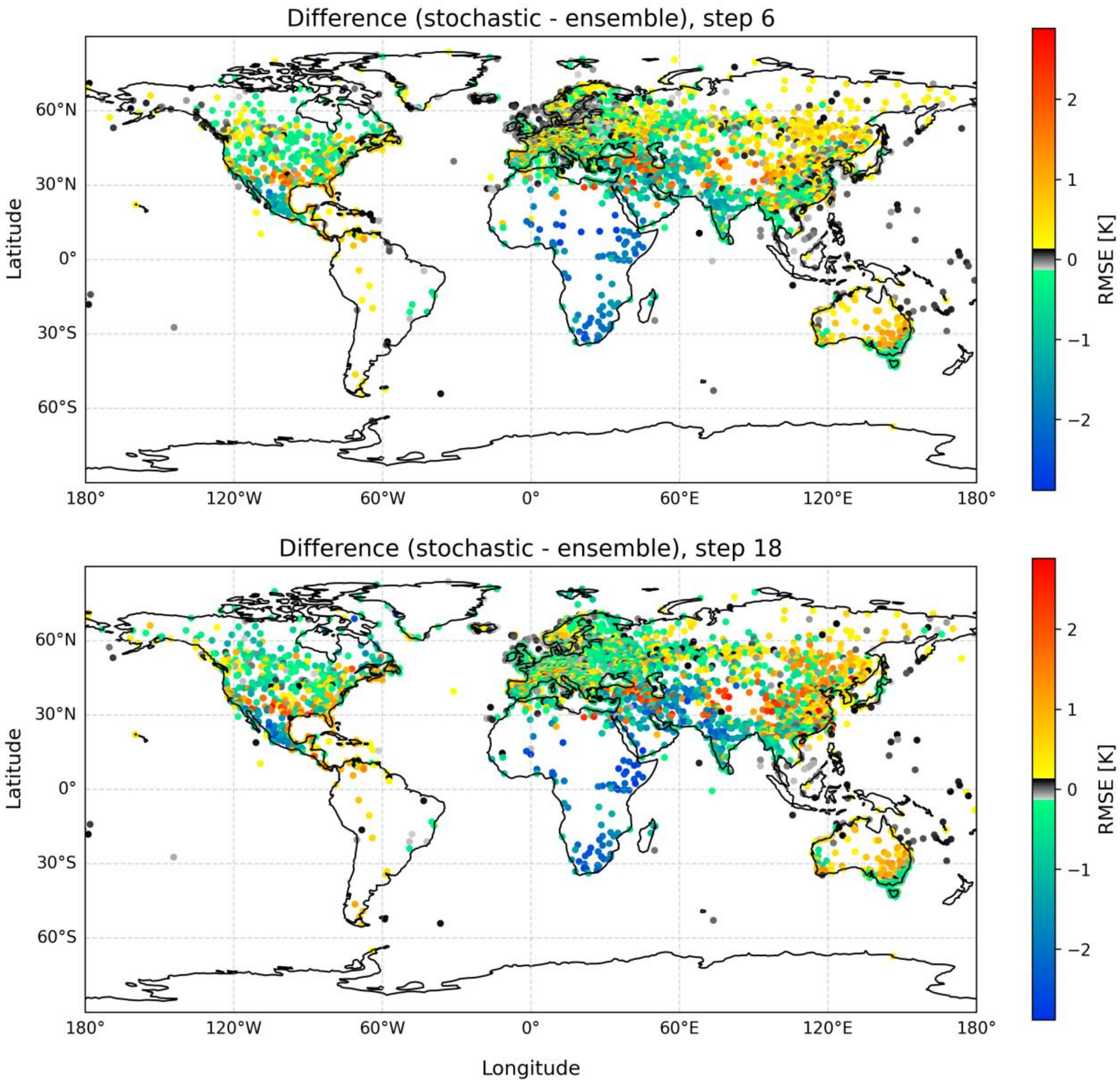
CEM – blue, standard ensemble red

CRPS – left, Spread-skill right



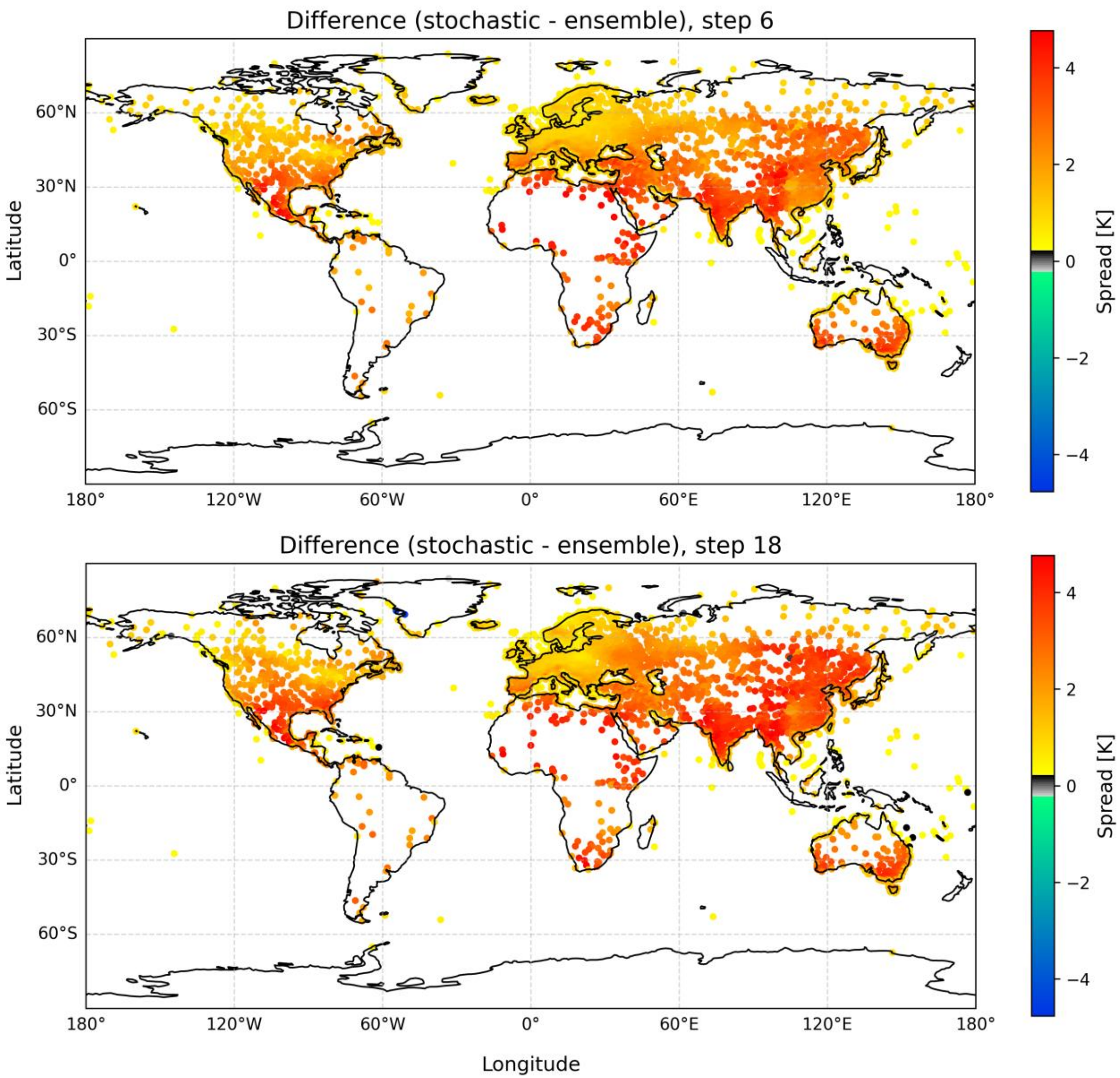
T2m verification against global **synop** data, 6th (top) and 18th timestep (bottom)

BIAS difference between CEM and standard ensemble system
Green – blue colors indicates better results for CEM method



T2m verification against global **synop** data, 6th (top) and 18th timestep (bottom)

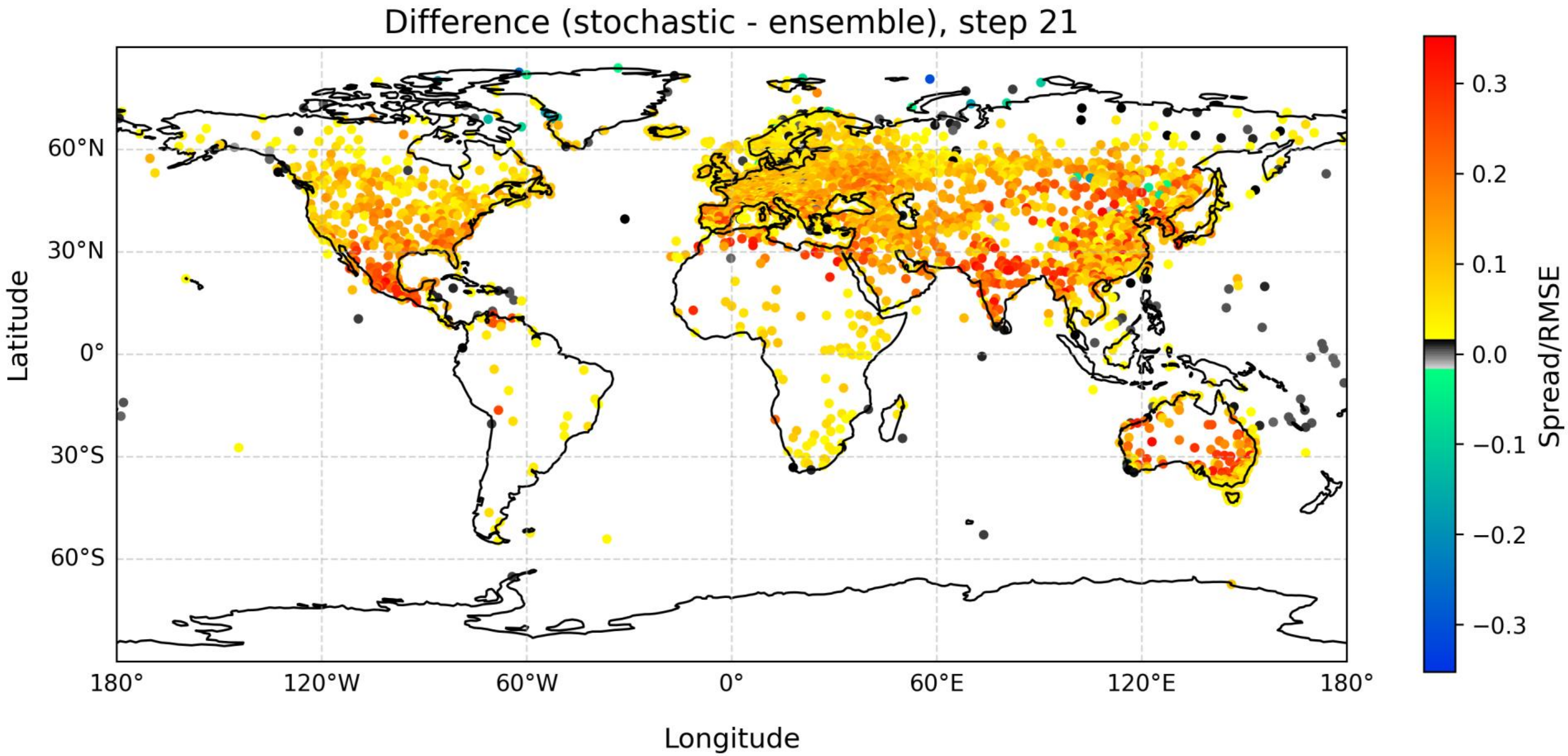
RMSE difference between CEM and standard ensemble system
Green – blue colors indicates better results for CEM method



T2m verification against global **synop** data, 6th (top) and 18th timestep (bottom)

Spread difference between CEM and standard ensemble system Yellow – red colors indicates better results for CEM method

METEO
IMGW-PIB
meteo.imgw.pl



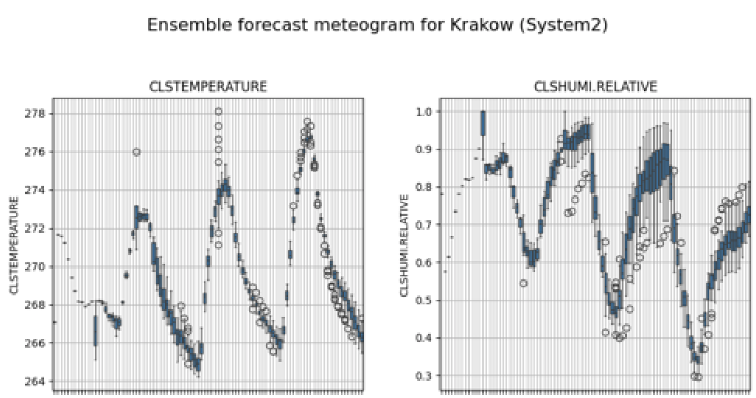
T2m verification against global **synop** data, last timestep

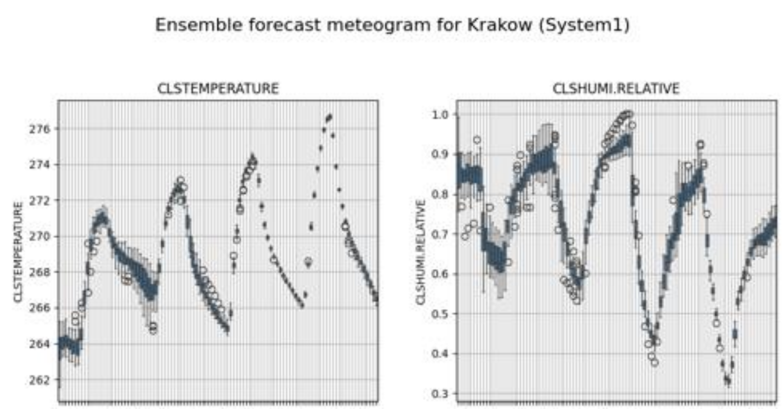
Spread-skill difference between CEM and standard ensemble system
Yellow – red colors indicates better results for CEM method

Tests of ensemble systems with increasing number of members:

2. ALARO 4 km, 102 hours forecast (**preliminary results**) 14-21 February 2025

Version with increasing number of members (starting with 1 member, and adding new members each 12 hours up to 30 members at +60 hours compared with standard system (starting with 30 members). Better spread for ½ of computer power and disc usage. Perturbing with gaussian random noise, with the same LBC for all members.





Ensgrams for 18th Feb 2025 for CLSTEMP and CLSHUMI.RELATIVE for CEM (left) and classical ensemble (right). Spread in CEM is increasing faster with time then in case of classical ensemble system.

Future work

Longer verification period, summer cases.

Smaller domain with higher resolution and shorter forecasts.

Better perturbation, adding also different boundary conditions for members.

Add perturbations in specific cases; Convection, freezing rain, inversion etc.: when deterministic model predicts such situation, add new member „on-demand” with perturbed atmospheric state.