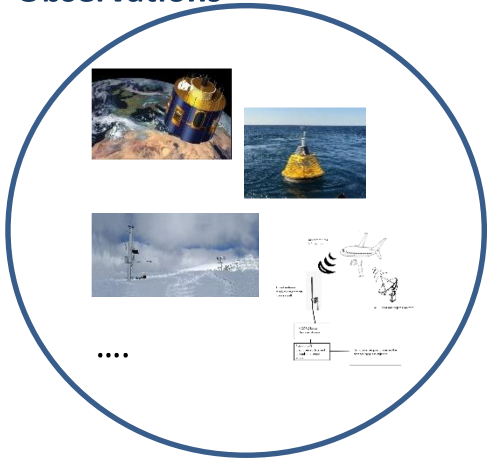
Machine Learning applications in meteorological forecasting - classics and where federated learning could be useful

I. Schicker ZAMG

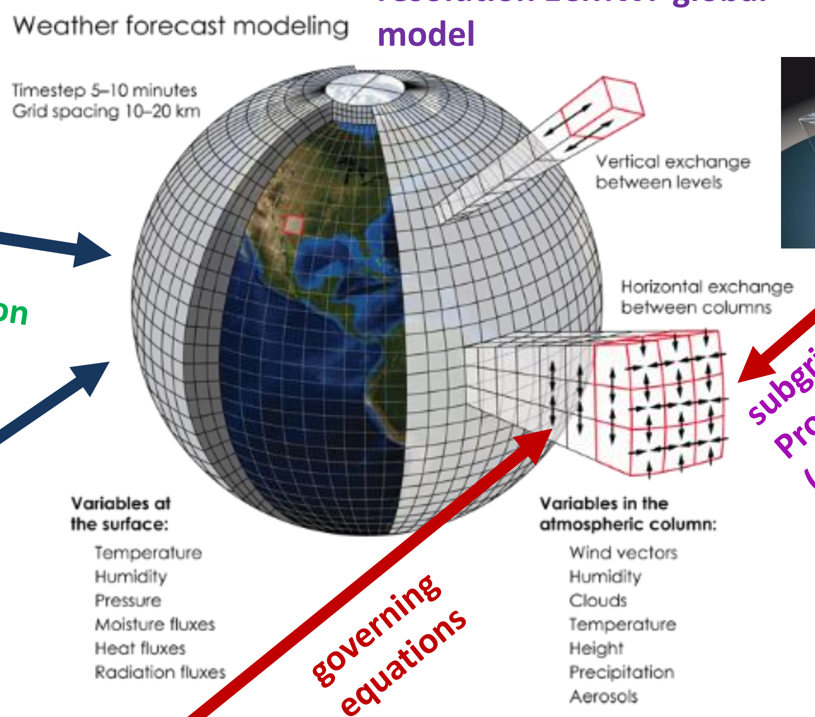
ZAMG
Zentralanstalt für
Meteorologie und
Geodynamik

Numerical weather prediction and weather forecasting

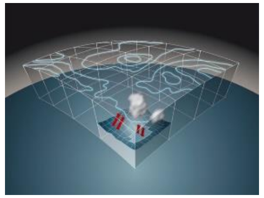
Observations



data assimilation



9 x 9 km / 11 x 11 km grid resolution ECMWF global model



subgrid scale Processes (also equations / parametrizations)

governing equations

"Primitive" Weather Forecasting Equations

$p = \rho R T$ *Ideal Gas Law* (Equation of State)

$\vec{a}_h = \sum\left(\vec{F}_h / m\right)$ *Newton's Second Law of Motion*

$\Delta p = -\rho g \Delta z$

$(PGA)_v = g$

$\vec{a}_v = \sum(\vec{F}_v / m) = (\vec{P}\vec{G}\vec{A})_v - \vec{g}$

Hydrostatic Law (Obtained from the Equation of Vertical Motion)

$\Delta T = \Delta q / c_p + (1/\rho)\Delta p$ *First Law of Thermodynamics*

$(1/\rho)\Delta\rho/\Delta t = -DIV$

Conservation of Mass Applied to the Atmosphere (Equation of Continuity)

Zonal wind:

$$\frac{\partial u}{\partial t} = \eta v - \frac{\partial \Phi}{\partial x} - c_p\theta\frac{\partial \pi}{\partial x} - z\frac{\partial u}{\partial \sigma} - \frac{\partial\left(\frac{u^2+v^2}{2}\right)}{\partial x}$$

Meridional wind:

$$\frac{\partial v}{\partial t} = -\eta\frac{u}{v} - \frac{\partial \Phi}{\partial y} - c_p\theta\frac{\partial \pi}{\partial y} - z\frac{\partial v}{\partial \sigma} - \frac{\partial\left(\frac{u^2+v^2}{2}\right)}{\partial y}$$

Temperature:

$$\frac{\delta T}{\partial t} = \frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} + w\frac{\partial T}{\partial z}$$

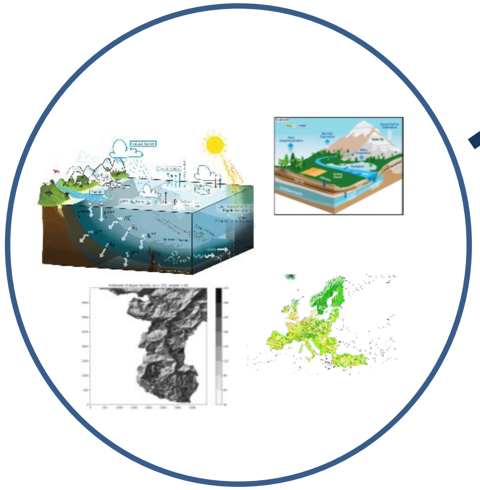
Precipitable water:

$$\frac{\delta W}{\partial t} = u\frac{\partial W}{\partial x} + v\frac{\partial W}{\partial y} + w\frac{\partial W}{\partial z}$$

Pressure thickness:

$$\frac{\partial}{\partial t}\frac{\partial p}{\partial \sigma} = u\frac{\partial}{\partial x}x\frac{\partial p}{\partial \sigma} + v\frac{\partial}{\partial y}y\frac{\partial p}{\partial \sigma} + w\frac{\partial}{\partial z}z\frac{\partial p}{\partial \sigma}$$

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} + \omega\left(\frac{\partial T}{\partial p} + \frac{RT}{pc_p}\right) = \frac{J}{c_p} \qquad \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial \omega}{\partial p} = 0 \qquad 0 = -\frac{\partial \phi}{\partial p} - \frac{RT}{p}$$



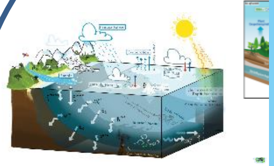
Ocean and surface model
Static fields

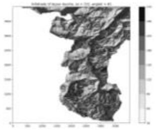
ZAMG
Zentralanstalt für Meteorologie und Geodynamik

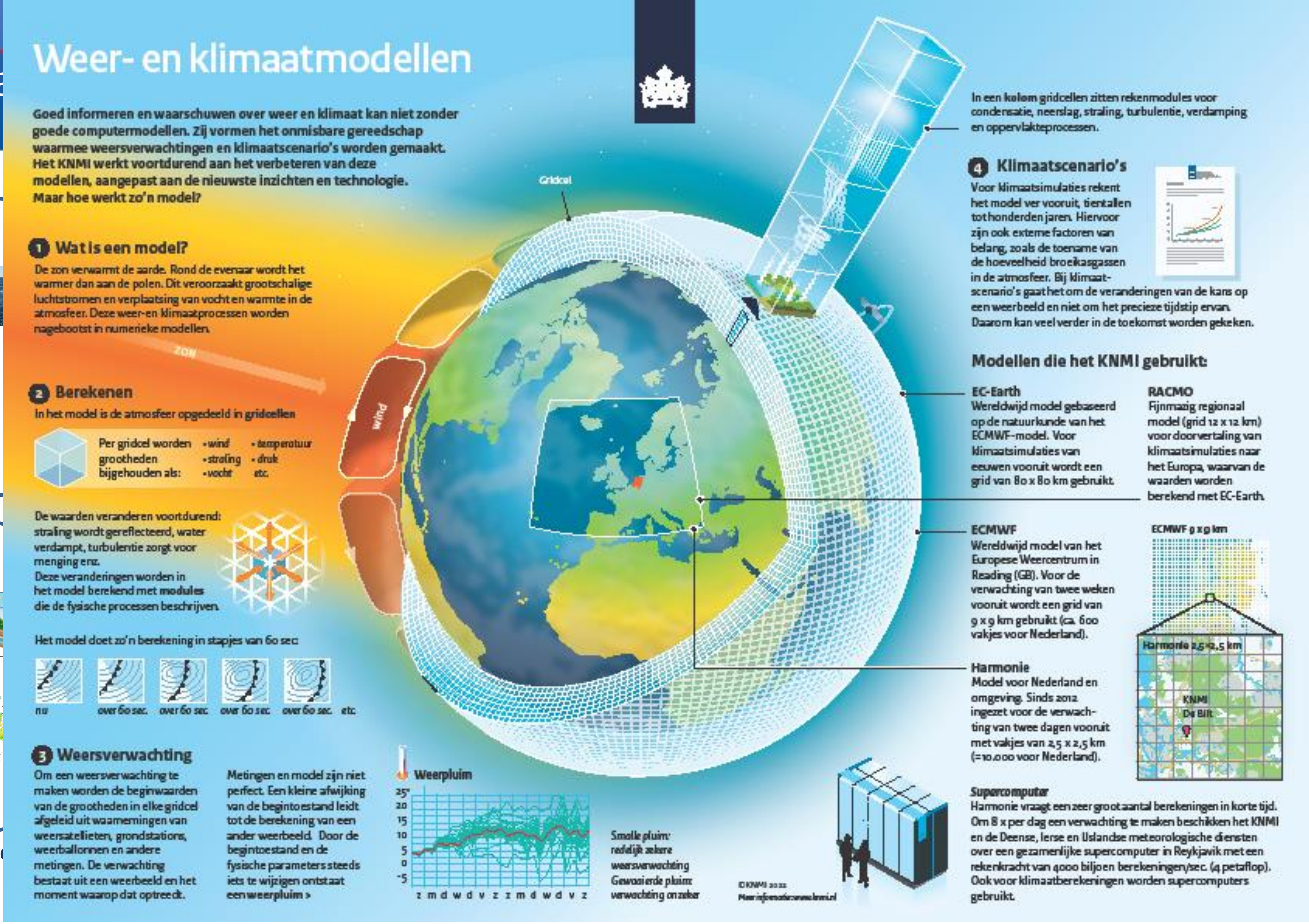
Numerica

Observations

....





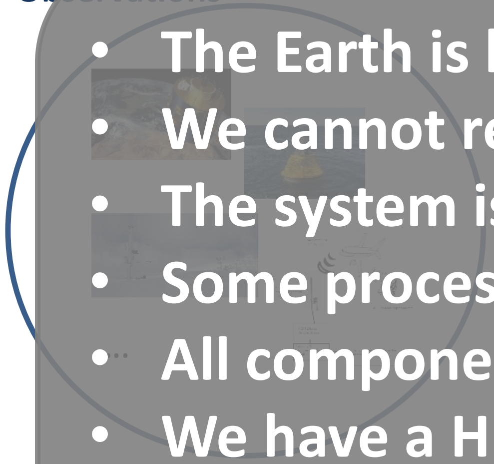


Ocean and surfac
Static fields

ZAMG
Zentralanstalt für Meteorologie und Geodynamik

Numerical weather prediction and weather forecasting

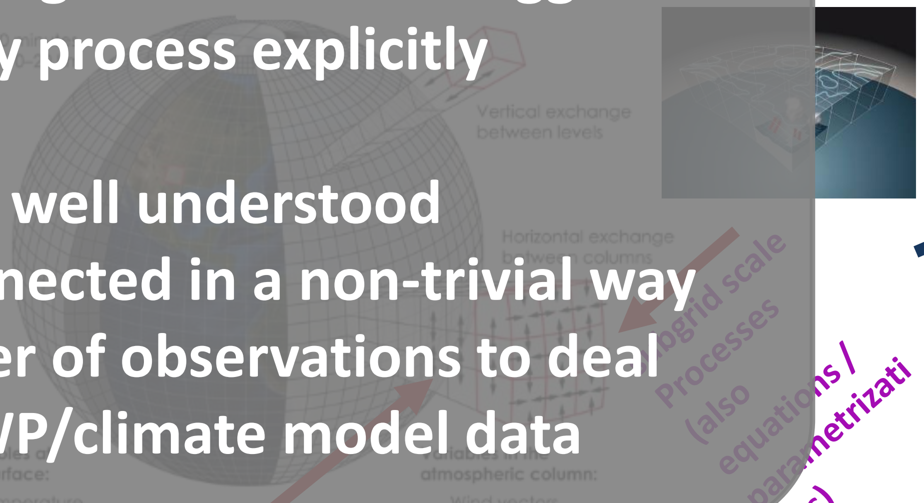
Observations



- The Earth is huge and ranges from flat to rugged
- We cannot resolve every process explicitly
- The system is chaotic
- Some processes are not well understood
- All components are connected in a non-trivial way
- We have a HUGE number of observations to deal with and even more NWP/climate model data

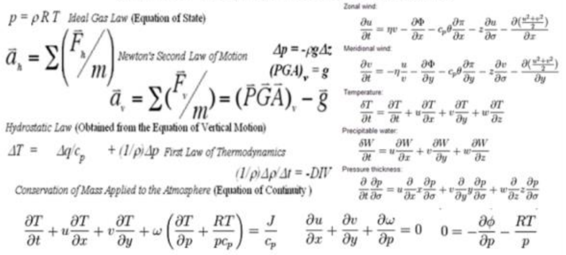
9 x 9 km / 11 x 11 km grid resolution ECMWF global model

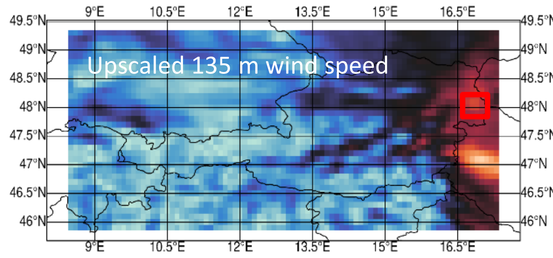
data assimilation



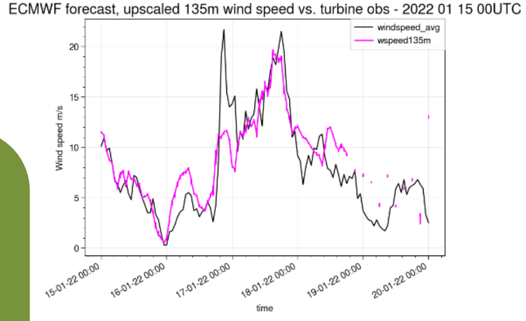
Subgrid scale Processes (also equations / parametrizations)

governing equations

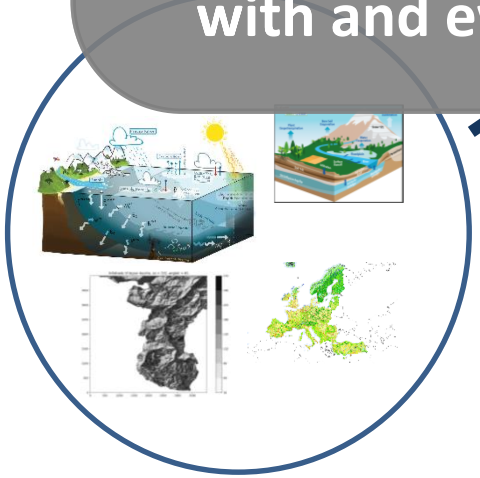




Post-processing



Machine Learning (?)

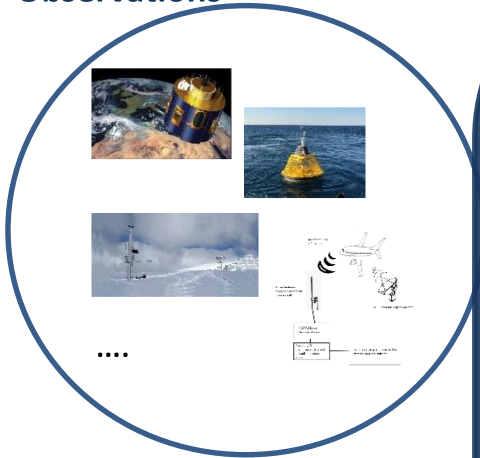


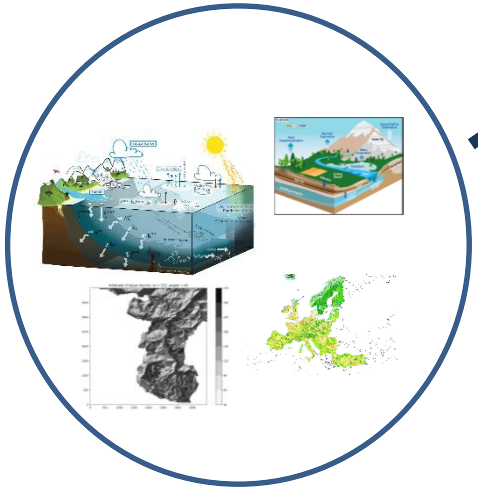
Ocean and surface model
Static fields

ZAMG Zentralanstalt für Meteorologie und Geodynamik

Numerical weather prediction and weather forecasting – and machine learning (?)

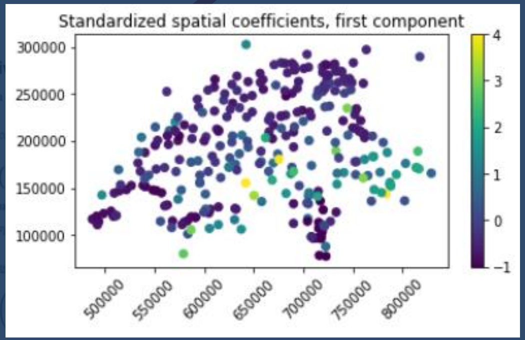
Observations

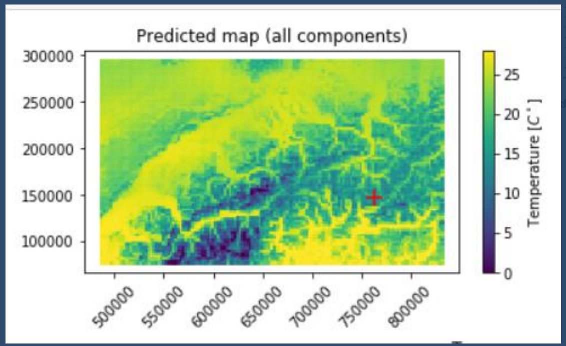




Ocean and surface model
Static fields

- **Data monitoring**
- **Real time quality control**
- **Anomaly detection**
- **Data cleaning and filtering for longer, historic time series**
- **Observation spatial interpolation / interpolation to unobserved areas**
- **Data fusion of different sources**
- **Guided decision making**
- **Correction of observation error**
- **Filling of missing values in time series**



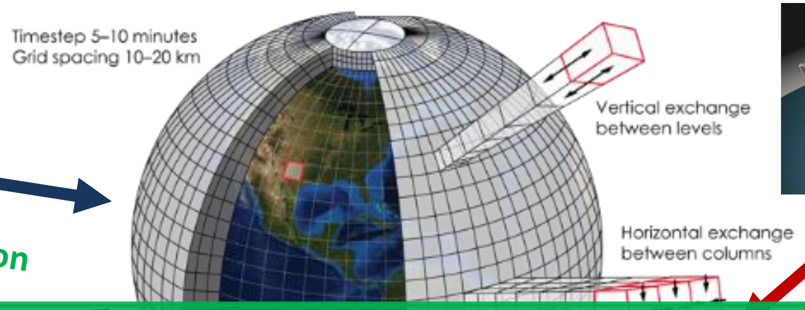


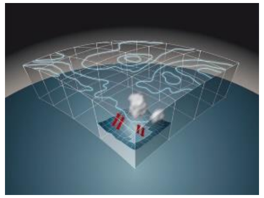
Numerical weather prediction and weather forecasting – and machine learning (?)

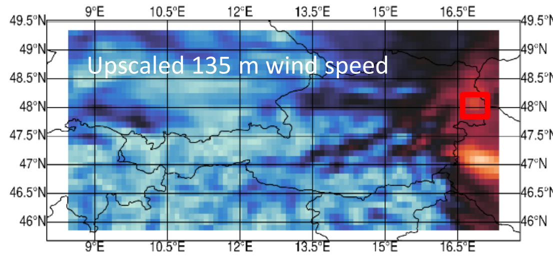
Observations

data assimilation

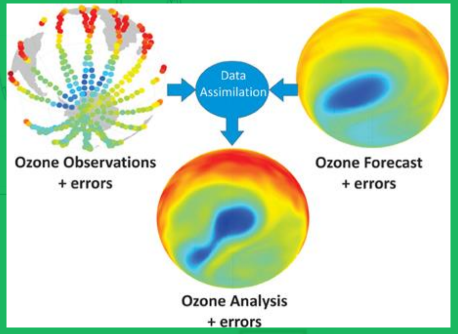
9 x 9 km / 11 x 11 km grid resolution ECMWF global model

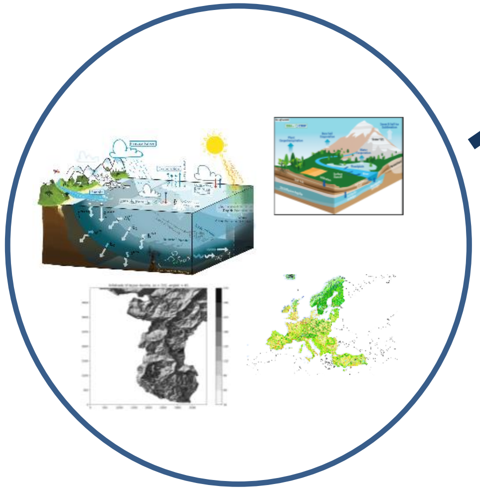






- **learn govering equations**
- **perform non-linear bias correction of observations**
- **Define Bias predictors**
- **Operational operators**
- **Define optical properties of hydrometeors / aerosols**
- **Emulate conventional tools to improve efficiency**
- **Perturb the data to generate an ensemble**



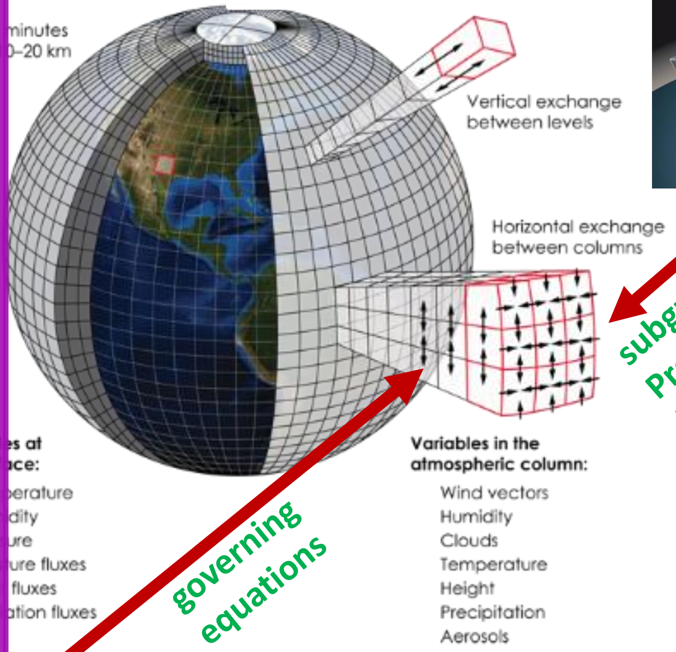


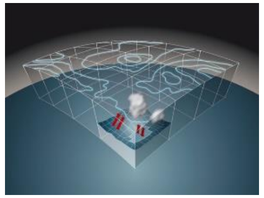
Ocean and surface model
Static fields

Numerical weather prediction and weather forecasting – and machine learning (?)

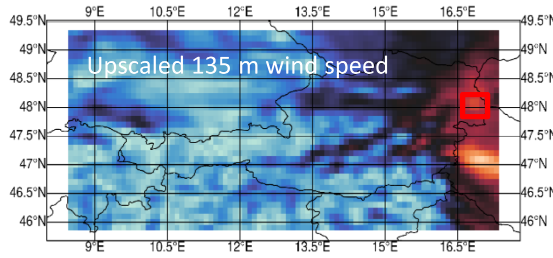
- Emulate model components
- Improve existing parametrizations
- Replace parts of parametrizations
- Better error models
- Learn equations of motion
- Generate conjoined code

9 x 9 km / 11 x 11 km grid resolution ECMWF global model

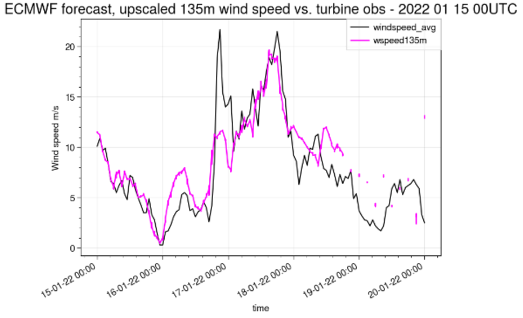




subgrid scale Processes (also equations / parametrizations)



Post-processing



"Primitive" Weather Forecasting Equations

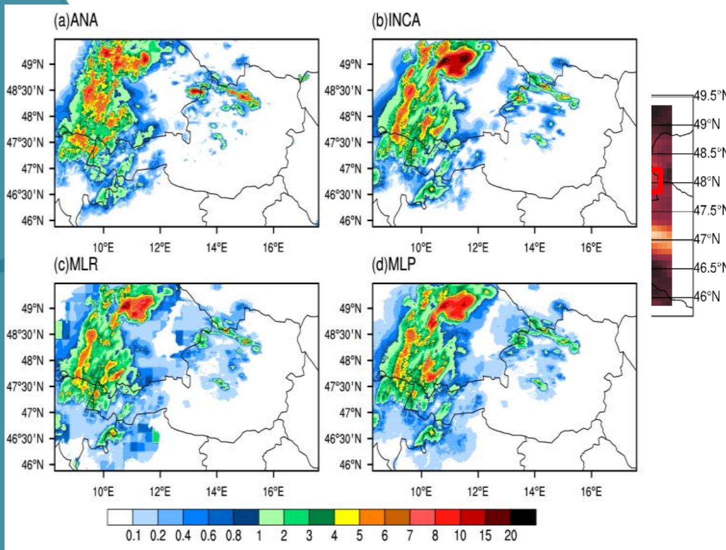
ZAMG Zentralanstalt für Meteorologie und Geodynamik

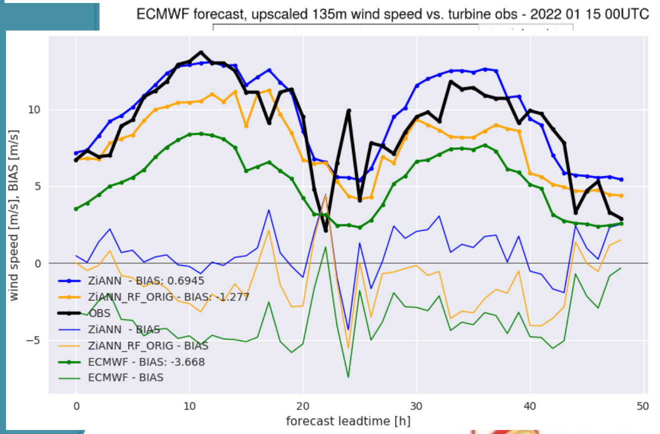
Numerical weather prediction and weather forecasting – and machine learning (?)

Observations

9 x 9 km / 11 x 11 km grid resolution ECMWF global model

- Adjustments of forecast products for renewable energy applications, nowcasting/forecasting of severe weather
- Improvements of forecast products for sub-seasonal to seasonal prediction
- Feature detection
- Uncertainty quantification and „cheap“ ensembling
- Low complexity models for research purposes
- Data driven forecasting
- Generation of synthetic data / data augmentation for algorithm training
- Increasing of spatial-temporal resolution (< 1 km, < hourly)

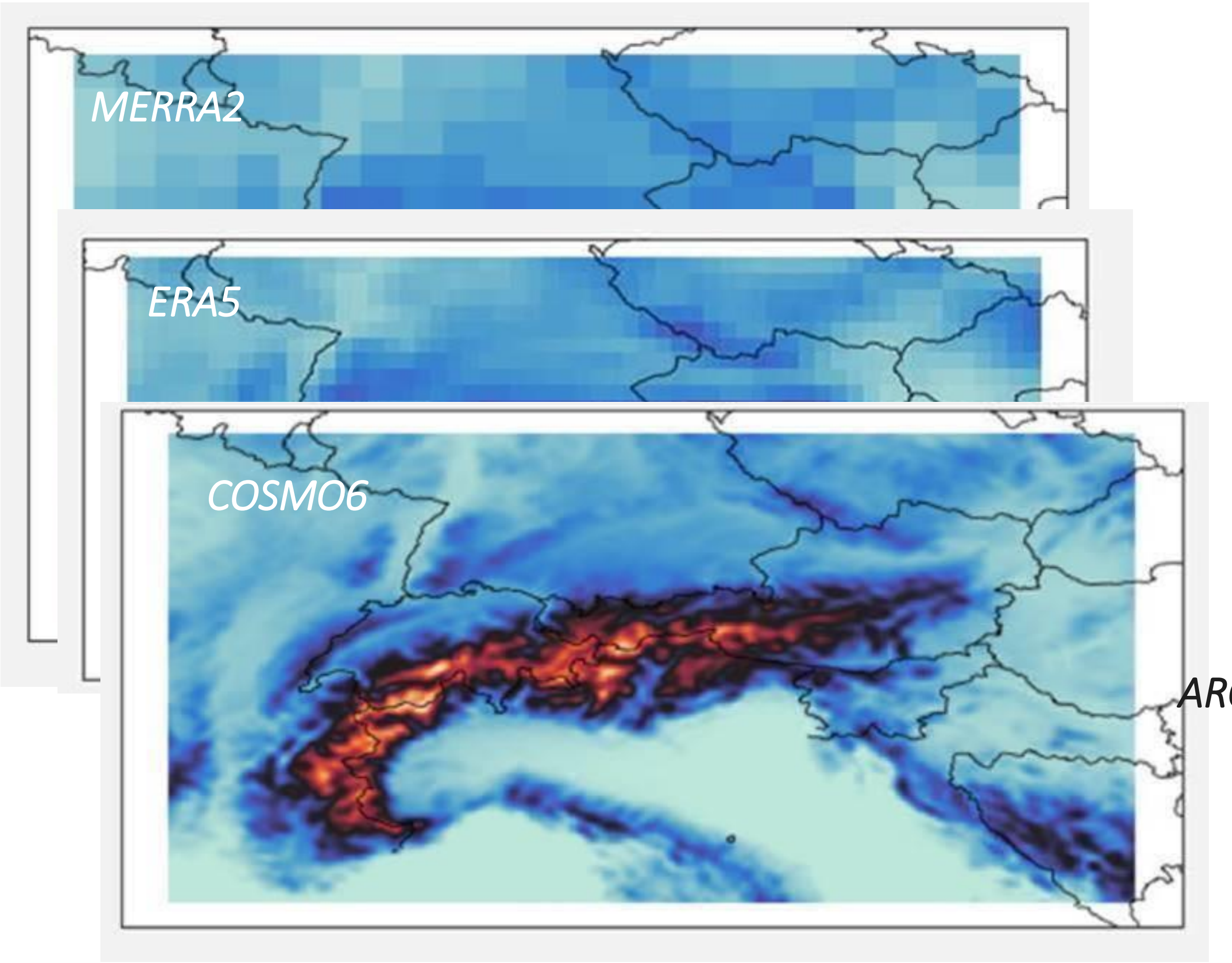




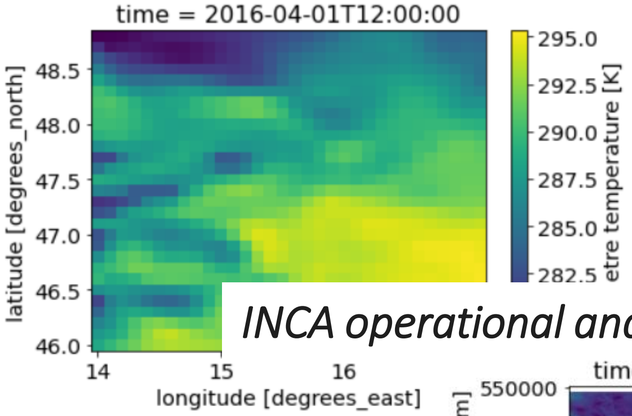
Ocean and surface model
Static fields

ZAMG Zentralanstalt für Meteorologie und Geodynamik

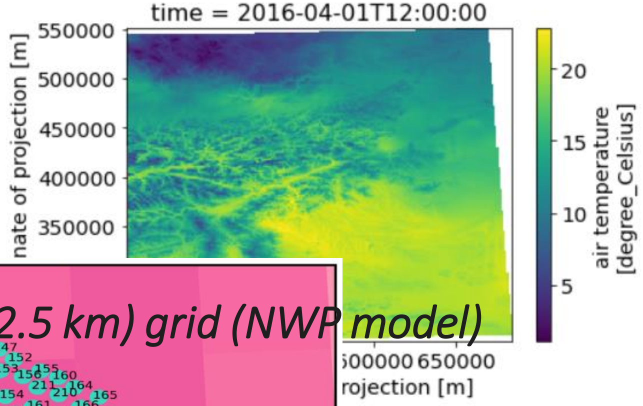
Post-processing – why is it important



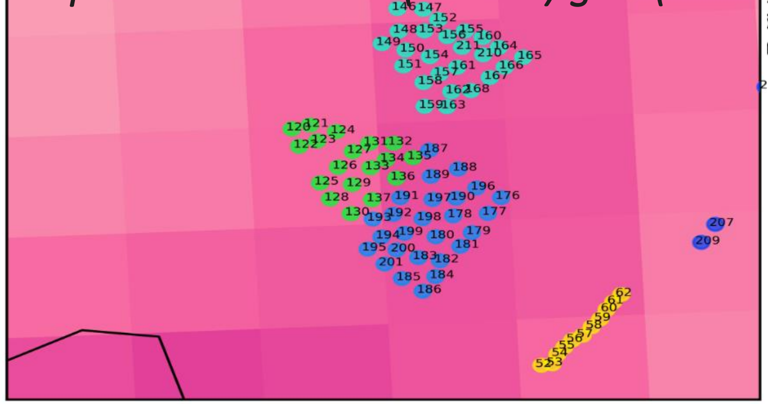
ECMWF operational global model (~ 9 km) – Eastern Austria



INCA operational analysis 1 km– Eastern Austria



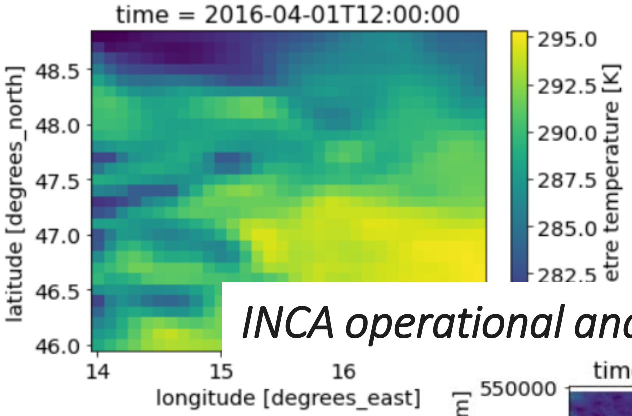
AROME operational model (2.5 km) grid (NWP model)



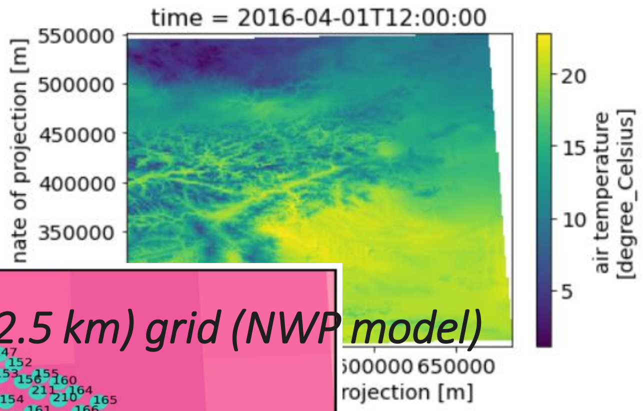
Post-processing – why is it important

- Grid resolution too coarse
- Temporal resolution for some applications too coarse
- Temporal horizon not long enough
- NWP models prone to different error sources (data, assimilation, parametrizations), bias / spread need to be corrected

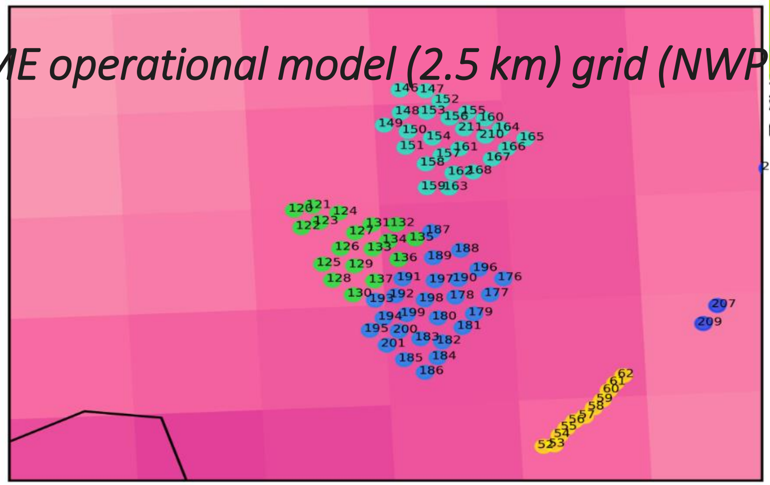
ECMWF operational global model (~ 9 km) – Eastern Austria



INCA operational analysis 1 km– Eastern Austria



AROME operational model (2.5 km) grid (NWP model)



ZAMG
Zentralanstalt für Meteorologie und Geodynamik

Post-processing – „classical“ methods

- „classical“ as age/idea of methods is >= 50 years dating back to the 1960ies
- Are based on (current) observations and/or recent weather conditions and don’t use numerical models (‘data driven’)
- Observations of the initial (predictor) and resultant (predictands) weather conditions are a must have
- Example: forecast the temperature for tomorrow with input consisting of observational data available at the time the forecast was issued:

$$\widehat{Y}_t = \sum f_C(X_0) \quad \dots\dots\dots\dots\dots \quad (1)$$

Y_t = predictand (dependent variable) at time ‘t’

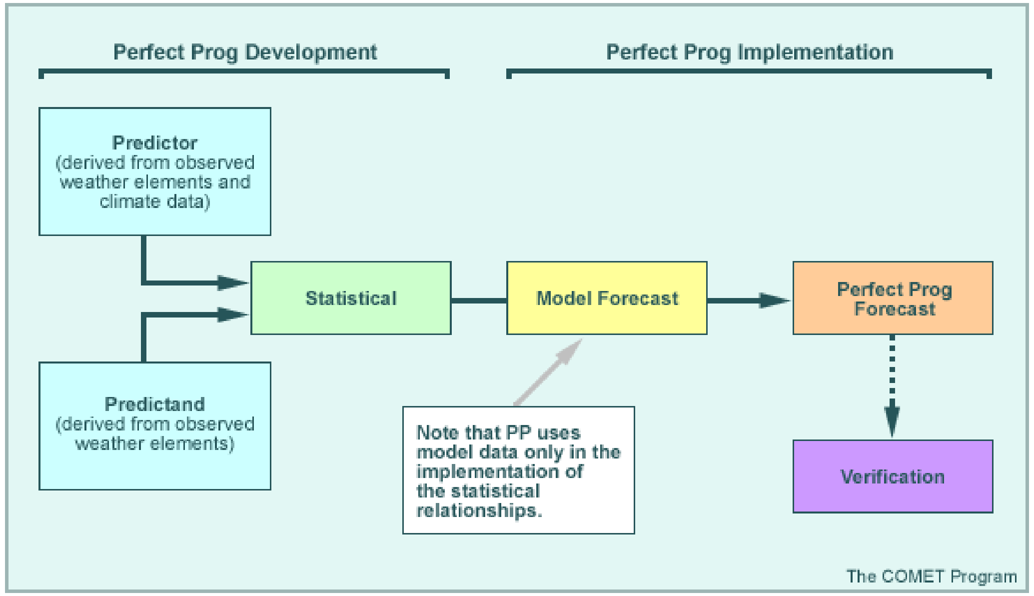
X_0 = predictor vector of observational data (independent variables) available at initial time 0

- Works good for short ranges and very long ranges, no skill in medium range (~24h to 10 days)
- Best under persistend weather conditions with low variability

Post-processing – „classical" methods

Perfect prognosis / perfect prog

- The need to accurately predict surface weather elements, led to the development of Perfect Prognosis Method (PPM) (Klien et al., 1959).
- objective method, in which, a concurrent relation is developed between the parameter to be predicted and the observed circulation around the location of interest, using several years of data
- based on the assumption that numerical model forecasts are "perfect"
- numerical models are not perfect but this approach gives an estimate of what to expect, if numerical models are correct
- does not require numerical model data for development, uses numerical output when equations are applied operationally
- make sure that variables used as 'Predictors' in development of Perfect Prognosis Equations will be available from NWP models for operational purposes



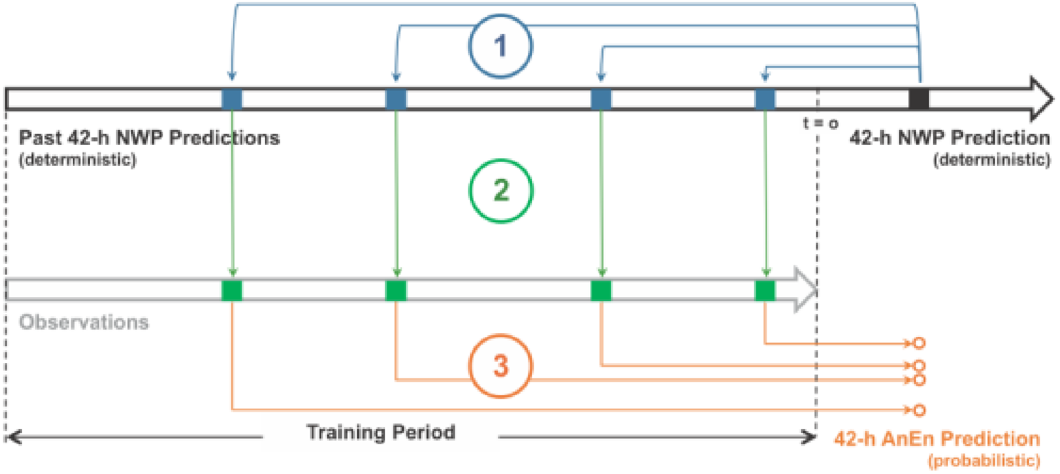
Post-processing – „classical“ methods

Analog-based methods - AnEn

- the current state of the atmosphere is compared with a repository of other, historical states of the atmosphere to determine the most similar scenario in the past (an analog) (Van den Dool, 1989; Hamill and Whitaker, 2006; Delle Monache et al., 2011; Delle Monache et al., 2013).
- Lorenz (1969): analogues refer to “two states of the atmosphere which resemble each other rather closely” and “Each state may then be looked upon as equivalent to the other state plus a reasonably small ‘error’.”
- Are used in meteorology, analogs primarily for pre- and post- processing of NWP forecasts (Hamill and Whitaker, 2006).

$$\|F_t, A_{t'}\| = \sum_{i=1}^{N_v} \frac{w_i}{\sigma_{f_i}} \sqrt{\sum_{j=-\tilde{t}}^{\tilde{t}} \left(F_{i,t+j} - A_{i,t'+j}\right)^2}$$

Ft = forecast to be corrected at a given time t and specific station location;
At = analog forecast at time t’ before Ft is issued and at the same location.
Nv, **wi** are the number of predictors and their weights, respectively;
σfi i= standard deviation of the time series of past forecasts of a given variable at the same location
t~ = an integer equal to half the width of the time window over which the metric is computed.
Fi,t+j and Ai,t+j = values of the prediction and the analog in the time window for a given variable.
This metric describes the quality of the analog chosen and is based upon the similarity of the current forecast window to the past forecast time windows available in the historical dataset. E.g., for a three-hour forecast the window would consist of three points, t-3hr, t, and t+3hr.



Delle Monache et al., 2013

Hybrid analogs – neural networks (e.g. CapsNet)

Post-processing – „classical“ methods

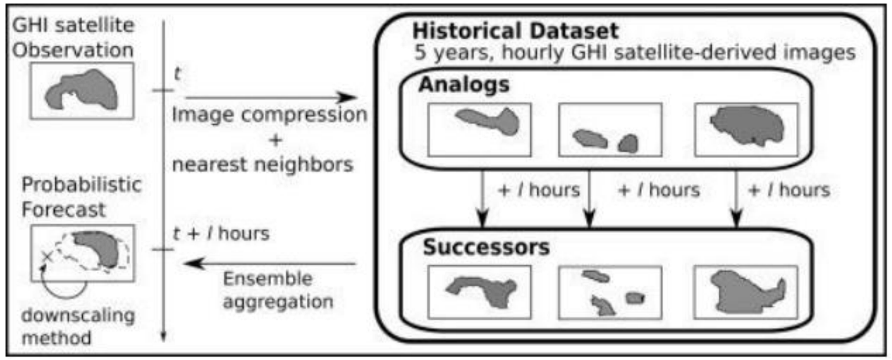
Example application solar energy

Analog-based methods – „data-driven“ AnEn with spatial search field (satellite/radar/analysis field)

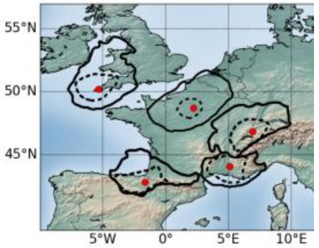
- Given a, e.g., satellite image of the current state of the atmosphere (the *observation* or *truth*), → search in a historical database N images that resemble the observation (the *analogs*)
- analogs are found running a k-nearest neighbors algorithm on compressed images (into four *features*)

```
def knn_analogs(self, obs, analog_zones, k=80):
    """
    Returns the k nearest neighbors of the observation obs from the dataset analog_zones. The results are cleaned to not include more t
    :param obs: observation data 4D feature representation
    :param analog_zones: analog neighbor candidates 4D feature representation
    :param k: num of neighbors
    :return: k nearest neighbors
    """
    neigh = NearestNeighbors(n_neighbors=k, metric='euclidean')
    neigh = neigh.fit(analog_zones.to_array().to_numpy().transpose())
    dists, indizes = neigh.kneighbors(
        obs.to_array().to_numpy().transpose(), 80, True)
    analog_time = analog_zones.coords['time']
    analogs = analog_time[indizes.flatten()]
    analogs = self.check_successors(analogs)
    analogs_filter = list()
    t_delta = np.timedelta64(12, 'h')
    for d in analogs.values:
        d_min = d - t_delta
        d_max = d + t_delta
        if not len(analogs_filter):
            analogs_filter.append(d)
        else:
            tmp = True
            for t in analogs_filter:
                if d_min <= t <= d_max:
                    tmp = False
            if tmp:
                analogs_filter.append(d)
    return self.ds_lim.sel(
        time=self.ds_lim.time.isin(analogs_filter)).load()
```

Gfähler, Schicker 2022



Ayet and Tandeo, 2018



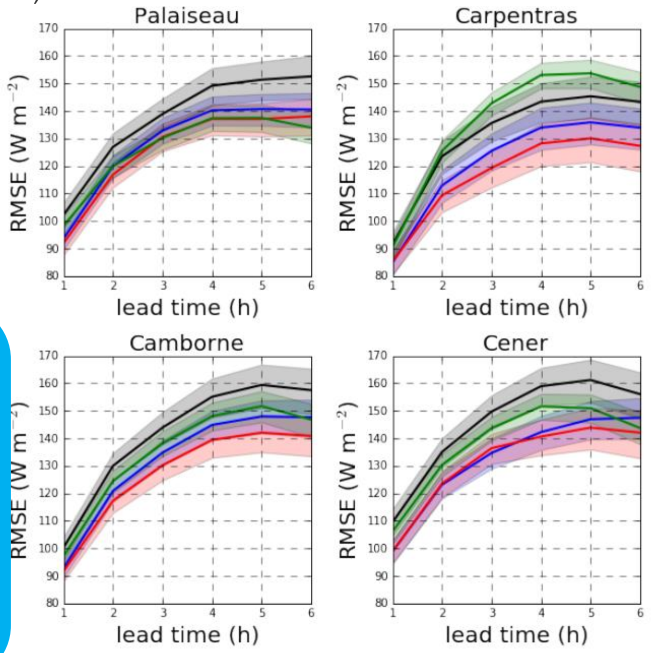


Figure 13: Normalized "ground" RMSE and corresponding 95% bootstrap confidence interval as a function of lead time for the analog method (blue), the post-processed analog method (red), the persistence method (black), and the adaptive VAR(1) model (green).

Not shown:
- MOS
- BMA
- ECC / Schaake Shuffle
- …

Implemented in test version, currently adapted for operational purposes

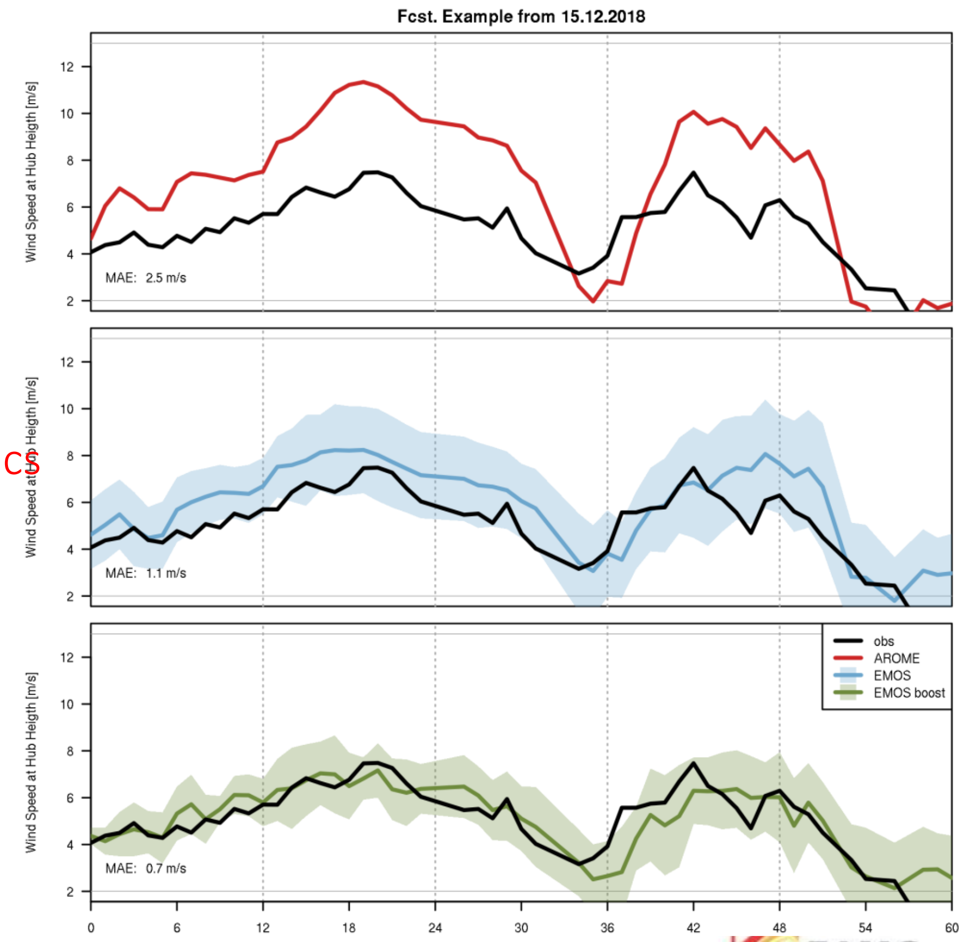
Post-processing – hybrid methods

EMOS – ensemble model output statistics on point and grid

Uses:

- numerical weather prediction data, determinisic/probabilisitc
- Observations of official weather obs site
- OR: gridded analysis fields
- Based on non-homogeneous gaussian regression
- Originally implemented in Fortran, rewritten in R and python

Boosting: rather machine learning than pure statistics



EMOS:

$$y \sim N(\mu, \sigma)$$
$$\mu = b_0 + b_1 \mathrm{m}$$
$$\sigma = c_0 + c_1 \mathrm{s}$$

EMOS boost:

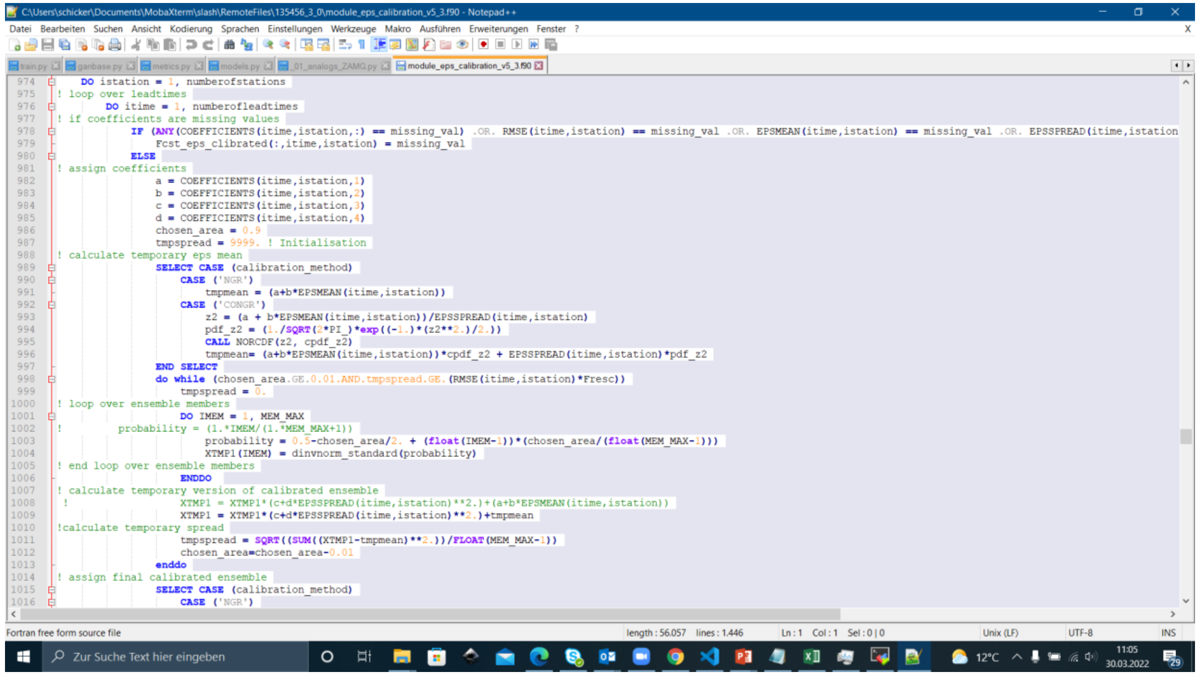
$$y \sim N(\mu, \sigma)$$
$$\mu = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + \ldots$$
$$\sigma = c_0 + c_1 z_1 + c_2 z_2 + c_3 z_3 + \ldots$$

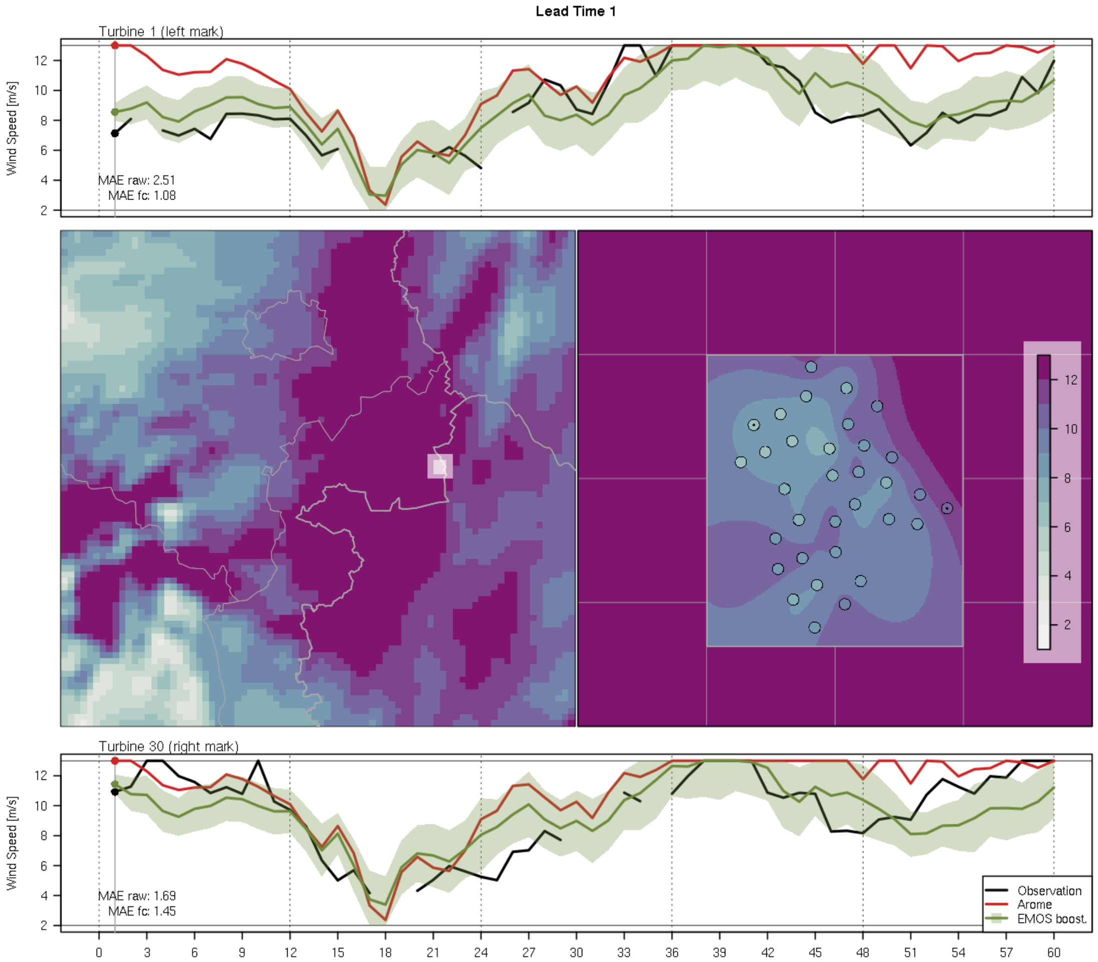
Advantage: we get an uncertainty estimation on-the-fly with the statistical-based method

Messner, J.W., G.J. Mayr, and A. Zeileis, 2017: Nonhomogeneous Boosting for Predictor Selection in Ensemble Postprocessing. *Mon. Wea. Rev.*, **145**, 137–147, https://doi.org/10.1175/MWR-D-16-0088.1

ZAMG Zentralanstalt für Meteorologie und Geodynamik

Post-processing – hybrid methods





[Python: https://github.com/slerch/ppnn/blob/master/nn_postprocessing/nn_src/emos_network_theano.py](https://github.com/slerch/ppnn/blob/master/nn_postprocessing/nn_src/emos_network_theano.py)

R package:
[ensembleMOS: EMOS modeling in ensembleMOS: Ensemble Model Output Statistics (rdrr.io)]()

ZAMG Zentralanstalt für Meteorologie und Geodynamik

Post-processing – hybrid methods

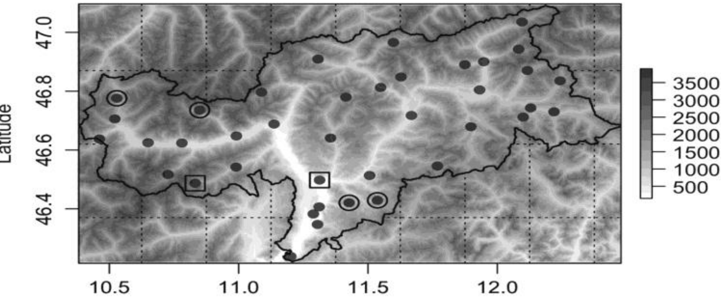
SAMOS – standardized anomalies based model output statistics

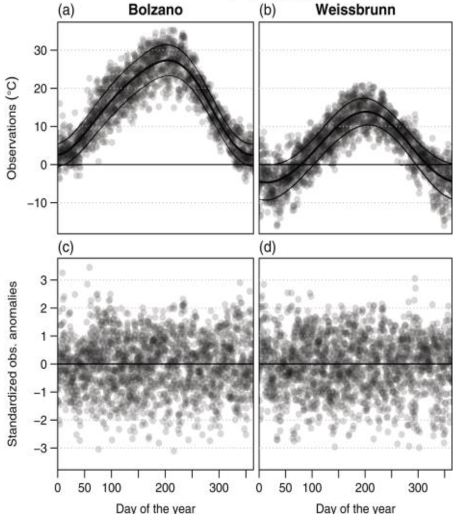
$$y \sim N(\mu, \sigma),$$
$$\mu = \beta_0 + f_1(\mathrm{doy}),$$
$$\log(\sigma) = \gamma_0 + g_1(\mathrm{doy}),$$

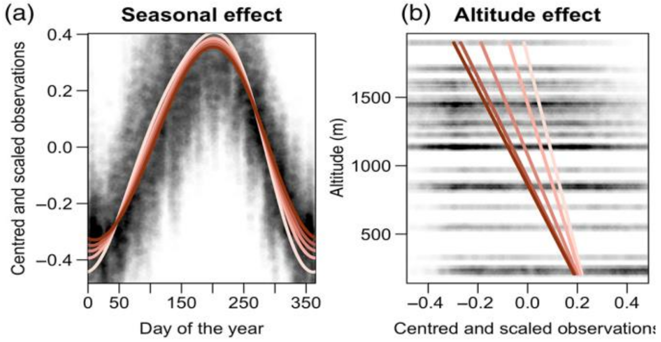
$$y^* = \frac{y - \mu_y}{\sigma_y},$$
$$m^* = \mathrm{mean}\left(\frac{\mathrm{ens} - \mu_{\mathrm{ens}}}{\sigma_{\mathrm{ens}}}\right),$$
$$s^* = \mathrm{std\ dev}\left(\frac{\mathrm{ens} - \mu_{\mathrm{ens}}}{\sigma_{\mathrm{ens}}}\right),$$

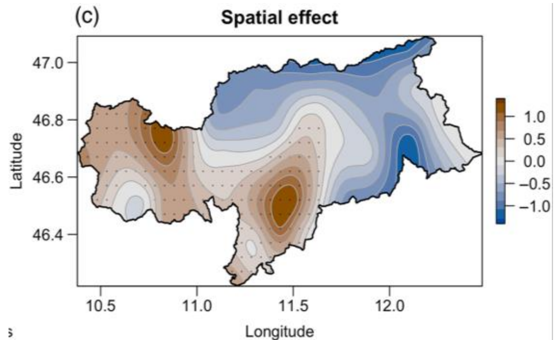
+ Boosting

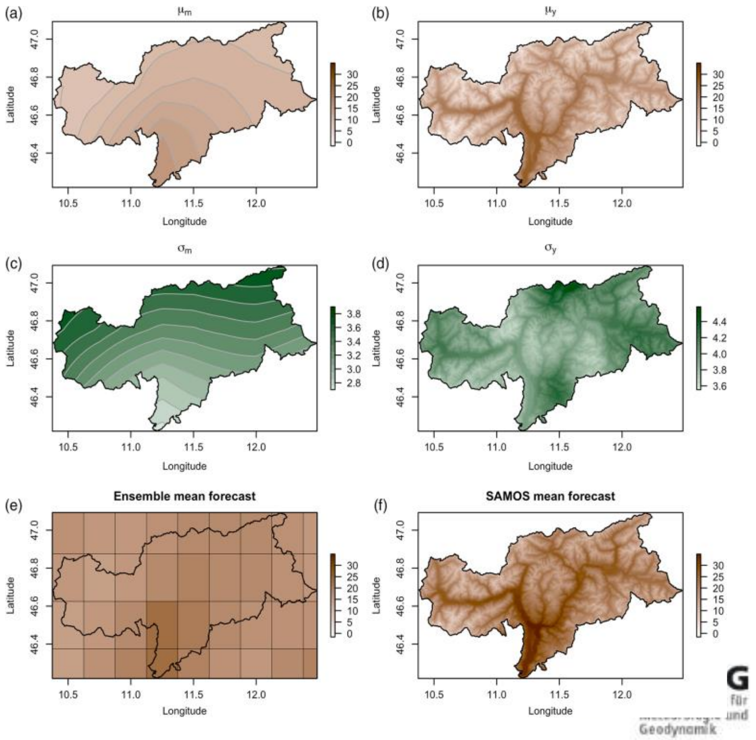
$$y^* \sim N(\mu^*, \sigma^*),$$
$$\mu^* = b_0 + b_1 m^*,$$
$$\log(\sigma^*) = c_0 + c_1 \log(s^*),$$











Post-processing – machine learning methods

ANN / CNN / ConvLSTM

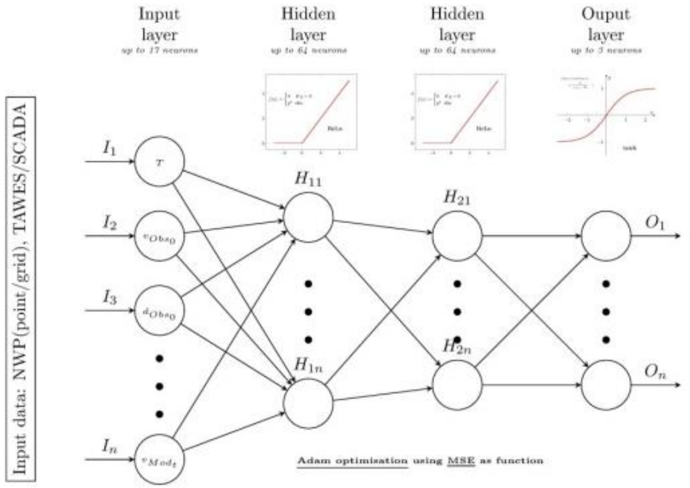
- Different applications:
  - meteorological forecasting grid/point
  - Downstream applications: renewables, agriculture, transportation, mobility, logistics, road maintanance…

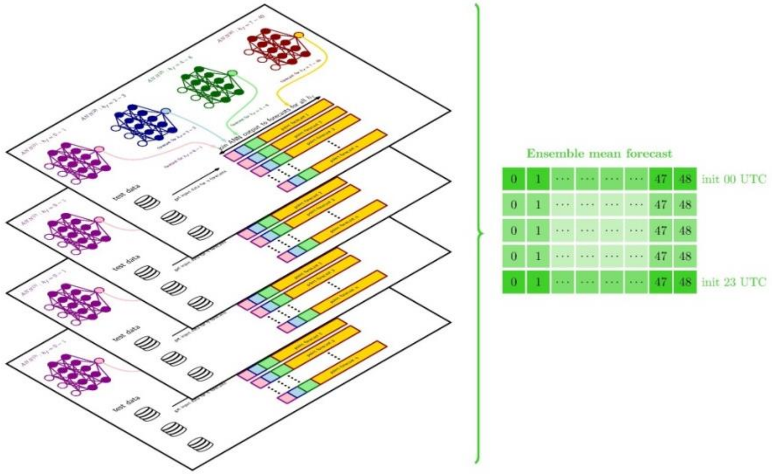
- Different types of data
  - Observations (standard WMO)
  - Satellite
  - Radar, lidar data
  - NWP models with varying quality, domain, grid size,…
  - IoT: private weather stations, GPRSS, microlink data, mobile devices

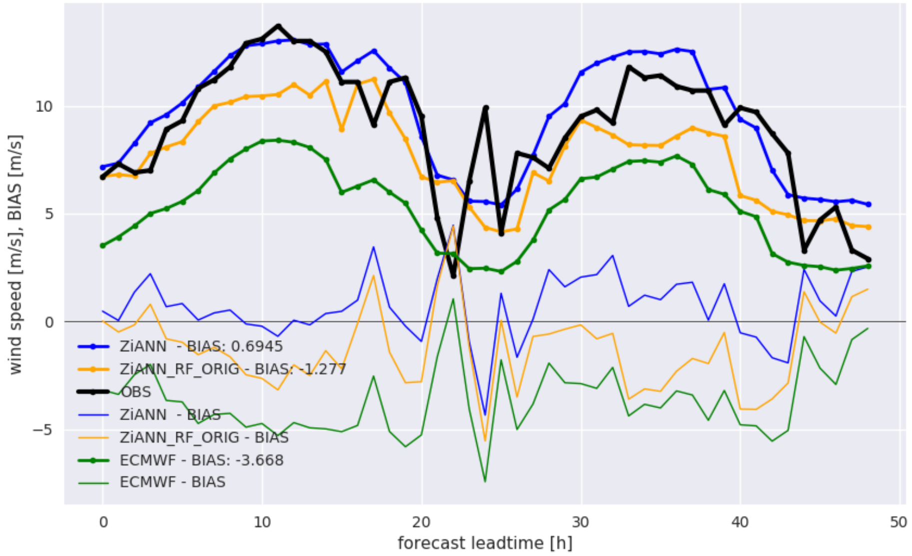
- Different types of AI methods
  - Simpler: MLP, Random Forest, SVM
  - Complex: CNN, ConvLSTM, Berstein Quantiles
  - Rather novel: NODEs, Graph (C)NN, SDEs/differential equations, physics-aware/inspired, GANs,…

ZAMG
Zentralanstalt für Meteorologie und Geodynamik

Post-processing – machine learning methods we use







- Hourly forecasts for the next 48 hours ahead
- Uses a neural network in "ensemble mode" (deterministic forecast) but can also switch to random forest forecast (future: good to have both)
- Subhourly added
- RF + LSTM component added
- Needed adjustments in pre-processing (scaling + transformation

Skills:

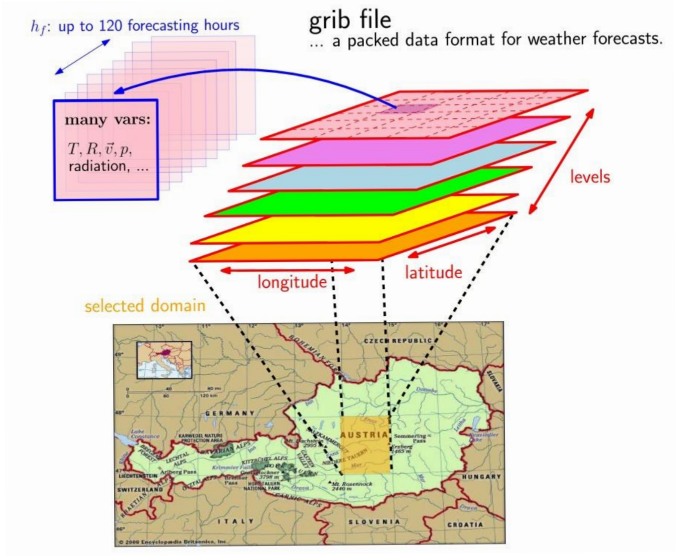
- Direct access to "online" SCADA data
- In-built QC
- Adjustable forecast intervals, neurons, layers, etc.
- Adjustable training length depending on data availability

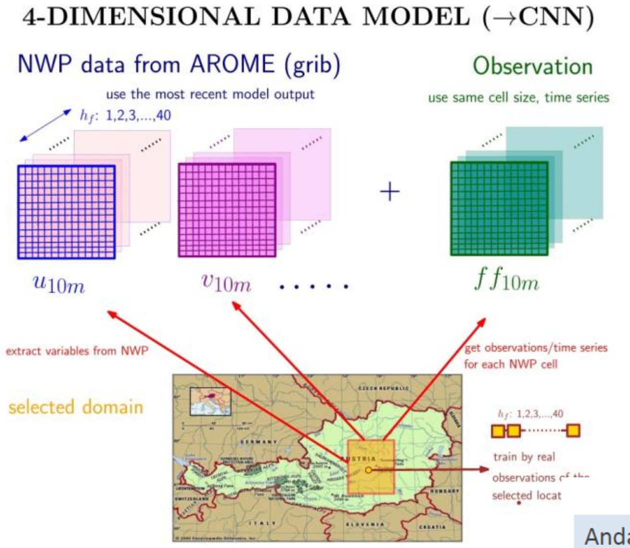
Challenges:

- "our" obs data available every 10-minutes
- NWP data so far with a large delay
- Non-convection permitting models are easy to learn of, don't need long time series of data – convection permitting models not, need lots of data
- Changes in the NWP model – how to deal with them? After 3 – 4 years a model changes nearly completely

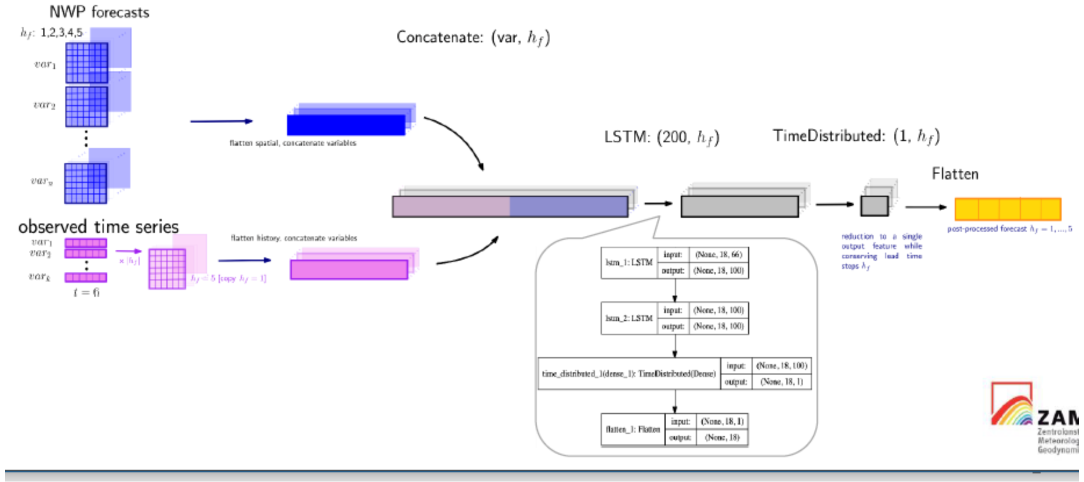
Post-processing – machine learning methods we use

Point forecast using complex neural network setup and multiple data sources (PhD project, AWAkE):



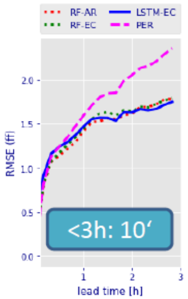


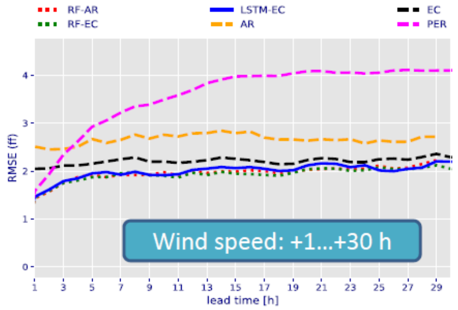
Semi-operational for solar and wind

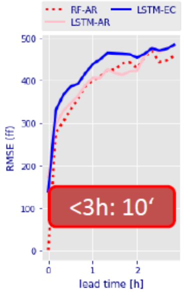


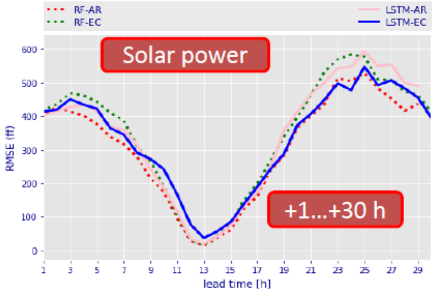
Andau, case study 2020:

- mean of 38 turbines
- SAMOS archieve for training (= AROME / EC at 00:00 UTC)
- runs each full hour (24 times a day) or suntime hours







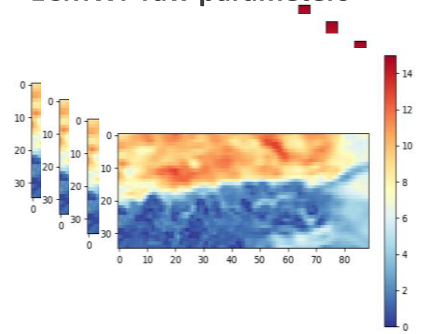


- best setup found used for training from SAMOS archieve (2018-2020)
- AI Models migrated to vmlearn
- queries of AROME, TAWES, Energie Burgeland etc. in real-time

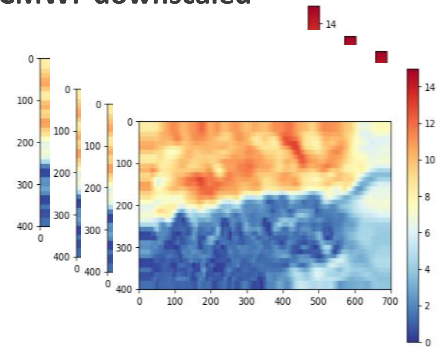
ZAMG Zentralanstalt für Meteorologie und Geodynamik

Post-processing – machine learning methods

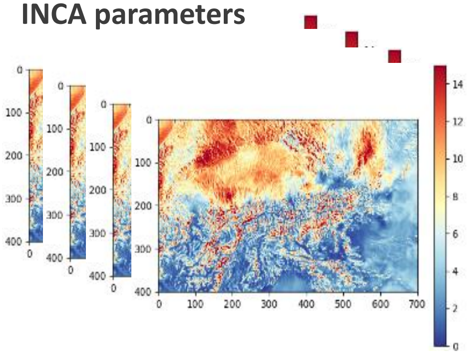
ECMWF raw parameters



ECMWF downscaled

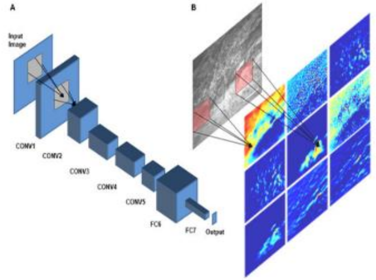


INCA parameters



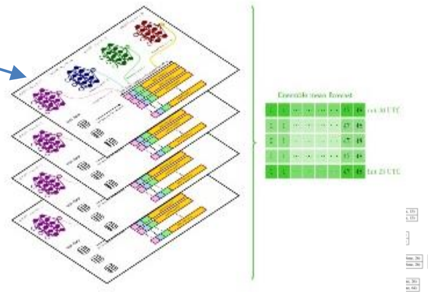
Static fields (z0, topography, TPI,..)

Convolutional neural network

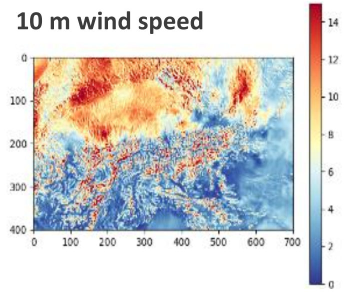


https://miro.medium.com/max/1214/1*AUBtwV3xkXW41Bbh2Uh1hw.png

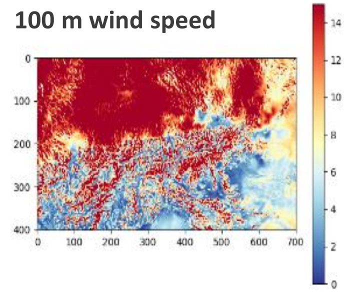
(Ensemble) neural network 2 input „branches“



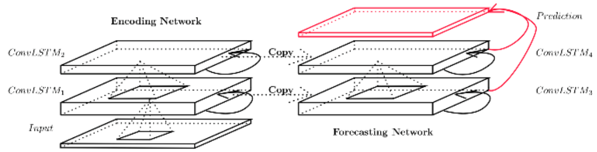
10 m wind speed



100 m wind speed



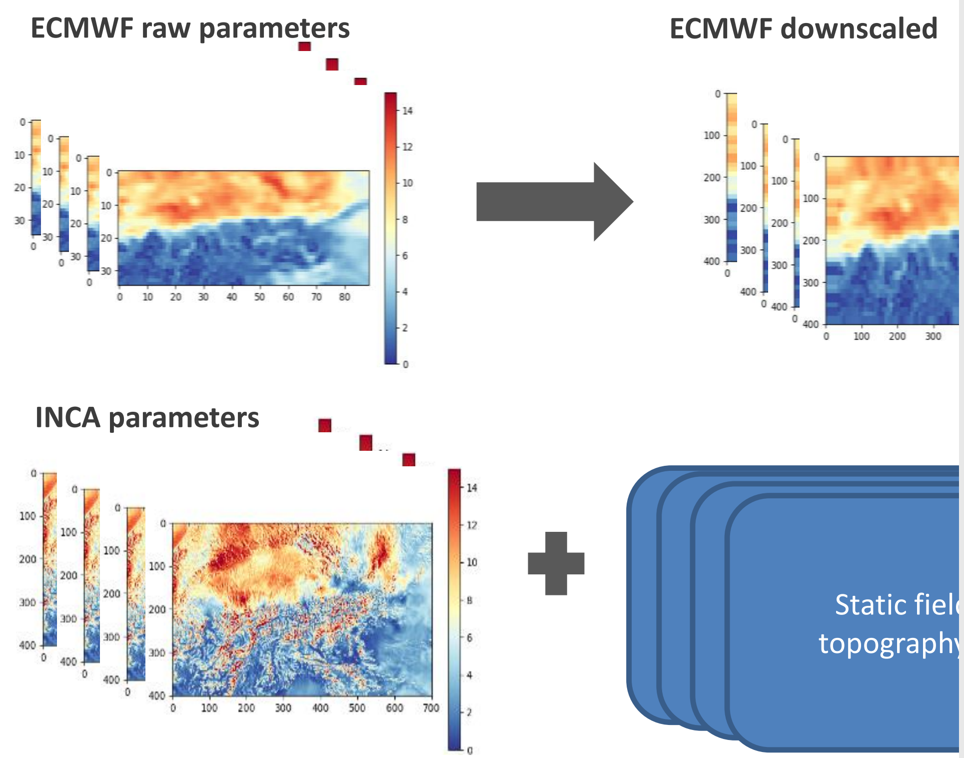
Core being replaced by:



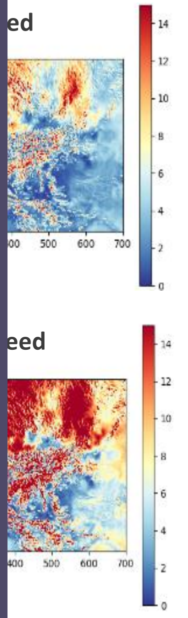
MEDEA
Renewables and meteorological extreme events

ZAMG
powered by klima+ energie fonds

Post-processing – machine learning methods



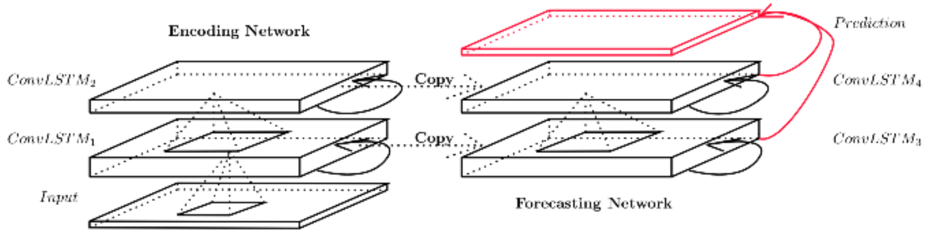

```
######################complex model setup#########################################################
def ML_two_branches_samedims(algo, Adim, Bdim, objective, lr, decay):

    if algo == 'Adam':
       algo = Adam(lr = lr, beta_1 = 0.9, beta_2 = 0.99, epsilon = 1e-08, decay = decay)
    lookback = 1

    print('shapes input: ', Adim, Bdim)
    # define two sets of inputs
    inputA = Input(shape=Adim)
    inputB = Input(shape=Bdim)
    # the first branch operates on the first input
    x = Dense(64, activation="relu", kernel_initializer="he_normal")(inputA)
    x = Dropout(0.2)(x)
    #x = Dense(64, activation="relu", kernel_initializer="he_normal")(x)
    #x = BatchNormalization()(x)
    x = GaussianNoise( 0.01 )(x)
    x = Model(inputs=inputA, outputs=x)

    # the second branch opreates on the second input
    y = Dense(64, activation="relu", kernel_initializer="he_normal")(inputB)

    #y = BatchNormalization()(y)
    y = Dropout(0.2)(y)
    #y = Dense(64, activation="relu", kernel_initializer="he_normal")(y)
    #y = BatchNormalization()(y)
    #y = Dense(64, activation="relu", kernel_initializer="he_normal")(y)
    #y = BatchNormalization()(y)
    y = GaussianNoise( 0.01 )(y)
    y = Model(inputs=inputB, outputs=y)

    # combine the output of the two branches
    combined = concatenate([x.output, y.output])
    # apply a FC layer and then a regression prediction on the
    # combined outputs
    #z = Dense(32, activation="tanh")(combined)
    z = Dense(1, activation="relu")(combined)

    # our model will accept the inputs of the two branches and
    # then output a single value
    model = Model(inputs=[inputA, inputB], outputs=z)
    model.compile(loss=objective, optimizer=algo, metrics=['mae',rmse])
    #plot_model(model, to_file='model_2dims.png', show_shapes=True, show_layer_names=True)
```

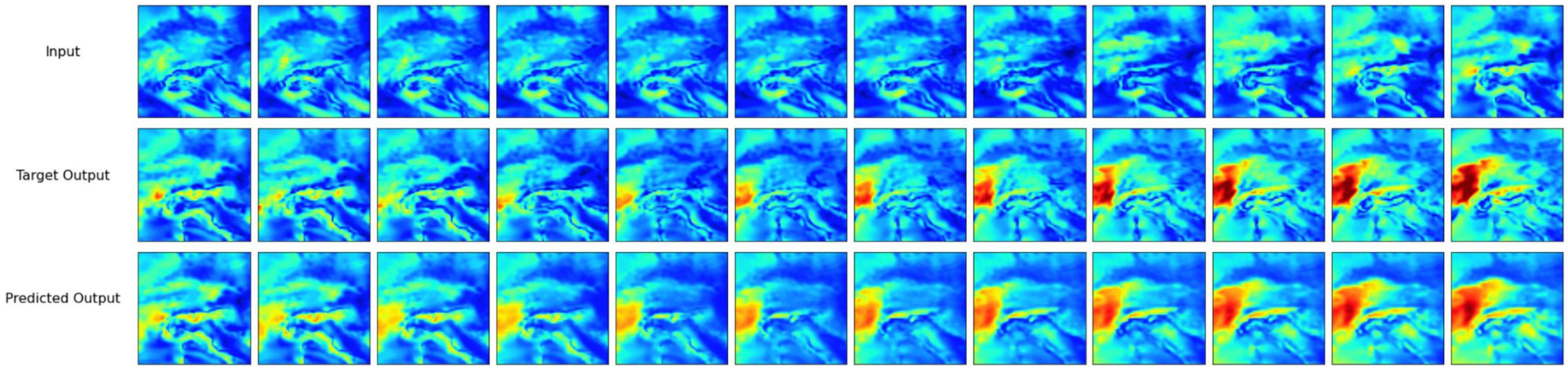


Post-processing – machine learning methods

ConvLSTM based model with adapted weighted loss function for different categories of wind speed

- Some sort of basic physics aware network
- Weighting of less frequent cases of wind speed („extremes“)
- Adapted metric function
- Data-driven using ERA5 as input





MEDEA
Renewables and meteorological extreme events

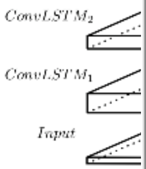
ZAMG Zentralanstalt für
powered by klima+ energie fonds

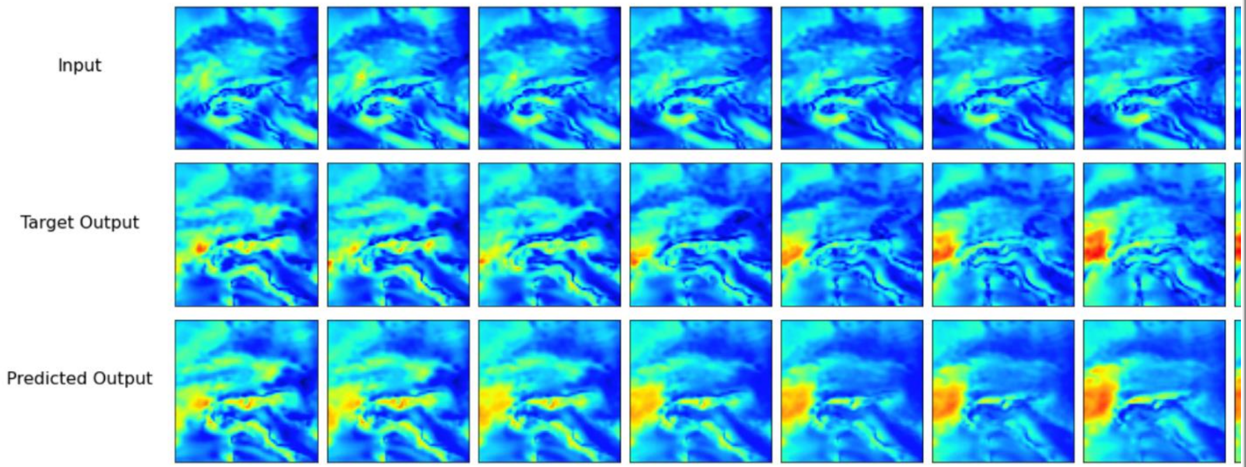
Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and Woo, W.-c. (2015). "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1. NIPS'15. Montreal, Canada: MIT Press, pp. 802–810.

Post-processing – machine learning methods

ConvLSTM based model with adapted weighted loss function

- Some sort of basic physics aware network
- Weighting of less frequent cases of wind speed („extremes“)
- Adapted metric function
- Data-driven using ERA5 as input






```python
class CLSTM_cell(nn.Module):
    """ConvLSTMCell
    """
    def __init__(self, shape, input_channels, filter_size, num_features, seq_len):
        super(CLSTM_cell, self).__init__()

        self.shape = shape  # H, W
        self.input_channels = input_channels
        self.filter_size = filter_size
        self.num_features = num_features
        self.seq_len = seq_len
        # in this way the output has the same size
        self.padding = (filter_size - 1) // 2
        self.conv = nn.Sequential(
            nn.Conv2d(self.input_channels + self.num_features,
                      4 * self.num_features, self.filter_size, 1,
                      self.padding),
            nn.GroupNorm(4 * self.num_features // 32, 4 * self.num_features))


    def forward(self, inputs=None, hidden_state=None, seq_len=None):
        seq_len=self.seq_len
        if hidden_state is None:
            hx = torch.zeros(inputs.size(1), self.num_features, self.shape[0],
                             self.shape[1]).cuda()
            cx = torch.zeros(inputs.size(1), self.num_features, self.shape[0],
                             self.shape[1]).cuda()
        else:
            hx, cx = hidden_state
        output_inner = []
        for index in range(seq_len):
            if inputs is None:
                x = torch.zeros(hx.size(0), self.input_channels, self.shape[0],
                                self.shape[1]).cuda()
            else:
                x = inputs[index, ...]

            combined = torch.cat((x, hx), 1)
            gates = self.conv(combined)  # gates: S, num_features*4, H, W
            # it should return 4 tensors: i,f,g,o
            ingate, forgetgate, cellgate, outgate = torch.split(
                gates, self.num_features, dim=1)
            ingate = torch.sigmoid(ingate)
            forgetgate = torch.sigmoid(forgetgate)
            cellgate = torch.tanh(cellgate)
            outgate = torch.sigmoid(outgate)

            cy = (forgetgate * cx) + (ingate * cellgate)
            hy = outgate * torch.tanh(cy)
            output_inner.append(hy)
            hx = hy
            cx = cy
        return torch.stack(output_inner), (hy, cy)
```

MEDEA
Renewables and meteorological extreme events

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and Woo, W.-c. (2015). “Convolutional LSTM Network: A Machine Learning Approach fc
Volume 1. NIPS’15. Montreal, Canada: MIT Press, pp. 802–810.

Post-processing – machine learning methods setup for a case study

- growing **renewable energy** source, can yield very different output for each location of interest
- effective integration to **power grid**: need **forecasts** of the **expected power curve** (e.g.: serves for grid stability, energy trading, scheduling of maintenance / energy transfer, ...)
- various **data sources available**: power generated, met. site observation, satellite, numeric prediction (NWP)
- strong **seasonal** and **diurnal variation** in the data → want these variations in the nowcasts

https://commons.wikimedia.org/wiki/File:Solar_PV_Austrian_Alps.jpg

→ investigate machine learning/ML such as Artificial Neural Nets, Random Forest as efficient forecast tool

ZAMG Zentralanstalt für Meteorologie und Geodynamik

Data for CASE STUDY 2021

We optimize **site specific models** and select data for each site from:

INPUT:

- **AROME:**
  forecasts in various p/z levels of solar radiation related parameters (e.g.: short-wave radiation, cloud cover, …)
- **CAMS – site interpolated radiation timeseries**:
  radiation related parameters
- **Observation site**:
  observed solarpower
- **TAWES/INCA** – closest observation/analysis at surface level:
  global radiation, temperature, wind, humidity

+ computed **climatology**

CASESTUDY

Training:
- ✓ **2015-2020** (incl. artificial)
- ✓ **2020** (real only)

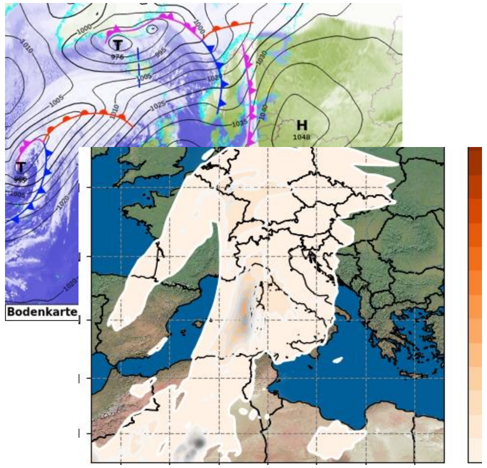
Testing:
- ✓ **2021**

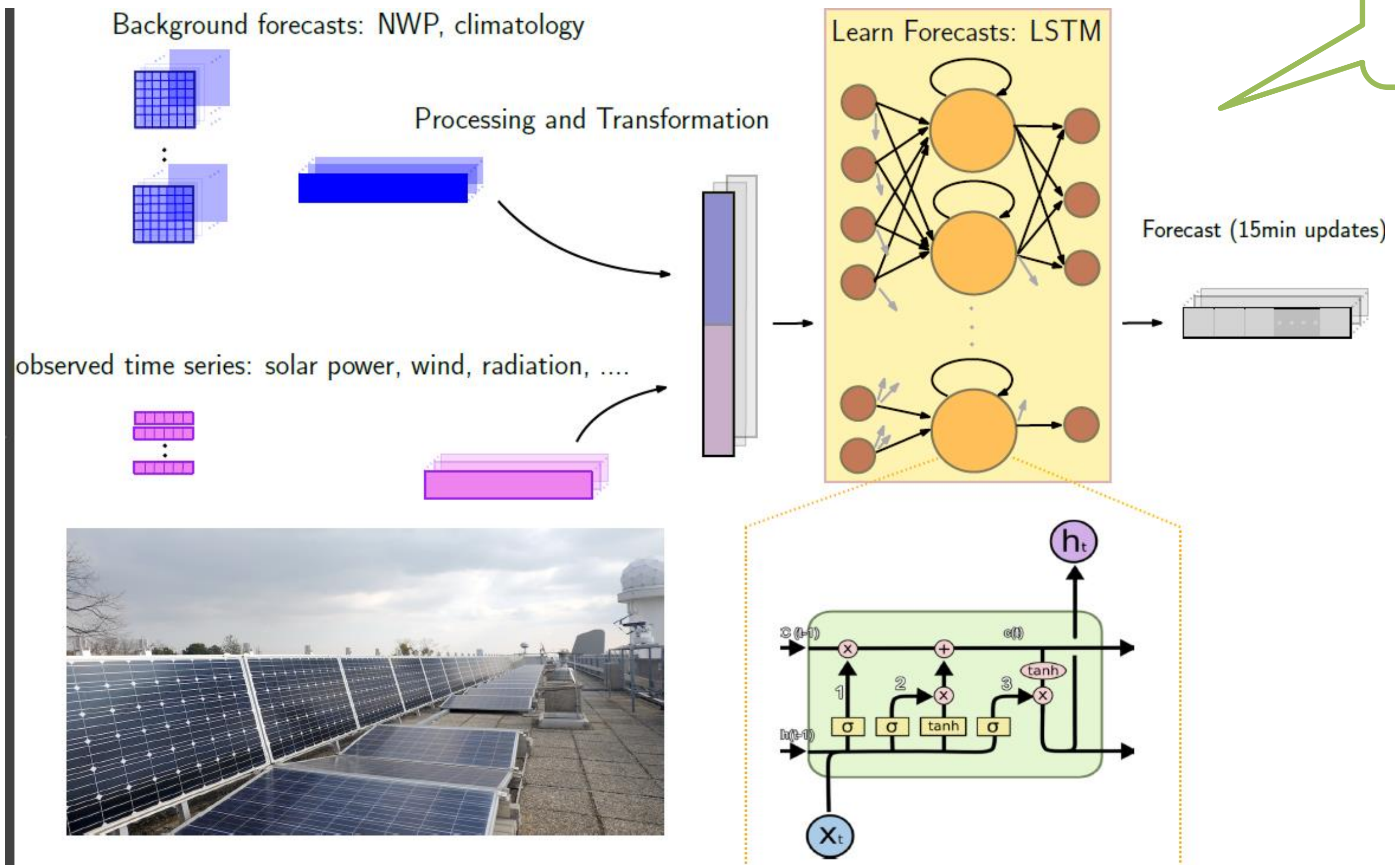
Check missing, normalize, etc.

OUTPUT: solar power forecasts
in 15 min. resolution
+6 hours, hourly runs

3 **Austrian sites** selected

Post-processing Methodology: update a Background Model(s) by ML





Alternative:
Random Forest

Issue: Short Observation Time-series of Power Plants

→ generation of **artificial training data**, as more is needed for complex models (LSTM etc.)

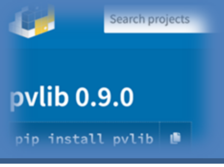
close observations

TAWES

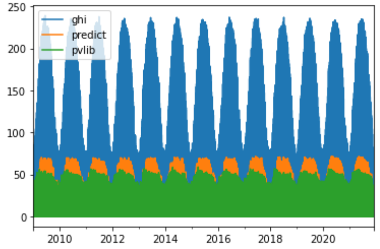
CAMS

satellite-derived
point interpolation

pvlib



https://pypi.org/project/pvlib/

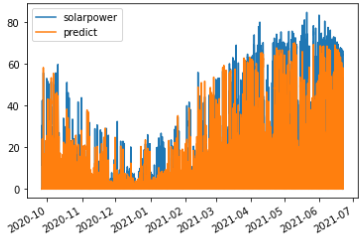


[pvlib-expected]
pv:
2015-2020

robust
very fast

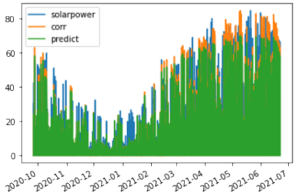
RF

pv OBS: *2020*



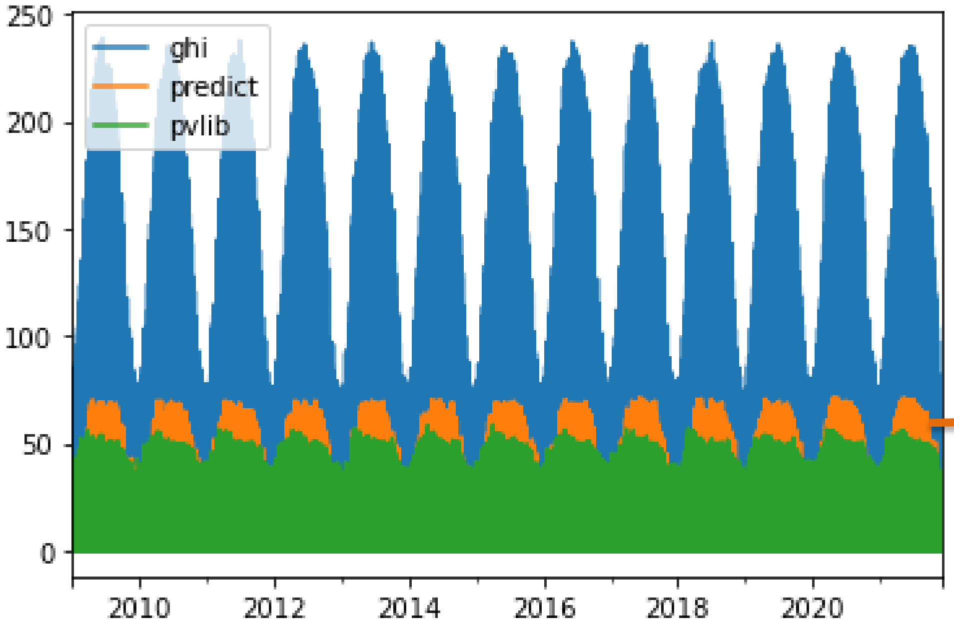
[RF-expected]
pv:
2015-2019

... further transforming ...

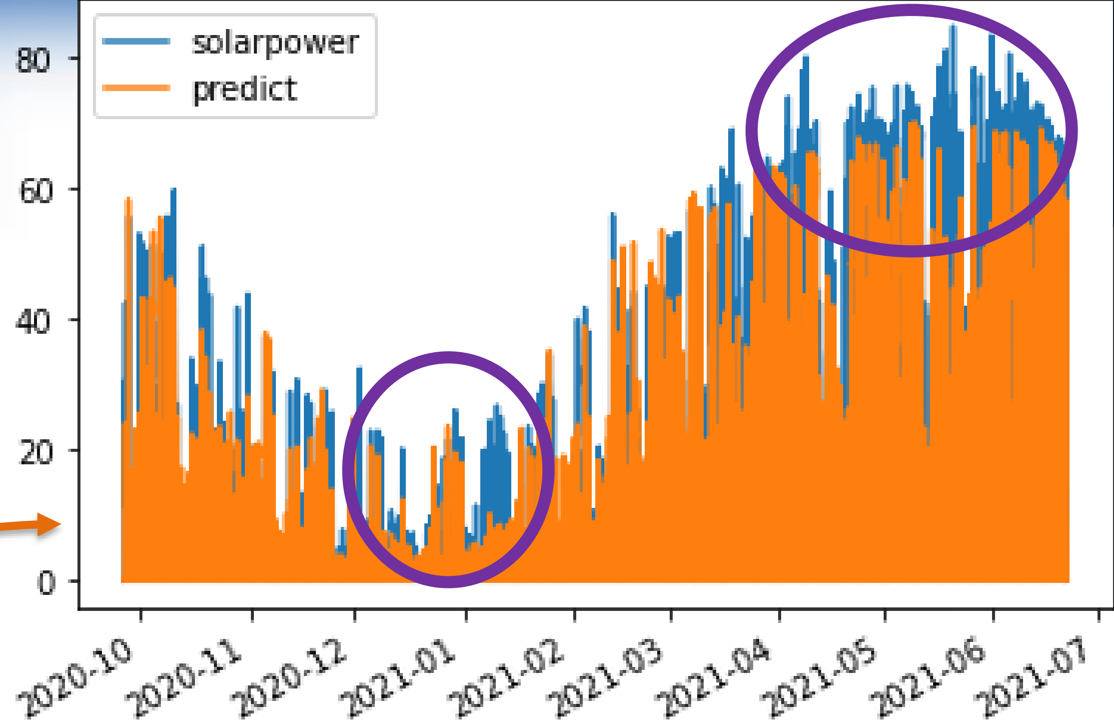


artificial pv:
2015-2020

Data: Obtaining Artificial Time-series

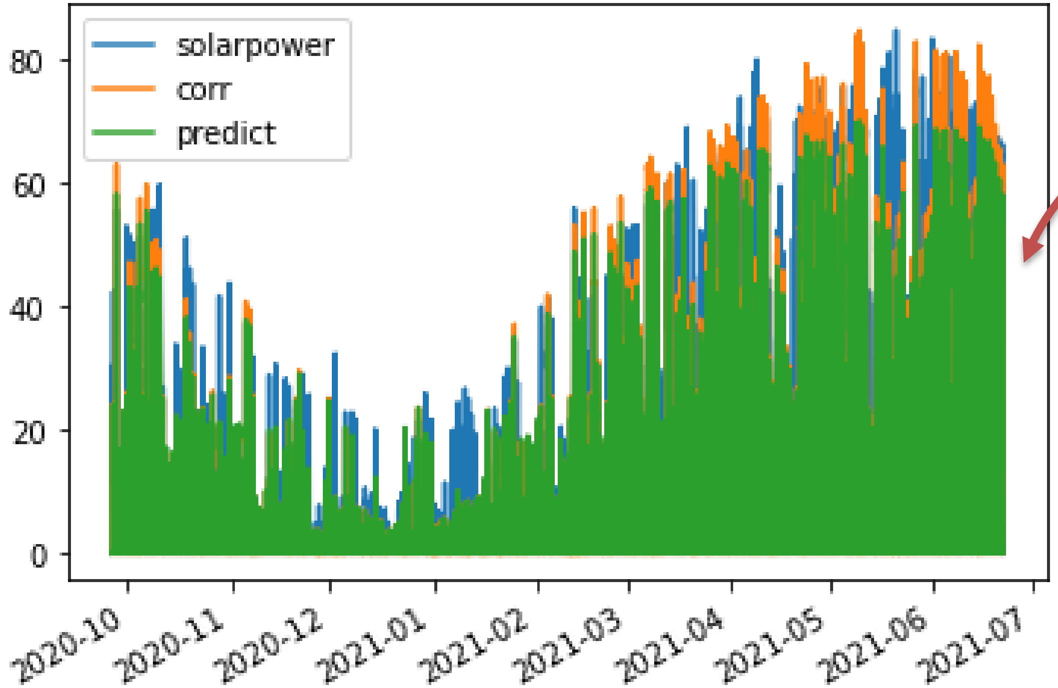


RF+pvlib good but often seems to underestimate real PV



Long time series obtained: ghi, pvlib

... deviate in scale for our specific site

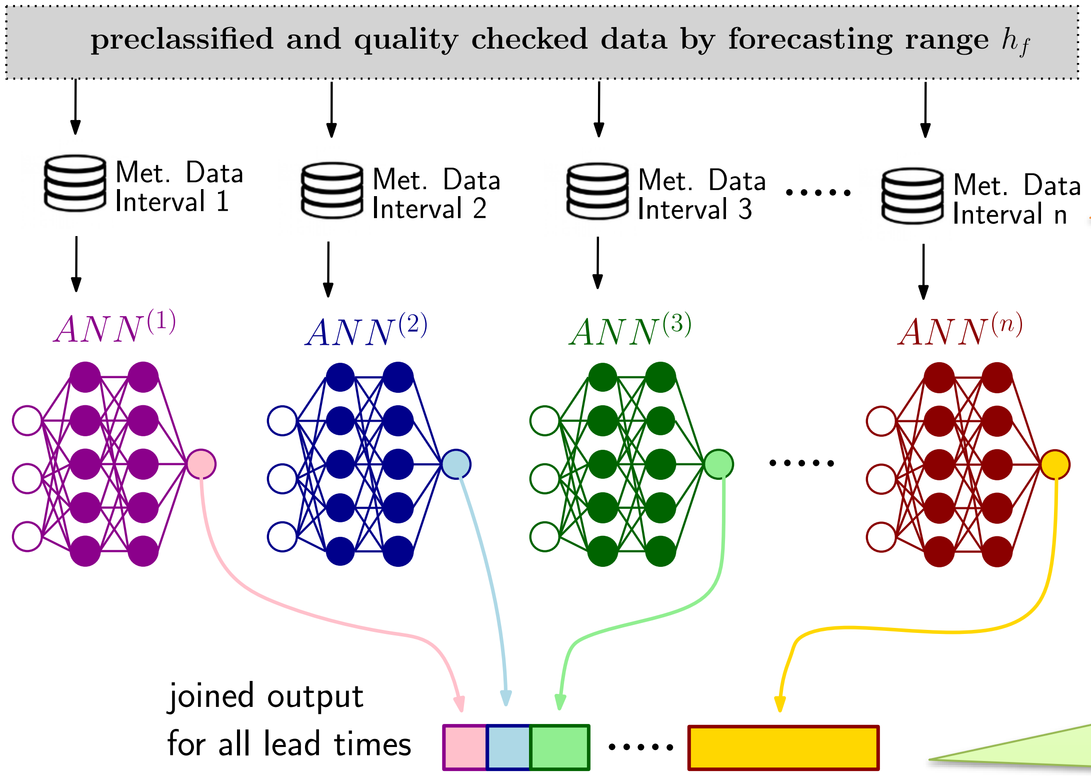


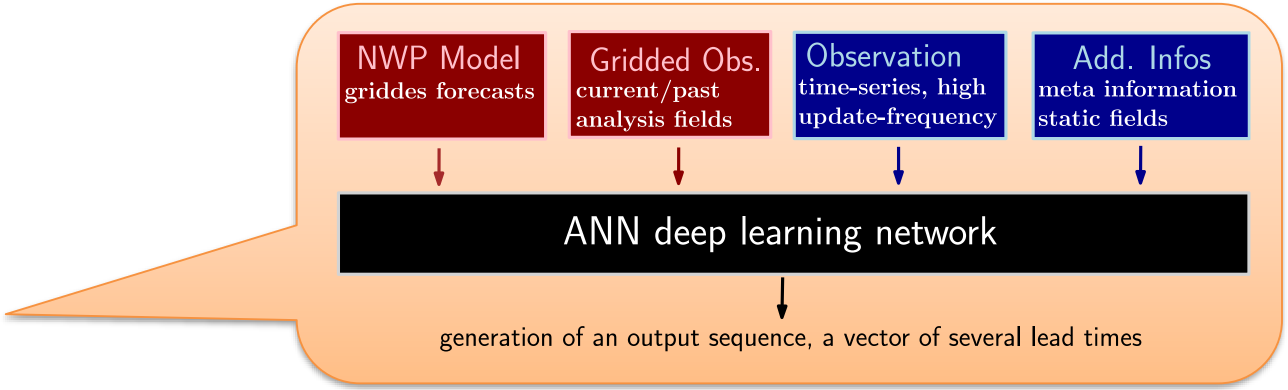
1st opti. approach: transform by percentiles of PV-OBS

predict :=RF output based on CAMS-radiation + TAWES observation + pvlib

Selection and Transformation of Inputs

1. **input feature X** selection: simple methods such as RF weights, Target Y: solar power
2. replace / remove **missing** values, check quality
3. 0-1 **normalization**, using (here hourly) climatological standards
4. for longer vectors / sequences: **intervalization** by lead time steps





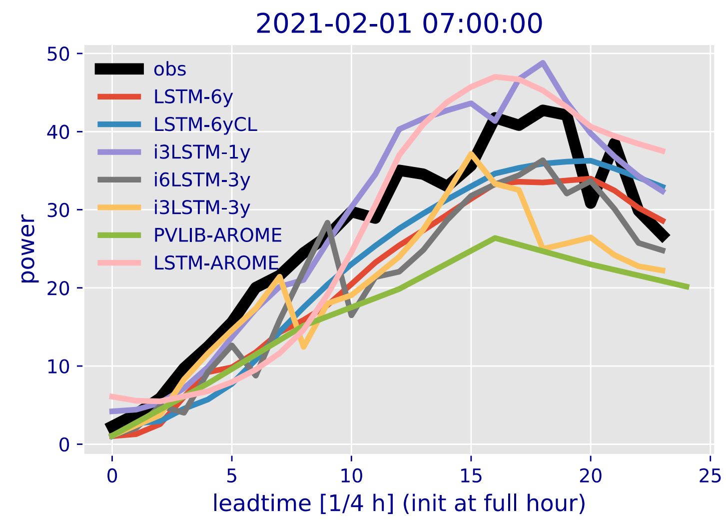
Basic climatological transformation from normalized X

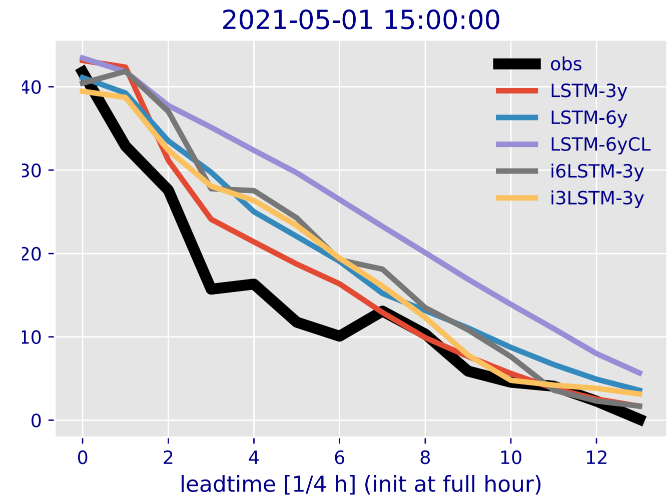
$$\Delta\, x_i(t) \;:=\; \frac{1 + norm\left(xi_{\{OBS\}}(t)\right) - norm\left(xi_{\{\,CLIM\}}(hour(t))\right)}{2}$$

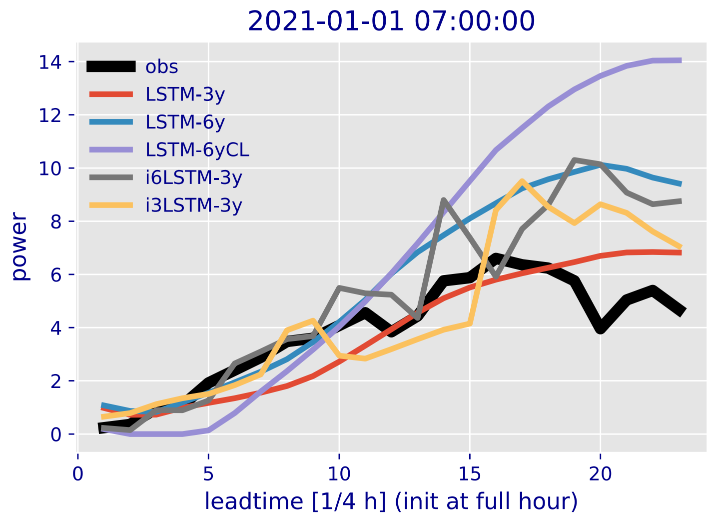
$$pv = \mathrm{denorm}(2\Delta\, pv - 1 + norm(pv_{clim}))$$

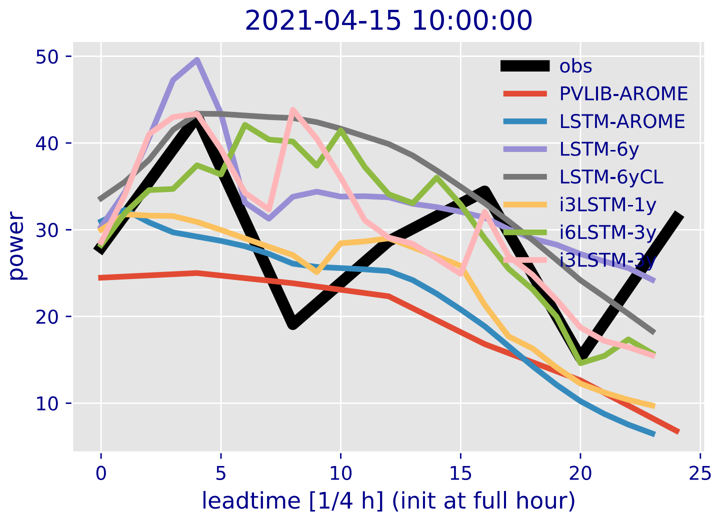
$$i = pv, rad, ff, u, v, T, CC, TOA, \ldots \quad t = 1,2,\ldots,24$$

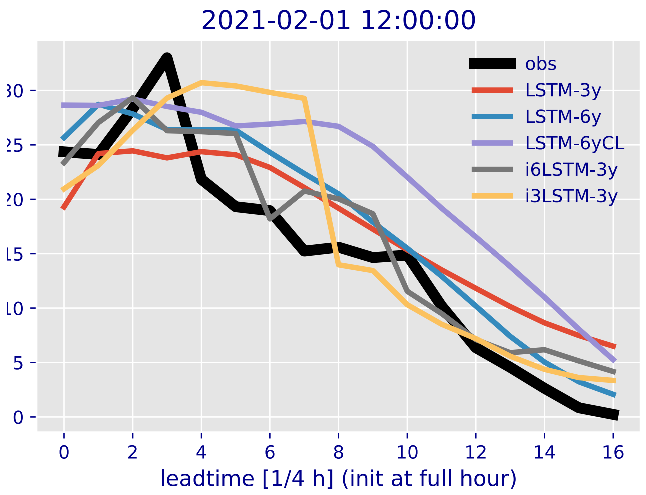
Case Study Results – Sample Forecasts

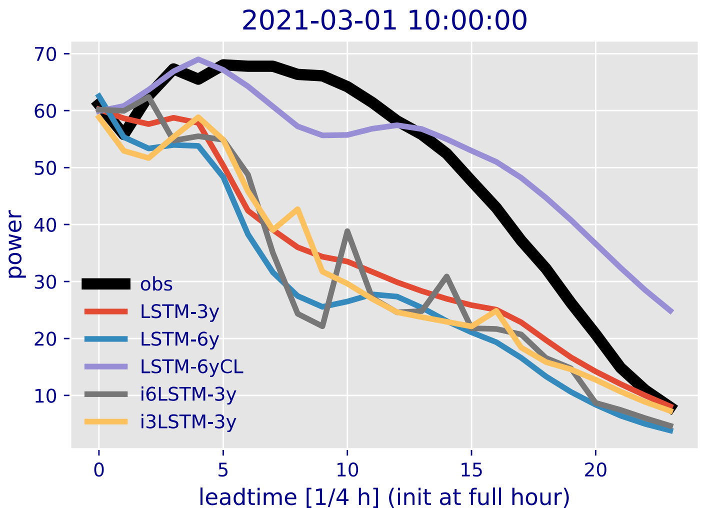






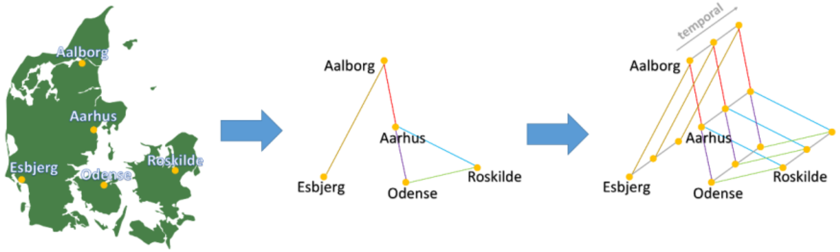




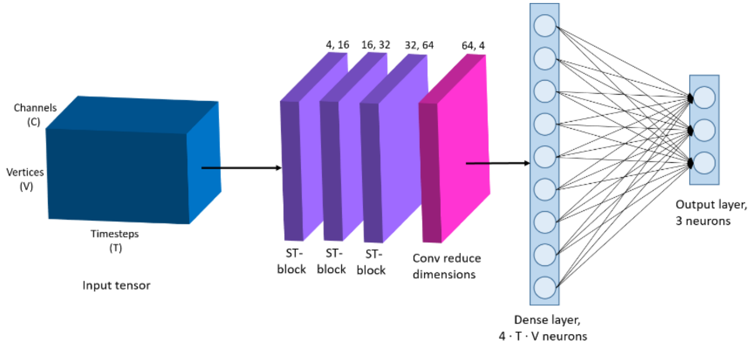


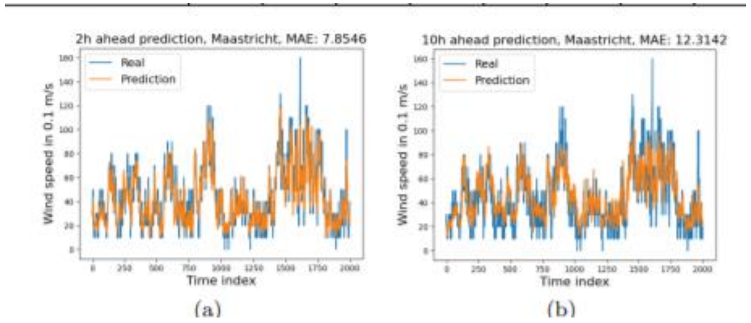
Post-processing – machine learning methods

Graph networks – for wind/solar energy prediction

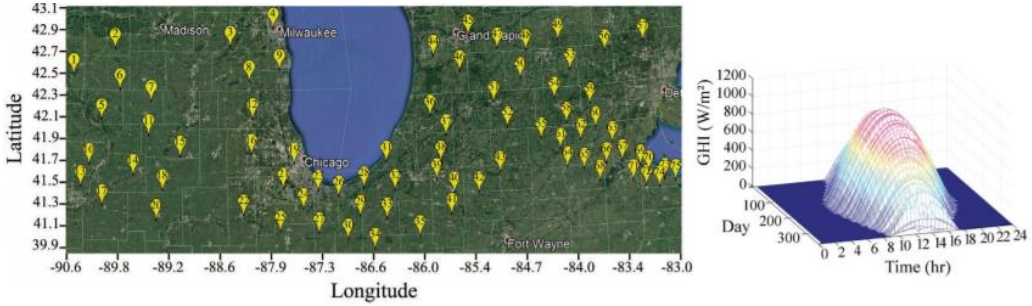


https://www.esann.org/sites/default/files/proceedings/2021/ES2021-25.pdf





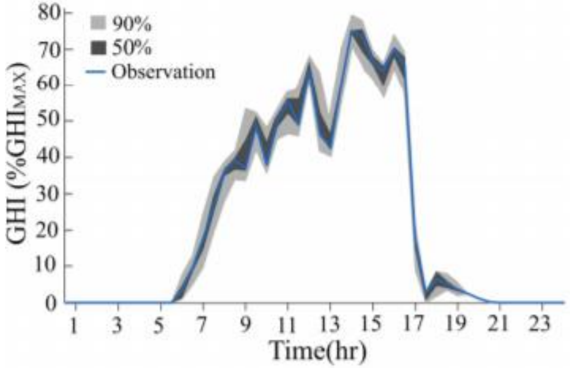
Something similar being implemented right now



(a) Latitude-Longitude map of 75 solar sites in the NSRDB (b) Solar Irradiance of 2015 at solar site 14

https://ieeexplore.ieee.org/ielaam/5165391/9043622/8663347-aam.pdf

measurements in the National Solar Radiation Database



ZAMG Zentralanstalt für Meteorologie und Geodynamik

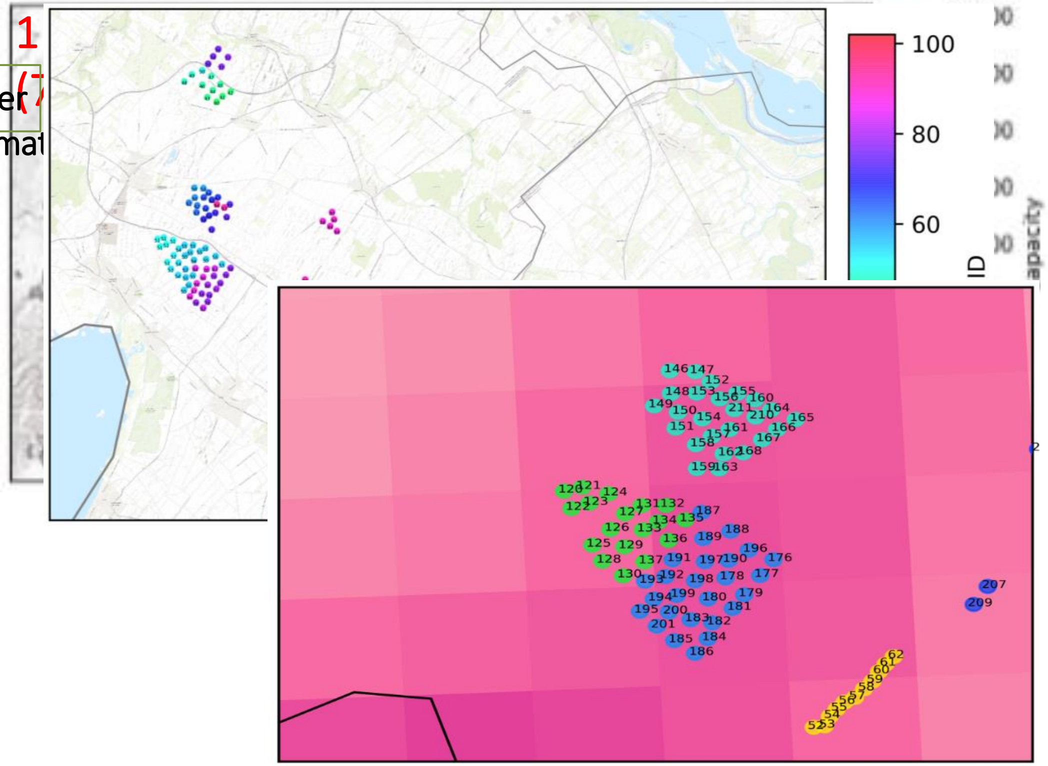
Post-processing – federated learning

Application fields for federated learning

- wind / solar energy: given the data policies of providers, TSOs, traders, etc. → distributed / network federated learning would definetively help improving forecasts

- Meteorology: forecasts for obs sites/sites using not only e.g. Austrian data but combine European observation network or even PWS sites (after quality control)

- Mobility: combine different sources, even car measurements
- Agriculture
- …



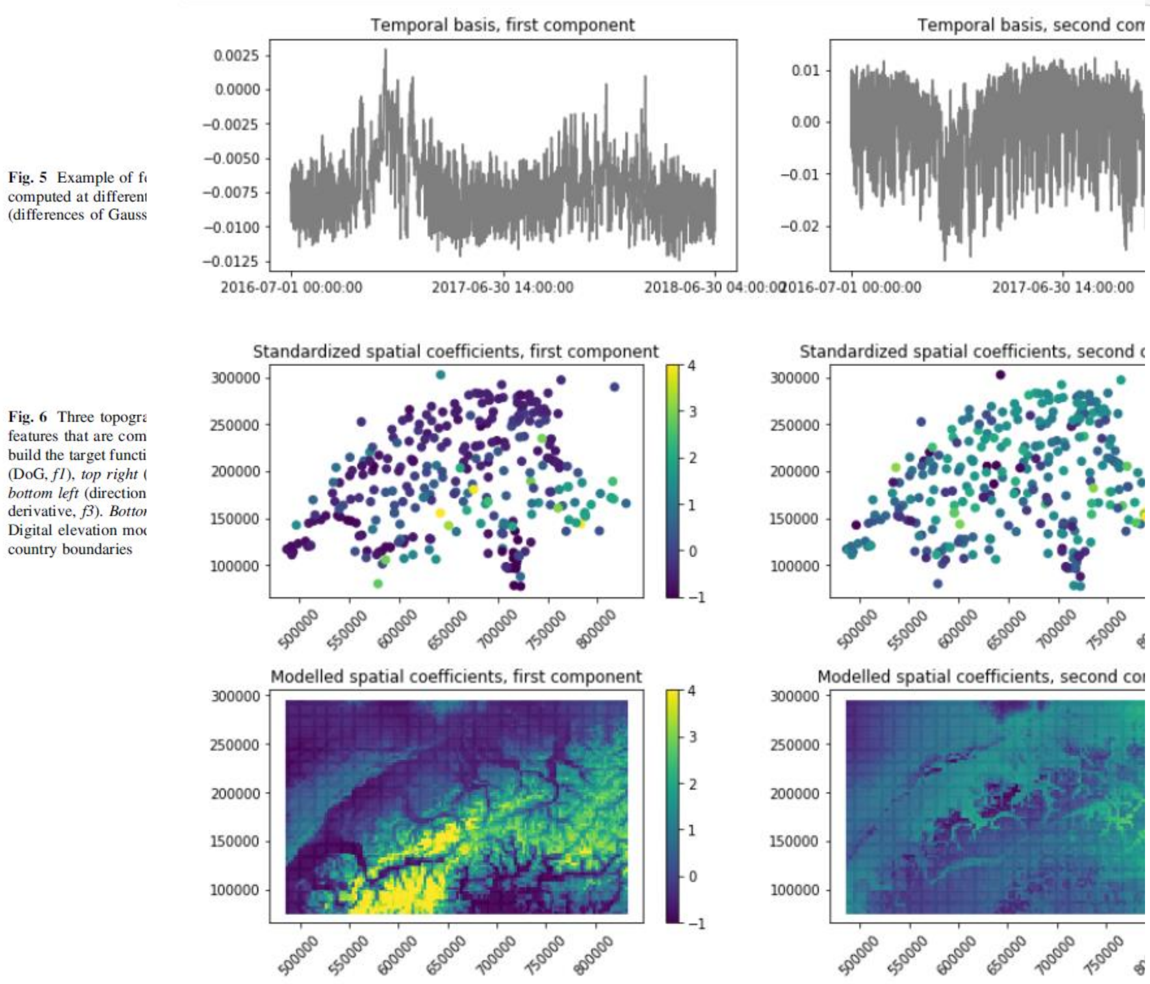
ZAMG Zentralanstalt für Meteorologie und Geodynamik

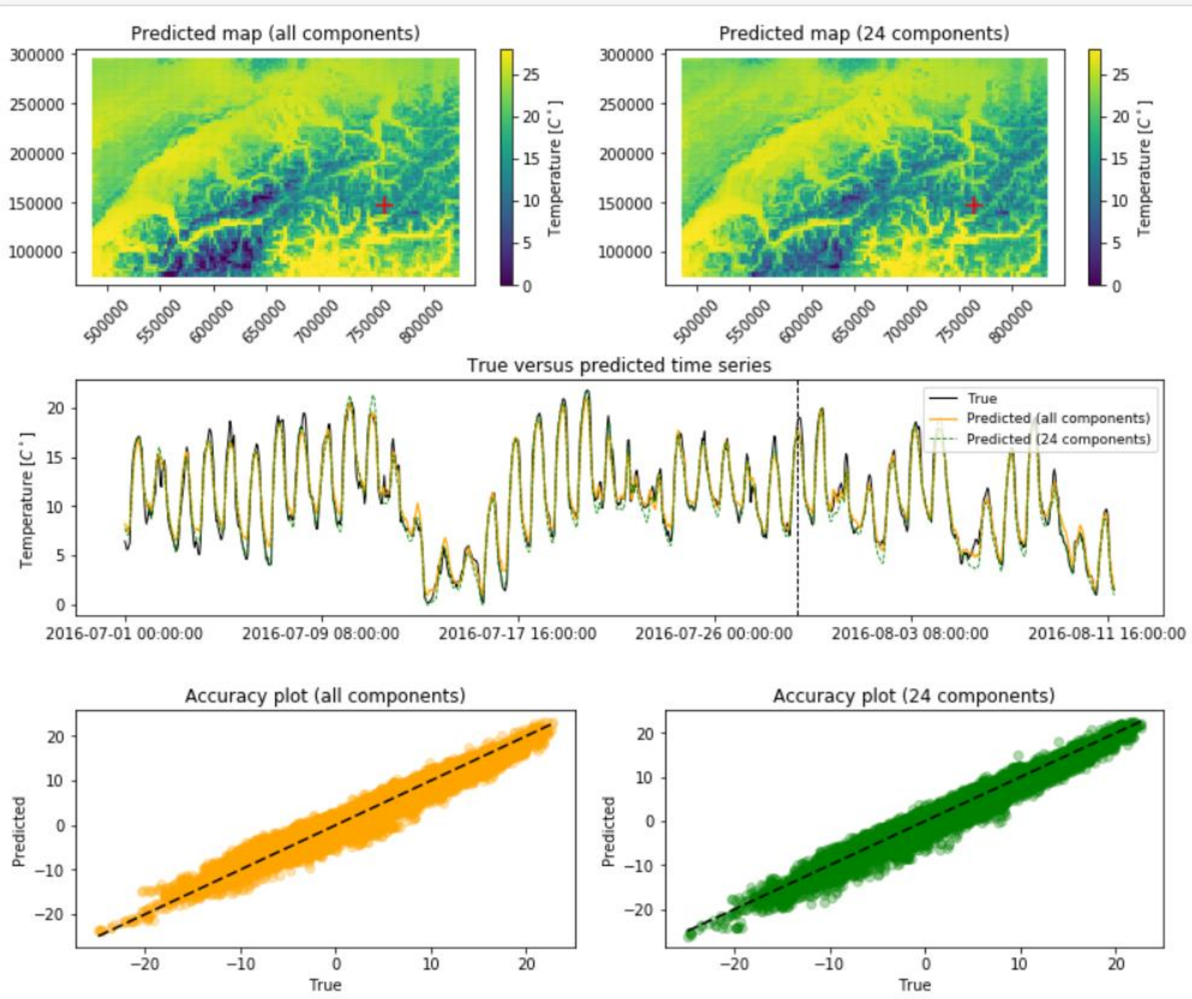
Post-processing – machine learning methods replacing gridded observations

Idea: use machine learning methods and/or statistics to "interpolate" in-situ observations of wind speed to a specified grid

Results: 100 m and 1 km analysis fields of wind speed using a different methodology. Add on: depending on used background fields (DEM etc.) resolution could be changed to higher/lower.

→ Can we use Graphs here? Would the work better? Can federated learning improve here





ZAMG
Zentralanstalt für Meteorologie und Geodynamik

Questions?

Recommendations?

Comments?