KØBENHAVNS UNIVERSITET

Interpolating satellite observations of sea surface temperatures

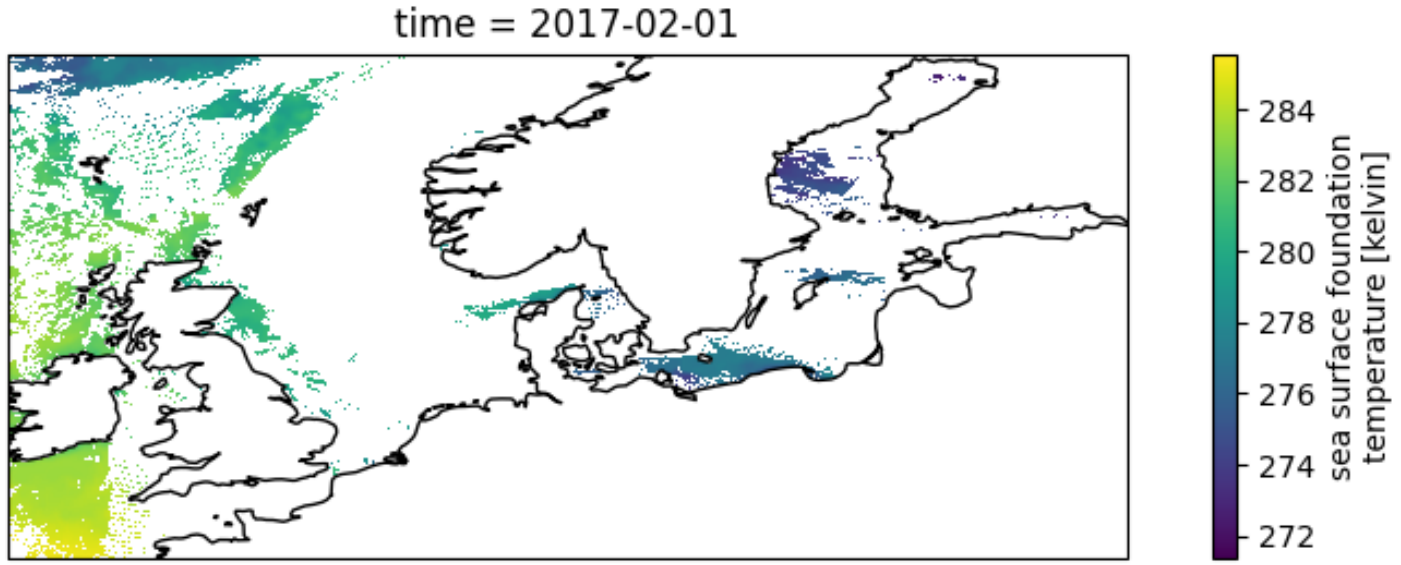
ANDRÉ, ASGER, CHRISTIAN & PUK

Outline

- Data
- Time: The Gated Recurrent Unit approach
- Space: The Convolutional Neural Network approach
- Conclusion

Introduction

- Satellite measurements of sea surface temperatures
- Missing data due to clouds
- Predict missing values



1: https://data.marine.copernicus.eu/product/SST_BAL_SST_L4_REP_OBSERVATIONS_010_016/description
2: https://data.marine.copernicus.eu/product/SST_BAL_PHY_L3S_MY_010_040/description

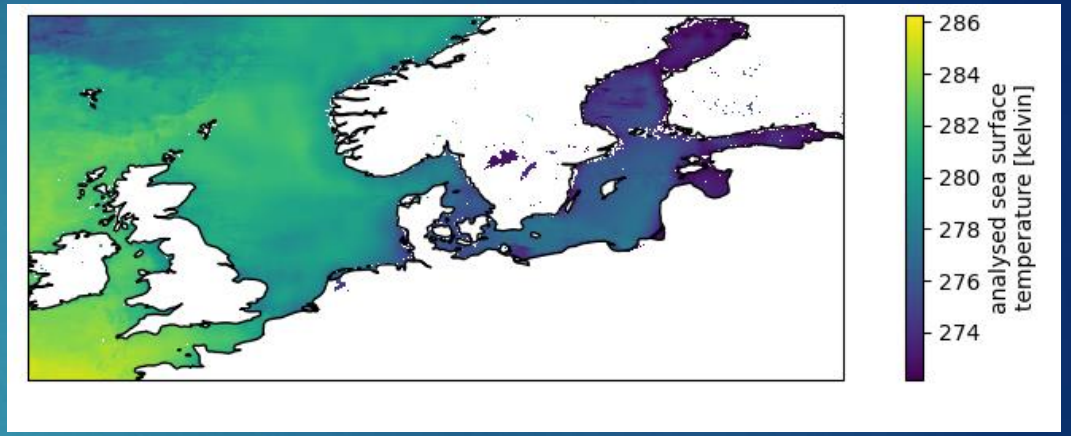
The Data

L3 and L4

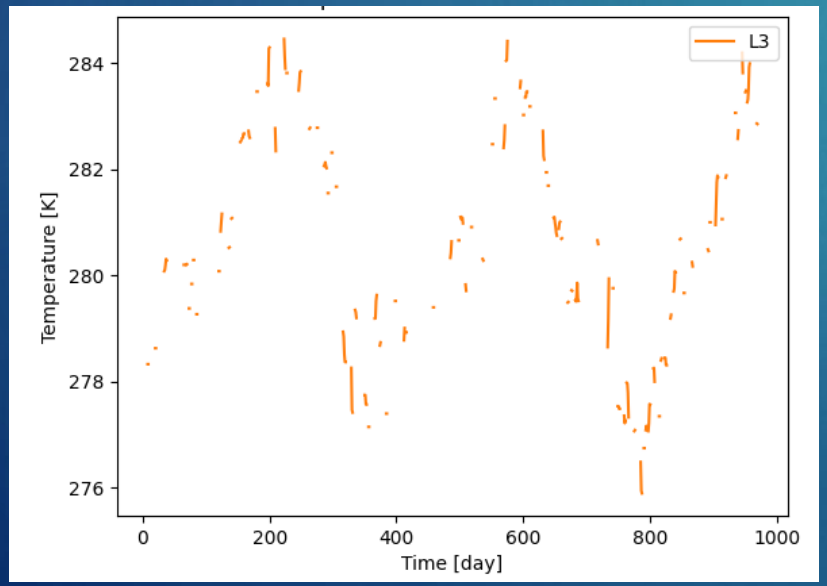
Challenges:

- Just two years of data take approx. 2 GB
- About 65 % data is missing
- No values over land

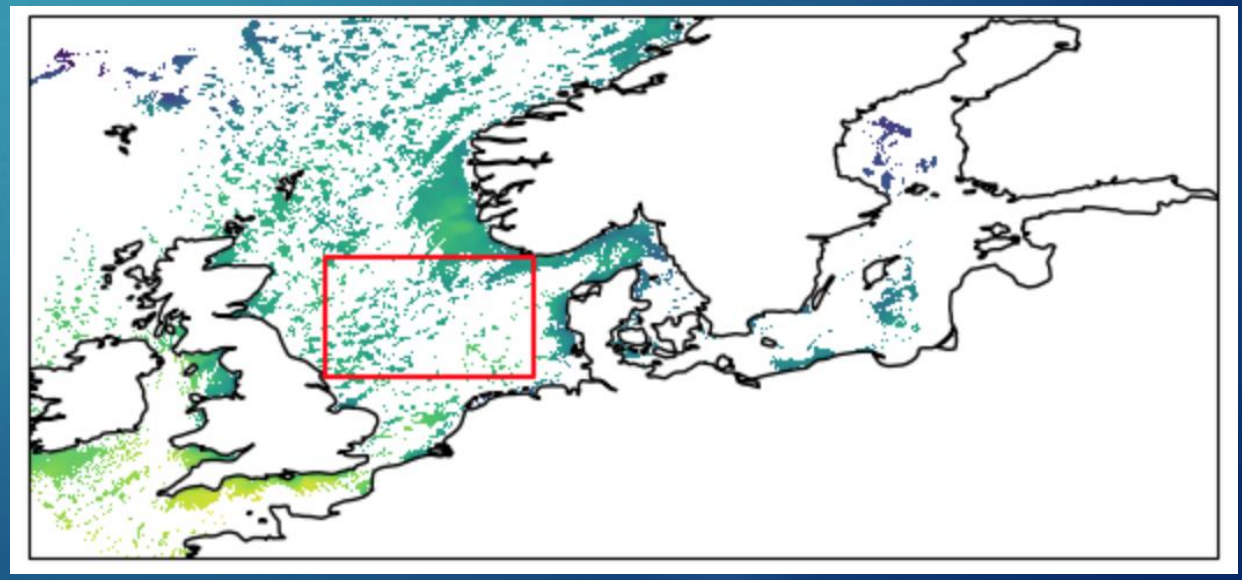
Level 4 (L4) daily image



Level 3 one pixel time series



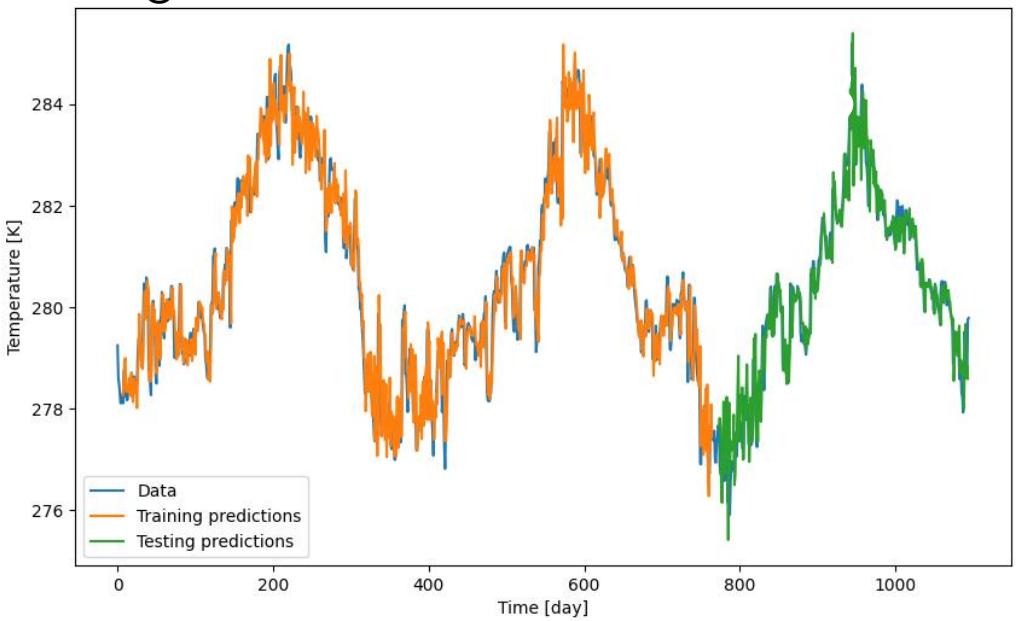
Level 3 (L3) daily image



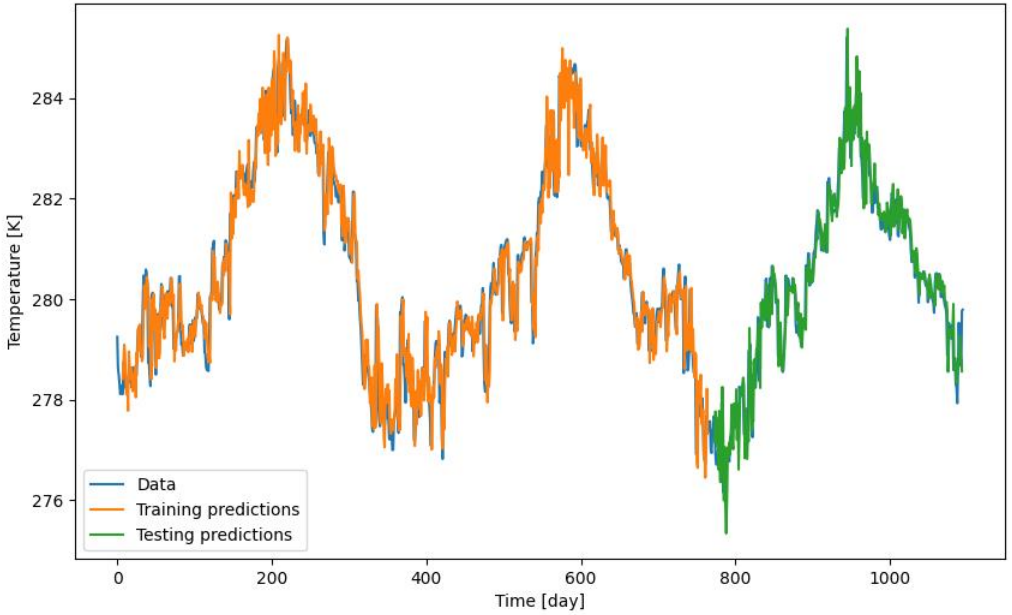
Gated Recurrent Unit

- GRU on L4 data
- Regular/unidirectional and bidirectional – not a big difference (RMSE of 0.54, and 0.47, respectively)

Regular GRU

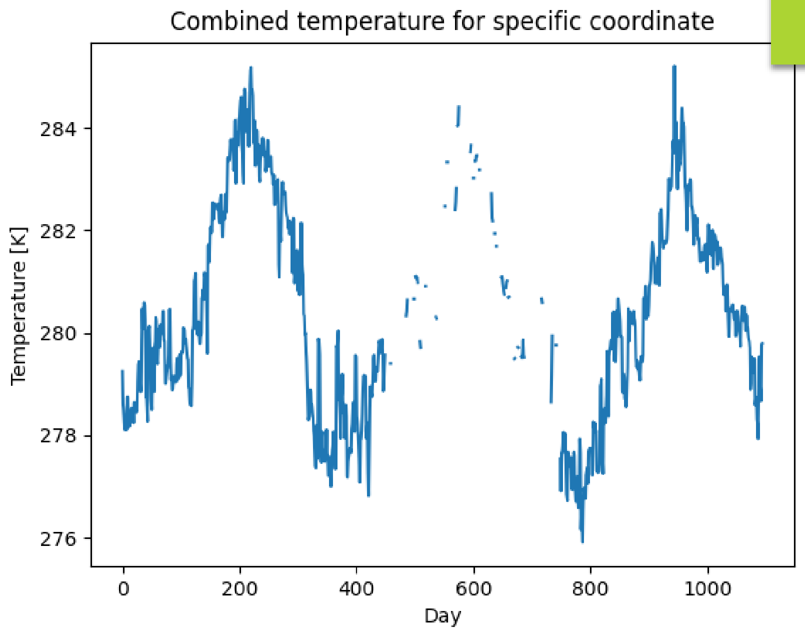


Bidirectional GRU



Gated Recurrent Unit

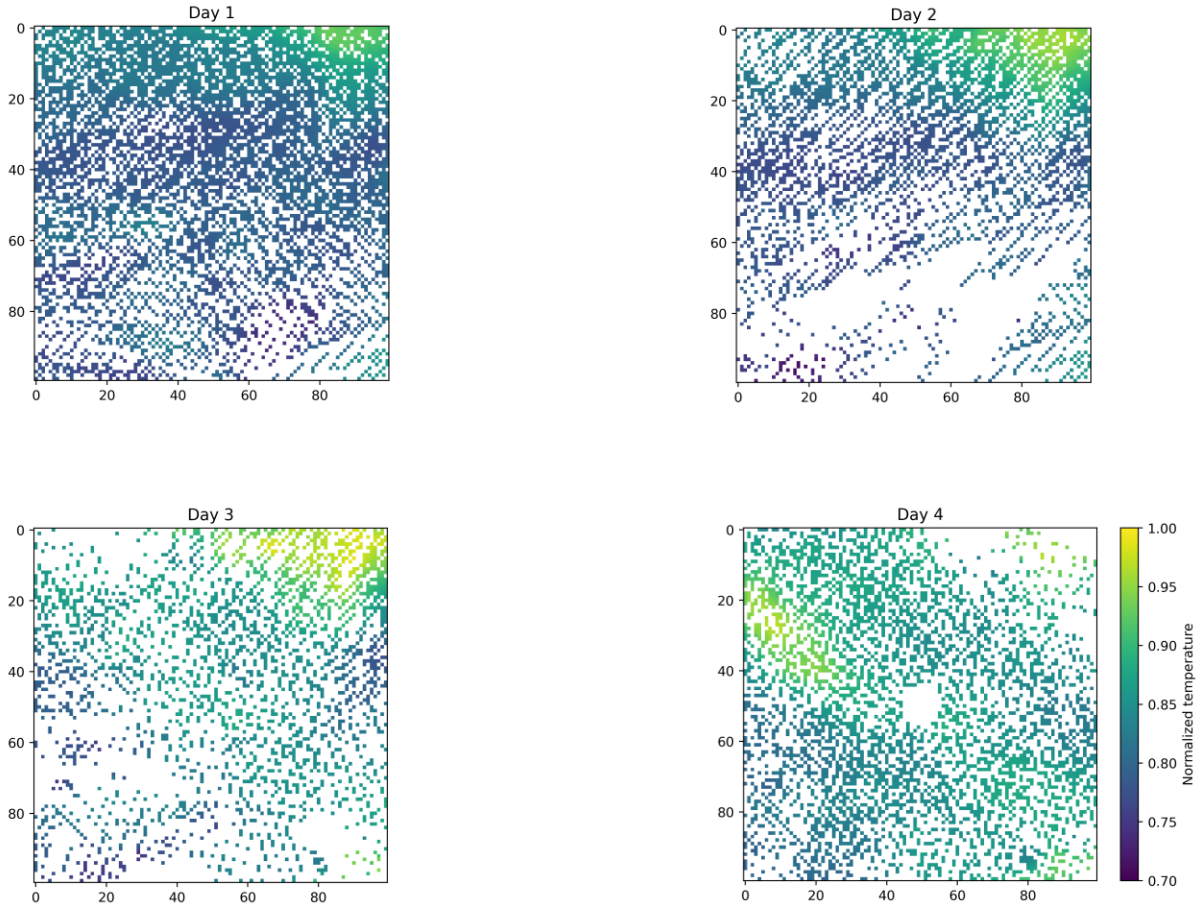
- Simple GRU on combined L3 and L4 data




```
[nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan]
```

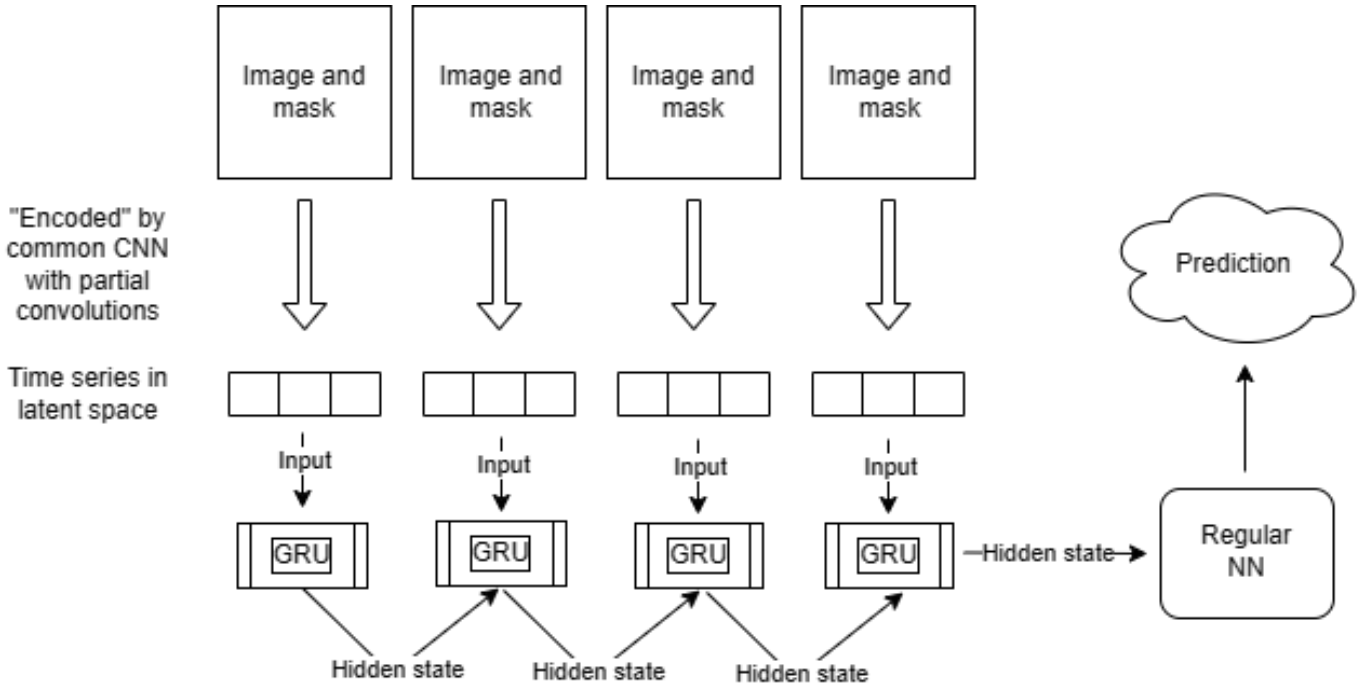
Combined model

- One point does not contain sufficient information
- New method: Look at a time series of images. Insert artificial "cloud" on last picture
- Target is L3 augmented with L4



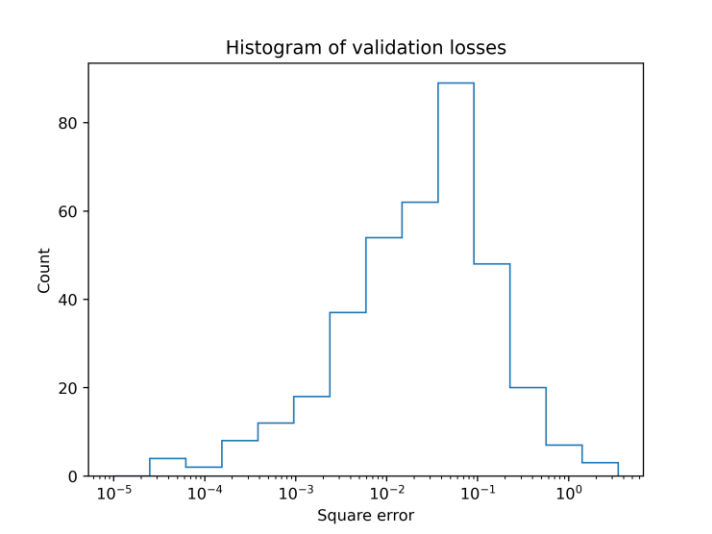
Combined model - structure

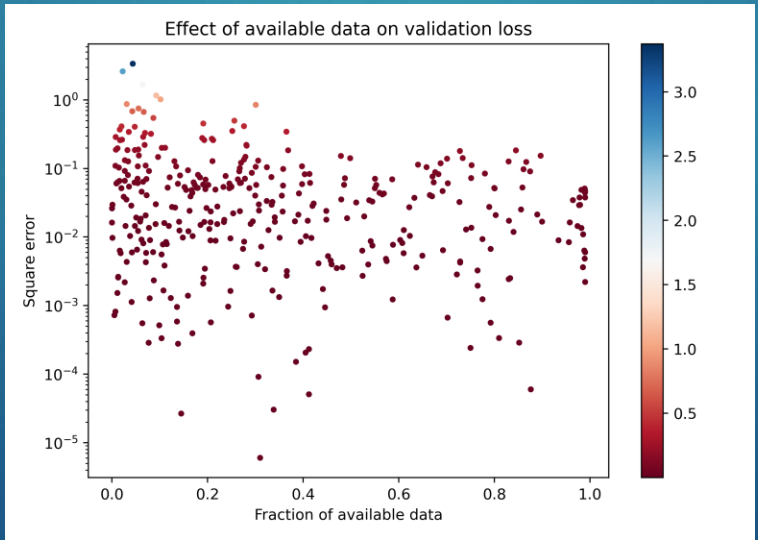
- Combination of CNN using "partial convolutions" (PCNN)
- Benchmarks:
  - Mean of today
  - Last measurement in point
  - Model without GRU and with only today's picture



Combined model - results

Model	Validation RMSE [K]
Benchmark: Mean of last day	0.53 K
Benchmark: Last valid measurement at point	0.65 K
Benchmark: Partial Convolutional Neural Network (PCNN)	0.30 K
Benchmark: GRU on L4 data	0.47 K
Combined PCNN and GRU	0.33 K



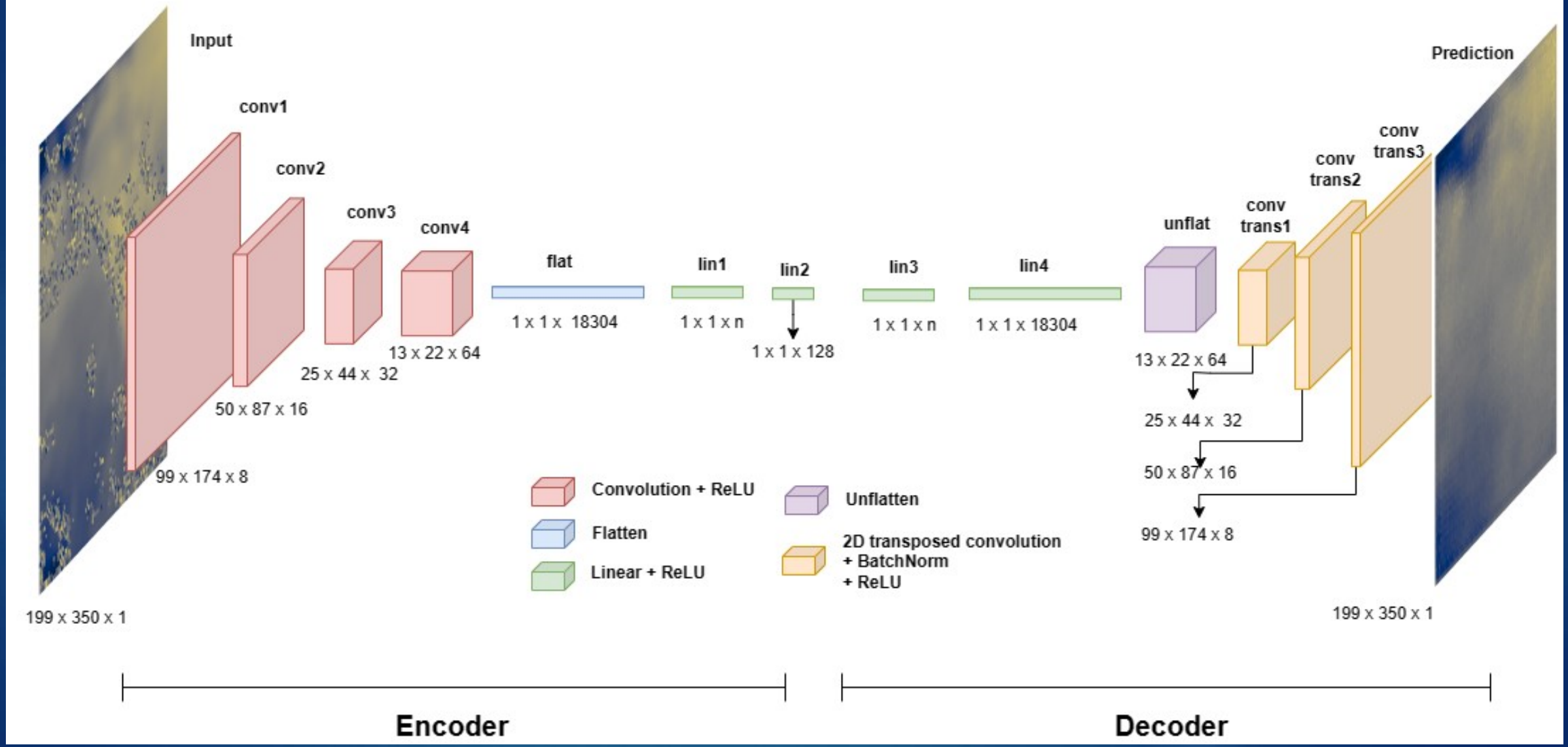


- Best model: Benchmark PCNN
- Convergence problems with the combined model
- Both PCNN's were able to overtrain significantly without dropout layers
- Losses span 5 orders of magnitude
- Huge impact by low-data days
- Possible information limit?

Convolutional Neural Network

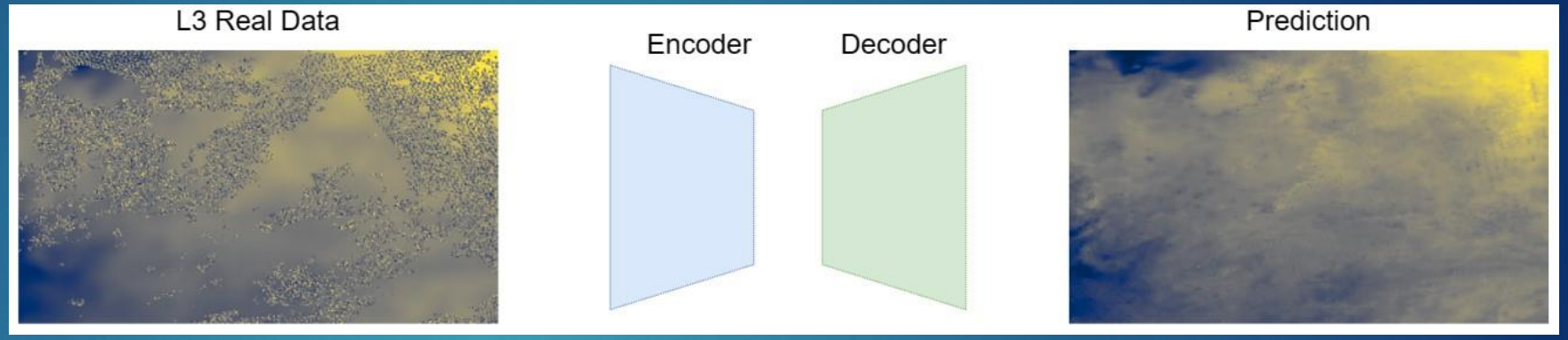
Predicting the full picture

CNN architecture

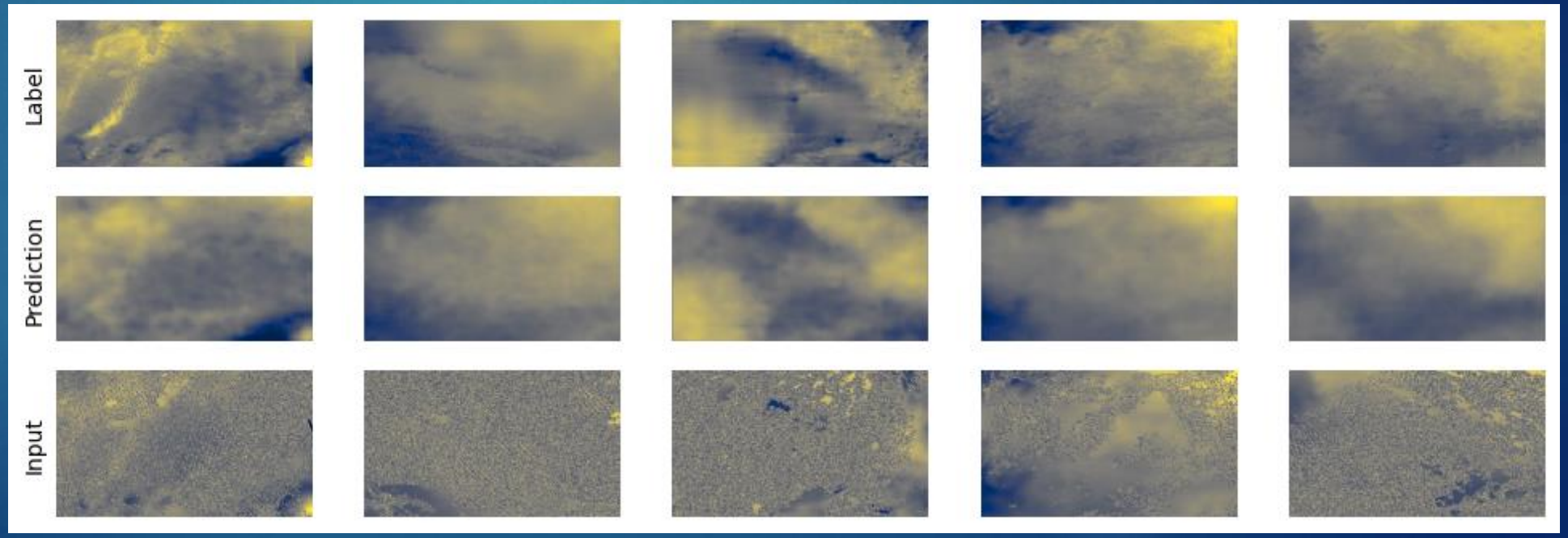


Inpainting autoencoder

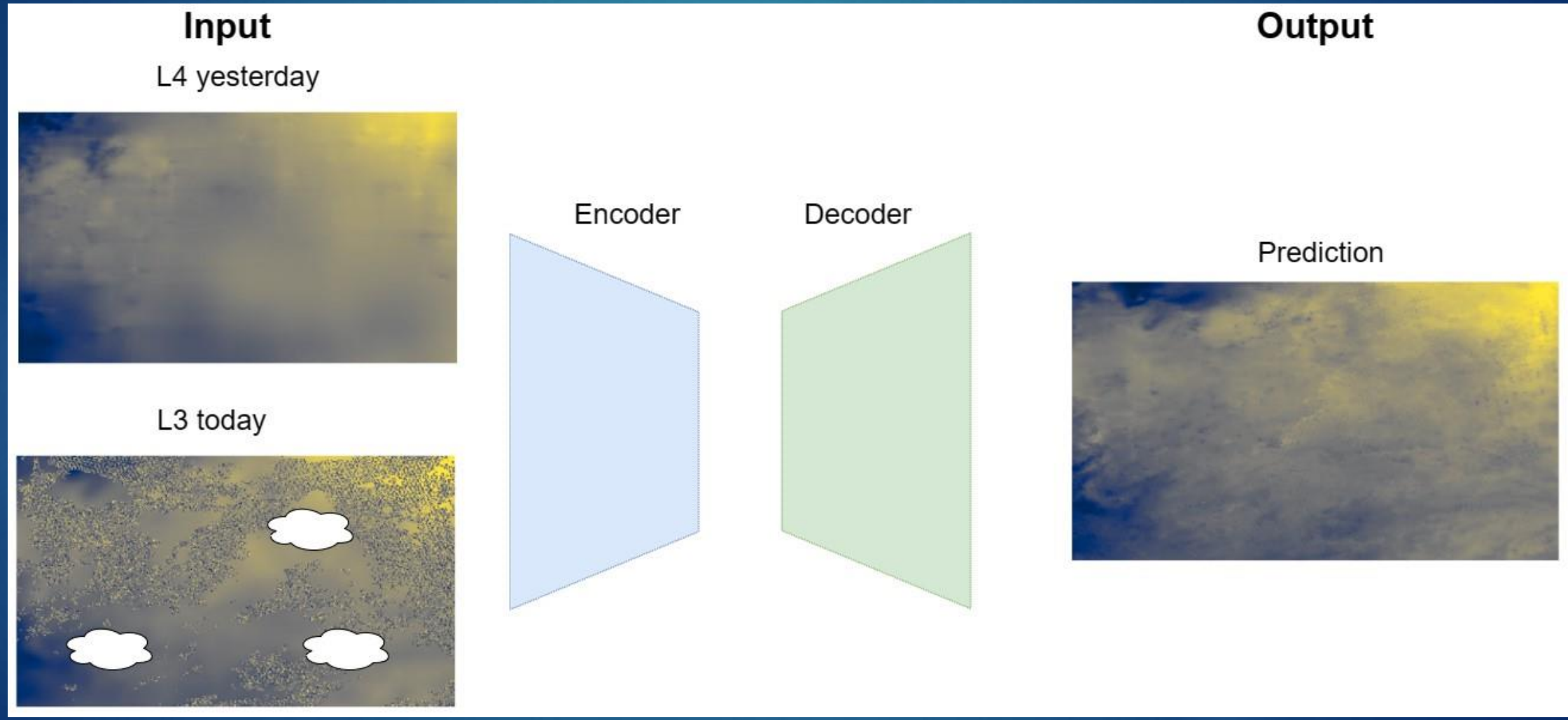
- Structure



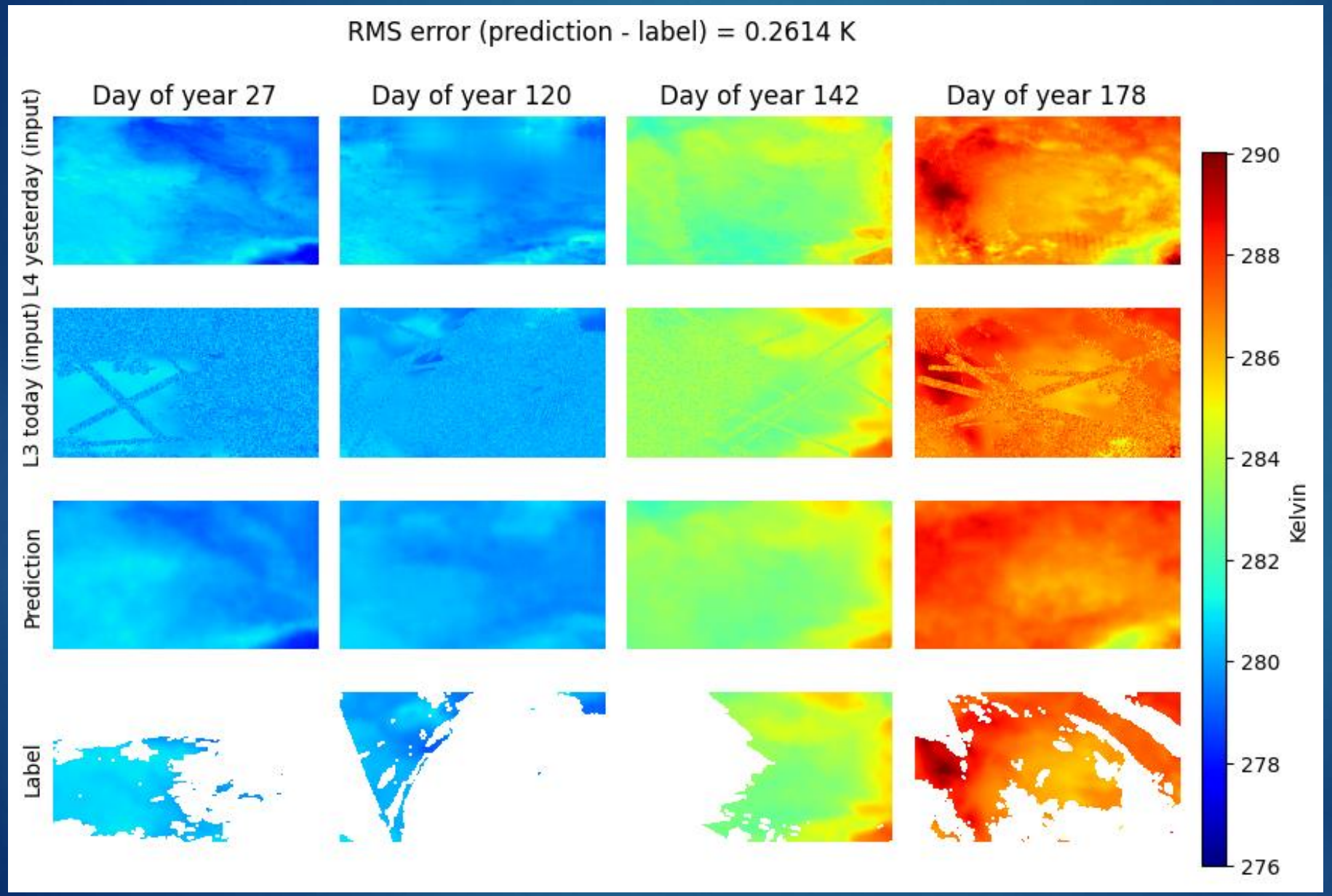
- Example results



Adding information from the day before



CNN interpolation simulating a real scenario



Concluding the project

- The combined GRU model should be modified to improve convergence properties. Unphysical properties of the artificial cloud
- The CNN needs proper evaluation

Further work

- Increase data size – both in time and space
- Combine the GRU and CNN methods
- Add variables: Pressure. Wind velocity. Precipitation. Etc.
- Include in situ observations in training

- Potential to beat the DMI algorithm

Appendix

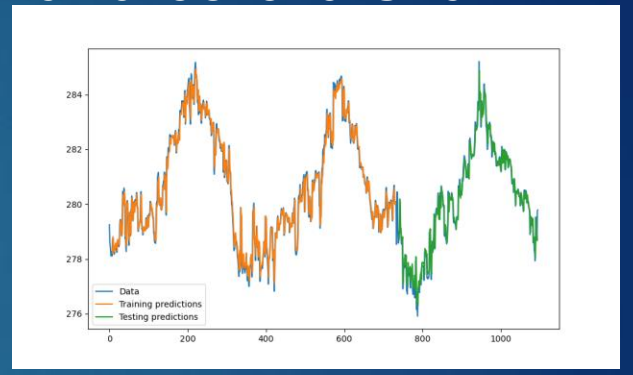
Appendix: GRU

Simple GRU

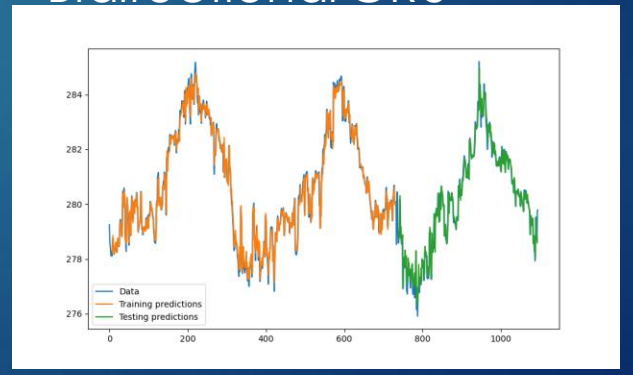
For the regular/unidirectional and bidirectional GRU:

- Lookback = 7
- Hidden layer = 7
- Data from January 1 2017 to December 31 2019 with a 70/20 split between training and validation
- Unidirectional RMSE: 0.54
- Bidirectional RMsE: 0.47
- Scaled with MinMaxScaler
- Dropout rate of 0.2 to prevent overfitting

Unidirectional GRU

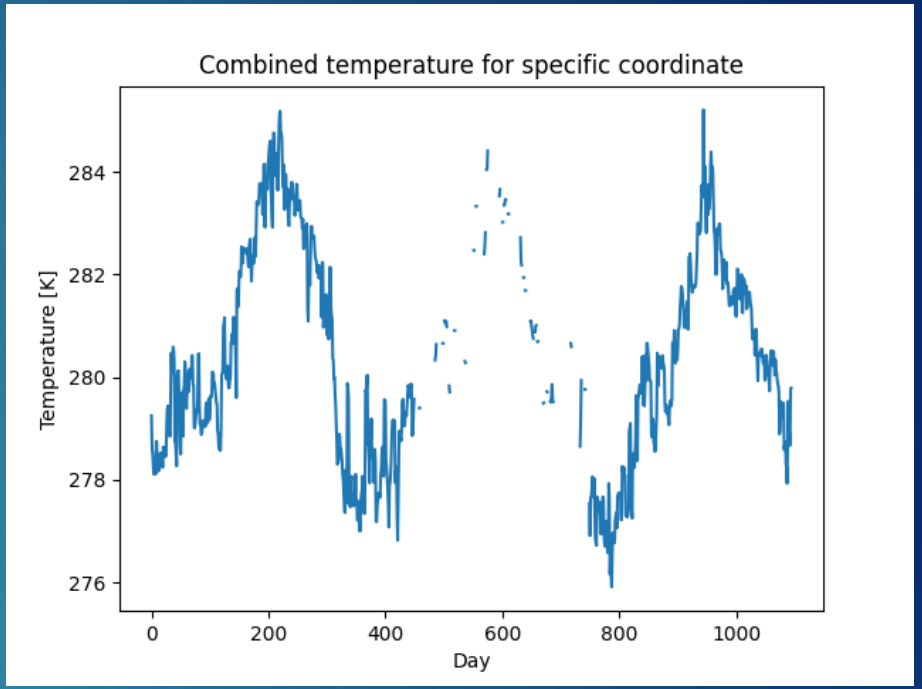


Bidirectional GRU



Simple GRU for combined L3 and L4 data

- Input features we a combination between the L3 and L4 data as well as time, and a nan-marker feature.
- Data from January 1 2017 to December 31 2019 with a 70/20 split between training and validation
- Scaled with MinMaxScaler
- Dropout rate of 0.2 to prevent overfitting
- Only predicted nan values



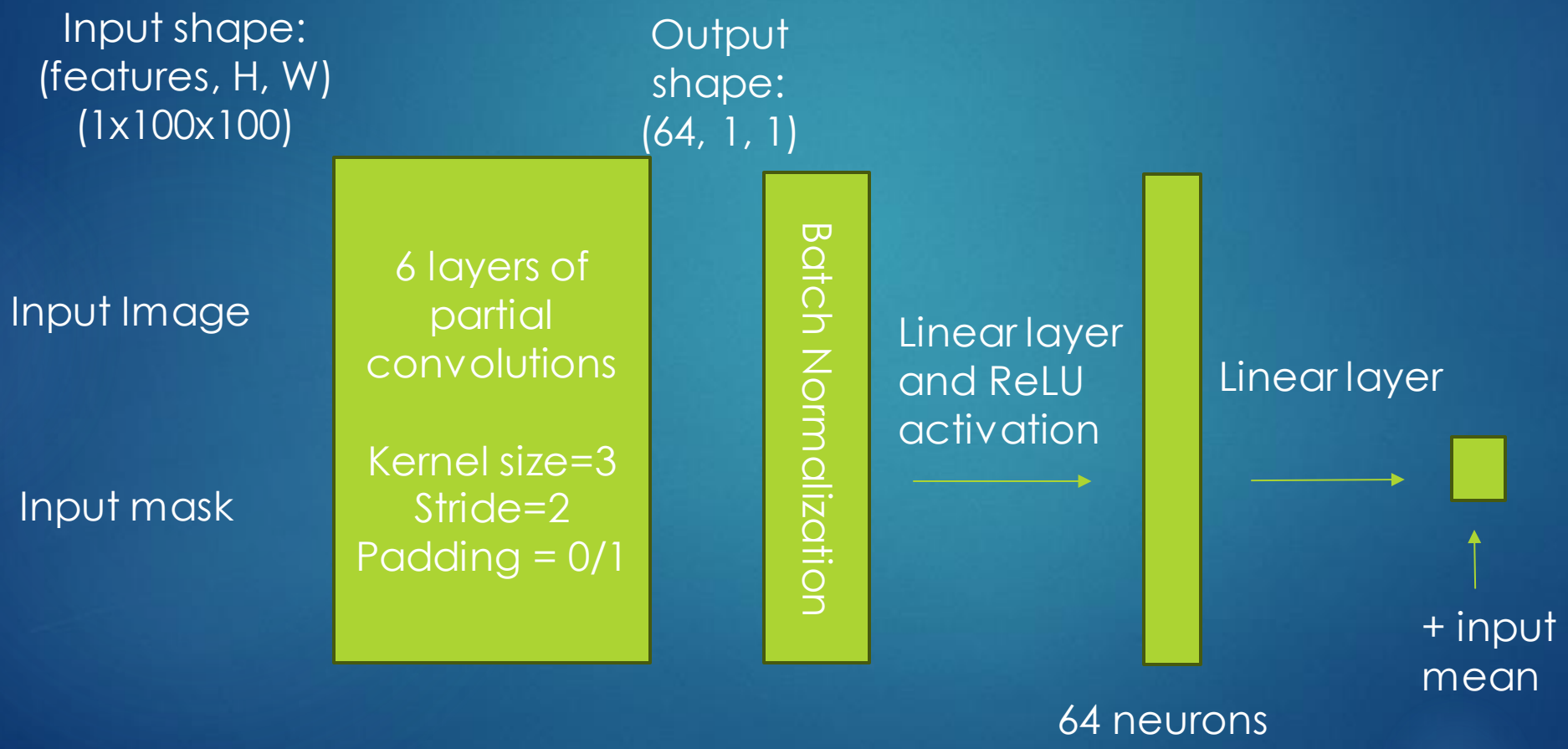
PCNN and GRU: Preprocessing

- L3 data from years 2016-2019 is taken in outlined region. Every second pixel is taken to reduce images to 100x100. Everything is normalized to between 0 and 1
- Training set: 2016-2018 data. Validation set: 2019 data
- The middle pixel is saved as a target. The available L3 data is augmented with L4 data where data is missing to create a more balanced dataset (would be biased towards summer weather if only using L3 data)
- Since the mean is a good guess, we want to make a "residual" model, meaning the mean is subtracted from each image, stored and is then to be added in the very end

PCNN and GRU: Simple PCNN

- Input to the benchmark PCNN is:
  - Image of today with mean subtracted and artificially inserted cloud in the middle
  - Binary mask indicating valid vs. NaN values
  - The mean of today
- Model is implemented in PyTorch. Partial convolutions is implemented using NVIDIA's "partialconv" (partialconv/models/partialconv2d.py at master · NVIDIA/partialconv · GitHub)

PCNN and GRU: Simple PCNN – Best Structure



PCNN and GRU
Simple PCNN – Optimization

- With no regularization, significant overtraining occurred
- Dropout layers with p=0.5 were inserted after batch normalization and after the first Linear + ReLU connection
- Learning rate was optimized with Adam with an initial learning rate of 1e-4 and tweaked later in training. Trained for 30 mins on google colab GPU until exhibiting signs of overtraining
- Many different combination of latent space sizes (64-256), placing and number of batch normalizations and dropouts were tried and the best solution is shown
- The number of convolutional layers is determined by when the image is reduced to (1x1). This process is inspired by an in-painting PCNN paper by NVIDIA and kept in the hope of adding a decoder later

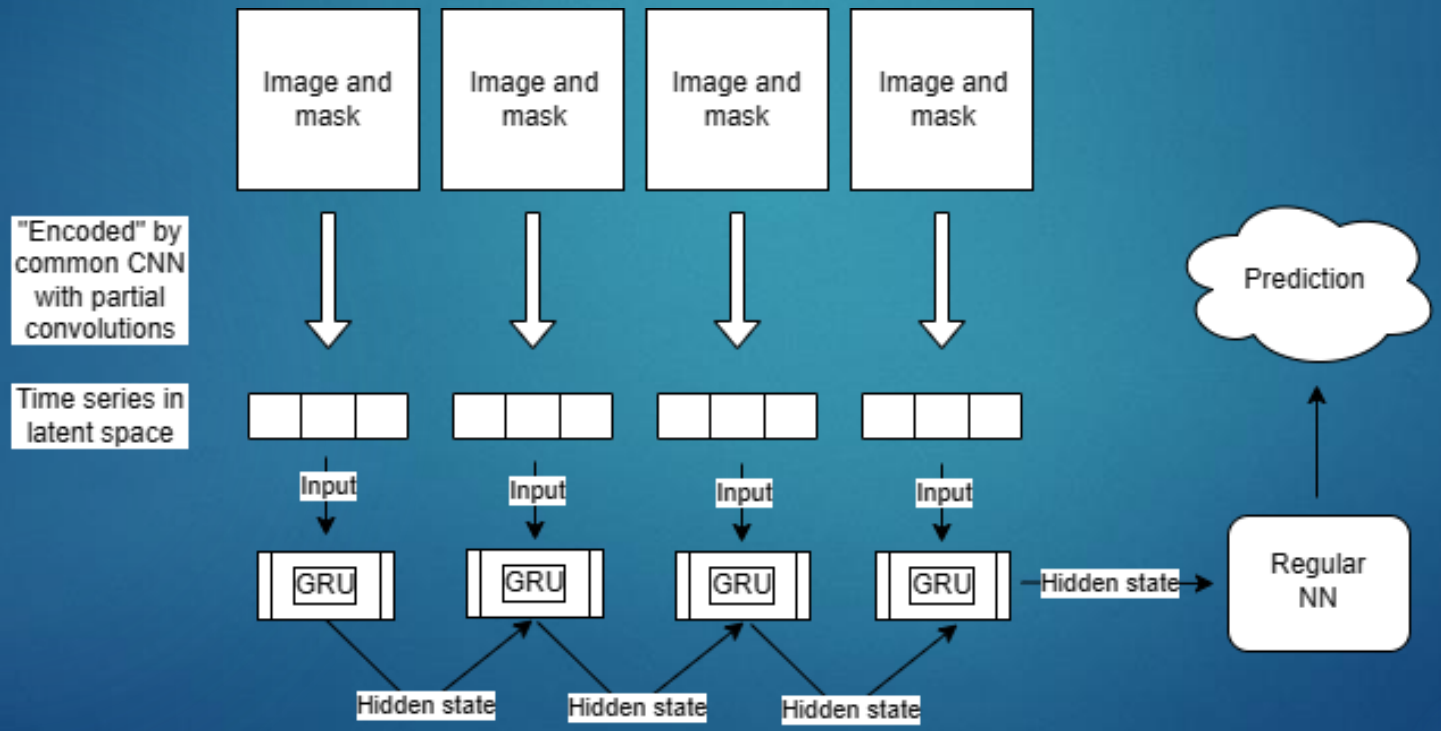
PCNN and GRU Combined Model

- Input to the combined model (PCNN + GRU) is:
  - Time series of 4 images with mean subtracted and artificially inserted cloud in the middle of the last image
  - Binary mask indicating valid vs. NaN values of all 4 images
  - The mean of all 4 images

- Model is implemented in PyTorch. Partial convolutions is implemented using NVIDIA's "partialconv" (partialconv/models/partialconv2d.py at master · NVIDIA/partialconv · GitHub)

PCNN and GRU: Combined model – Best Structure

- The PCNN encoding is identical to the simple PCNN up to and including the batch normalization and first dropout layer



PCNN and GRU: Combined model – Best Structure

- Details:
  - To the 64 latent space features is appended 2 things: The number of valid pixels in today's image and the change in mean (today – yesterday)
  - The GRU has 128 hidden states
  - The NN has 2 hidden layers (both size 64)
  - A skip connection is introduced between the latent space representation of the last image and the first hidden layer in the NN (essentially the "memories" of GRU should represent pertubations to the last latent space representation)
  - In the end, the mean of today is added to the prediction

PCNN and GRU: Combined model – Optimization

- If no dropout layers were introduced, the model overtrained significantly. We introduced extra dropouts with p=0.5 between the GRU and the regular NN and after the final hidden layer of the NN
- When doing the first dropout (after the PCNN encoding), we found that care had to be taken so that the SAME dropout was made on all 4 images. This made a significant difference
- We started without skip connections and just appending the mean and number of valid pixels. Then, we tried out several model architectures: Having the GRU look at DIFFERENCES in latent space, adding and subtracting hidden states, increasing number of hidden layers in the regular NN, modifying dropouts and batch normalizations. The best obtained result is shown

PCNN and GRU: Combined model – Optimization

- The model was optimized with Adam, and the learning rate of the different parts of the network was tweaked during training.
- The model was trained for 50 min on google colab free GPU until loss settled around a stable value

Appendix: CNN

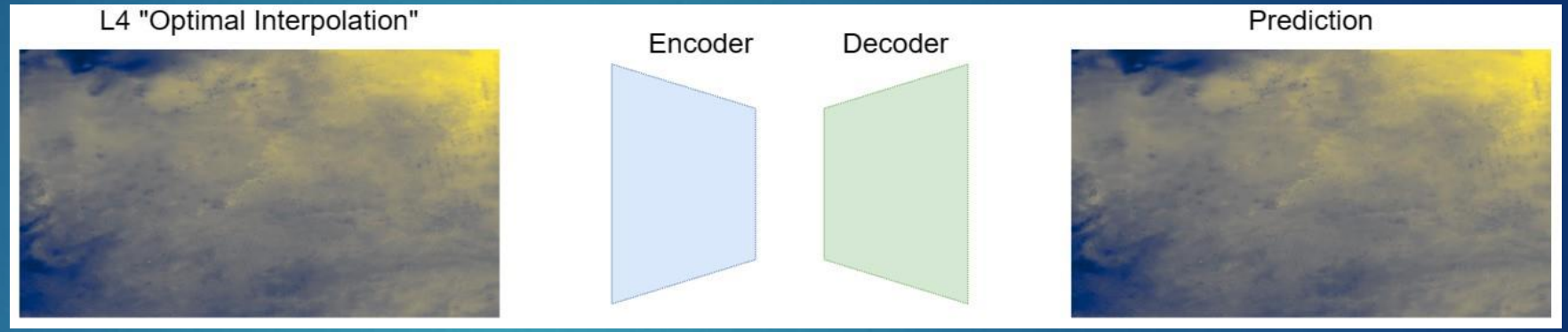
- Normalizing the images for CNN
- Auto encoder (testing the architecture when there is no noise)
- CNN interpolation using L4 from yesterday and L3 from today
- CNN interpolation simulating a real scenario (alternativ version)
- CNN error evalutation

Normalizing the images for CNN

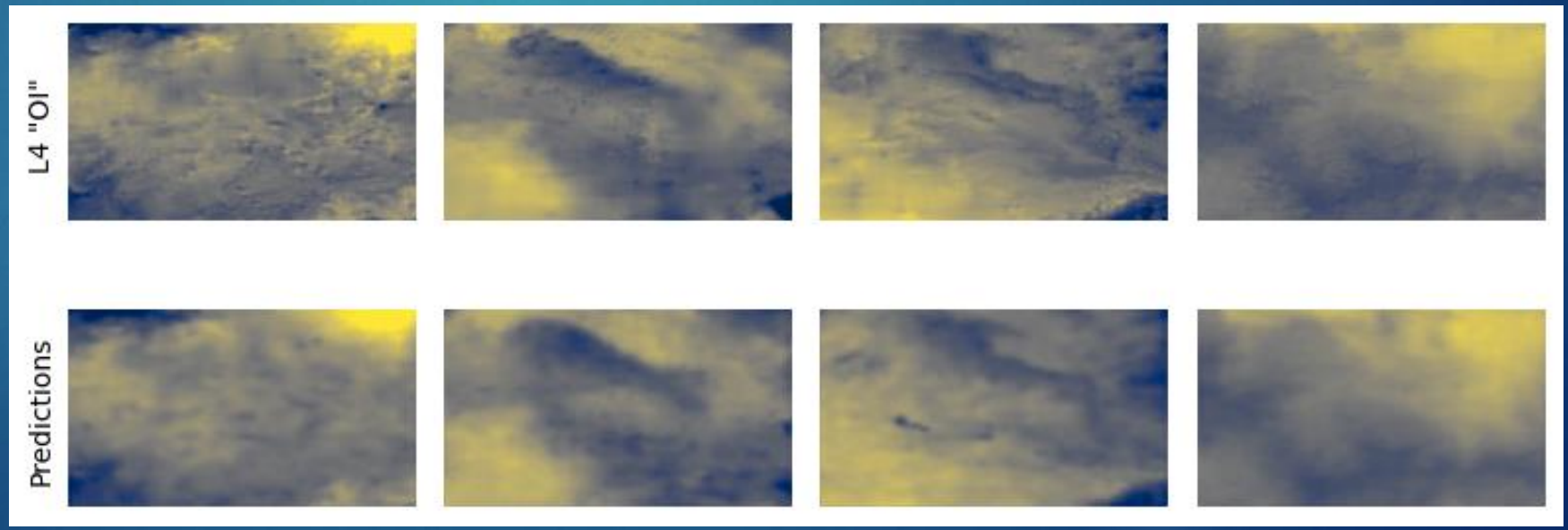
- The temperature data is given in Kelvin and can thus only take positive values. Naturally the sea surface temperatures lay in the range 270 to 300 Kelvin in which most variability can be explained by the season.
- Normalizing by a similar approach in every season will e.g. course winter images to take only low values and summer images to take only high values. Furthermore most days will have a low spatial variability.
- Normalizing was done by subtracting the mean and dividing my the standard deviation of the level 4 data of the previous day. This way most images have data distributed approximately normally.

A regular autoencoder

- Structure



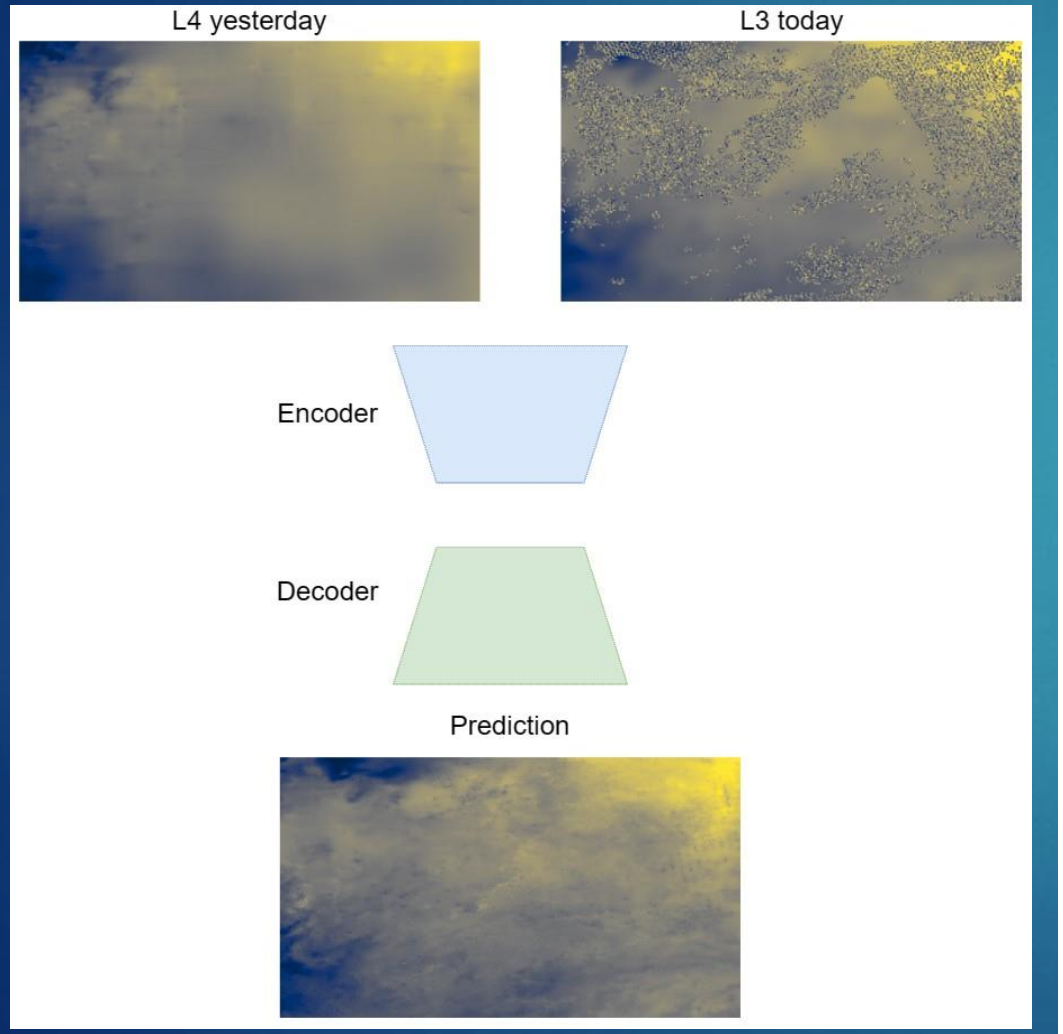
- Example results



Validation loss = 0.09

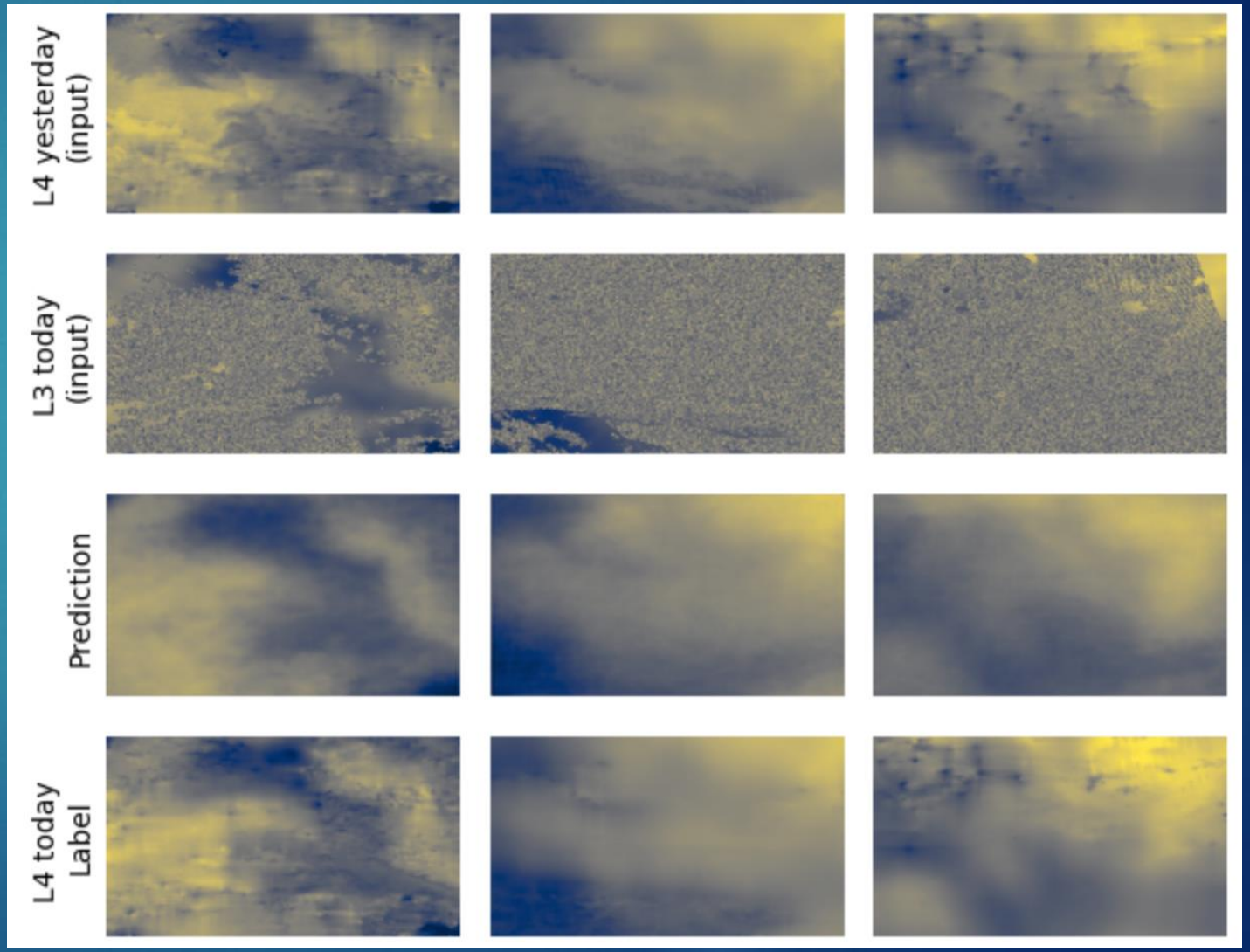
CNN interpolation using L4 from yesterday and L3 from today

- Structure: L4



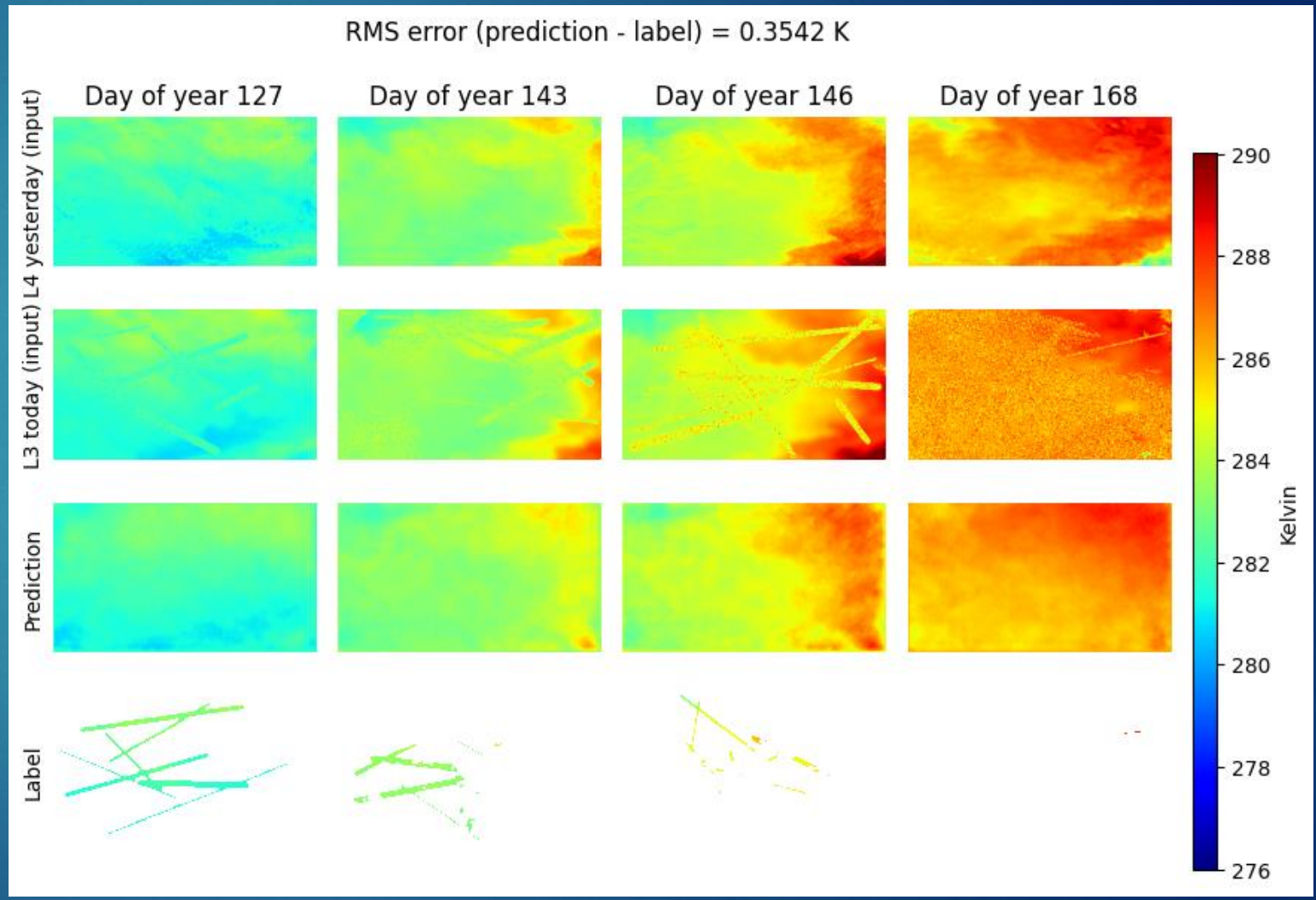
- Results

Validation loss = 0.17



CNN interpolation simulating a real scenario (alternativ version)

- In this verison of the CNN interpolation, where fake clouds are added to the L3 input data, the training pictures are masked, so that only pixels in which fake clouds have removed L3 data are used for training and evaluation.



CNN error evalutation

- The error of the "Optimal Interpolation" DMI method is evaluated by comparing the L4 product to in situ data from buoys and ships. The RMS error is approximately 0.7 Kelvin.[3]
- The error of our algorithm is evaluated by taking the RMS of the error between L3 and predictions yielding approximately 0.3 Kelvin. The RMS error in pixels where fake clouds were added was 0.4 Kelvin.
- Since a different method of evaluation is used nothing can be concluded, but the CNN method do shows promising results.

3: Høyer, J. L., Le Borgne, P. and Eastwood, S. 2014. A bias correction method for Arctic satellite sea surface temperature observations, Remote Sensing of Environment