Predicting the Past

Vasileios Dalampiras, Carla Mailin Friedrich,
Aikaterini Karathanasi, Peter Resch

Our Motivation

Do you need rainpants for your bike ride from uni??

- **Reliable weather predition mechanism for precipitation**

OUR AIM

What is even important when trying to predict weather??

Data

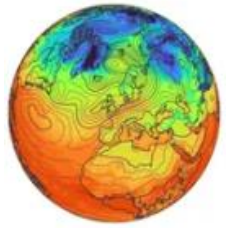
Reanalyis climate data

Time resolution:
Hourly from 1940

Variables available:
>200

Dataformat: GRIB

Product used from:
ERA5 hourly data on single levels from 1940 to present

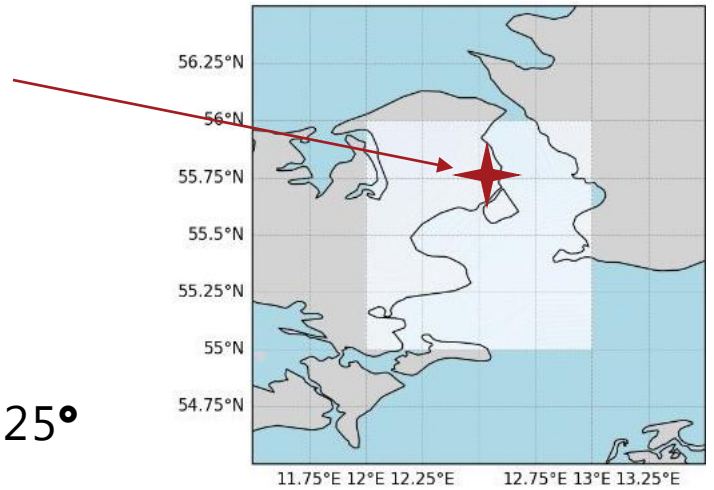


Climate Data Store

Single Point

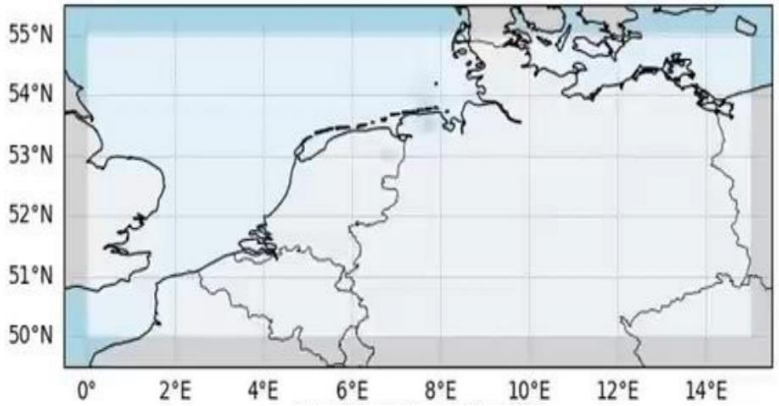
Small Grid

Lat x lon: 5 x 5
Grid Resolution: 0.25°



Big grid

Lat x lon: 21 x 61



Selection of Variables

(the ones we judged relevant for precipitation)

10m u-component of wind

10m v-component of wind

surface pressure

temperature 2m

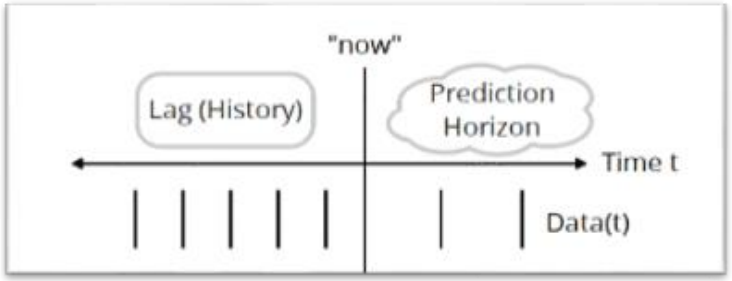
total precipitation

Surface solar radiation downwards

Total cloud cover

Spatial Precipitation Forecasting over Northern Europe Using XGBoost

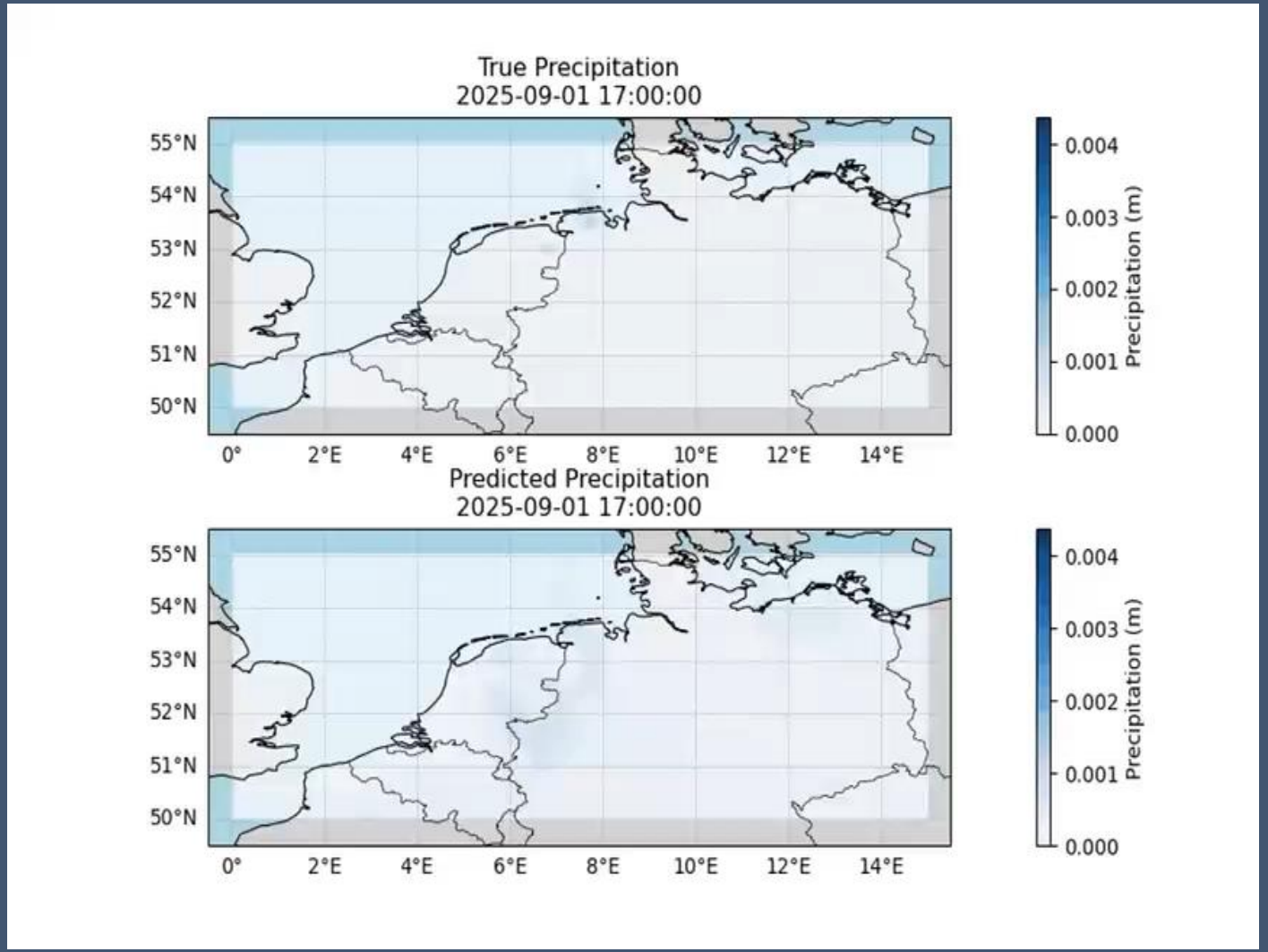
- **Coordinates:** North: 55°, South: 50°, West: 0°, East: 15°
- **Time Period:** Hourly data from March to September for the years 2023–2025
- **Method Used:** XGBoost Regressor
- **Prediction Horizon:** 1 hour into the future
- **Training Strategy:** Undersampling of dry events



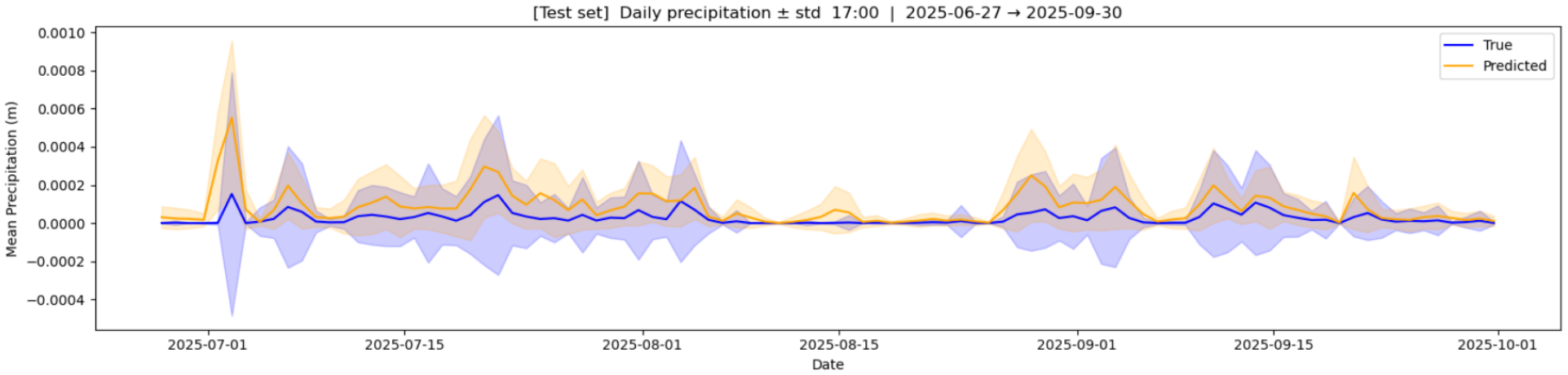
Observed vs Predicted daily Precipitation for 09/2025 using XGBoost

- Correct representation of the overall rainfall pattern
- Underestimation of rainfall intensity
- Occasional false precipitation alarms

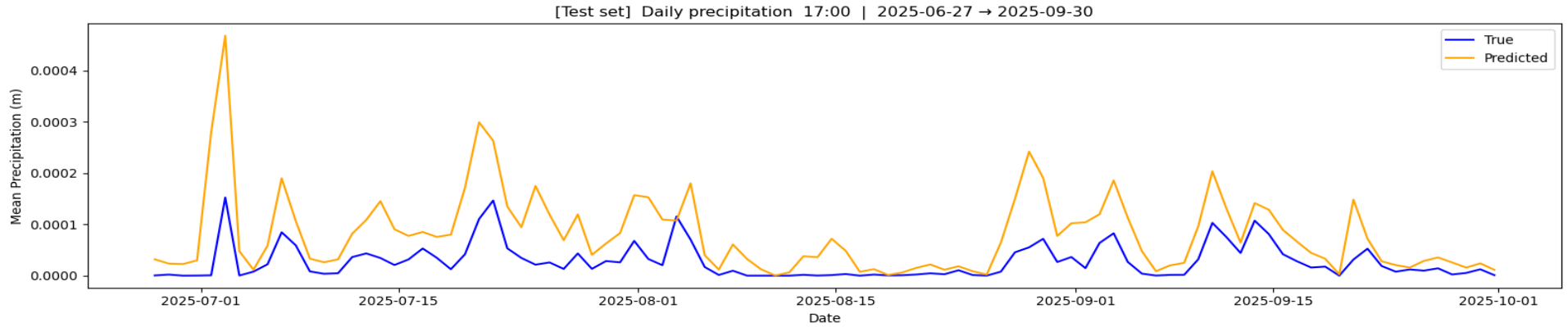
Film:



True Vs Predicted Precipitation for the Test Set using XGBoost



Daily prediction at 17:00



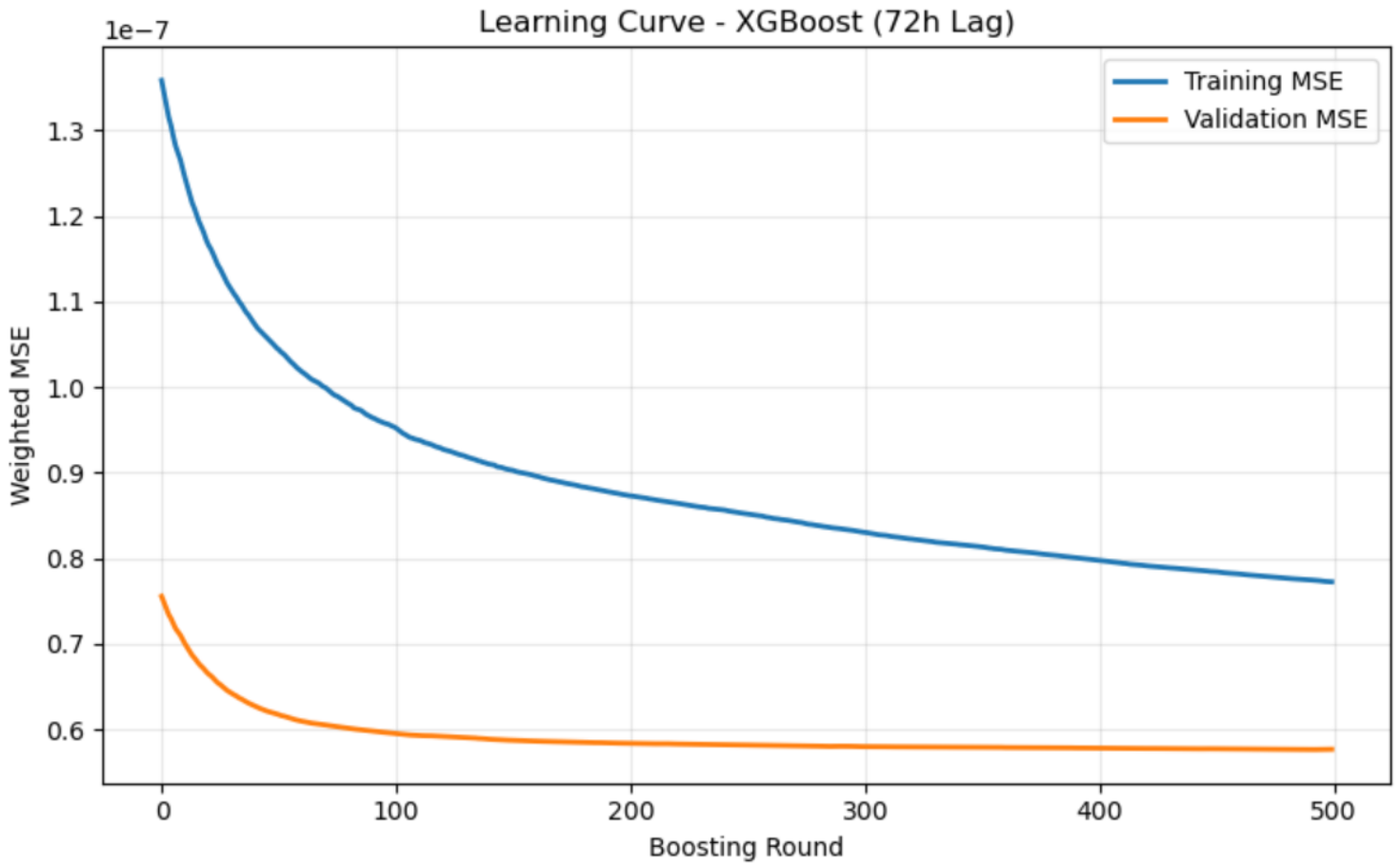
Hourly prediction

Prediction target: "tp"
Prediction horizon: 1h
Lag: (72, 48, 24, 12, 6, 3, 1h) x (6 features)
applied at each grid point

Model Performance

Metric	Result
RMSE	0.197 mm
MAE	0.079 mm
FAR	0.781

- Training and validation errors **converge** => **stable model** learning.
- **Lower validation error**: more balanced and challenging training dataset because **undersampling**.

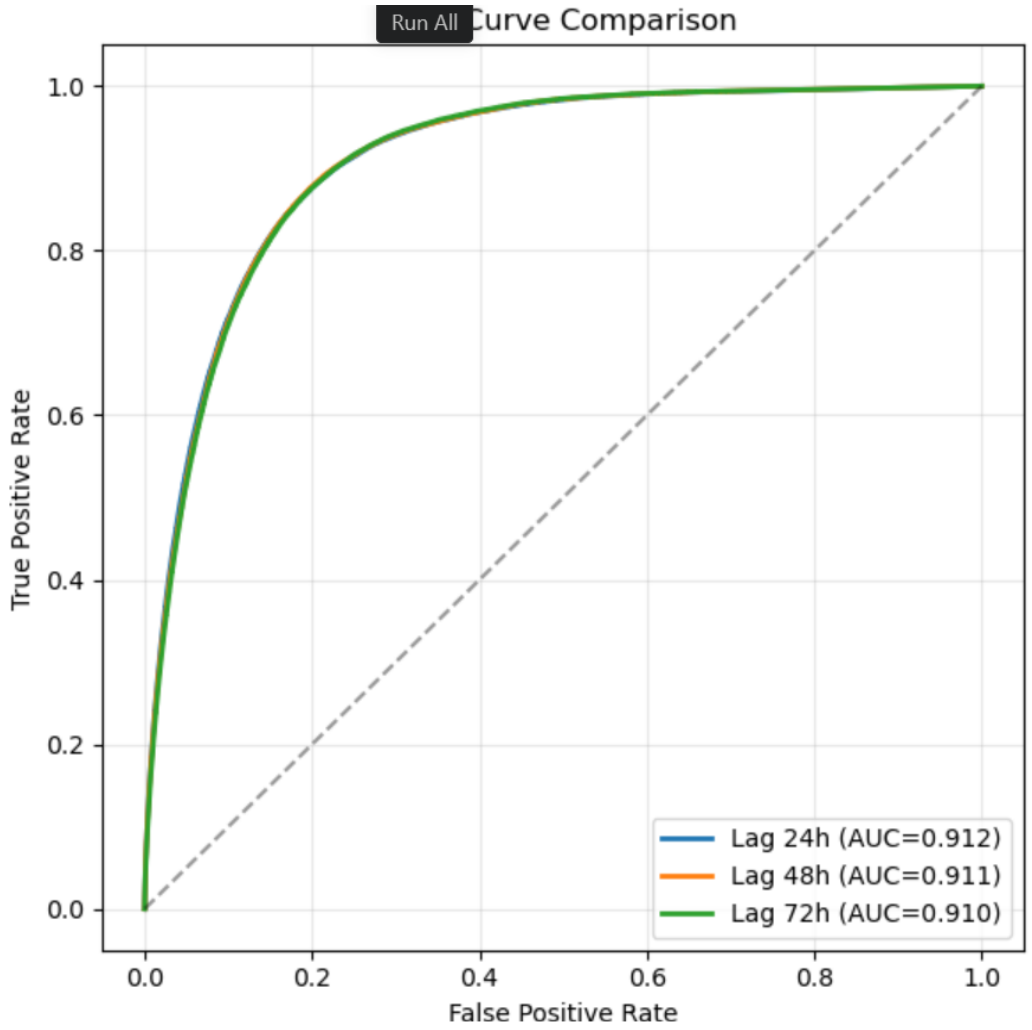


Prediction target: "tp"
Prediction horizon: 1h
Lag: (72, 48, 24, 12, 6, 3, 1h) x (6 features) applied at each grid point

Comparison of Different Time Lags

Model	MAE (mm)	RMS(mm)	Correlation
24 h lag	0.076	0.193	0.390822
48 h lag	0.078	0.195	0.385328
72 h lag	0.079	0.198	0.374042

- Increasing the past time lag **does not** significantly affect forecasting performance.
- **Shorter** lag windows: saves computational time with **similar results**.

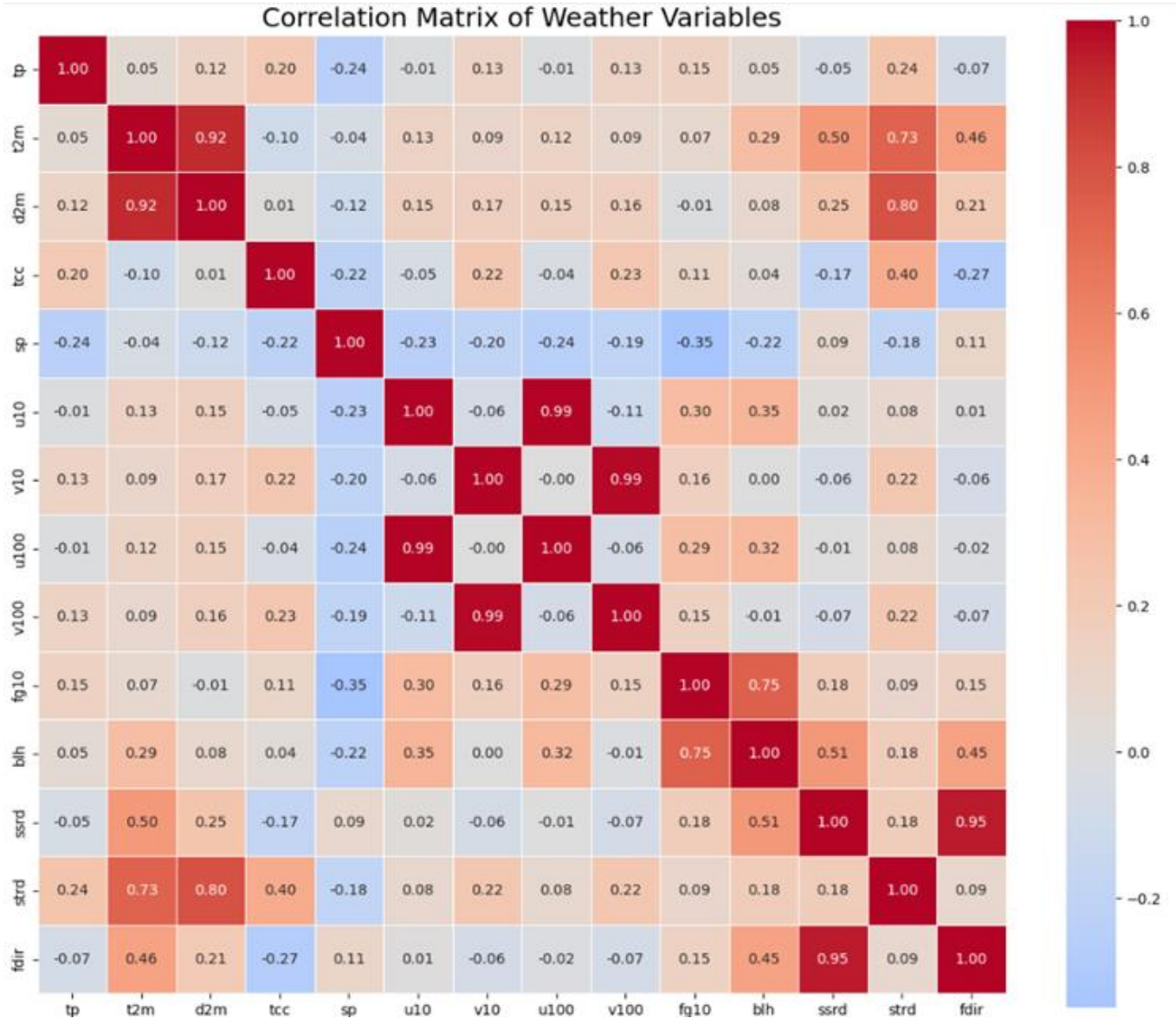


Prediction target: "tp"
Prediction horizon: 1h
Lag: (72, 48, 24, 12, 6, 3, 1h) x (6 features)
applied at each grid point

How Well Can ML Predict Precipitation at a Single Point, and Which Features Are Most Important?

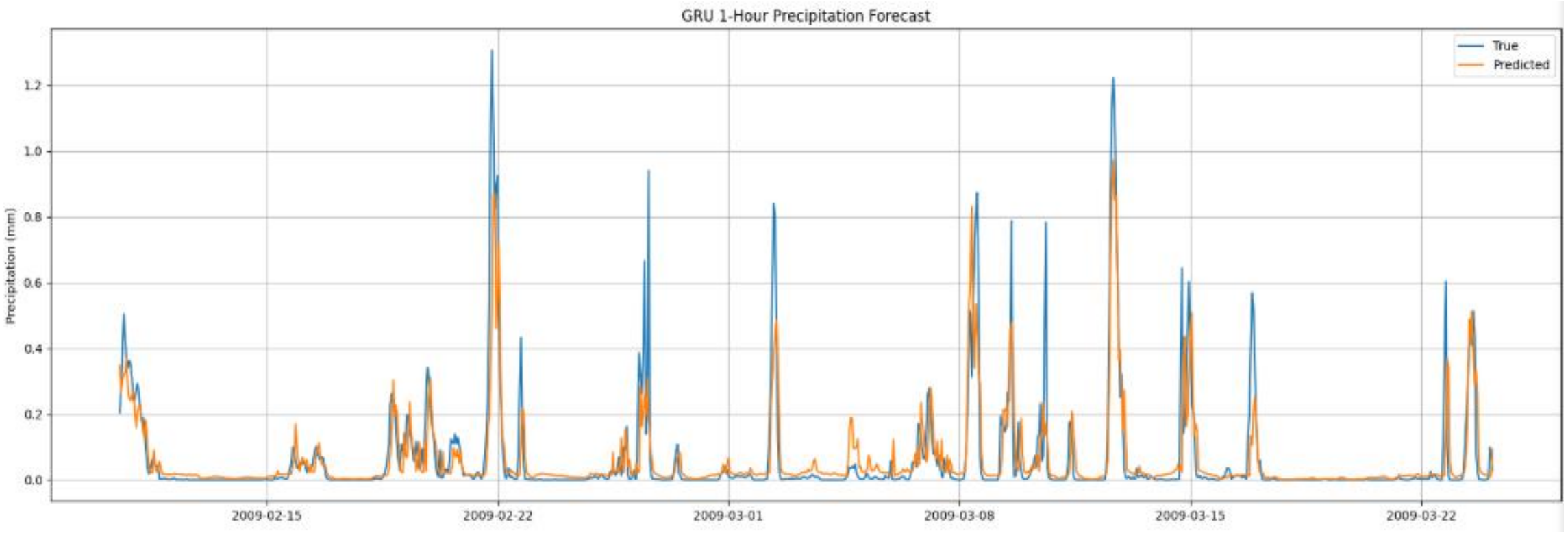
Correlation Matrix

Variable	Description
tp	Total Precipitation
t2m	2 m Temperature
d2m	2 m Dew Point Temperature
u10	10 m Zonal Wind
v10	10 m Meridional Wind
u100	100 m Zonal Wind
v100	100 m Meridional Wind
fg10	10 m Wind Gust
tcc	Total Cloud Cover
sp	Surface Pressure
blh	Boundary Layer Height
ssrd	Solar Radiation Downwards
strd	Thermal Radiation Downwards
fdir	Direct Solar Radiation



GRU 1-Hour Prediction Forecast

Prediction target: "tp"
Prediction horizon: 1h
Lag: (24 consecutive h × 21 features)



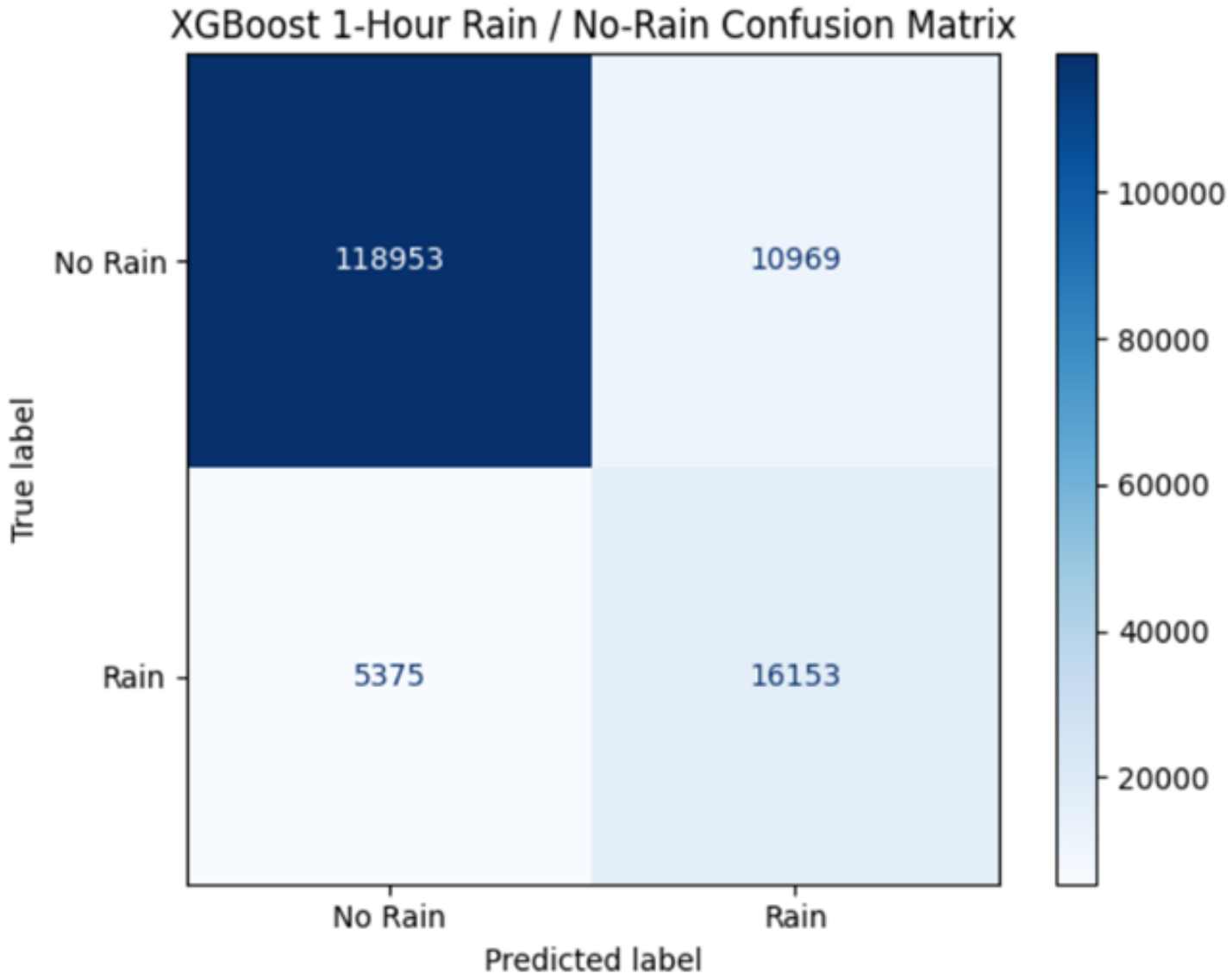
XGboost vs LSTM vs GRU Metrics Comparison

Model	RMSE (mm)	MAE (mm)	R^2
XGBoost	0.236	0.070	0.382
LSTM	0.236	0.071	0.382
GRU	**0.234**	**0.068**	**0.394**

- **GRU** achieves the best overall performance
- XGBoost and LSTM have very similar perfomance
- Differences between models are relatively small
- All models capture **short term precipitation** reasonably well

Rain/No Rain Analysis

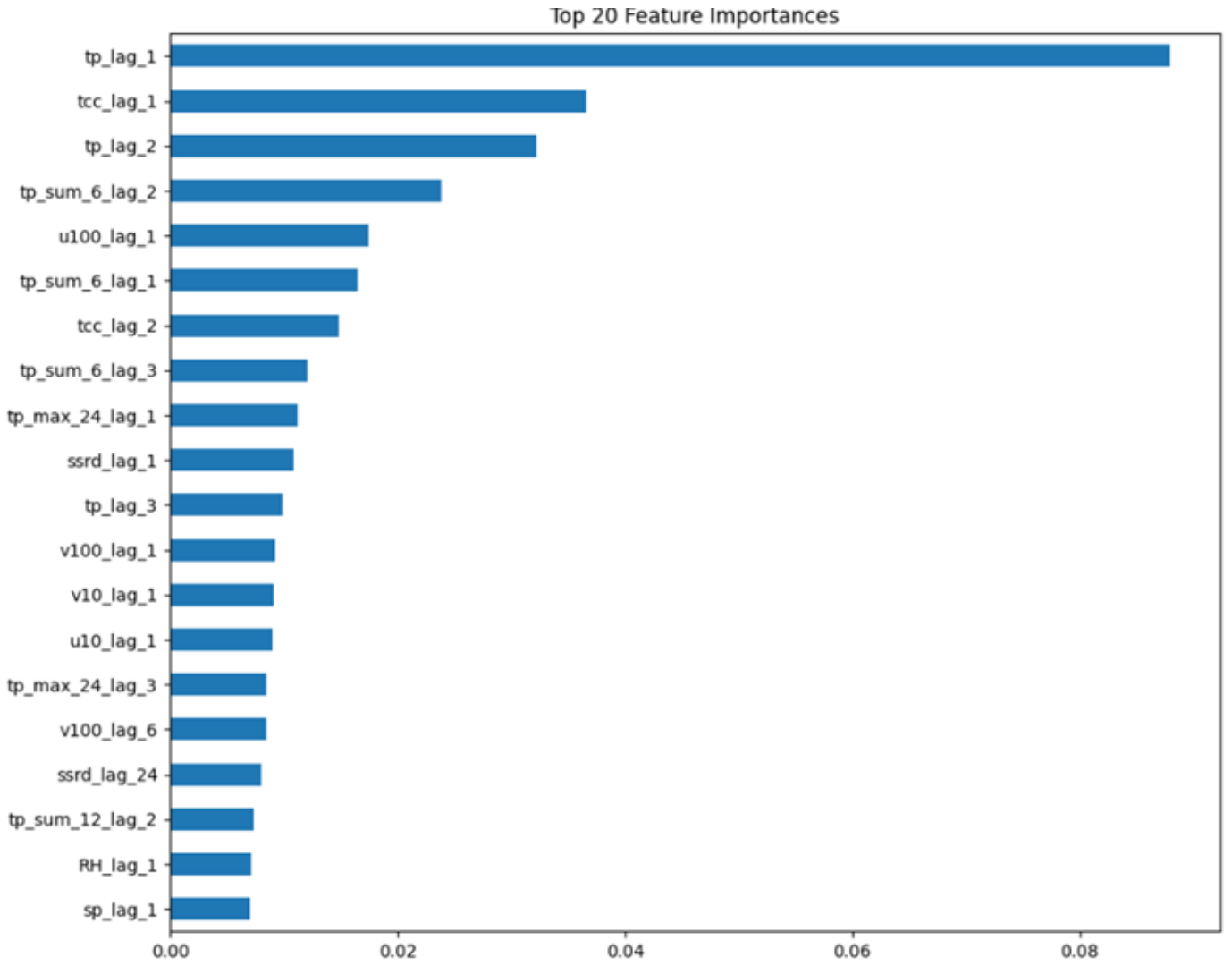
- Accuracy: 0.89
- Precision: 0.60
- Recall: 0.75
- F1 Score: 0.66



Feature Importance

XGBoost 1h prediction

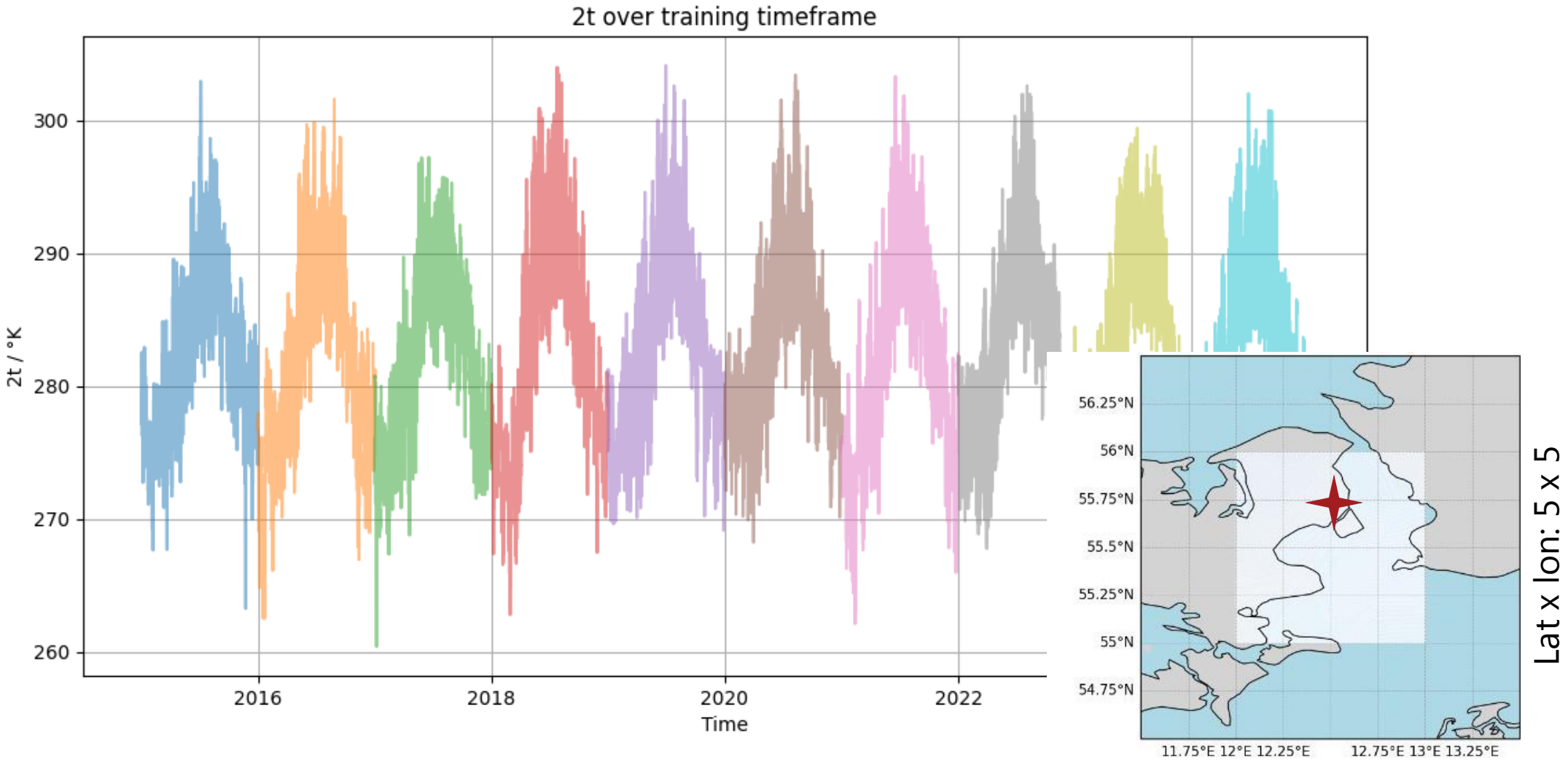
- **Previous precipitation** values are the strongest predictors of future rainfall
- **Cloud cover** is also influential as expected
- Humidity, radiation, and wind variables provide additional predictive information.



Does it matter what season we train on?

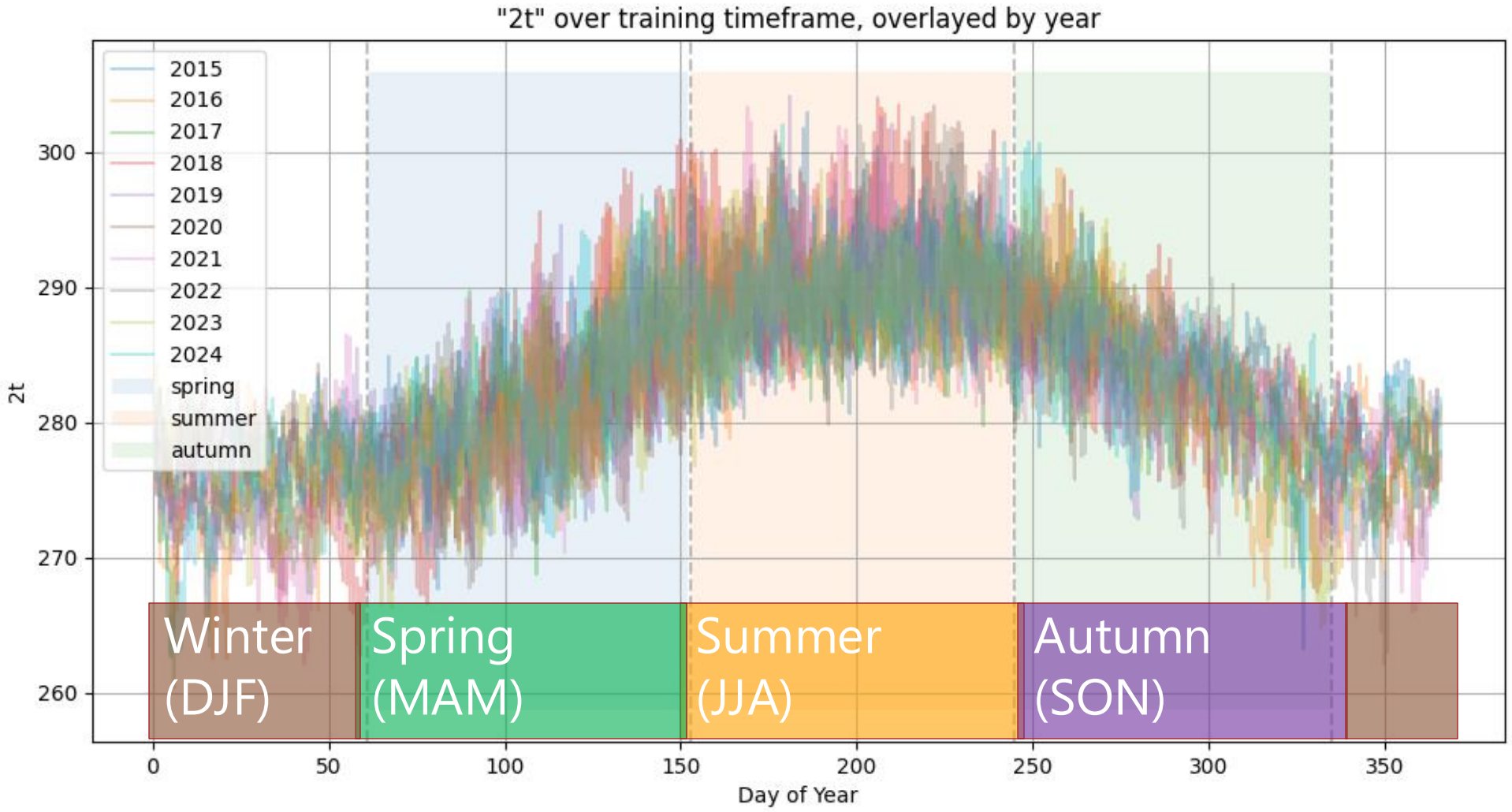
Every season is different...

"2t": temperature 2m over ground



Every season is different...

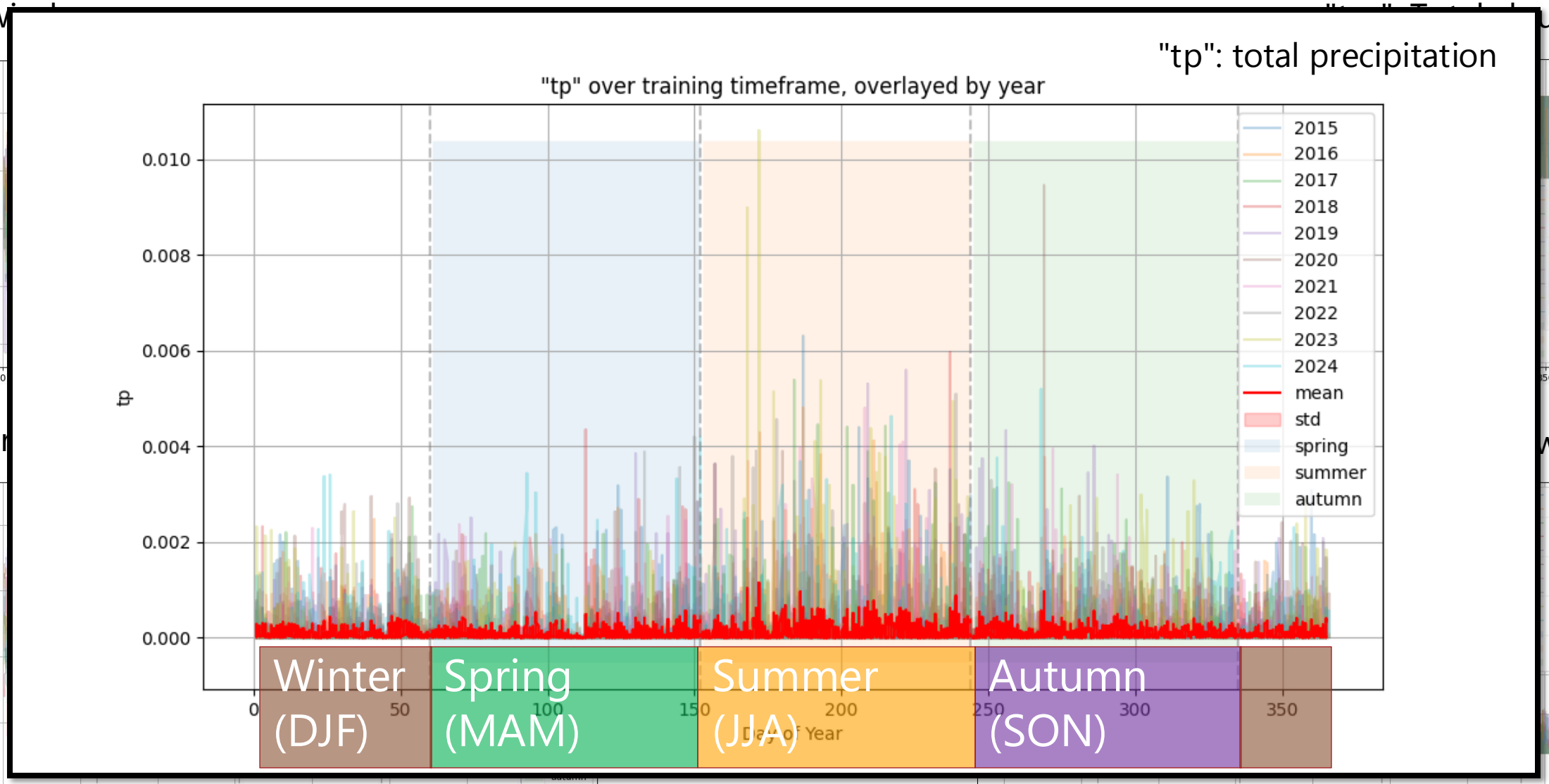
"2t": temperature 2m over ground



Every season is different...

"10v": w ud cover

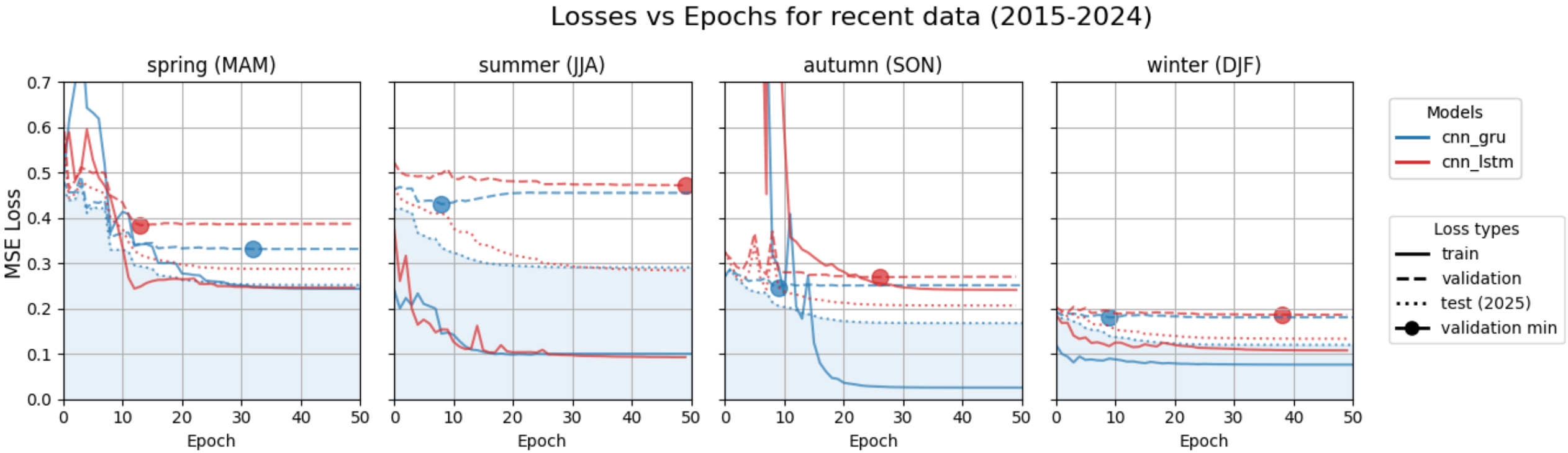
"tp": total precipitation



"sp": sur wnwards

Training on the recent seasons

Prediction target: "tp"
Prediction horizon: 1h x grid
Lag: (72, 48, 24, 12, 6, 3, 1h) x (7 features) x grid

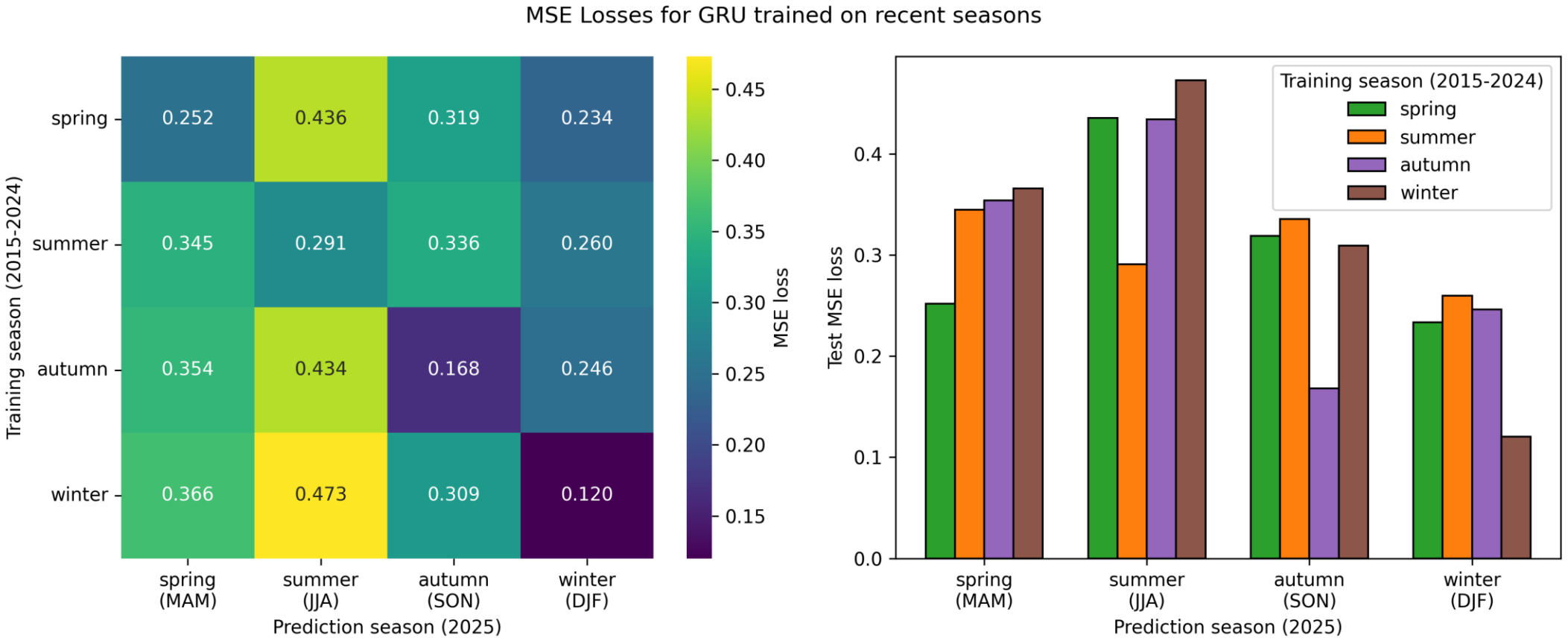


Data trained on for all 4 seasons	GRU	LSTM	LSTM HPT
"Recent" (2015-2024)	●	●	●
"Past" (1950-2010 every 10 years)			
"Combined"	●		●

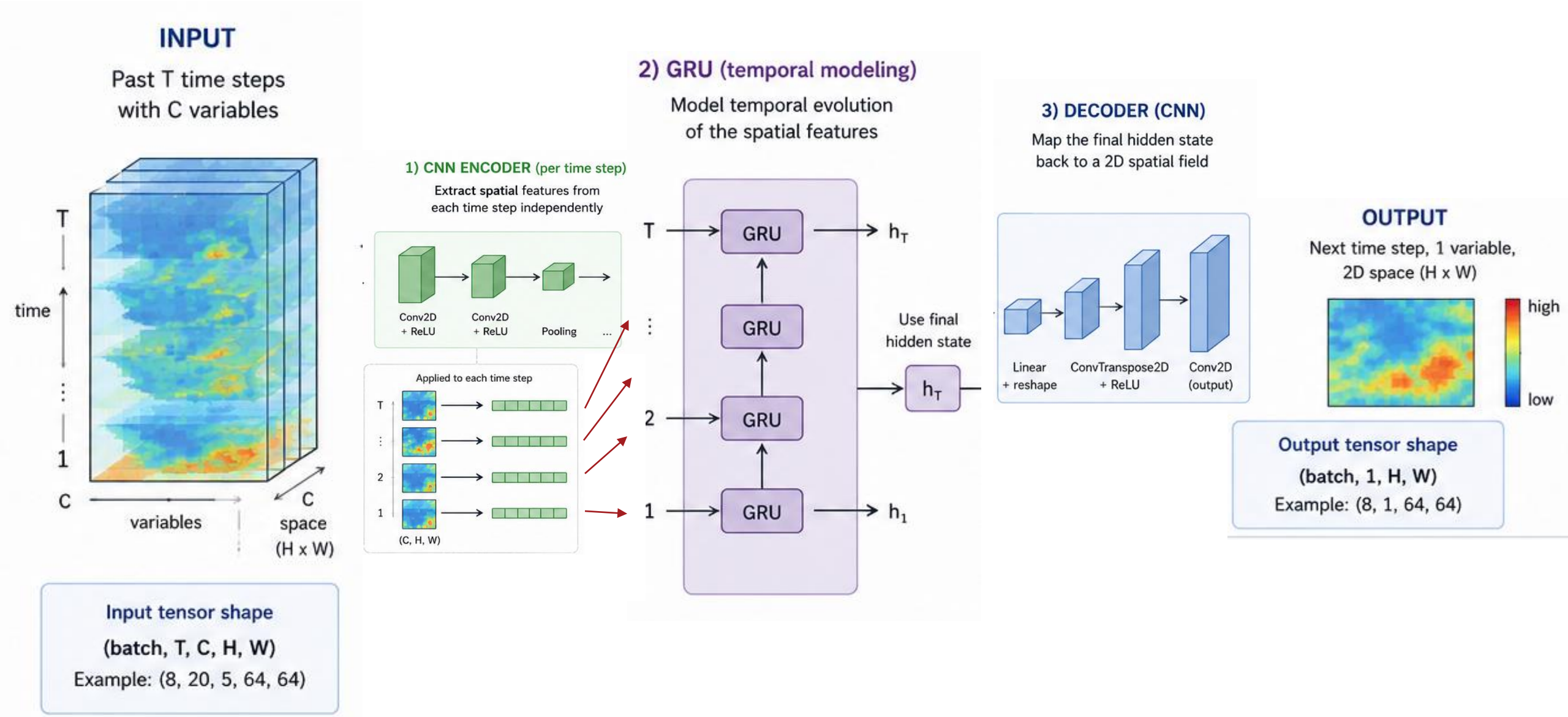
*see Appendix

Loss Matrix: Prediction season vs. Training season

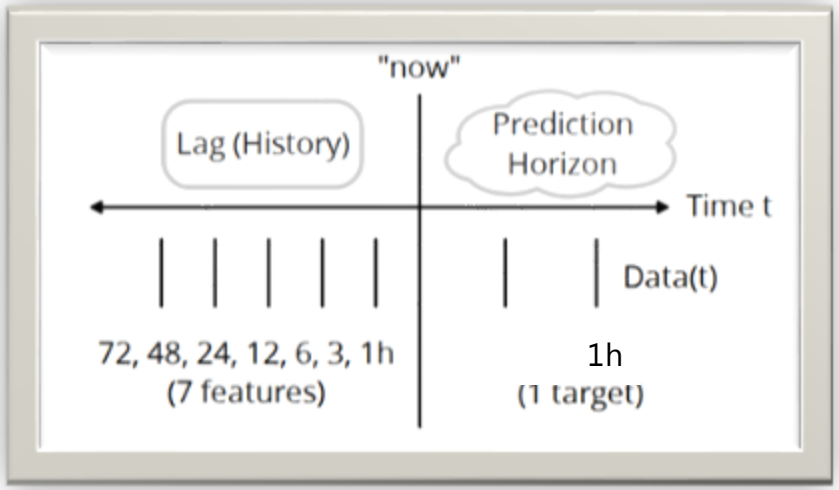
Prediction target: "tp"
Prediction horizon: 1h x grid
Lag: (72, 48, 24, 12, 6, 3, 1h) x (7 features) x grid

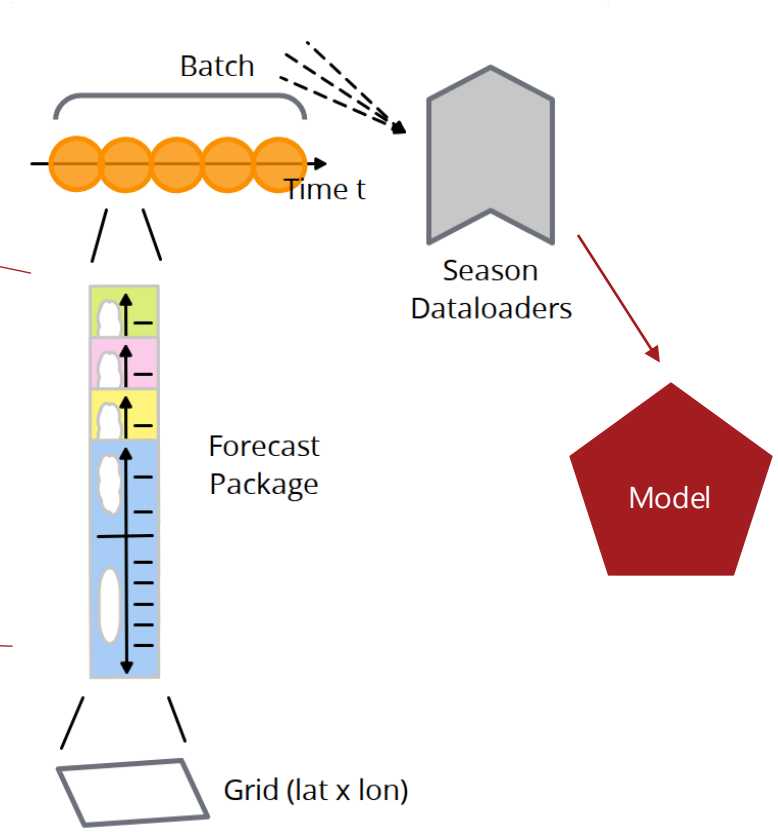


Predicting 2D Timeseries: CNN+GRU/LSTM



Dataprocessing and -flow for CNN+GRU/LSTM





Flow of Dataprocessing for model digestion

Does it matter what years we train on?

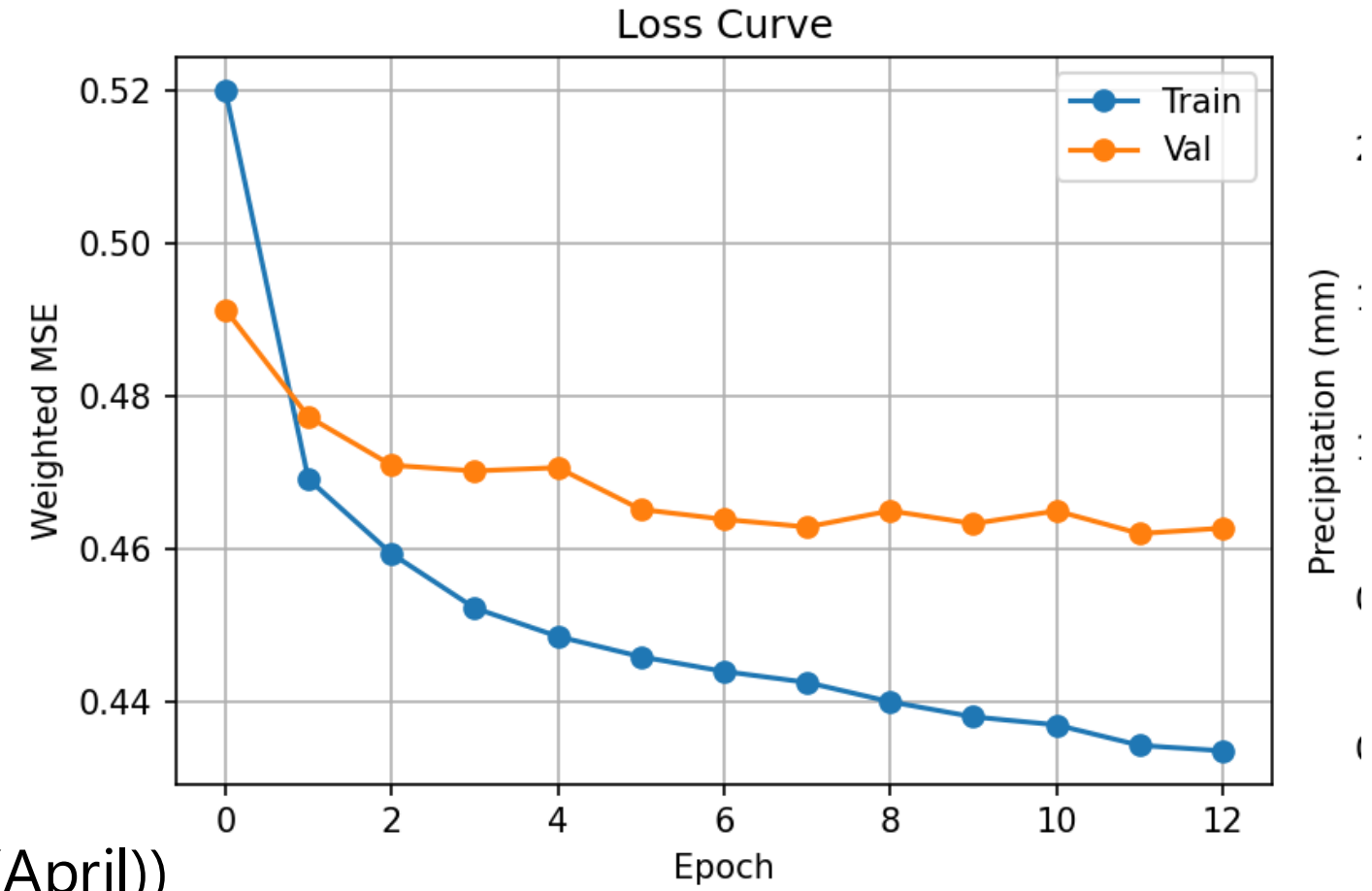
Predicting 6 hours in the future
(forecasting)

CNN-LGBM Model Performance

Prediction target: "tp"
Prediction horizon: 6h x grid
Lag: (12 (consecutive) x (7 features) x grid

Metric	Result
RMSE	0.167 mm
MAE	0.085 mm
FAR	0.787

FAR: fraction of rain alarms that are false

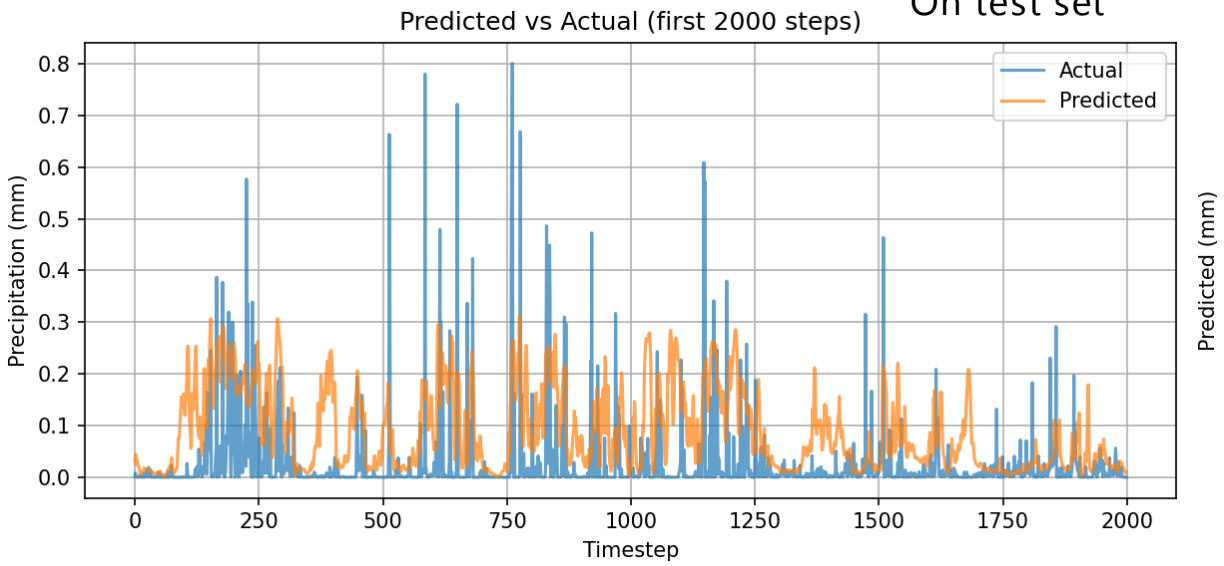


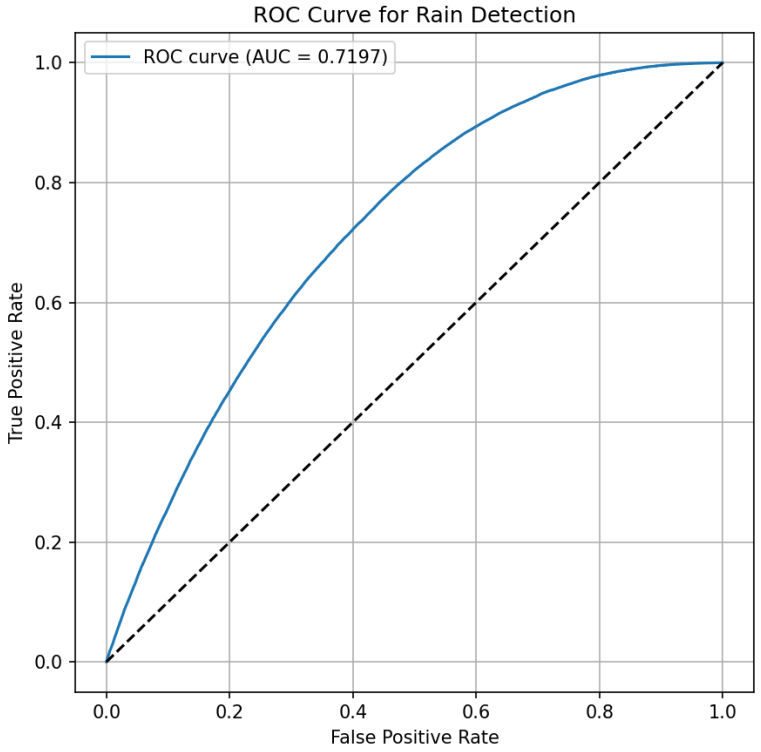
Training period: (2015-2026 (April))

Model Training

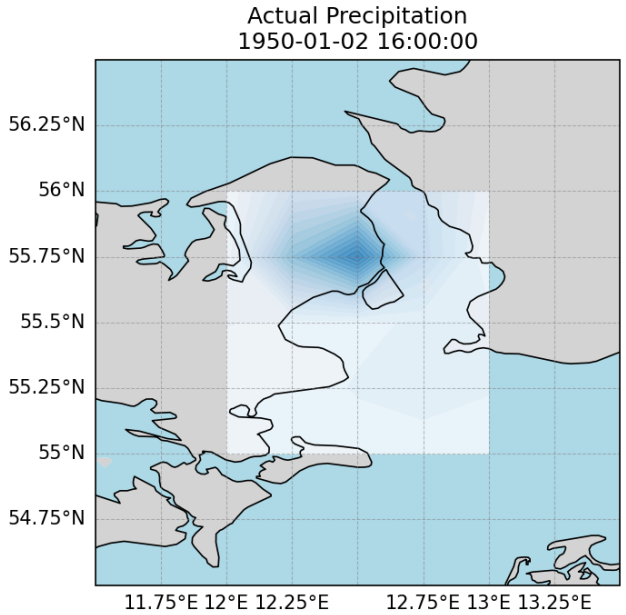
Prediction target: "tp"
Prediction horizon: 6h x grid
Lag: (12 (consecutive) x (7 features) x grid

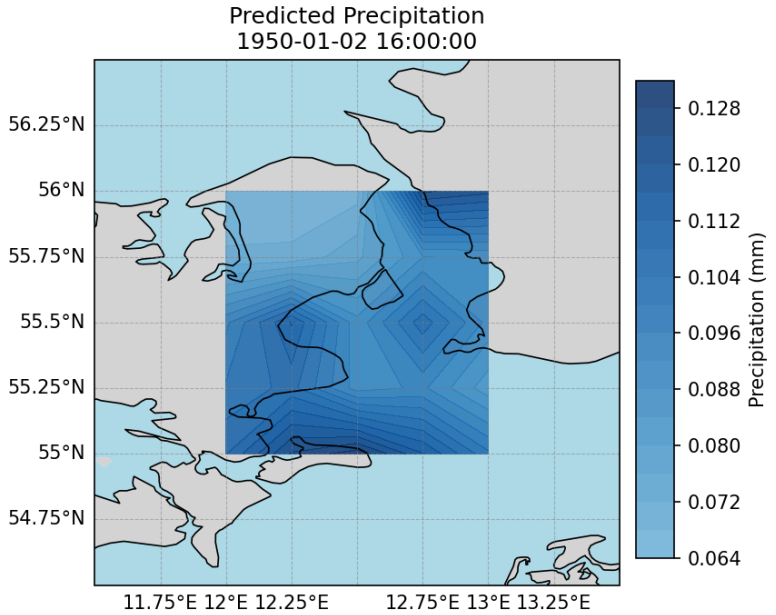
On test set





- We are catching the rain events!
- -> a lot of false positive alarms





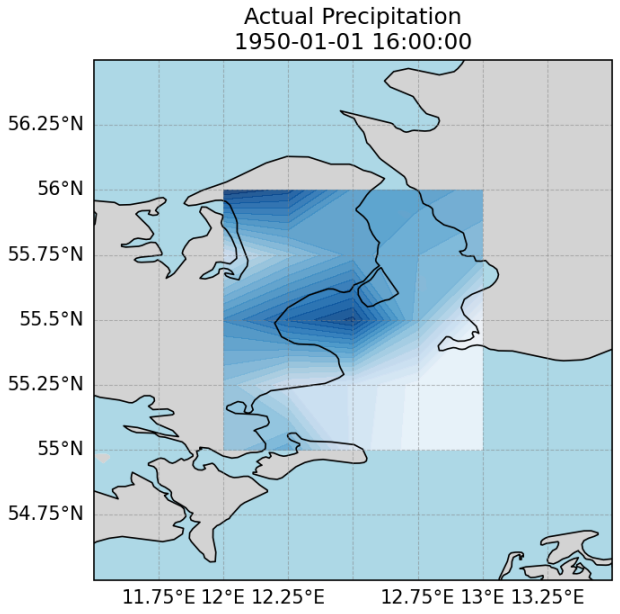
Prediction target: "tp"
Prediction horizon: 6h x grid
Lag: (12 (consecutive) x (7 features) x grid

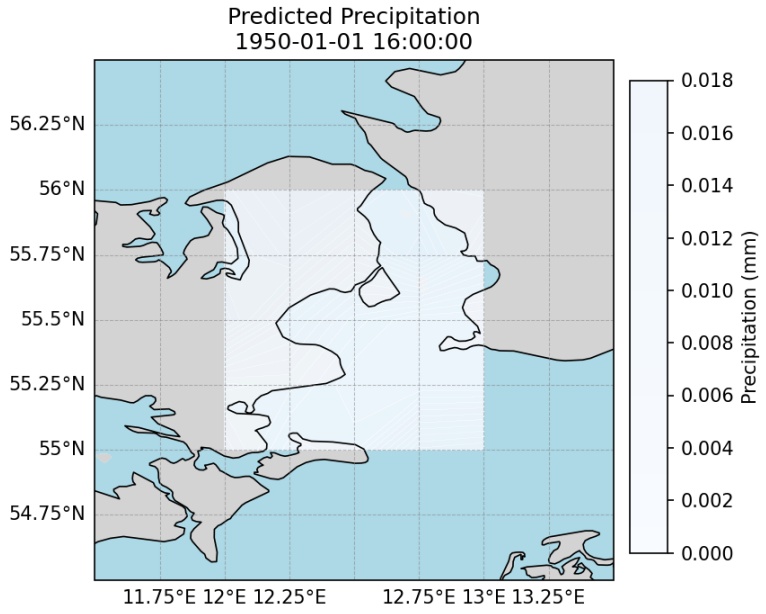
01.01.1950

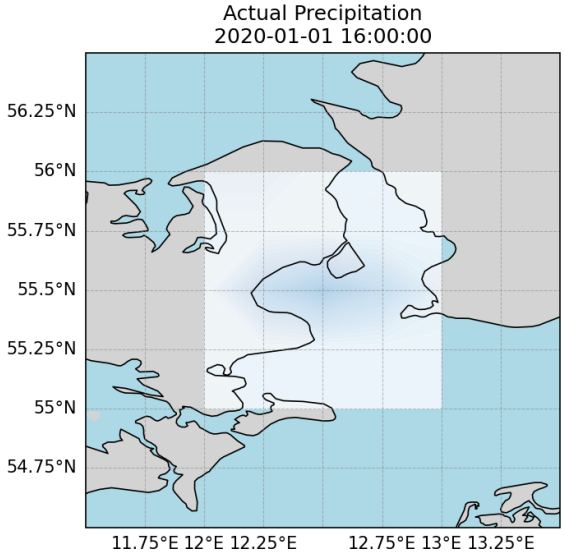
January

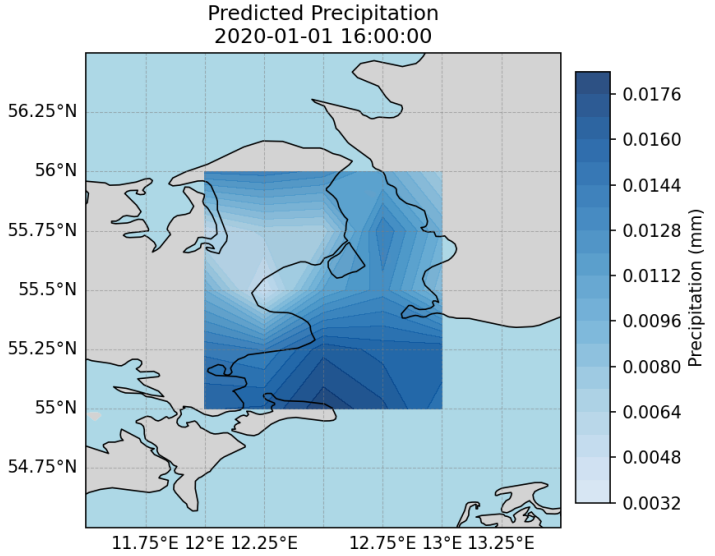
1950

02.01.1950





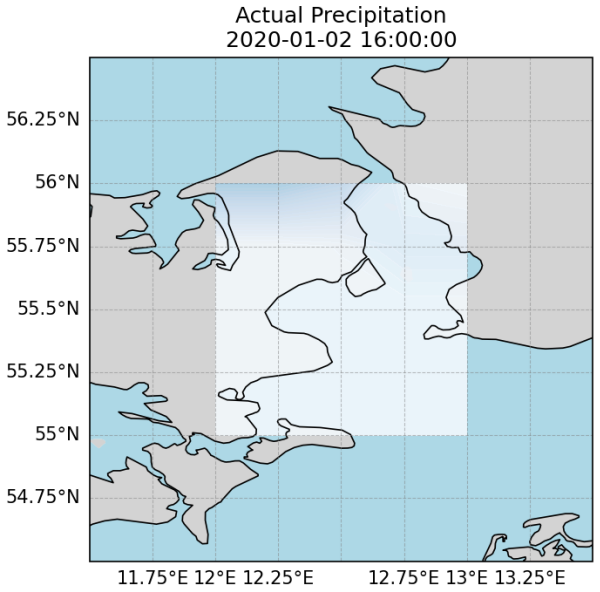


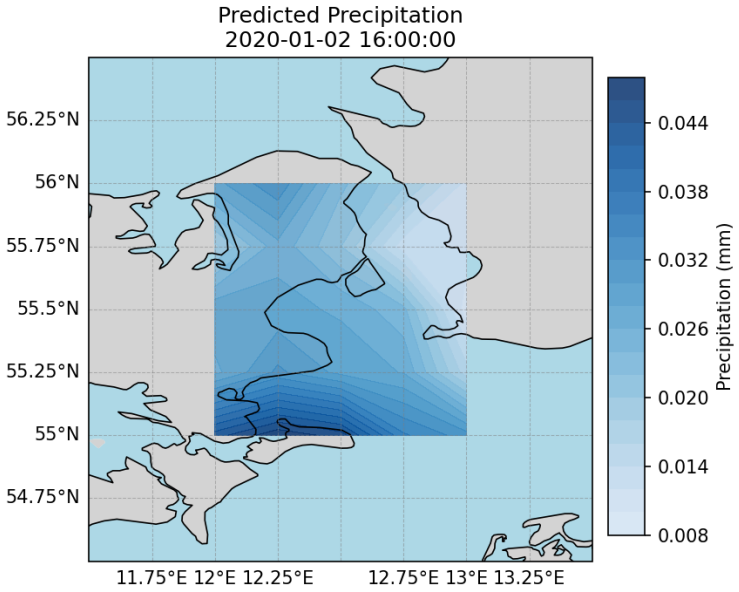


Prediction target: "tp"
Prediction horizon: 6h x grid
Lag: (12 (consecutive) x (7 features) x grid

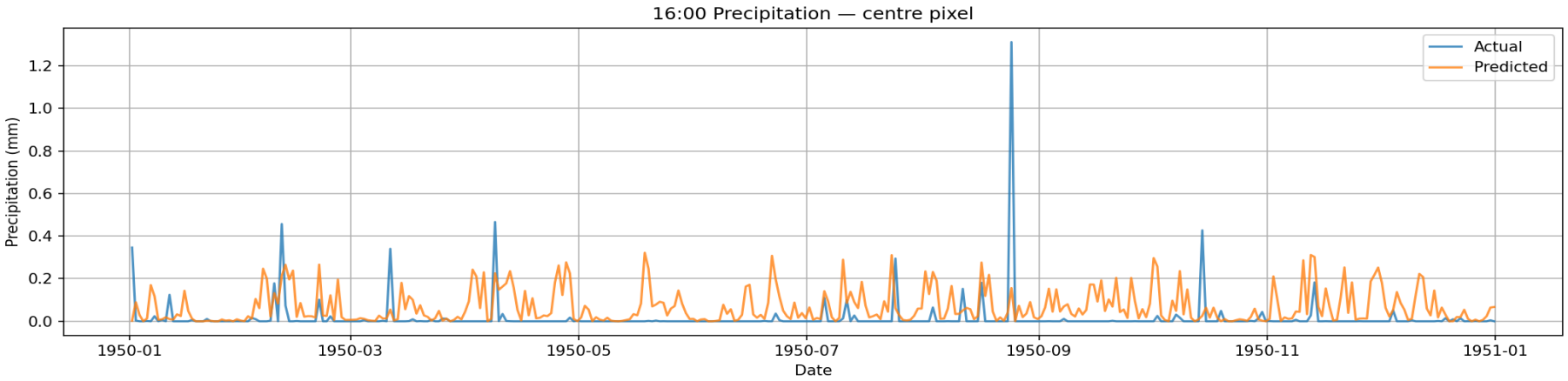
01.01.1950

January
2020

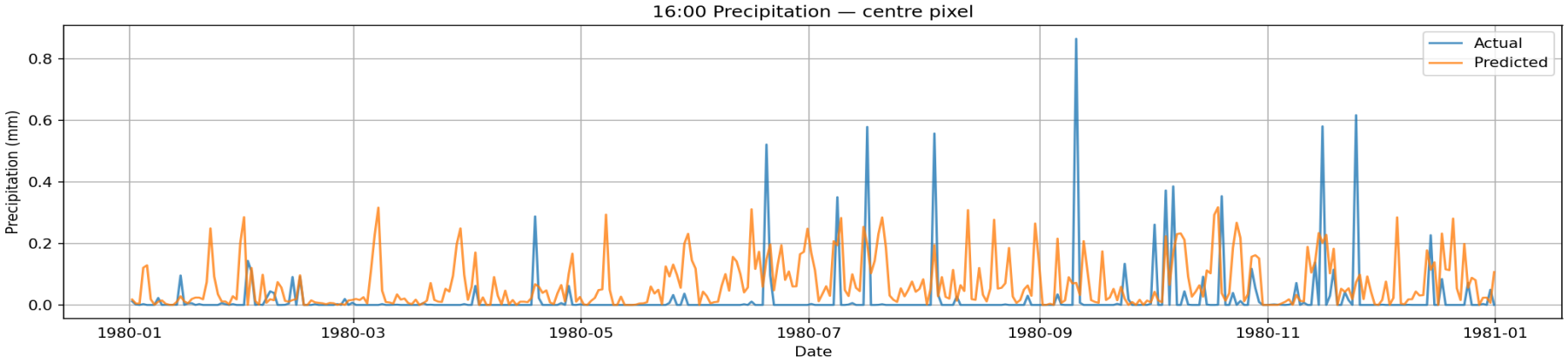




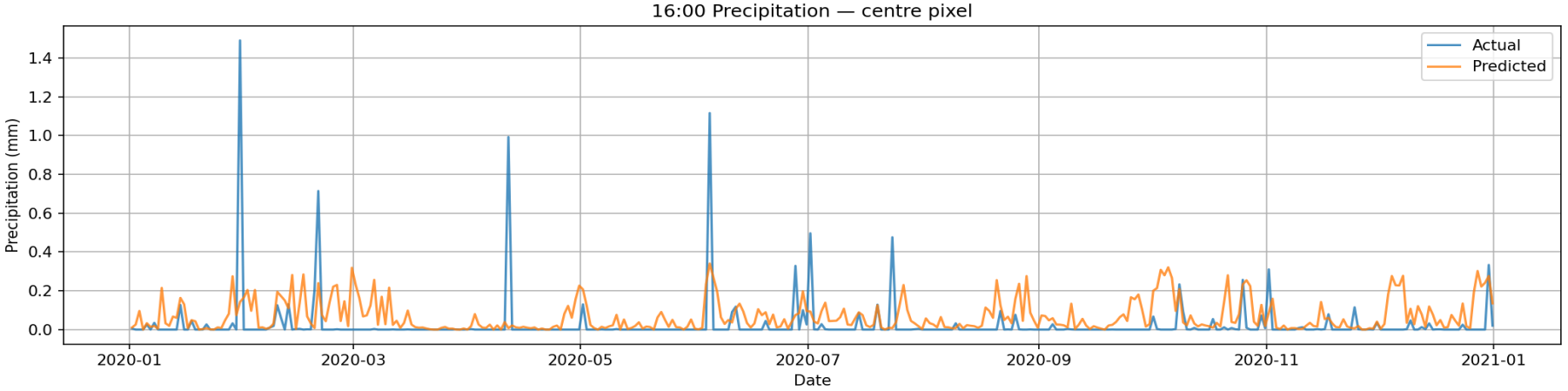
02.01.1950



1950



1980



2020

How to continue:

Tune loss function more and decade specific!

What is even important when trying to predict weather??

- Feature importance
- Lag importance
- Seasonality
- Horzion choice vs. accuracy
- Time range used for training?

Thanks!

QR-Code
to our
github repo

https://github.com/CarlaMailin/Predicting-the-Future---Machine-Learning-Weather-Prediction-Model/tree/main

Contribution Distribution

- Katerina: XGboost (Big Grid)
  - Trained on 3 years (summer+spring) of data for predicting 1 hour ahead
  - Compared with different lag times
- Peter: CNN GRU/LSTM (small Grid)
  - Trained on seasons and predicted for different seasons for different timesections
  - fetch and convert Data
  - Meeting Coordination
- Vasilis: XGBoost, LSTM, GRU (single Grid Poinr)
  - Trained on 66 years of data for predicting 1 hour ahead
  - Feauture Importance
- Mailin: CNN LSTM Model (small Grid)
  - Trained on 11.5 years of data for forecasting 6 hours ahead
  - Compared with mulitiple decades

Appendix

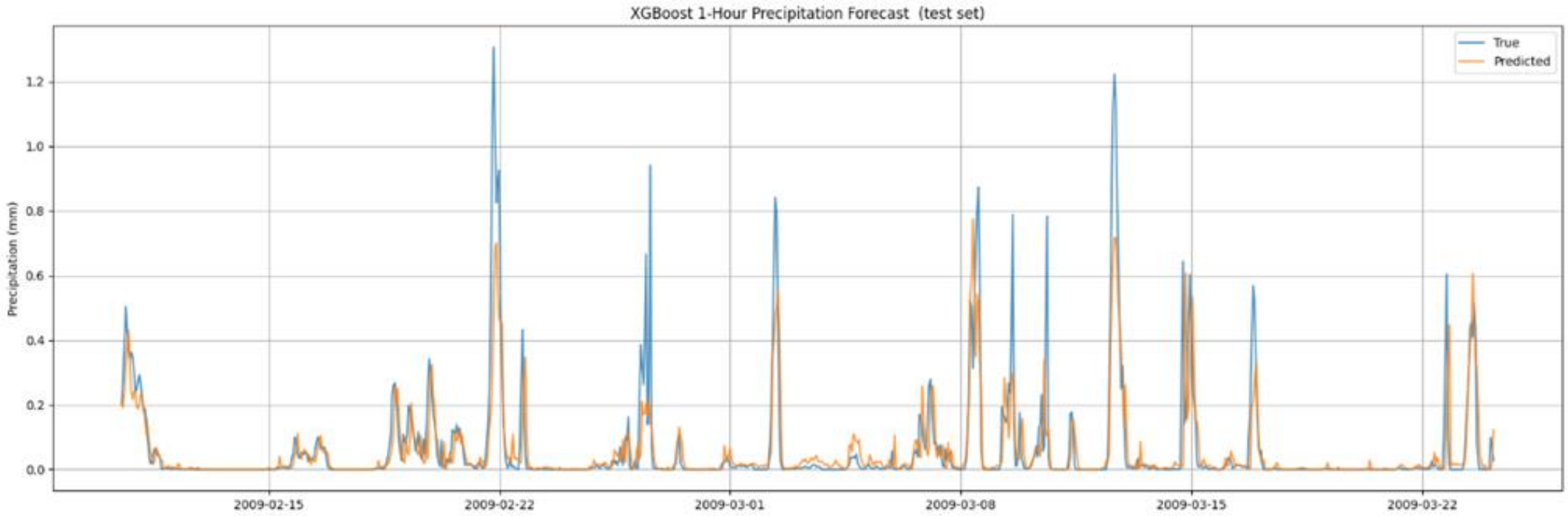
Spatial Precipitation Forecasting over Northern Europe Using XGBoost

- **Coordinates:** North: 55°, South: 50°, West: 0°, East: 15°
- **Time Period:** Hourly data from March to September for the years 2023–2025
- **Method Used:** XGBoost Regressor
- **Hyperparameters:** Number of trees (`n_estimators`): 500, Learning rate: 0.03, Maximum tree depth: 6, Subsample ratio: 0.8, Column sampling, ratio (`colsample_bytree`): 0.8, Random seed: 42
- **Data Split:** Training Set->70%, Validation Set->15%, Test Set->15%
- **Forecasting Horizon:** 1 hour into the future
- **Sample Weighting:** 1+1000×precipitation
- **Training Strategy:** Undersampling of dry events

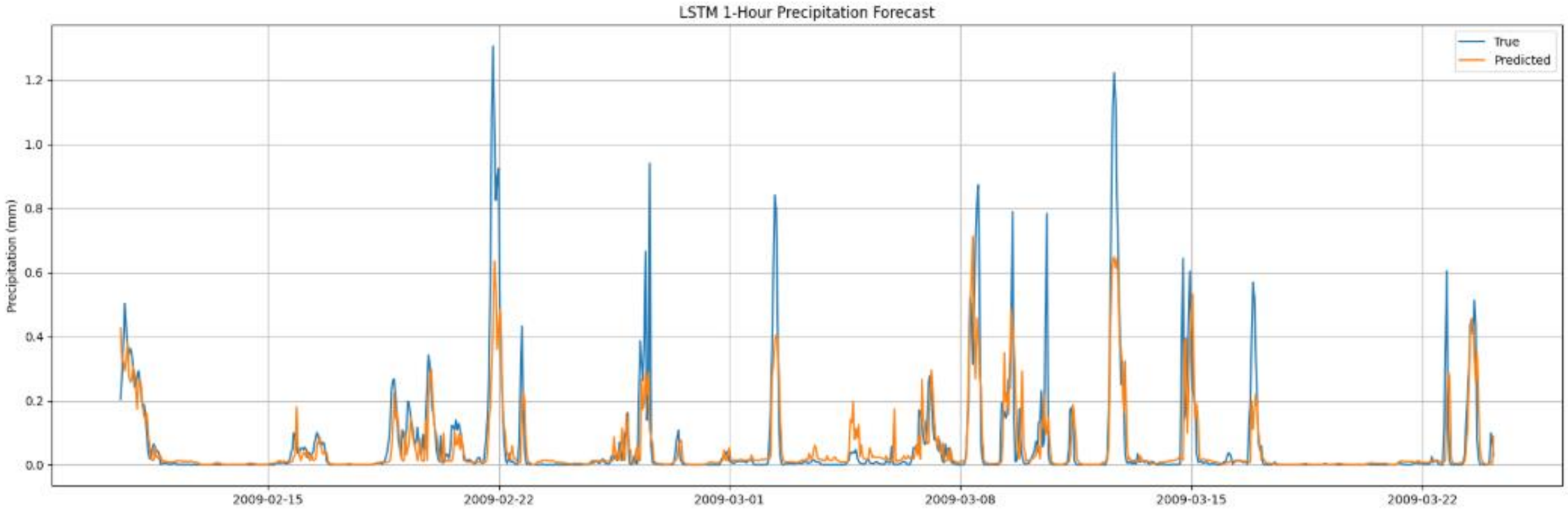
Single Point Precipitation Forecasting

- **Objective**: predict future precipitation using historical weather observations.
- **Dataset**: hourly meteorological data from 1940–2026.
- **Models**: XGBoost, LSTM, GRU
- 1-hour into the future prediction
- Rain/No Rain Classification
- Feature Importance

XGBoost 1-hour Precipitation Forecast

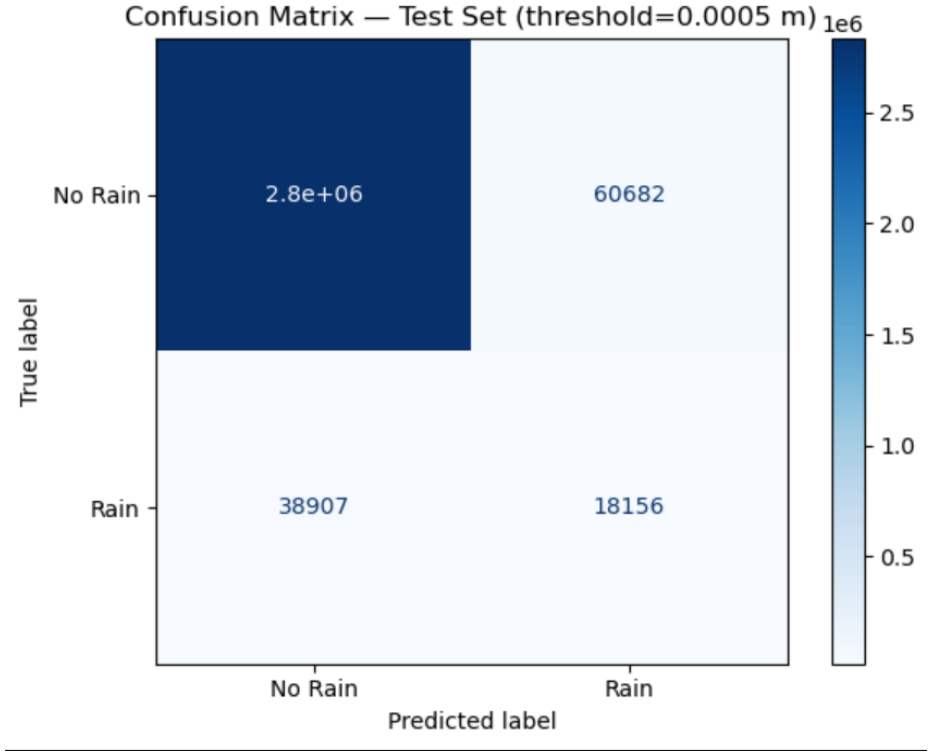


LSTM 1-hour Precipitation Forecast



XGBoost Big grid Performance

Metric	Result
RMSE	0.197 mm
MAE	0.079 mm
Bias	0.042 mm
FAR	0.781
CSI	0.204



- **Correlation:** Agreement in variability between predictions and observations.
- **Bias:** Average overestimation or underestimation.
- **FAR** (false alarm ratio): Fraction of false rainfall predictions.
- **CSI** (Critical Success Index): Overall rainfall event detection accuracy.

Prediction target: "tp"
Prediction horizon: 1h
Lag: (72, 48, 24, 12, 6, 3, 1h) x (6 features) x applied at each grid point

Data Processing and Training Strategy (XGBoost big grid model)

Precipitation Processing

- ERA5 precipitation (tp) is provided as accumulated precipitation within each forecast cycle.
- Hourly precipitation was derived by differencing consecutive forecast steps.
- Negative values were clipped to zero.

Prediction target: "tp"
Prediction horizon: 1h
Lag: (72, 48, 24, 12, 6, 3, 1h) x (6 features) x applied at each grid point

```python
df_list = []

for f in files:
    datasets = cfgrib.open_datasets(f)
    ds_features = datasets[0]
    ds_tp       = datasets[1]

    # Features
    df_features = ds_features.to_dataframe().reset_index()
    df_features = df_features[["time","latitude","longitude","sp","tcc","u10","v10","t2m"]]
    df_features = df_features.rename(columns={"time":"valid_time"})

    # Precipitation
    tp = ds_tp["tp"]
    tp_deacc = tp.diff("time").clip(min=0)
    tp_df = tp.to_series().reset_index()
    tp_df["valid_time"] = tp_df["time"] + tp_df["step"]
    tp_df["tp_hourly"] = tp_df.groupby(["time","latitude","longitude"])["tp"].diff()
    tp_df["tp_hourly"] = tp_df["tp_hourly"].fillna(tp_df["tp"]).clip(lower=0)
    tp_df = tp_df[["valid_time","latitude","longitude","tp_hourly"]].sort_values("valid_time")

    # Merge features + tp
    df = pd.merge(df_features, tp_df, on=["valid_time","latitude","longitude"], how="inner")
    df = df.rename(columns={"tp_hourly":"tp"})

    df_list.append(df)
```

Data Processing and Training Strategy (XGBoost Big grid model)

Prediction target: "tp"
Prediction horizon: 1h
Lag: (72, 48, 24, 12, 6, 3, 1h) x (6 features)
applied at each grid point

```
df_all["tp_target"] = df_all.groupby(["latitude","longitude"])["tp"].shift(-1)
df_all = df_all.dropna().copy()  # drop rows with NaNs due to lags/rolling/target

# ========================
# 6 Create X, y, weights
# ========================
y_all = np.log1p(df_all["tp_target"])
X_all = df_all.drop(columns=["valid_time", "tp_target"])
X_all = X_all[[col for col in X_all.columns if not col.startswith("tp_target")]]

weights_all = 1 + 100*np.expm1(y_all)
```

Target Transformation

- The prediction target **y=log(1+tp).**

  Reduces the skewness of precipitation data

Sample Weighting

- Rainfall events were assigned higher importance during training:

  w=1+1000·tp

  Encourages the model to focus on rainfall intensity .

Data Processing and Training Strategy (XGBoost Big grid model)

Training Data Balancing

- **All rainfall** samples were retained.
- A subset of dry samples was selected through **undersampling**.
- Reduces the dominance of **non-rainfall** observations in the training set.

Prediction target: "tp"
Prediction horizon: 1h
Lag: (72, 48, 24, 12, 6, 3, 1h) x(6features)
applied at each grid point

```
#  Undersampling
# ========================
rain_mask = np.expm1(y_train_full) >= 0.0001
dry_mask  = ~rain_mask

rain_idx = np.where(rain_mask)[0]
dry_idx  = np.where(dry_mask)[0]

# keep all the rain + 3x dry samples
np.random.seed(42)
keep_dry = np.random.choice(dry_idx, size=len(rain_idx)*10, replace=False)
keep_idx = np.sort(np.concatenate([rain_idx, keep_dry]))

X_train = X_train_full.iloc[keep_idx]
y_train = y_train_full.iloc[keep_idx]
weights_train = weights_full.iloc[keep_idx]

print(f"Original train size:  {len(X_train_full)}")
print(f"Balanced train size:  {len(X_train)}")
print(f"Rain %: {(np.expm1(y_train)>=0.0001).mean()*100:.1f}%")
```

CNN-LSTM Model for Grid Region Sjælland

- Combines CNN (spatial patterns) and LSTM (temporal memory)
  - LSTM gates applied as convolutions, not matrix multiplications
  - Feature maps passed through gates
  - Preservation of spatial structure
- Spare rain
  - Loss function used to tune model's rain prediction
- Long tail of rain distribution
  - Model learns in log space
- Rain is deaccumulated -> only looking at rain at that point in time

CNN-LSTM GRID Model

- Data processing
  - ERA5 forecast initialized every 12 hours
  - Precipitation and solar radiataion accumulate over this time (12 time steps)
    - For this model data is deaccumulated by differentiating so each time step
    - flattened into a single continuous time series
- ERA5 preciptation in m -> before training converted to mm
- Hyperparameters: hidden size = 32, learning rate= 1e-4, dropout = 0.3
  - Early stopping with patience = 5
- Right skewedness of distribution -> rain amounts vary
  - Model trained in log space
- Rain events are rare -> weighted loss function used

CNN LSTM: Model Parameters

Timesteps for window

How far do we look in the future

Number of windows processed together

What do we count as rain

CNN-LSTM: Rain loss function

Makes rain problem binary

no rain is penalized more

Errors on rainy pixels have more weight

Punish outliers

```python
def rain_loss(prob_logits, amount, target):
    rain_mask = (target > RAIN_THRESH_LOG).float()
    pos_w = torch.tensor(BCE_POS_WEIGHT, device=prob_logits.device)
    bce   = nn.functional.binary_cross_entropy_with_logits(
                prob_logits, rain_mask, pos_weight=pos_w)
    weight   = 1.0 + RAIN_WEIGHT * rain_mask
    mse      = (weight * (amount - target) ** 2).mean()
    mae      = (weight * torch.abs(amount - target)).mean()
    amt_loss = 0.7 * mse + 0.3 * mae

    return bce + amt_loss, bce, amt_loss
```

Grid CNN-LSTM: Oversampling Rain Events

- Since rain events are sparse
  - IDEA: oversample them and model is better trained at predicting them
- After comparison of evaluation metrics
  - Oversampling does not catch more rain events
    - just creates additional false ones

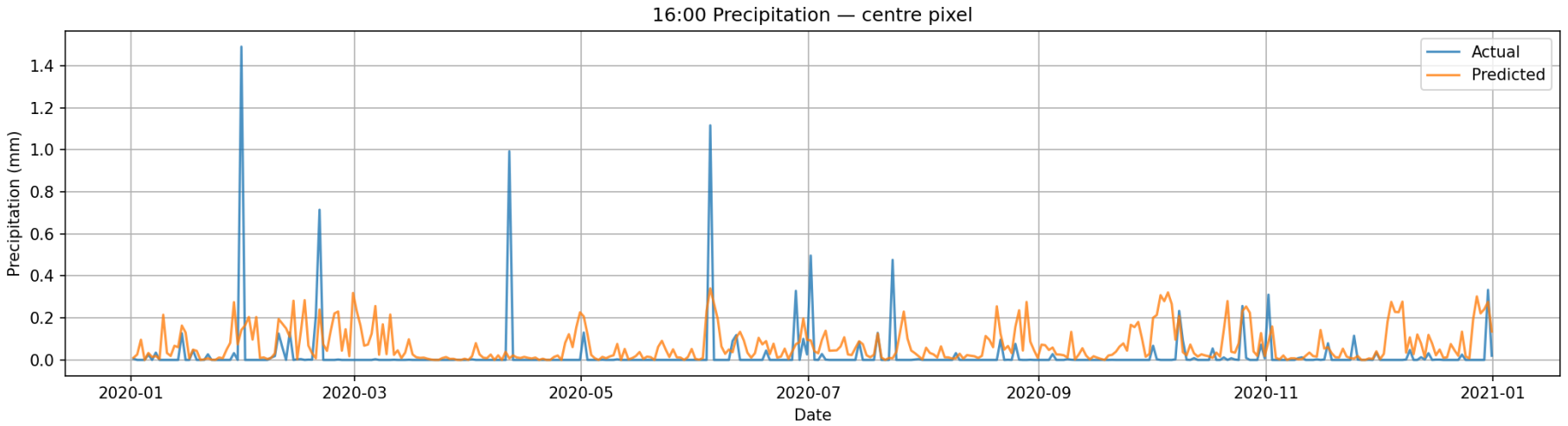
Comparison Oversampling Rain Events

-POD: fraction of rain events caught

-FAR: fraction of rain alarms that are false

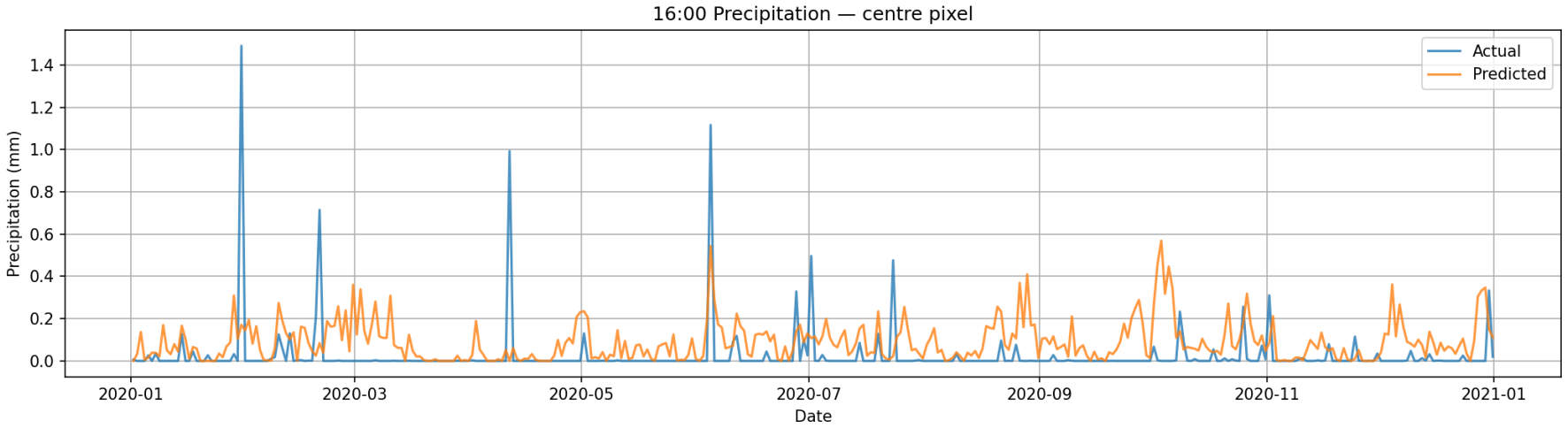
1x

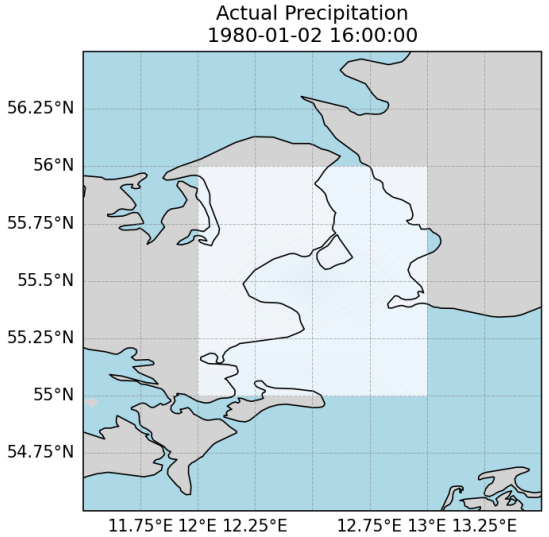
Metric	Result
RMSE	0.143 mm
MAE	0.074 mm
FAR	0.851
POD	0.65

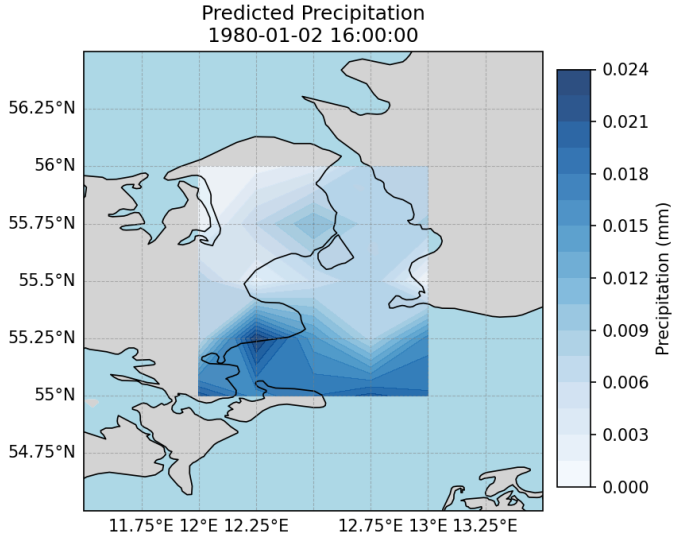


2x

Metric	Result
RMSE	0.159 mm
MAE	0.095 mm
FAR	0.898
POD	0.65





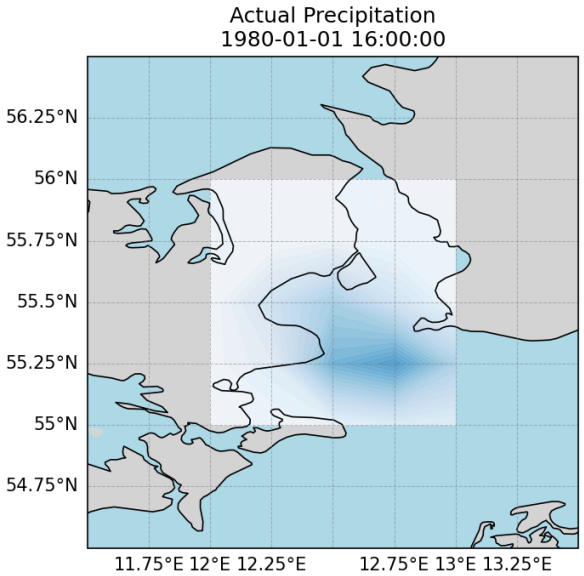


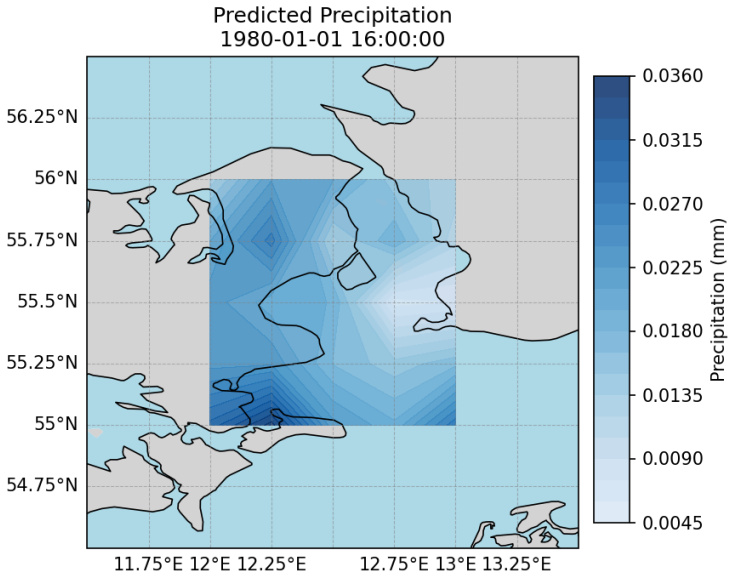
Prediction target: "tp"
Prediction horizon: 6h x grid
Lag: (12) x (7 features) x grid

CNN-LSTM Grid Model

01.01.1950

January
1980

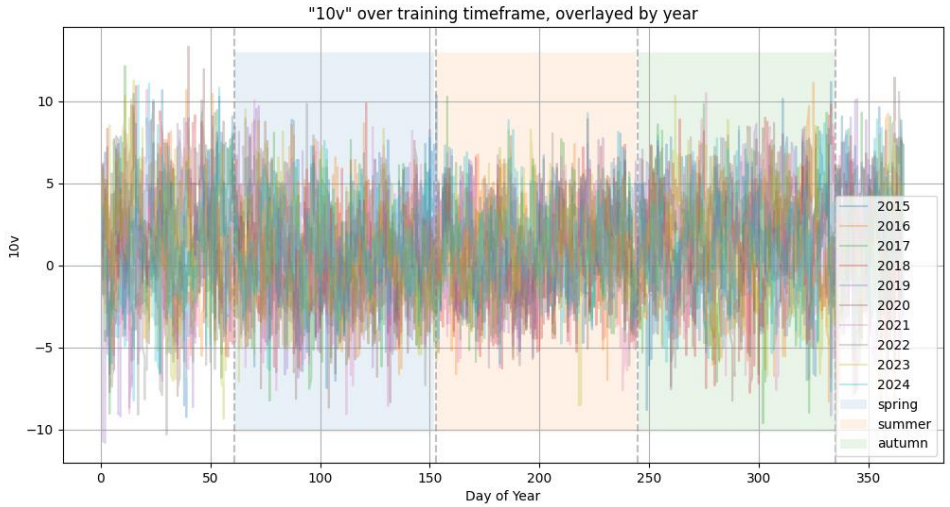




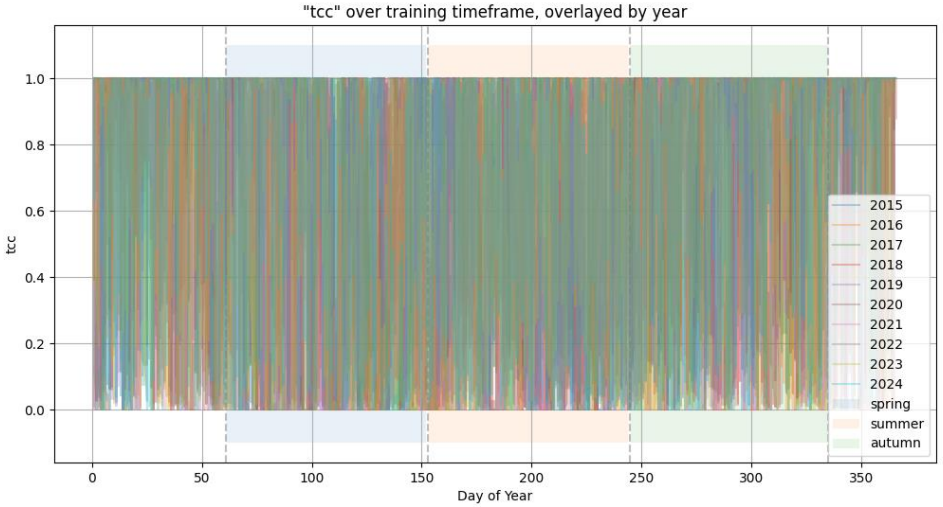
02.01.1950

Every season is different...

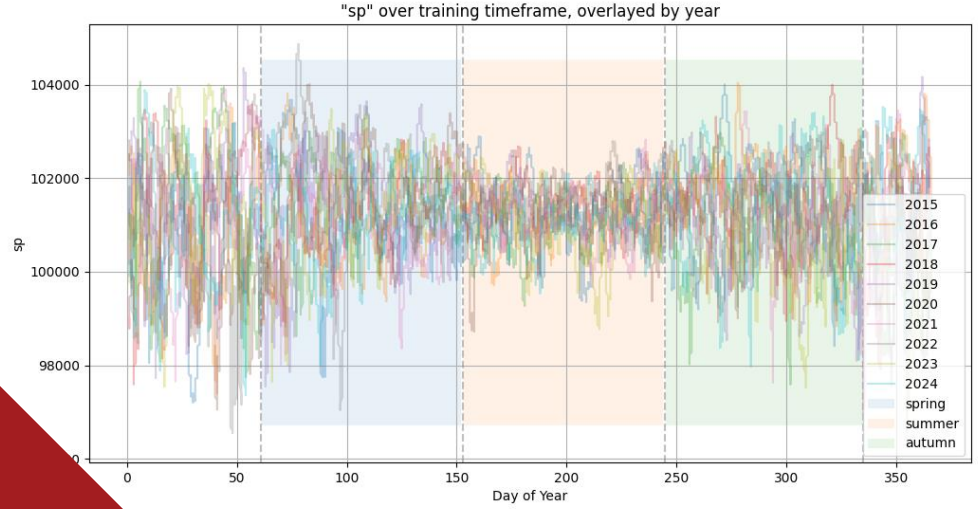
"10v": wind



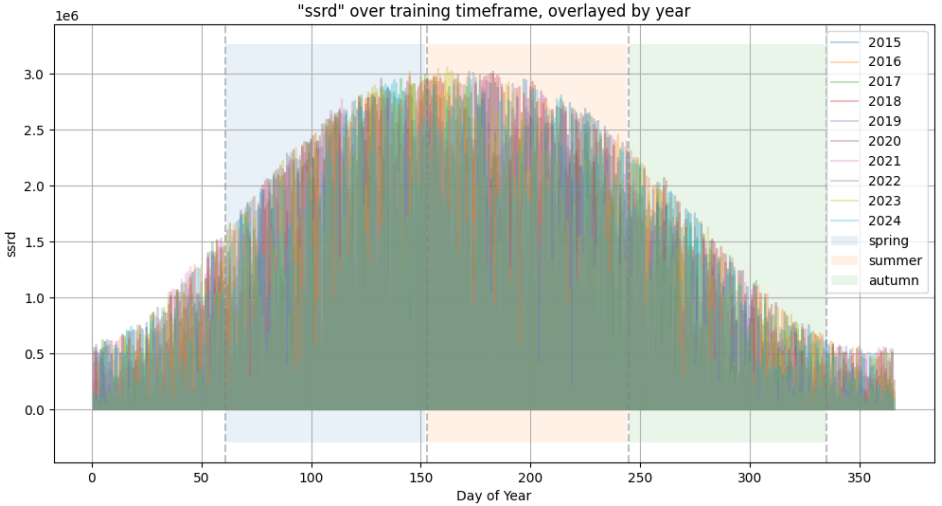
"tcc": Total cloud cover



"sp": surface pressure

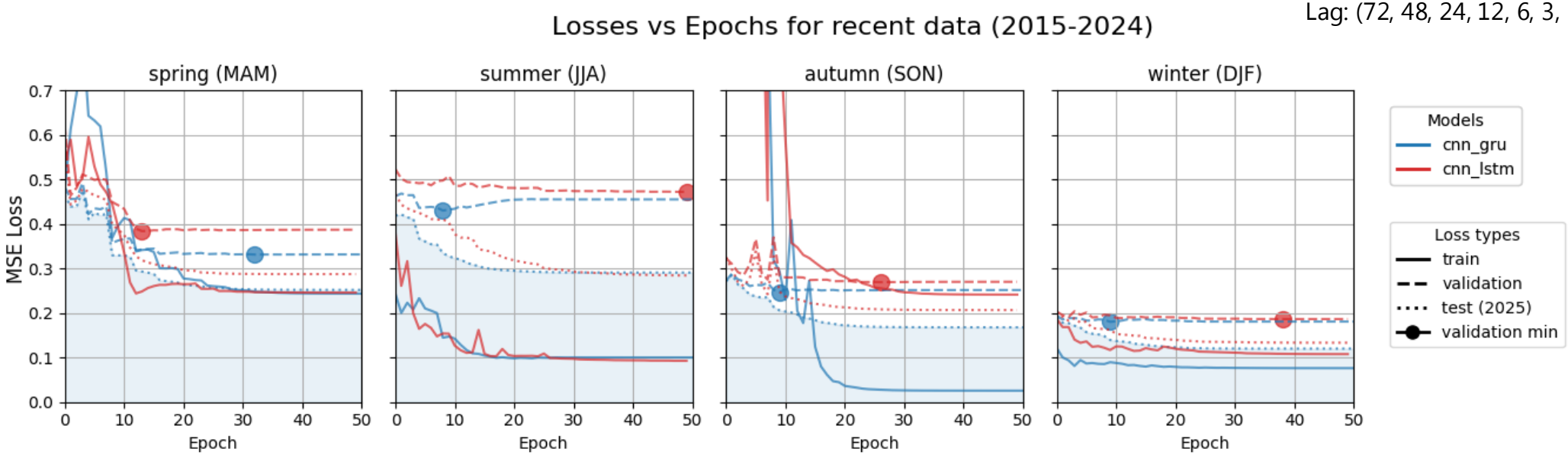


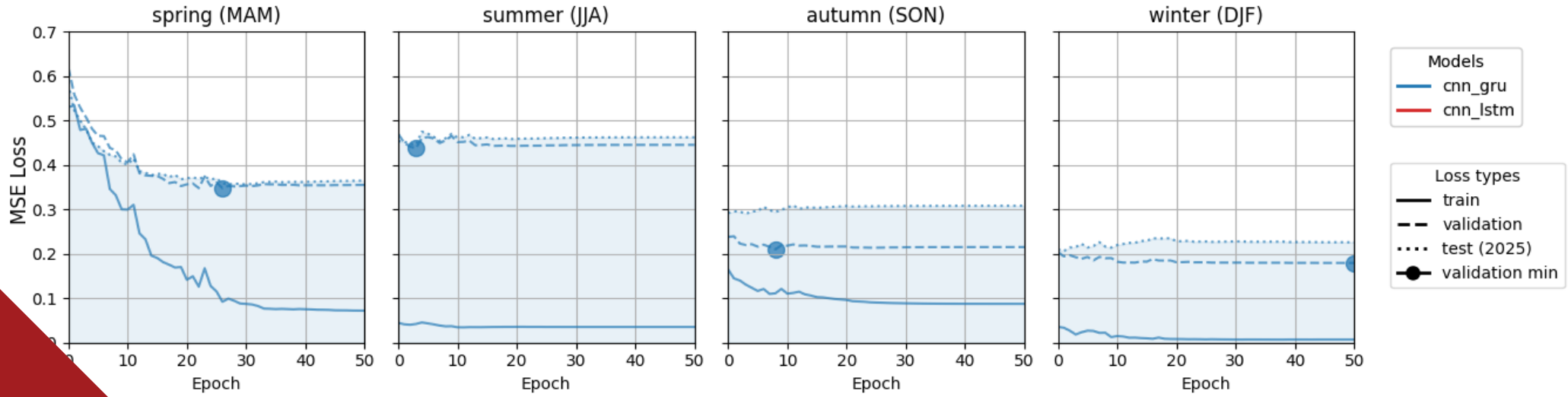
"ssrd": solar surface radiation downwards



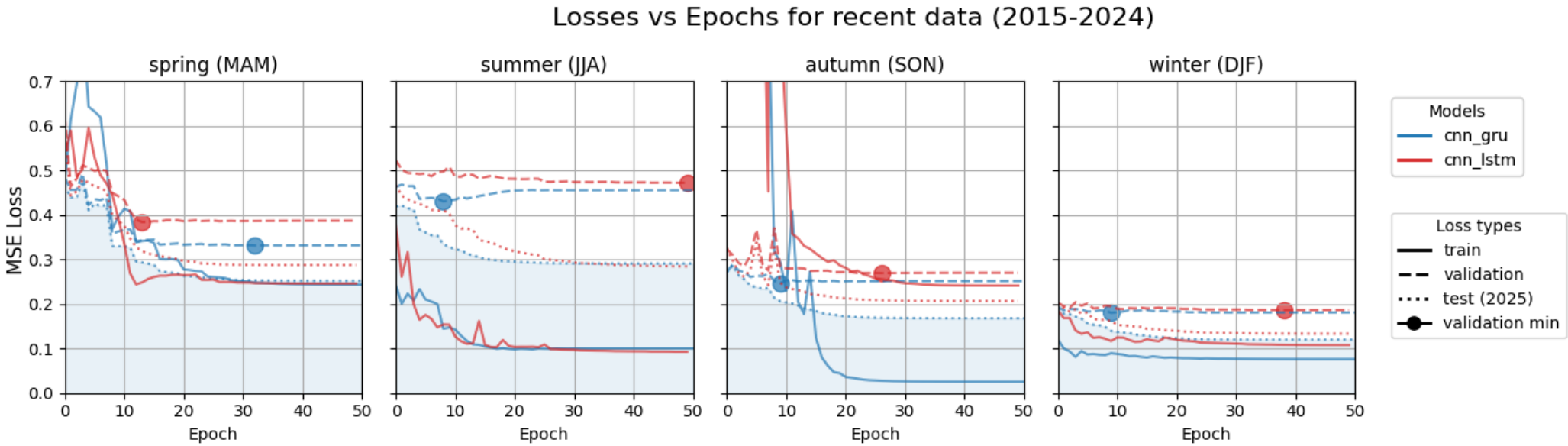
Training on past seasons

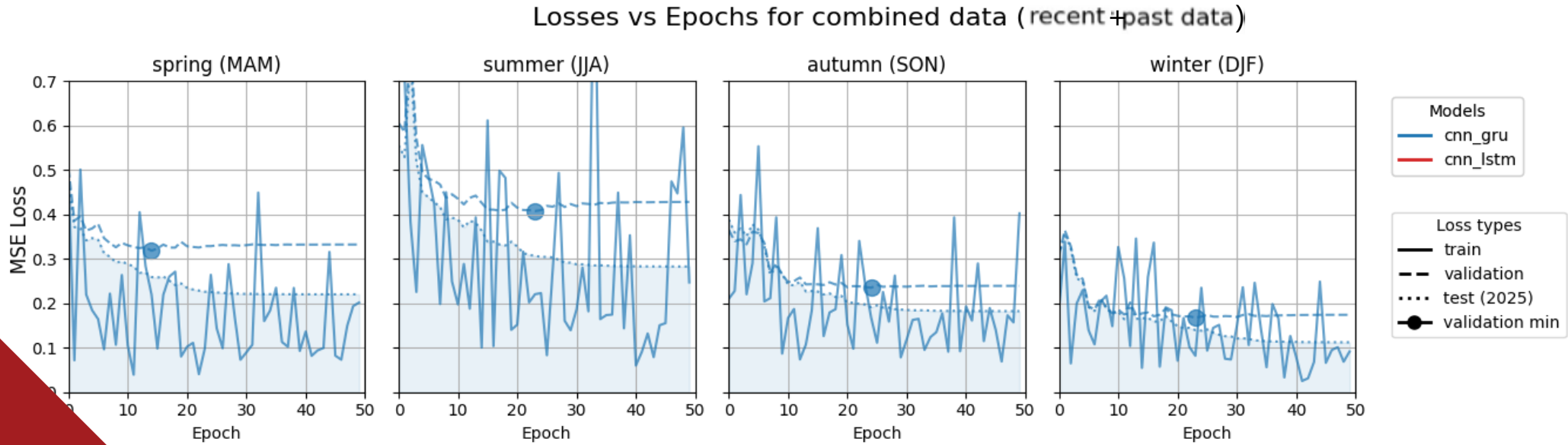
Prediction target: "tp"
Prediction horizon: 1h x grid
Lag: (72, 48, 24, 12, 6, 3, 1h) x (7 features) x grid



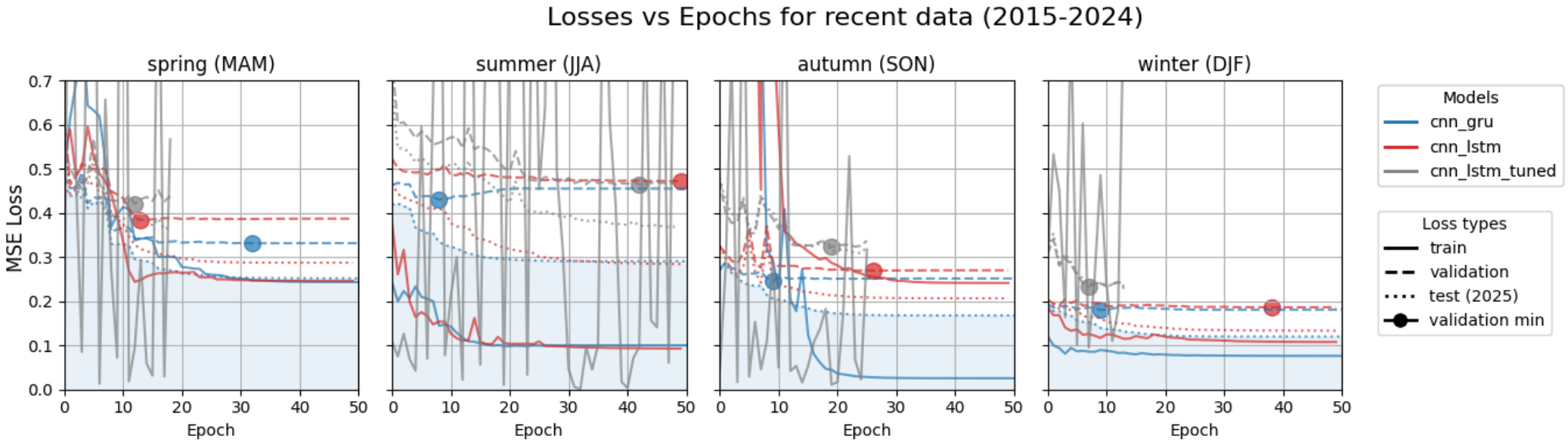


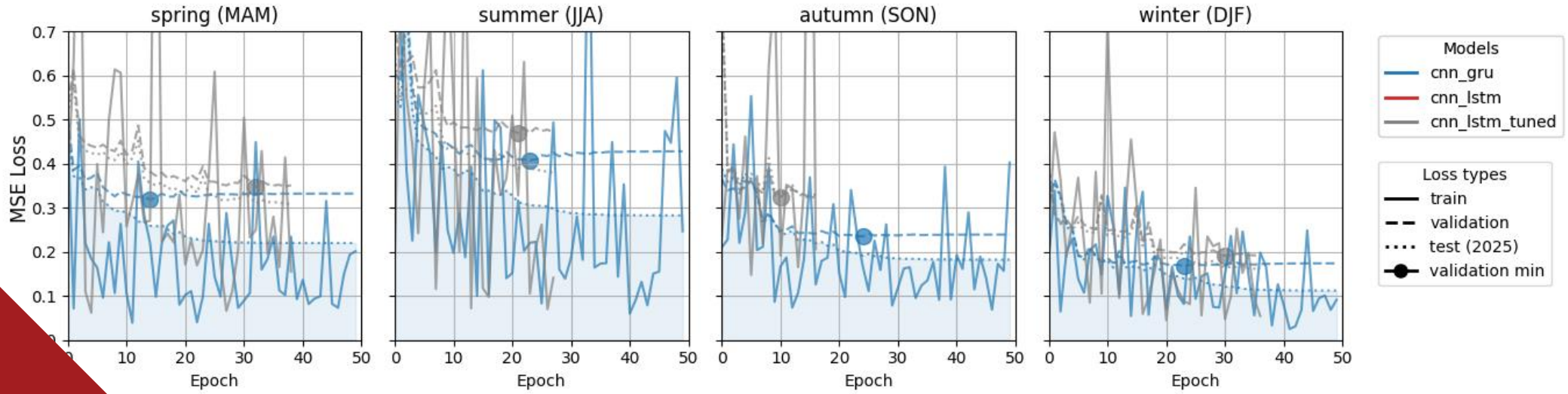
Training on more data





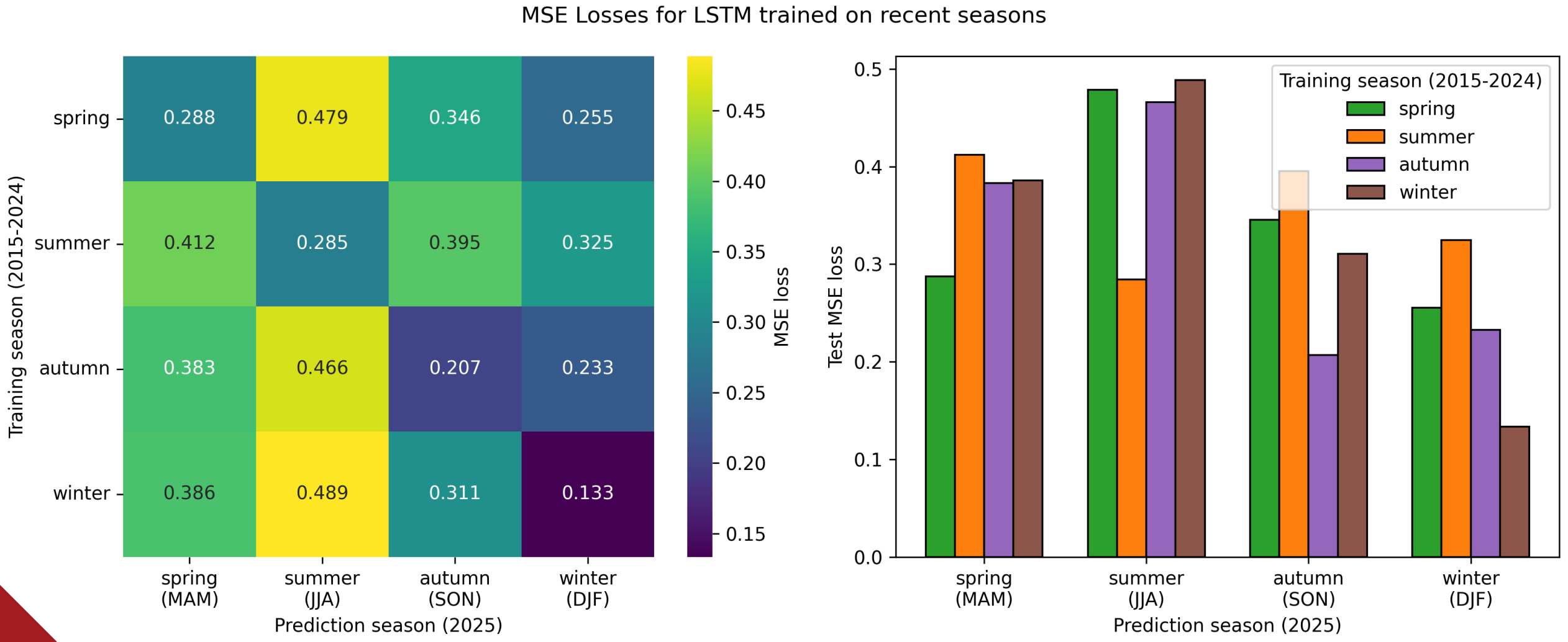
Hyperparametertuned models





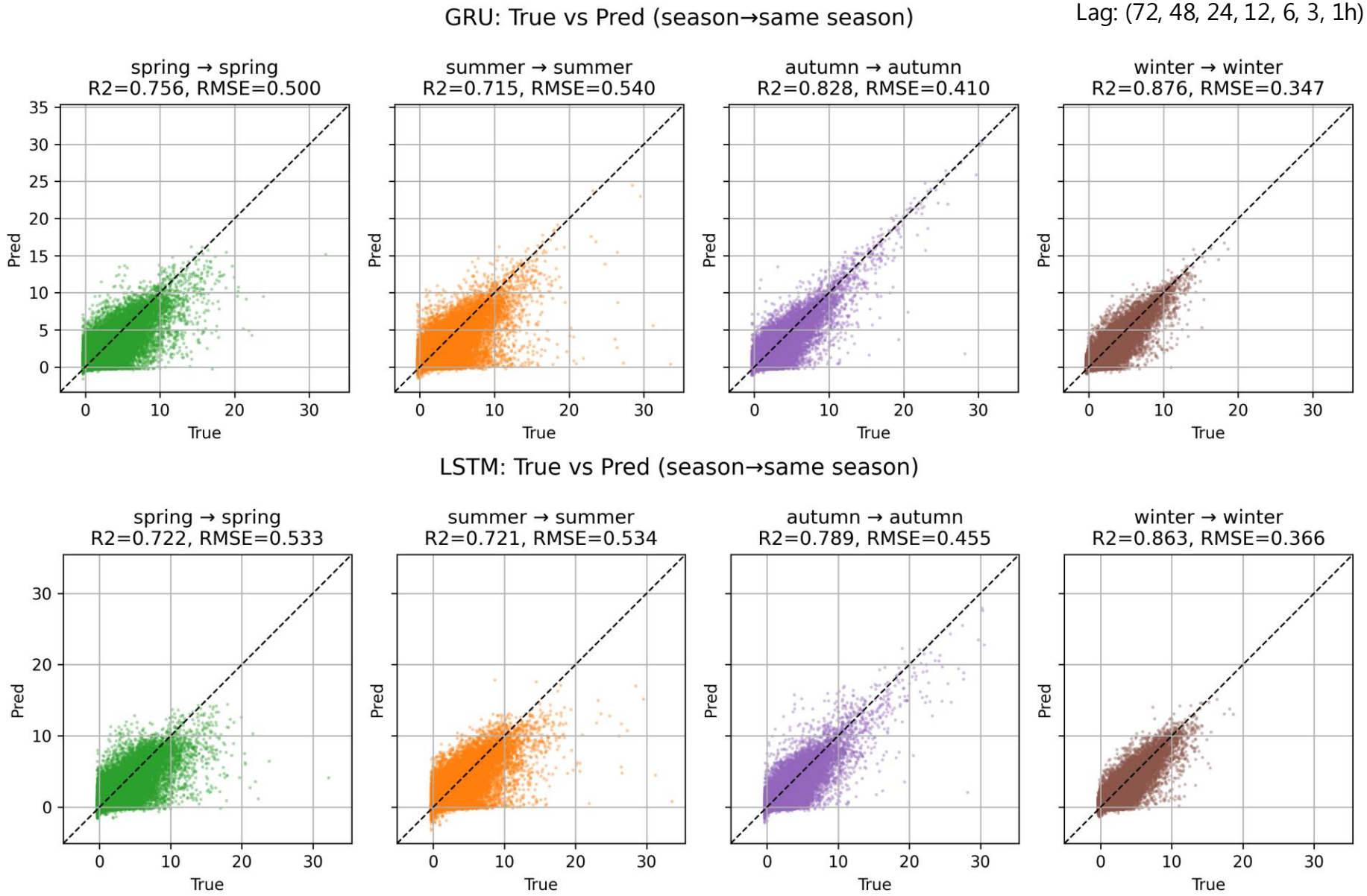
Loss Matrix: Prediction season vs. Training season

"tp": total precipitation



Regression: same Prediction & Training season

Prediction target: "tp"
Prediction horizon: 1h x grid
Lag: (72, 48, 24, 12, 6, 3, 1h) x (7 features) x grid



Preprocessing data for seasonal models

dataloader_generation_local.ipynb

1. Loading raw .nc data for each year
2. Concatenate years and split in seasons
3. Feeding data in Pipeline object to generate dataloaders scaled on train

-> dataloaders for CNN-GRU/LSTM models

Save them as

- Past (1950-2010 every 10 years) train, val
- Recent (2015-2024) train, val
- Combined (1950-2024) train, val
- Now (2025) - test

my_dataloader_module.py

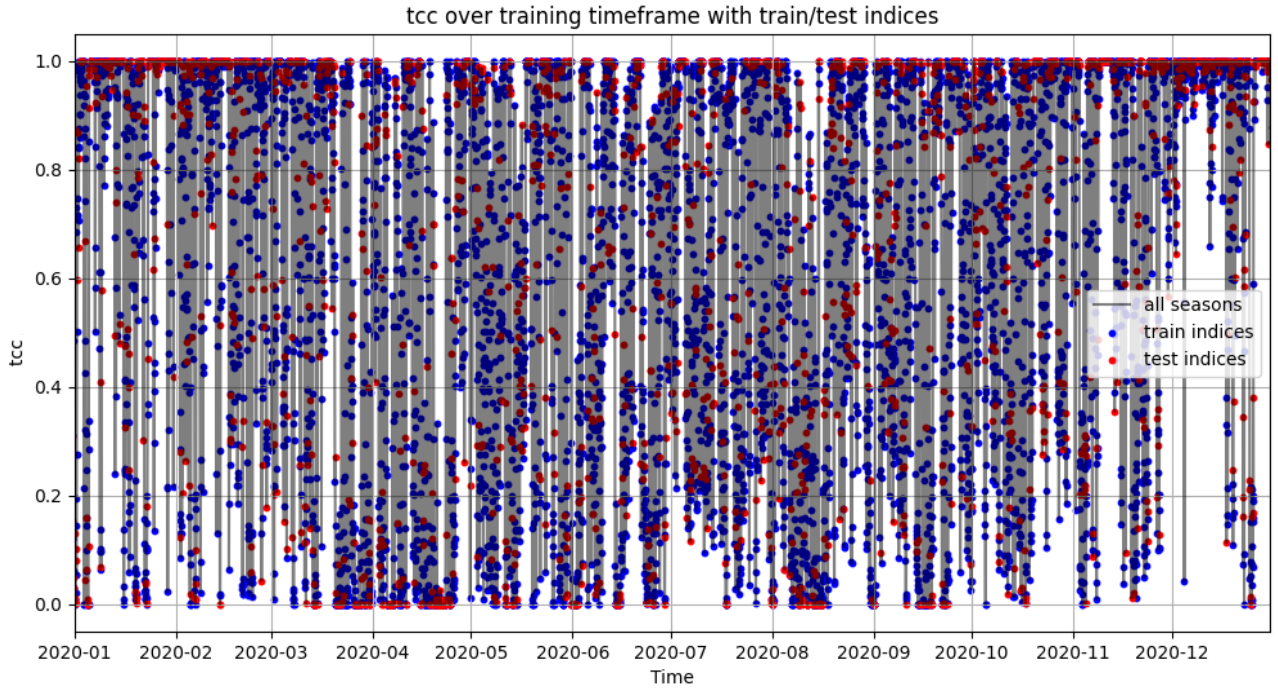
Pipeline for train+val:

1. Loading data
2. bring in shape for lag processing (time, var, lat,lon)
3. Create shuffled train,val idx list for usable indices (enough data available for lag+pred)
4. Window method for lag generation
5. Split in X, y for train and val
6. Scale with train scaler
7. Get dataloaders + used scalers + list of skipped idx

Pipeline for test:

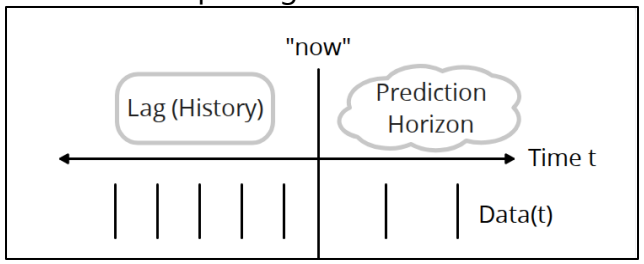
3. Create idx list for usable indices (enough data available for lag+pred)

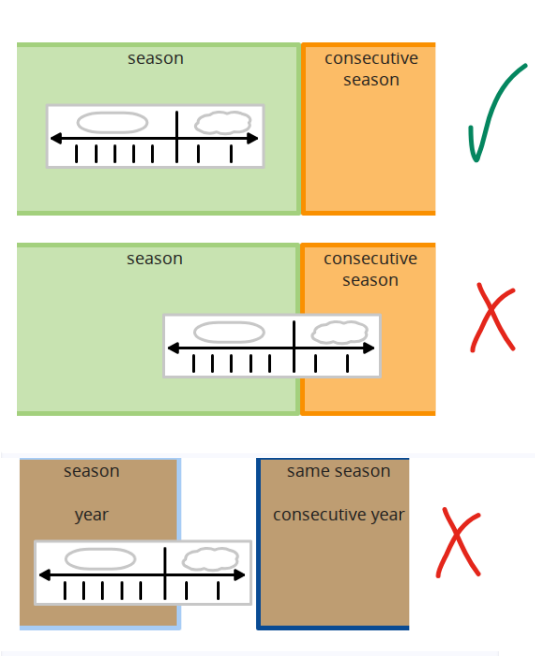
Preprocessing and splitting training and validation



Plot of used indices colored in blue for training-forecast-package and red for validation-forecast-package over one year – no seasonal breaks introduces here

Forecast data package

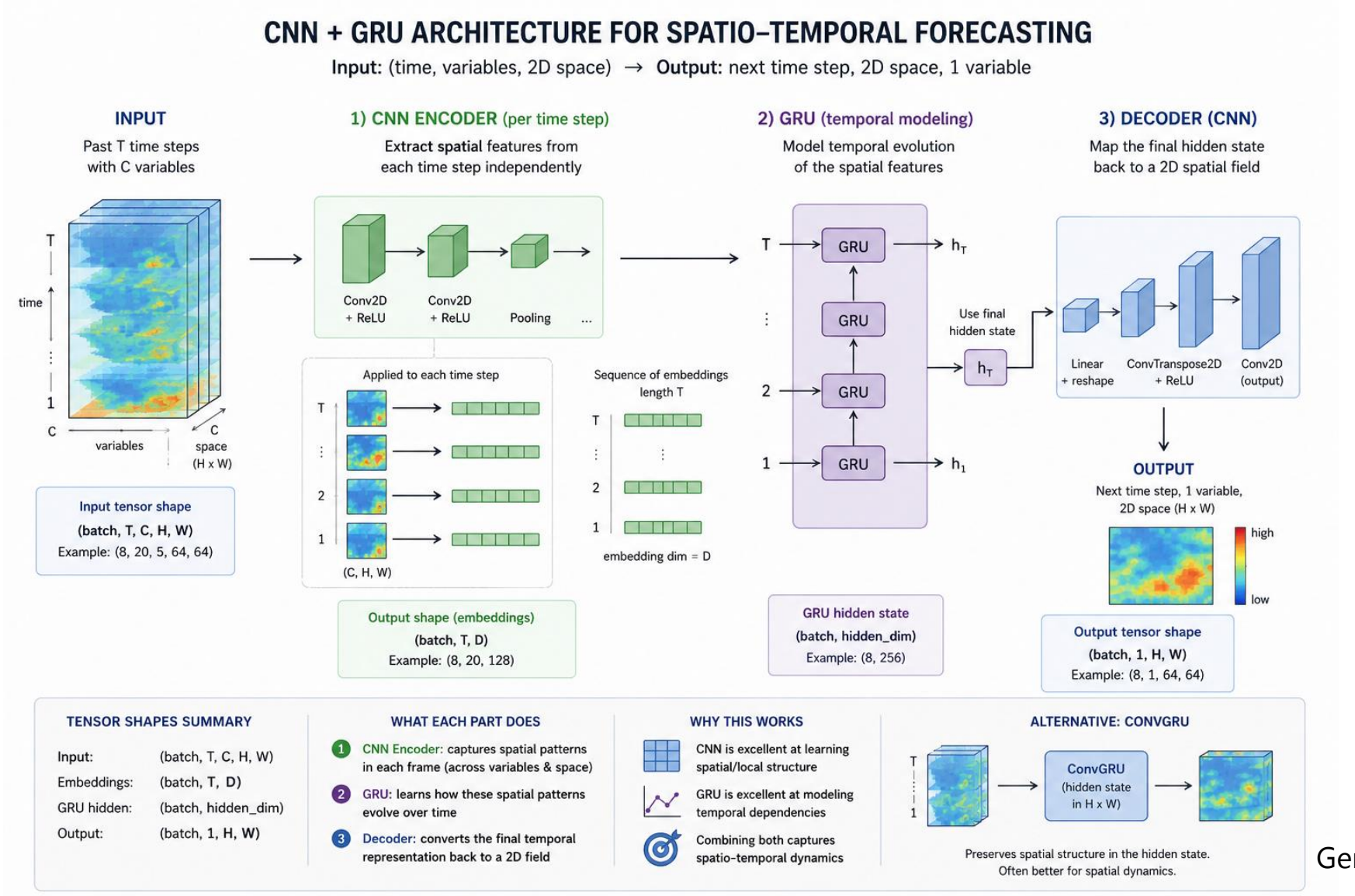




Datasets were split seasonally and saved, so that some timepoints could not be used as prediction data packages

Fx. for the "recent" train dataloader, there were 603 idx skipped for winter due to the both rules above

Predicting 2D Timeseries: CNN+GRU/LSTM



Generated by ChatGPT

Our **(main)** takeaways

- **Meetings** – enter them with an aim and go out with a goal until next time
- **GitHub** – learn it as a whole group in the beginning to prevent frustation and dataloss
- Get your **data** right – really… look at it in the beginning
- **Use colab** for training your models with code prepared locally
- …