rainymotion &

RainNet

optical flow and deep learning models

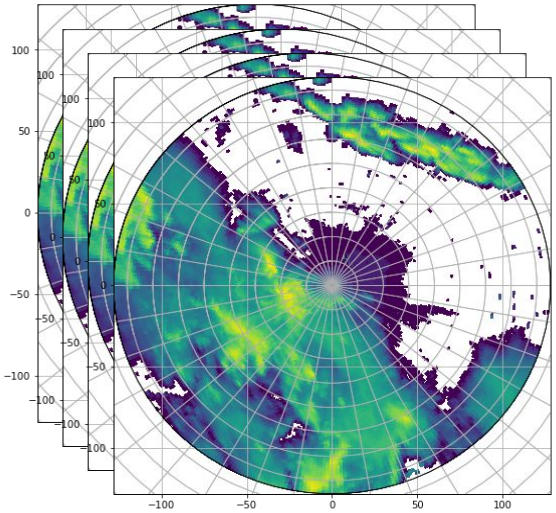
for radar-based precipitation nowcasting

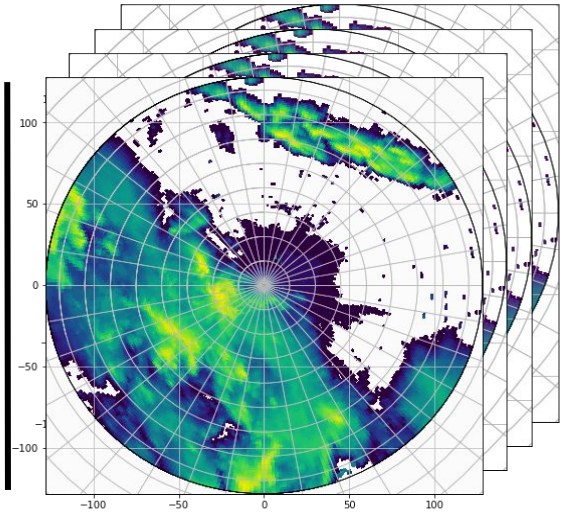
DWD

Georgy Ayzel, Maik Heistermann, Tanja Winterrath,

Niels Landwehr, Tobias Scheffer

Nowcasting





Model

0

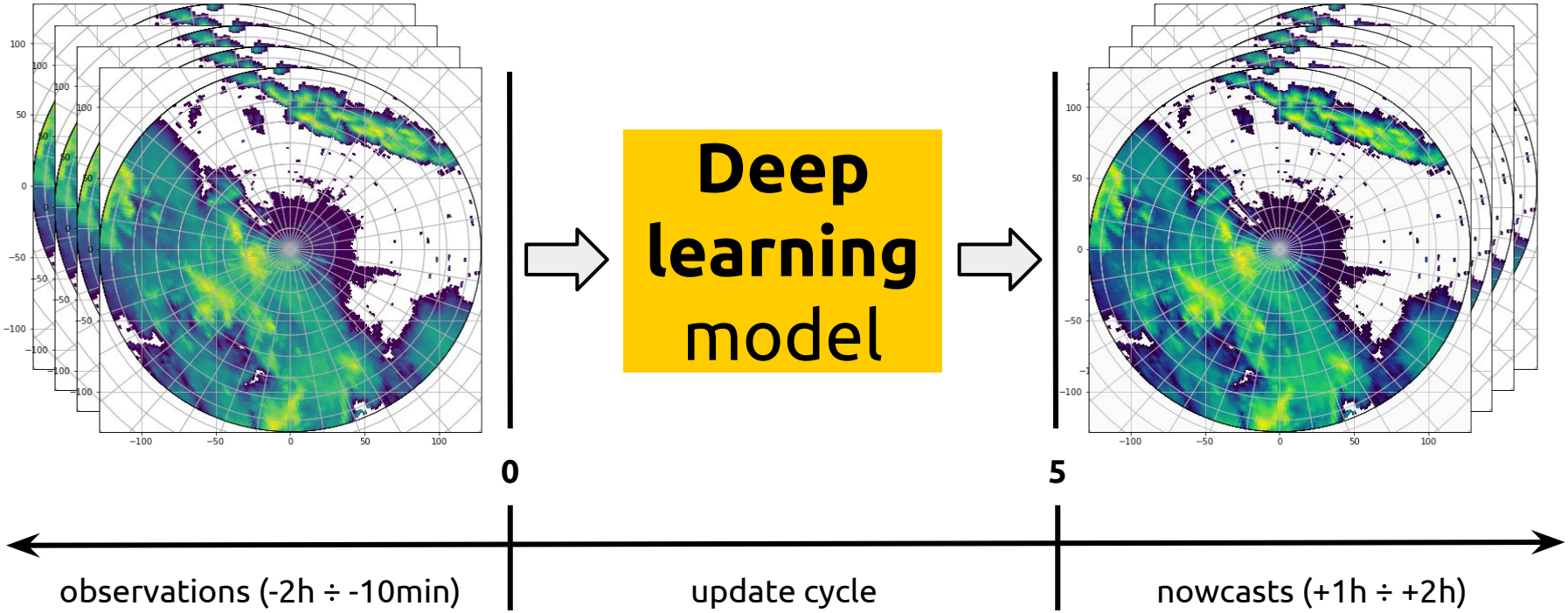
5

observations (-2h ÷ -10min)

update cycle

nowcasts (+1h ÷ +2h)

Research topic



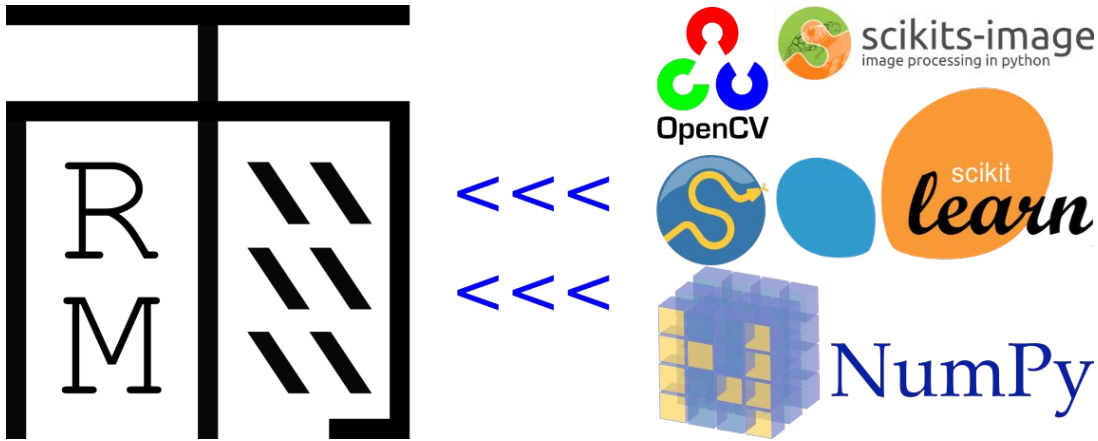
Baseline

- Persistence, … ?

No open tool for radar-based precipitation nowcasting

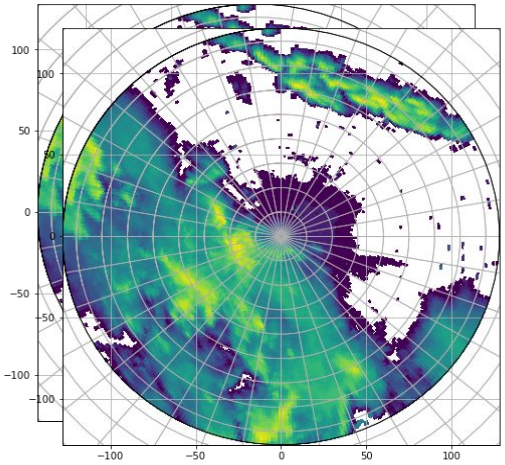
- Clear gap to fill

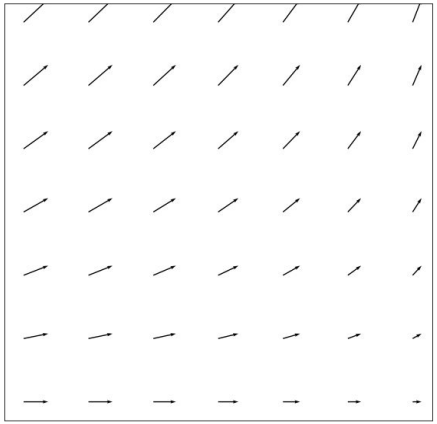
rainymotion



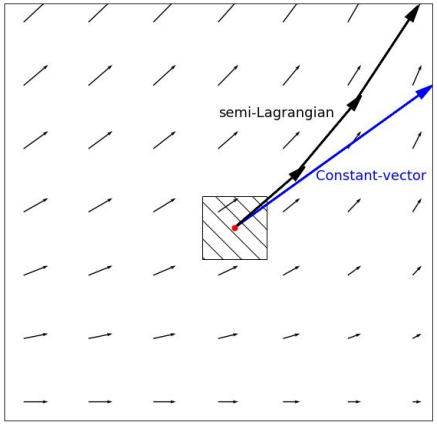
Idea: don’t reinvent wheels, *just* combine them!

Conventional approach





Tracking



Extrapolation

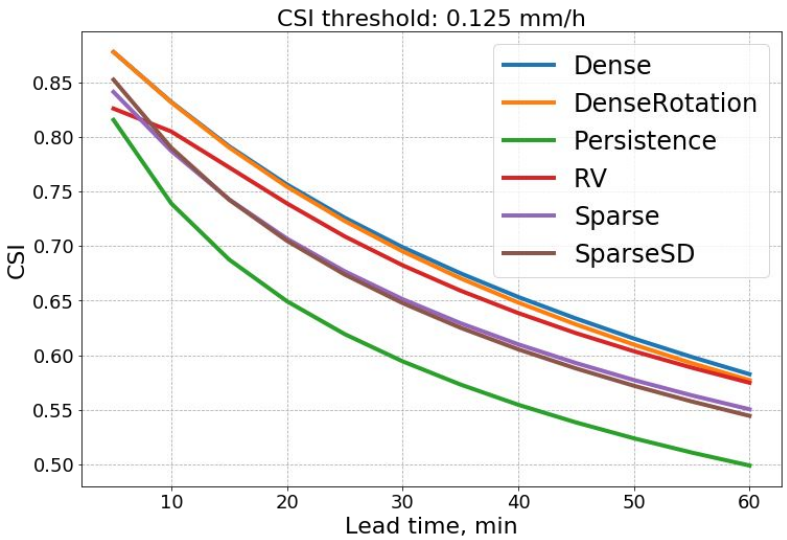
rainymotion models

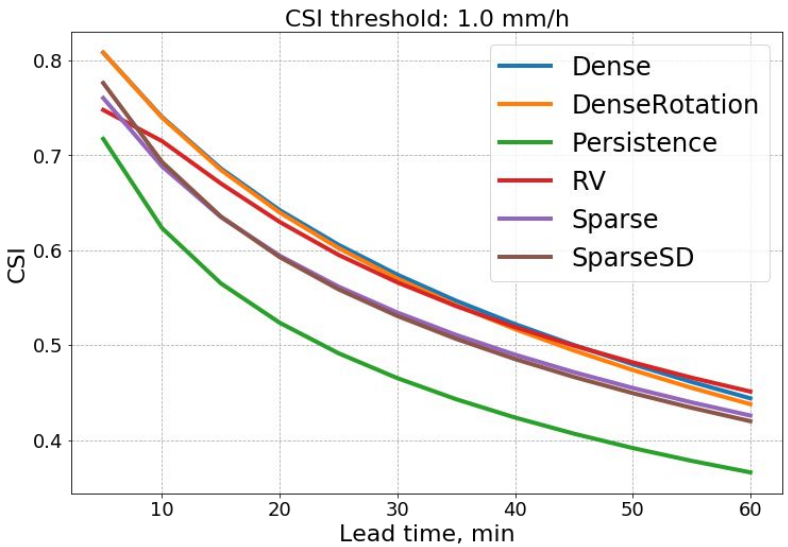
Name	Tracking	Extrapolation
SparseSD	• Shi-Tomasi detector (opencv) • Lukas-Kanade optical flow (opencv)	• Constant delta-change • Affine warping (skimage)
Sparse	• Shi-Tomasi detector (opencv) • Lukas-Kanade optical flow (opencv)	• Linear regression (sklearn) • Affine warping (skimage)
Dense	• Farnebäck optical flow (opencv)	• Constant-vector advection
DenseRotation	• Farnebäck optical flow (opencv)	• Semi-Lagrangian advection

rainymotion usage scenario

```
# import rainymotion model
from rainymotion.models import Dense

# initialize model instance
model = Dense()

# load the data using your custom DataLoader function
model.input_data = DataLoader("/path/to/data")

# run the model
nowcasts = model.run()
```

rainymotion for Germany

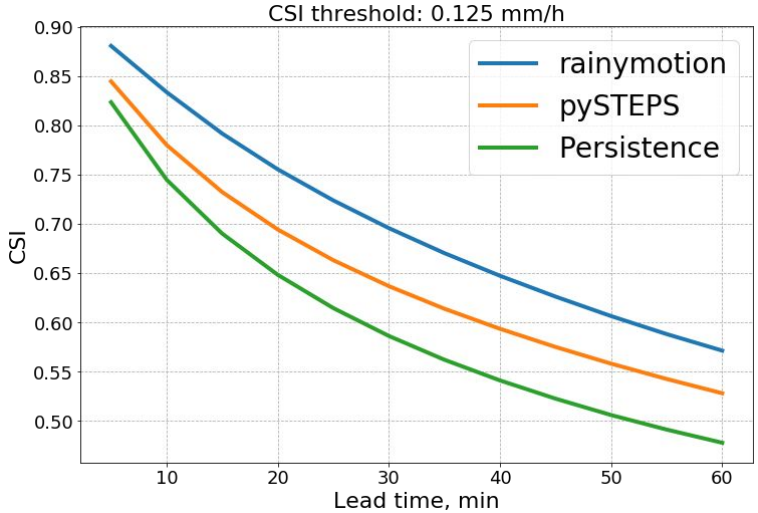


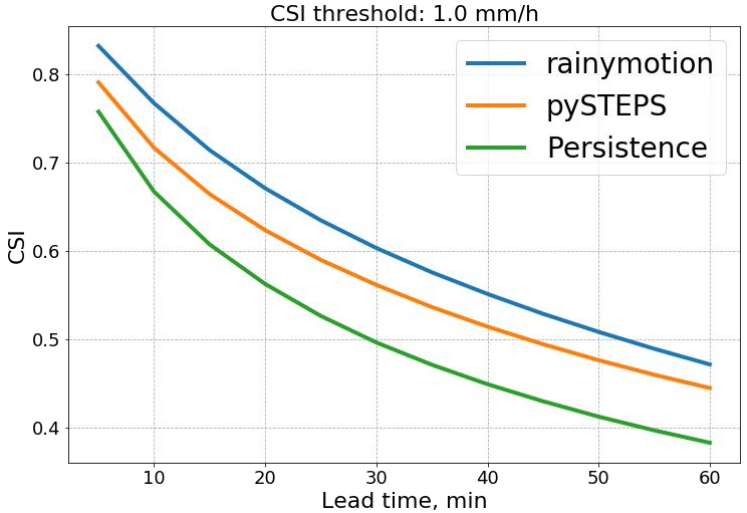


- Establishes a solid baseline
- Outperforms the operational model

rainymotion vs. pySTEPS

- GitHub Stars: 13 vs. 12
- Calculation speed: pySTEPS is faster
- Nowcasting efficiency: **rainymotion** is better





rainymotion

Code

github.com/hydrogo/rainymotion

1404 lines

Utils and metrics are included

Sample data is included

Installation from source:

```
$ python setup.py install
```

Documentation

rainymotion.readthedocs.io

Sphinx (sphinx-doc.org)

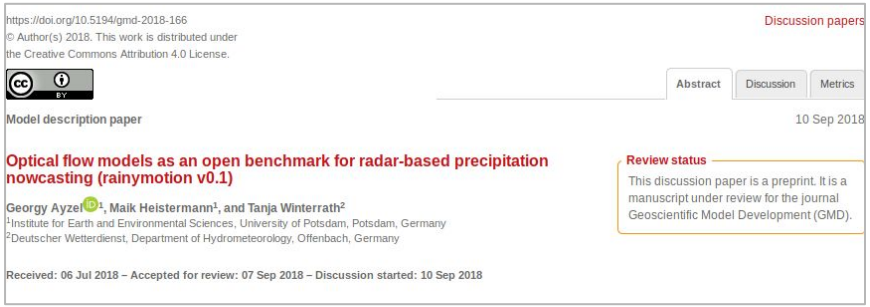
Installation, overview (.rst)

Tutorials, examples (.ipynb)

The most challenging part

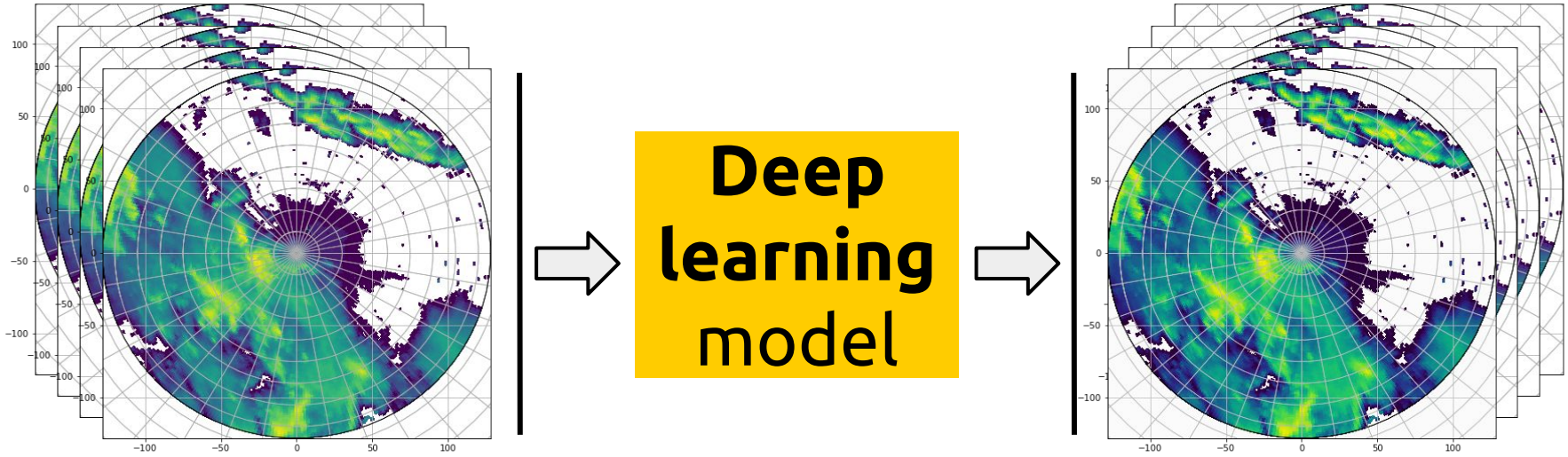
rainymotion paper

Ayzel, G., Heistermann, M., and Winterrath, T.:
Optical flow models as an open benchmark for radar-based precipitation nowcasting (rainymotion v0.1), **Geosci. Model Dev. Discuss.**, doi: 10.5194/gmd-2018-166, in review, 2018.



- Model description
- Verification for Germany
- Comparison with the operational model
- Discussion is open **until 5th November**

RainNet



- Deep learning is the new black
- Is it ready to conquer nowcasting?
  - Yes

RainNet development challenges

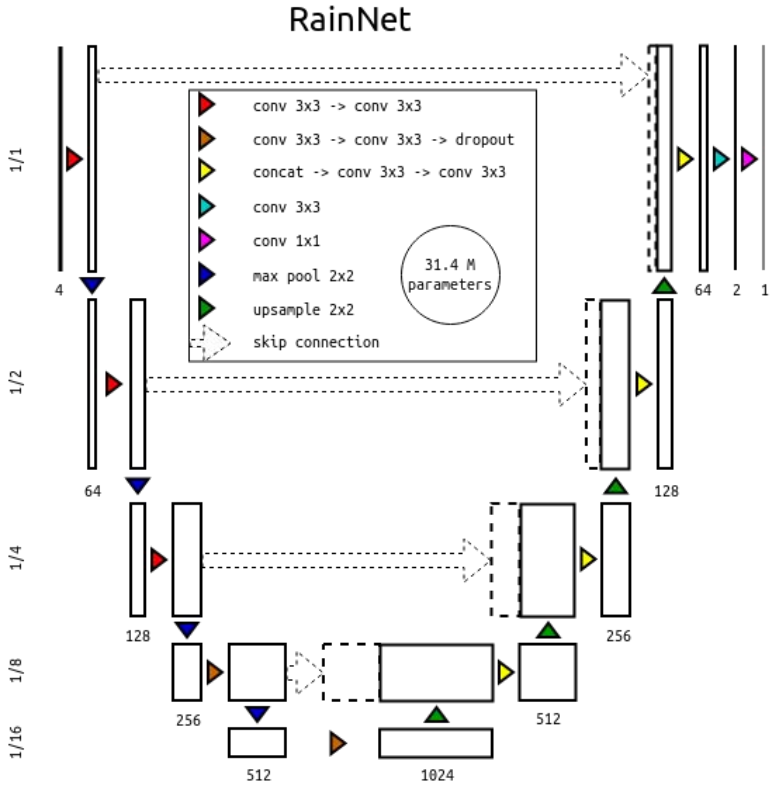
- We have
  - Big data
    - 10 years. Temporal resolution: 5 min
    - 900x900 km. Spatial resolution: 1 km
  - Open software
    - Tensorflow, Keras, PyTorch
  - Hardware
    - Clusters with Nvidia GPUs
- We have no idea about
  - Efficient I/O
  - Preprocessing
  - Neural network architecture
  - Training (optimizer, loss)
  - Consequences for humanity

RainNet development challenges

Efficient I/O	Individual .npy files (40Gb -> 2Tb)
Preprocessing	Crop, log(X+0.01)
Neural network architecture	Decoder-encoder, skip connections
Training (optimizer, loss)	Adam, logcosh
Consequences for humanity	Extends predictiability

We spent **more than one year** for this trial-and-error study

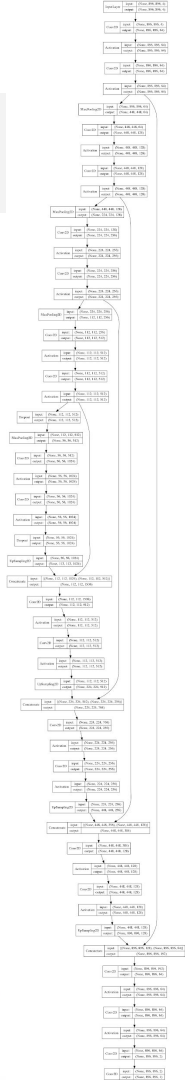
RainNet architecture



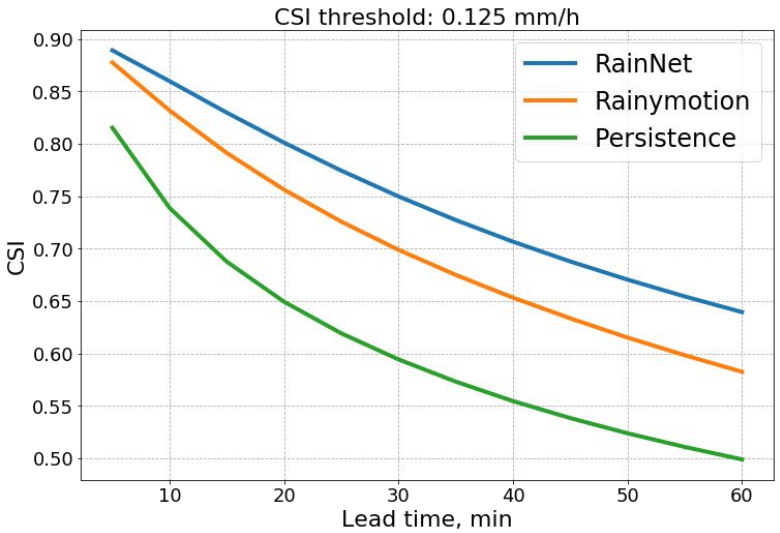
RainNet

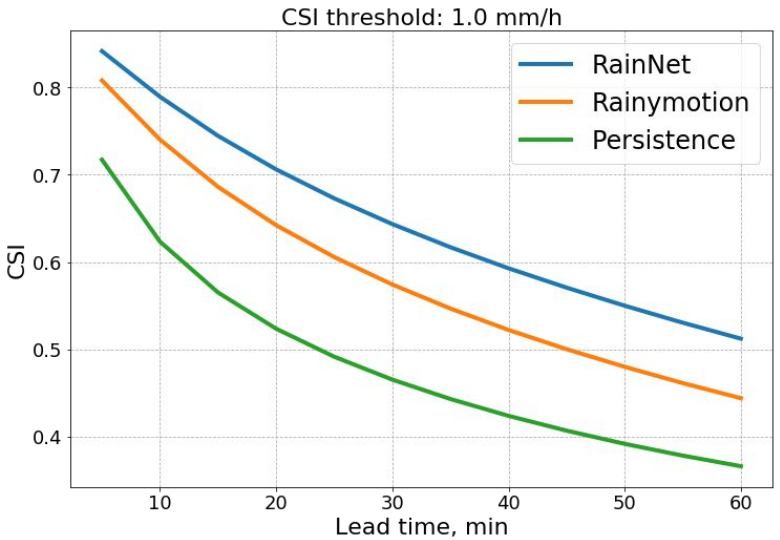
- Keras functional API
- 60 lines of code
- ~ 31.4M parameters
- Requires a lot of RAM
- 1 training epoch ~ 10h

```python
def rainnet(input_size=(896, 896, 4)):

    inputs = Input(input_size)

    conv1f = Conv2D(64, 3, padding='same', kernel_initializer='he_normal')(inputs)
    conv1f = Activation("relu")(conv1f)
    conv1s = Conv2D(64, 3, padding='same', kernel_initializer='he_normal')(conv1f)
    conv1s = Activation("relu")(conv1s)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1s)

    conv2f = Conv2D(128, 3, padding='same', kernel_initializer='he_normal')(pool1)
    conv2f = Activation("relu")(conv2f)
    conv2s = Conv2D(128, 3, padding='same', kernel_initializer='he_normal')(conv2f)
    conv2s = Activation("relu")(conv2s)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2s)

    conv3f = Conv2D(256, 3, padding='same', kernel_initializer='he_normal')(pool2)
    conv3f = Activation("relu")(conv3f)
    conv3s = Conv2D(256, 3, padding='same', kernel_initializer='he_normal')(conv3f)
    conv3s = Activation("relu")(conv3s)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3s)

    conv4f = Conv2D(512, 3, padding='same', kernel_initializer='he_normal')(pool3)
    conv4f = Activation("relu")(conv4f)
    conv4s = Conv2D(512, 3, padding='same', kernel_initializer='he_normal')(conv4f)
    conv4s = Activation("relu")(conv4s)
    drop4 = Dropout(0.5)(conv4s)
    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

    conv5f = Conv2D(1024, 3, padding='same', kernel_initializer='he_normal')(pool4)
    conv5f = Activation("relu")(conv5f)
    conv5s = Conv2D(1024, 3, padding='same', kernel_initializer='he_normal')(conv5f)
    conv5s = Activation("relu")(conv5s)
    drop5 = Dropout(0.5)(conv5s)

    up6 = concatenate([UpSampling2D(size=(2, 2))(drop5), conv4s], axis=3)
    conv6 = Conv2D(512, 3, padding='same', kernel_initializer='he_normal')(up6)
    conv6 = Activation("relu")(conv6)
    conv6 = Conv2D(512, 3, padding='same', kernel_initializer='he_normal')(conv6)
    conv6 = Activation("relu")(conv6)

    up7 = concatenate([UpSampling2D(size=(2, 2))(conv6), conv3s], axis=3)
    conv7 = Conv2D(256, 3, padding='same', kernel_initializer='he_normal')(up7)
    conv7 = Activation("relu")(conv7)
    conv7 = Conv2D(256, 3, padding='same', kernel_initializer='he_normal')(conv7)
    conv7 = Activation("relu")(conv7)

    up8 = concatenate([UpSampling2D(size=(2, 2))(conv7), conv2s], axis=3)
    conv8 = Conv2D(128, 3, padding='same', kernel_initializer='he_normal')(up8)
    conv8 = Activation("relu")(conv8)
    conv8 = Conv2D(128, 3, padding='same', kernel_initializer='he_normal')(conv8)
    conv8 = Activation("relu")(conv8)

    up9 = concatenate([UpSampling2D(size=(2, 2))(conv8), conv1s], axis=3)
    conv9 = Conv2D(64, 3, padding='same', kernel_initializer='he_normal')(up9)
    conv9 = Activation("relu")(conv9)
    conv9 = Conv2D(64, 3, padding='same', kernel_initializer='he_normal')(conv9)
    conv9 = Activation("relu")(conv9)
    conv9 = Conv2D(2, 3, activation='relu', padding='same', kernel_initializer='he_normal')(conv9)

    outputs = Conv2D(1, 1, activation='linear')(conv9)

    model = Model(inputs=inputs, outputs=outputs)

    return model
```



RainNet vs. rainymotion

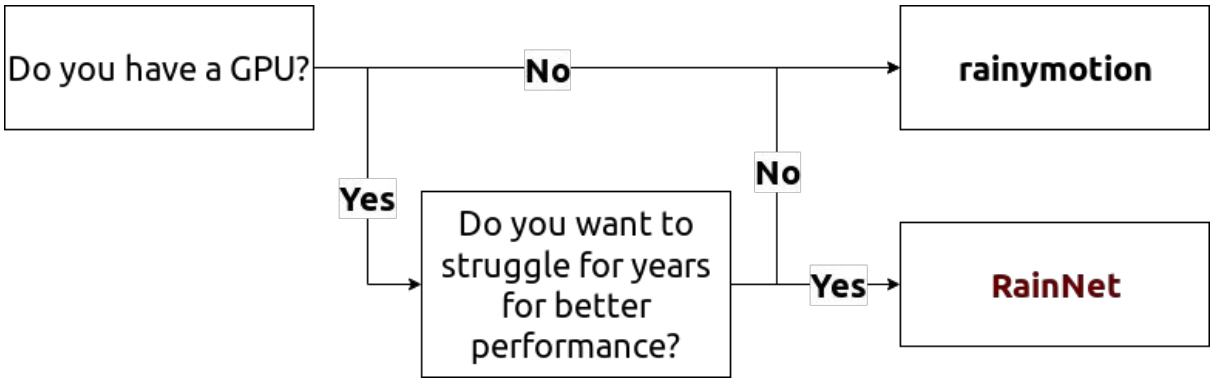




- **RainNet** is better
- What are the reasons behind?
- RainNet learns…

RainNet vs. rainymotion

- What should I choose?



Summary

- Python and open source software push forward scientific research
- Easy to stay on giants' shoulders (NumPy, SciPy, Xarray, Keras, OpenCV…)

- **rainymotion:** fast, free and transparent benchmark
- **RainNet:** extends precipitation predictability

- Deep learning research has strong hardware limits

Aggressive promotion

- Participate in a discussion around our **rainymotion** paper in GMDD

- Contact and follow me

@hydrogo89

github.com/hydrogo

t.me/hydrogo