FLOODS 4

Final Presentation

CEGM2003 U3: Project

By:
Kevin de Bruijn
Patrick de Bruijn
Mats Kerver
Han-Yu Wu

Content

Introduction

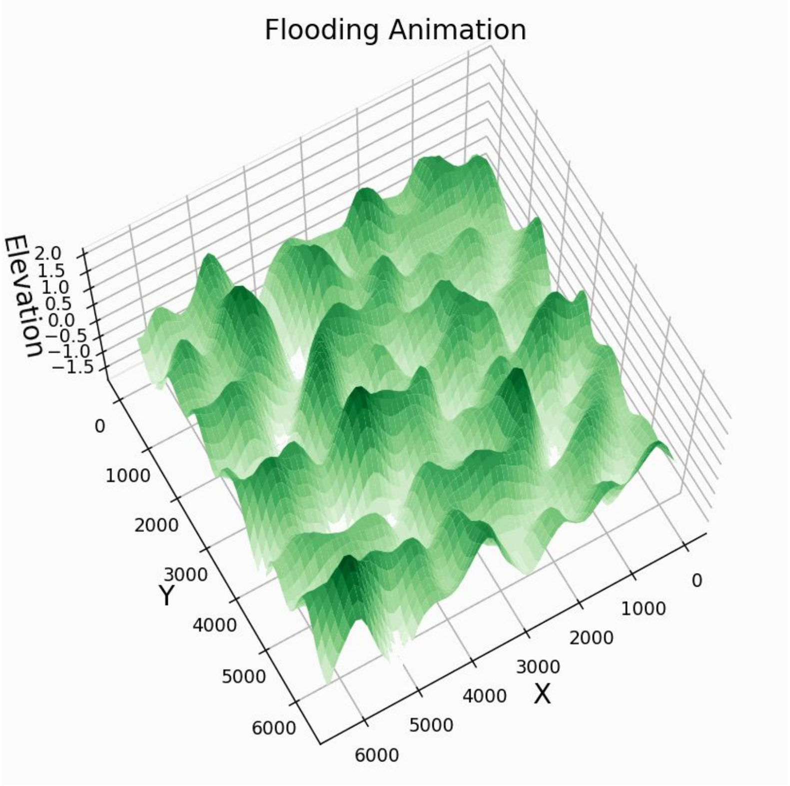
- “Your model should take topographical and hydraulic inputs to produce the evolution at different time steps of the hydraulic variables.”

  ⇒ Predict water level

- Faster than numerical modelling
  - Important for time-sensitive predictions
- Application:
  - Evacuation strategies
  - Mitigation measures
  - Probabilistic flood interpretation

Data

- Numerical ‘reference’ solutions/simulations, with fixed discharge
- Used variables:
  - water depth
  - topography
- 80 training/validation simulations, and 3 test sets
- Data augmentation applied (mirroring and rotation) ⇒ 400 training/validation simulations
- Normalization:
  - MinMax for water depth
  - Standard Gaussian for topography



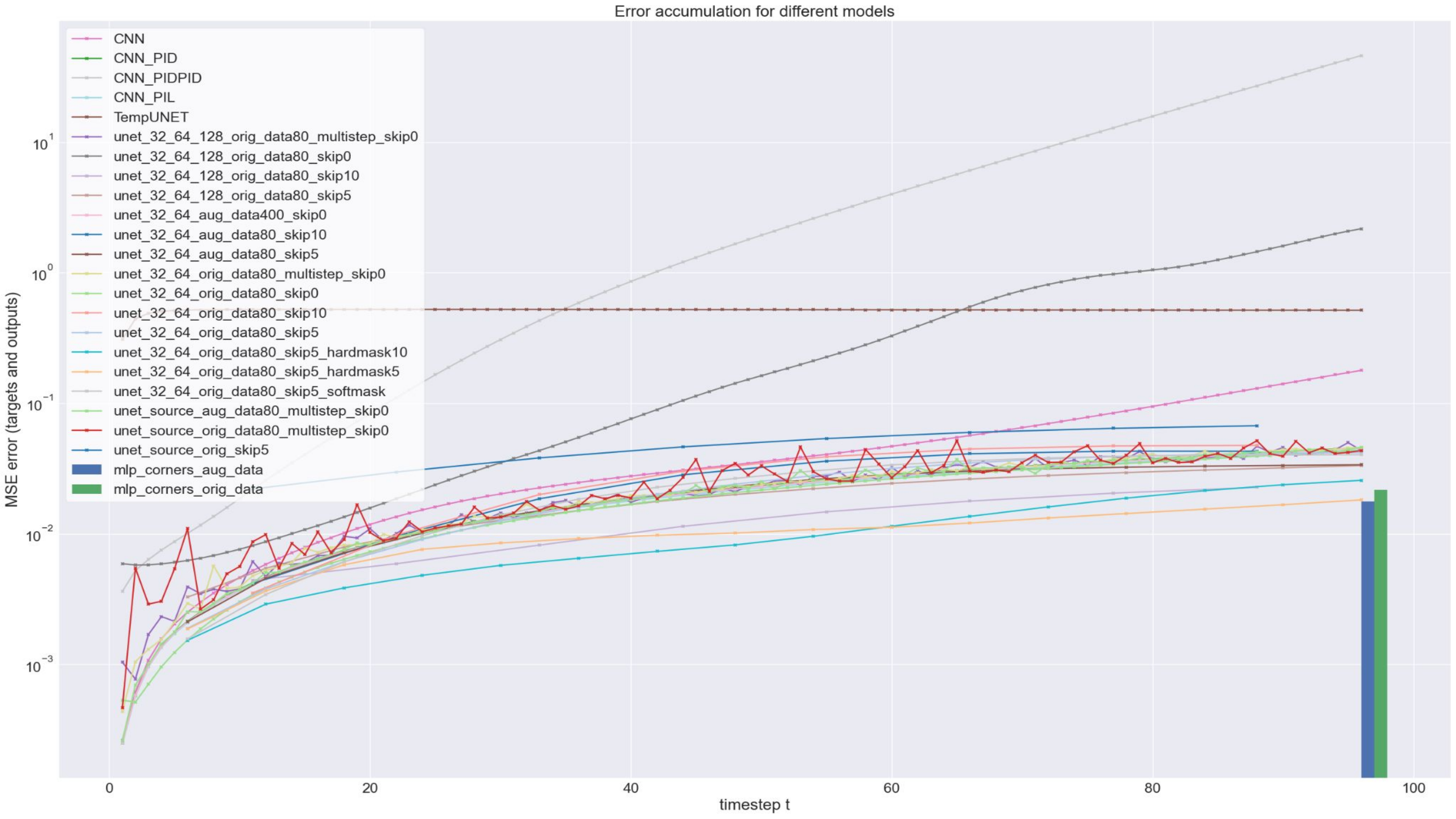
Description of models: inductive bias

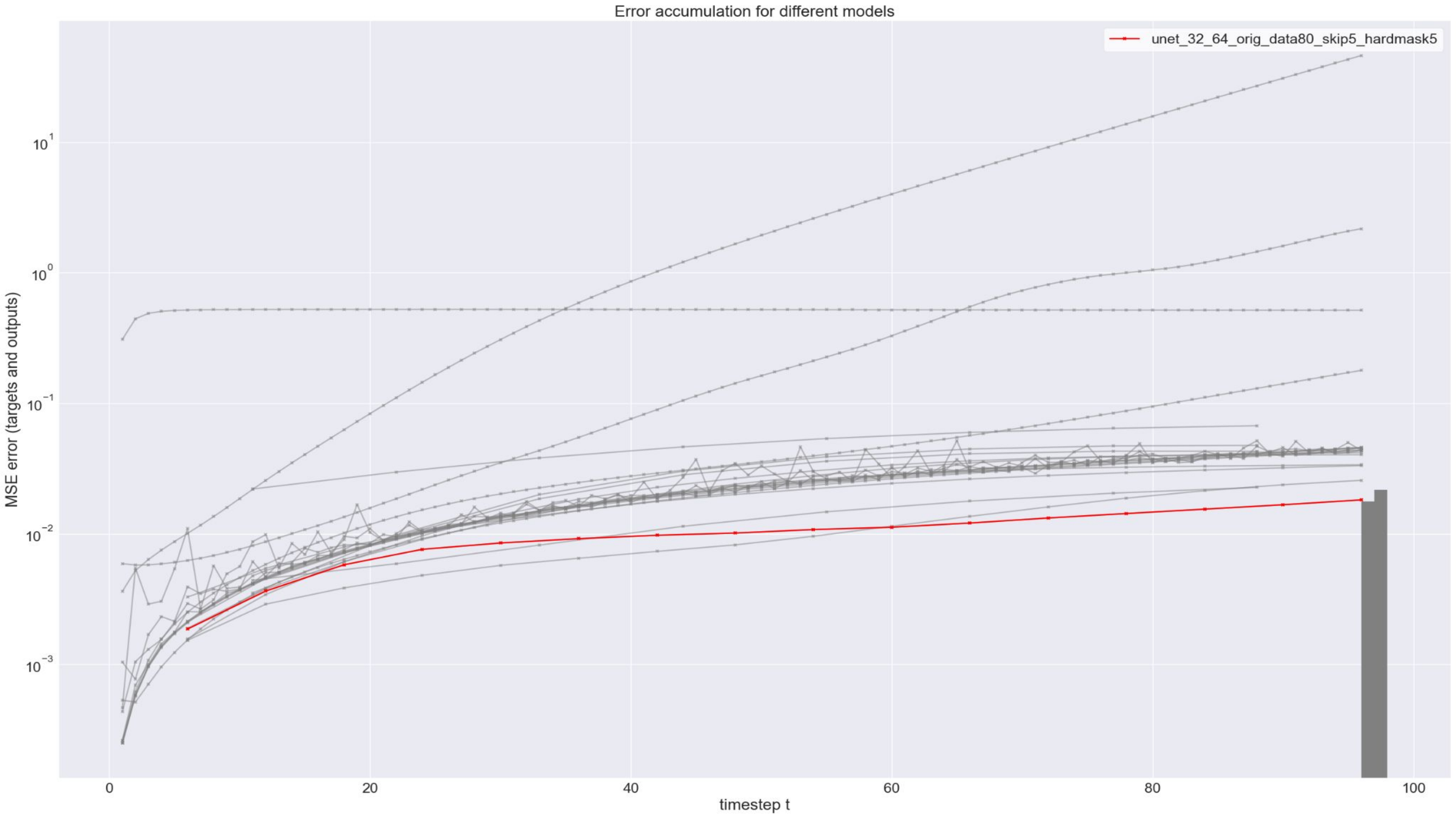
- Previously, we wanted to build an RNN
    - Temporal inductive bias
    - Requires time series to generate forecast
        - We might not have data available in this time sensitive application!
- But we need spatial inductive bias
    - Grid cells within close proximity should be highly correlated
    - Does not require time series, only an initial condition

⇒ Move towards CNN architecture in order to introduce spatial bias

Description of models: overview

Model	Type	Number of trainable parameters	Number of models
MLP with(out) data augmentation	MLP	264,193	2
CNN, CNN PIL, CNN PID, CNN PIDPID	CNN	112,577	4
TempCNN	CNN	1984	1
U-NET (original)	CNN, all time-steps and autoregressive	31,037,057	4
U-NET	CNN, all time-steps and autoregressive	101,505 — 466,881	9
U-Net mask	CNN, autoregressive	101,506 (1 extra for the distance)	3
U-Net dropout	CNN, autoregressive	101,505 — 466,881	4


Error accumulation for different models
CNN
CNN_PID
CNN_PIDPID
CNN_PIL
TempUNET
unet_32_64_128_orig_data80_multistep_skip0
unet_32_64_128_orig_data80_skip0
unet_32_64_128_orig_data80_skip10
unet_32_64_128_orig_data80_skip5
unet_32_64_aug_data400_skip0
unet_32_64_aug_data80_skip10
unet_32_64_aug_data80_skip5
unet_32_64_orig_data80_multistep_skip0
unet_32_64_orig_data80_skip0
unet_32_64_orig_data80_skip10
unet_32_64_orig_data80_skip5
unet_32_64_orig_data80_skip5_hardmask10
unet_32_64_orig_data80_skip5_hardmask5
unet_32_64_orig_data80_skip5_softmask
unet_source_aug_data80_multistep_skip0
unet_source_orig_data80_multistep_skip0
unet_source_orig_skip5
mlp_corners_aug_data
mlp_corners_orig_data
MSE error (targets and outputs)
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
0
20
40
60
80
100
timestep t



Error accumulation for different models
unet_32_64_orig_data80_skip5_hardmask5
MSE error (targets and outputs)
timestep t
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
0
20
40
60
80
100

U-NET: description (1)

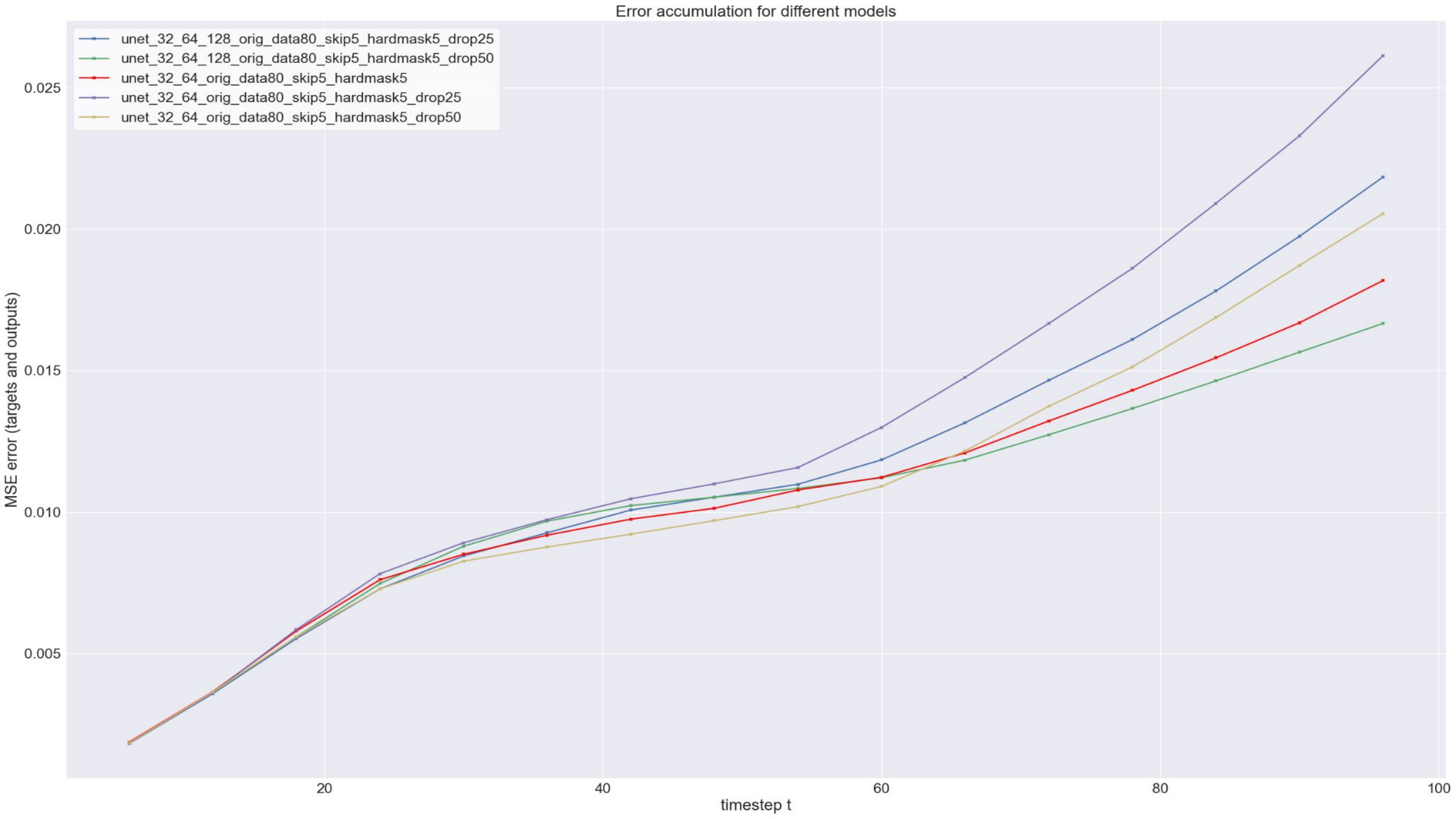
- Custom implementation with flexible hidden size
- Skip-connections (needs image to explain)
- Trained with time-skips

U-NET: description (2)

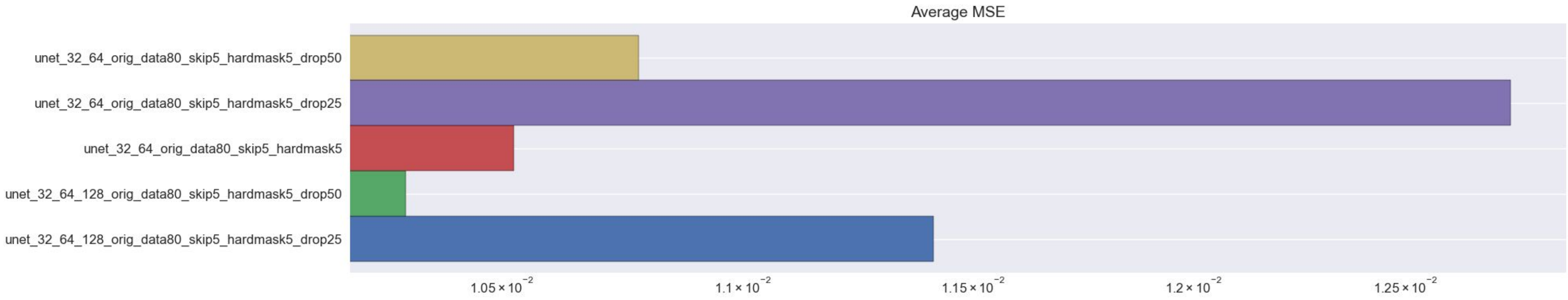
1. Custom layer amount

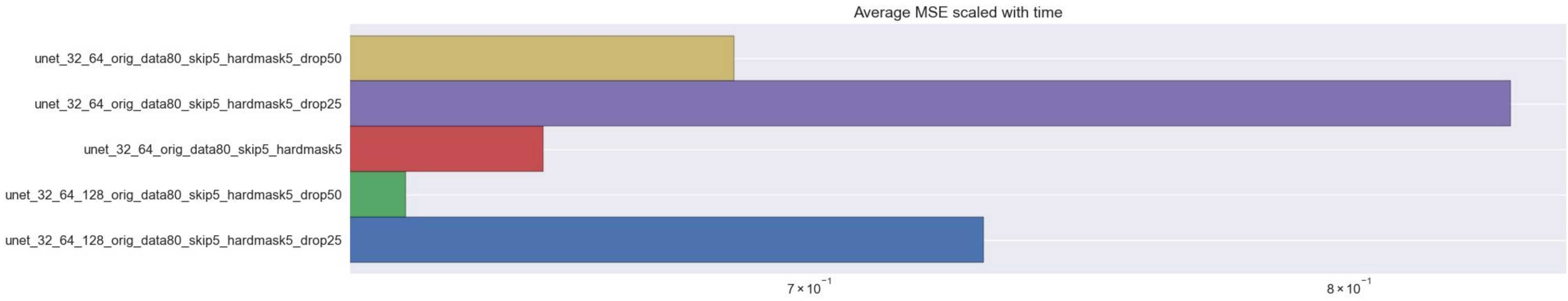
2. Mask (based on distance from wet pixel)
    a. output = x * distance_matrix(input) ** -p, where p is a trainable parameter
    b. output = x * mask → booleans, non trainable

3. Dropout


Error accumulation for different models
unet_32_64_128_orig_data80_skip5_hardmask5_drop25
unet_32_64_128_orig_data80_skip5_hardmask5_drop50
unet_32_64_orig_data80_skip5_hardmask5
unet_32_64_orig_data80_skip5_hardmask5_drop25
unet_32_64_orig_data80_skip5_hardmask5_drop50
MSE error (targets and outputs)
0.025
0.020
0.015
0.010
0.005
20
40
60
80
100
timestep t

U-NET: dropout

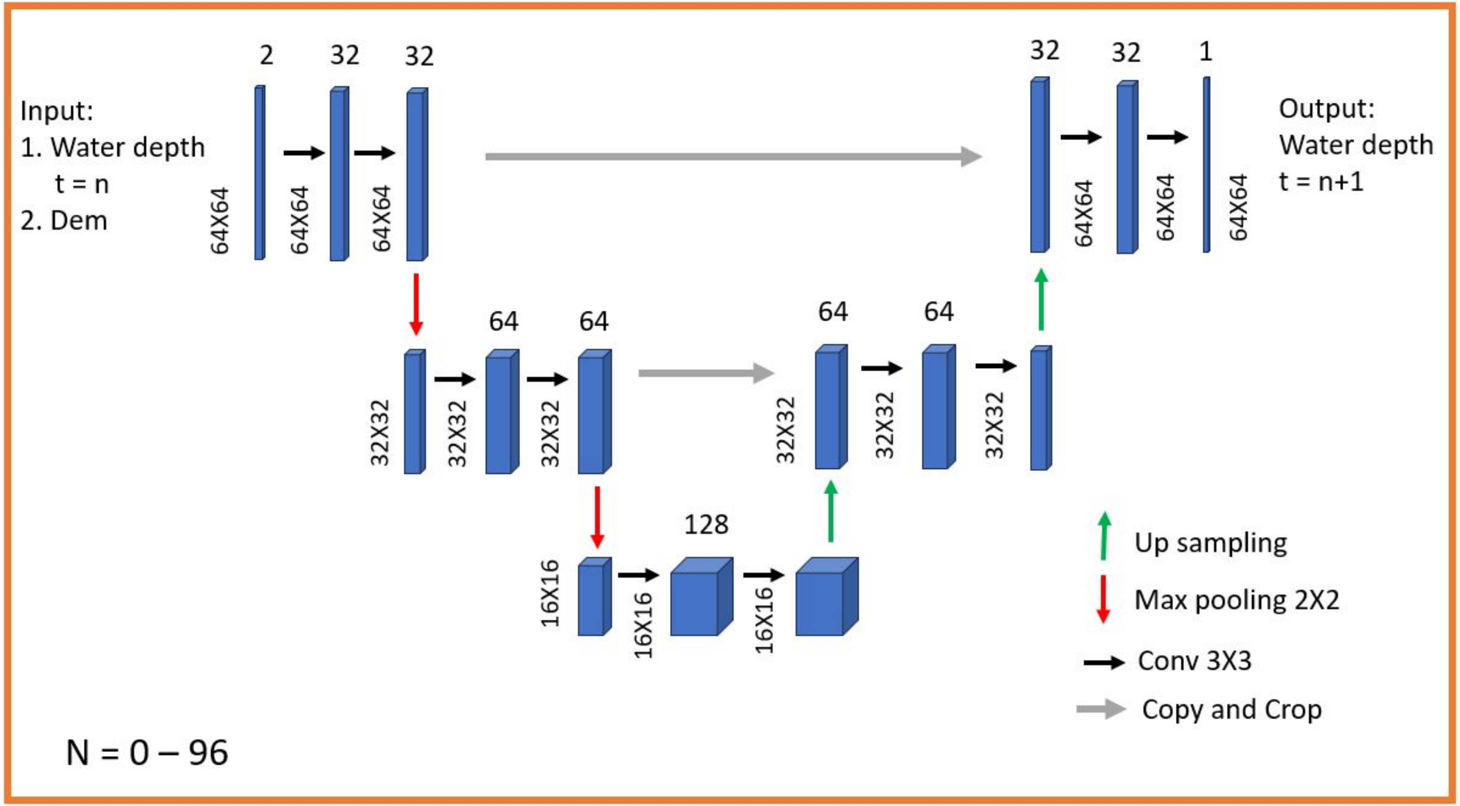




Final model: decisions

- No augmented data
- Timeskip = 5
- Encoder [2, 32, 64, 128] $\rightarrow$ Decoder [128, 64, 32, 1]
- Hard mask of 5 pixels
- Dropout rate 0.50

Final model: architecture

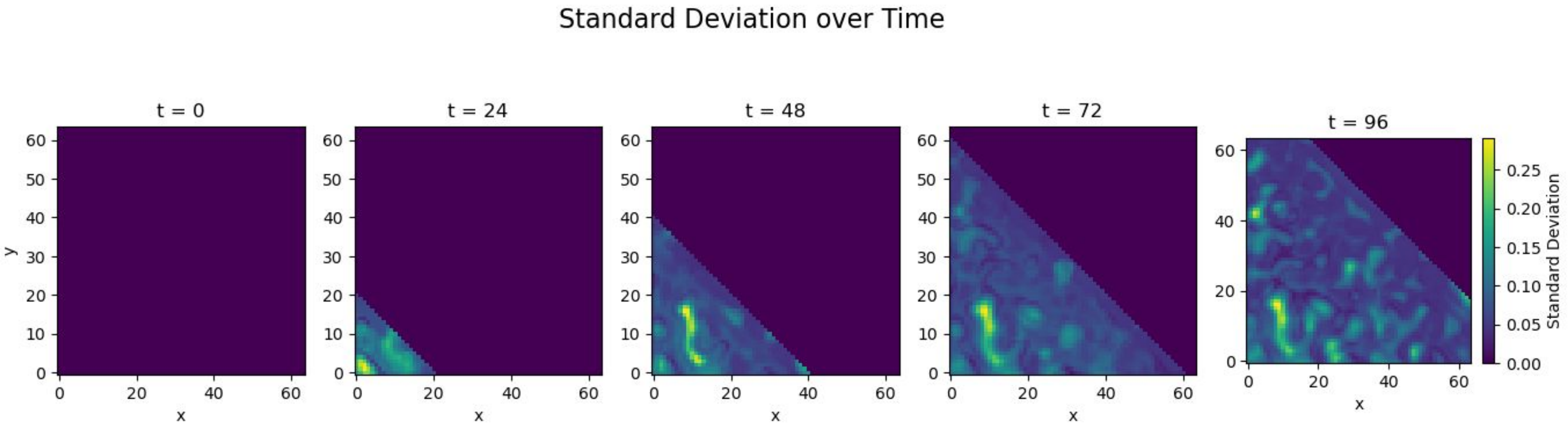


Final model: demo

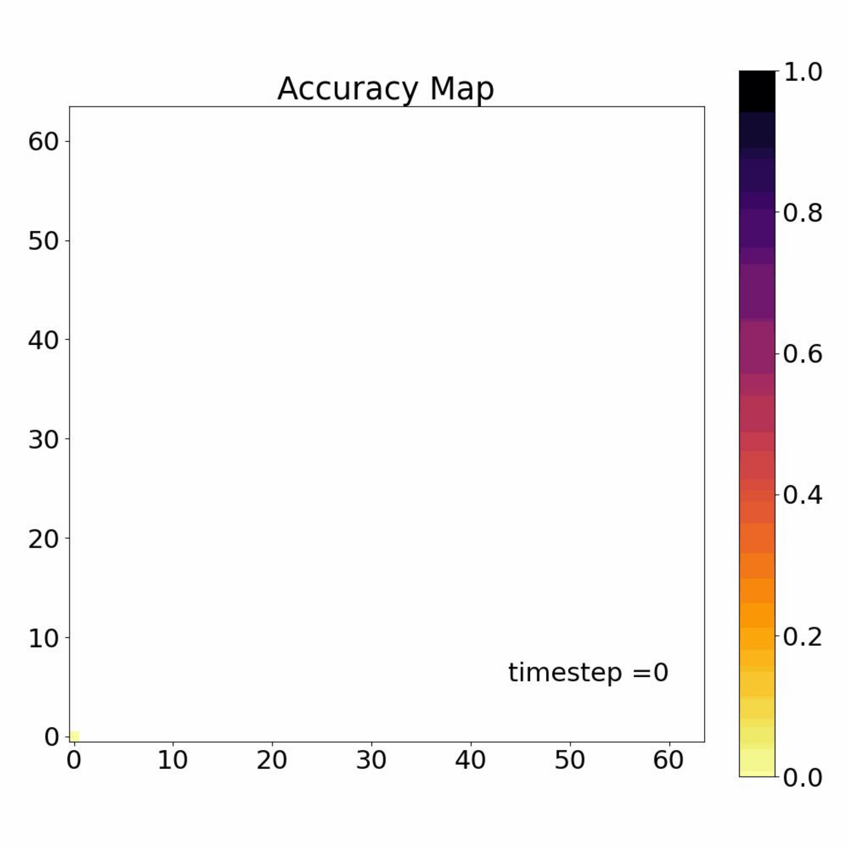
- Let’s do a demo of the model!

Final model: advantages

- Much faster than numerical model
- Includes uncertainty through dropout



Final model: accuracy



$$accuracy = 1 - \frac{|target - prediction|}{target}$$

Final model: limitations

- Temporal resolution of 5+1 steps, i.e. one prediction every 3 hours
- Weaker prediction shortly after the breach (in comparison to other U-NET models)
- Not trained on augmented data, so not very generalizable to other flood orientations
- Predictions as fast as possible, since the masking method is not optimized

Possible improvements

- Train the model on augmented data to make it applicable for different locations of breach
- Improve temporal resolution
- Optimize the number of parameters
- Improve masking method (optimized and more flexible mask)
- Make the network physics-informed:
    - Implement a graph neural network to force the model to adhere to mass balance (Finite Volume Method)
    - Train with flow velocities, and use SWE in loss function

THANK YOU

By:
Kevin de Bruijn
Patrick de Bruijn
Mats Kerver
Han-Yu Wu

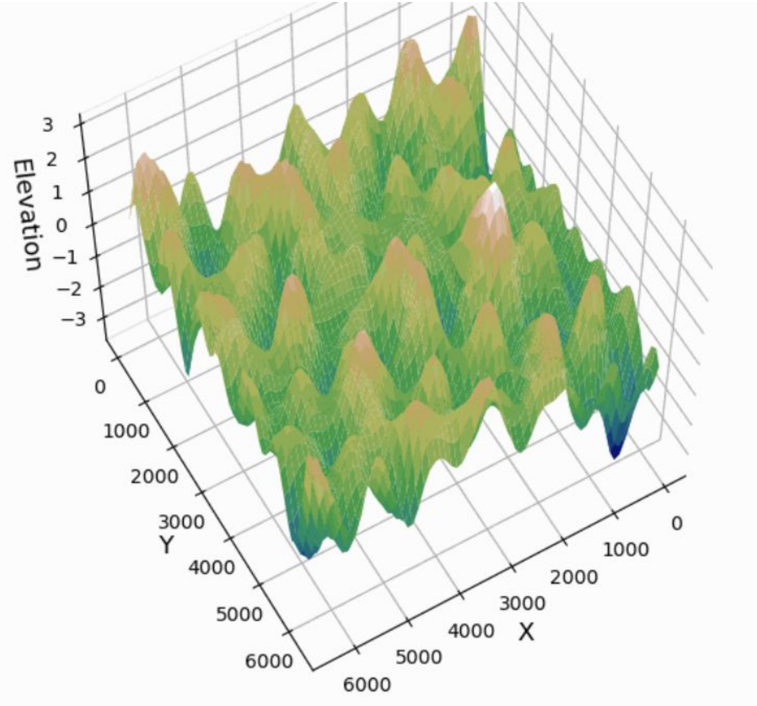
EXTRA SLIDES

- **Data**
  - **Description**
  - **Augmentation**
  - **Normalization**
- **Different models**
  - **MLP**
  - **CNN**
  - **TempCNN**
- **Comparison models**
- **Confusion matrix**
- **Reflection**

Data: description, augmentation, and normalization

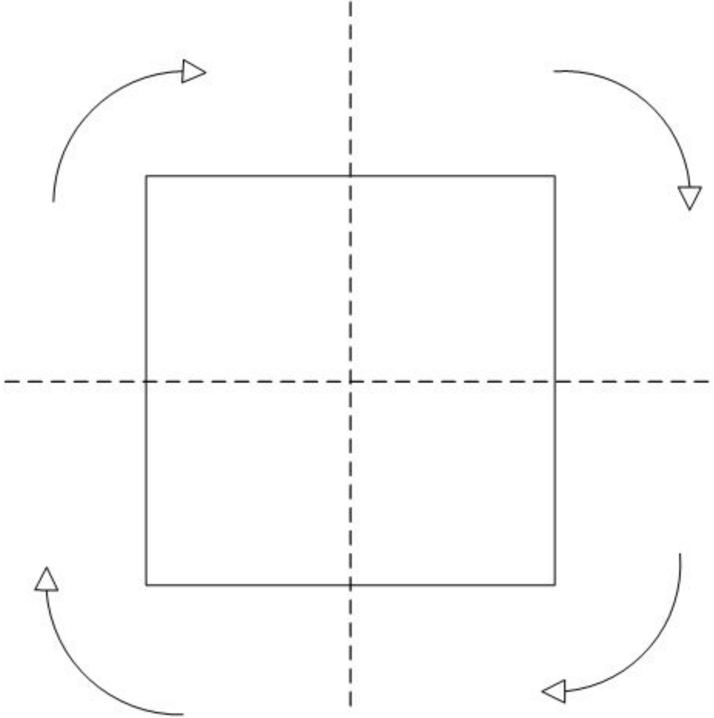
Data: description

- Numerical ‘reference’ solutions/simulations
- Grid:
    - Resolution: 100m x 100m
    - Size: 64 x 64 (and some 128 x 128)
- Temporal resolution:
    - Resolution: 0.5h
    - Total simulation length: 48h (and some 120h)
- Available variables:
    - water depth
    - horizontal flow velocities (x- and y-direction) → not used
    - topography
- Constant discharge at fixed location into the domain



Data: augmentation

- 80 simulations (70% training, 30% validation)
- Used to increase number of available datasets for training, validation and testing.
- Grid is square, so *rotation* by [0°, 90°, 180°, 270°] is possible.
- Additionally, *horizontal and vertical mirroring* is available.
- In total 4*2*2 = 16
    - 1 dataset → 16 datasets (some of which overlap) →
    - Randomly choose 5 from these
- 400 simulations for training/validation!



Data: normalization

- Each simulation normalized separately
- Topography:
  - Calculate mean and standard deviation to fit standard Gaussian
- Water depth:
  - Should be positive definite
    ⇒ MinMax normalization
  - Minimum and Maximum calculated from final time step of each simulation
- Ensures consistent scaling of input data

$$z = \frac{x - \mu}{\sigma}$$

$\mu =$ Mean

$\sigma =$ Standard Deviation

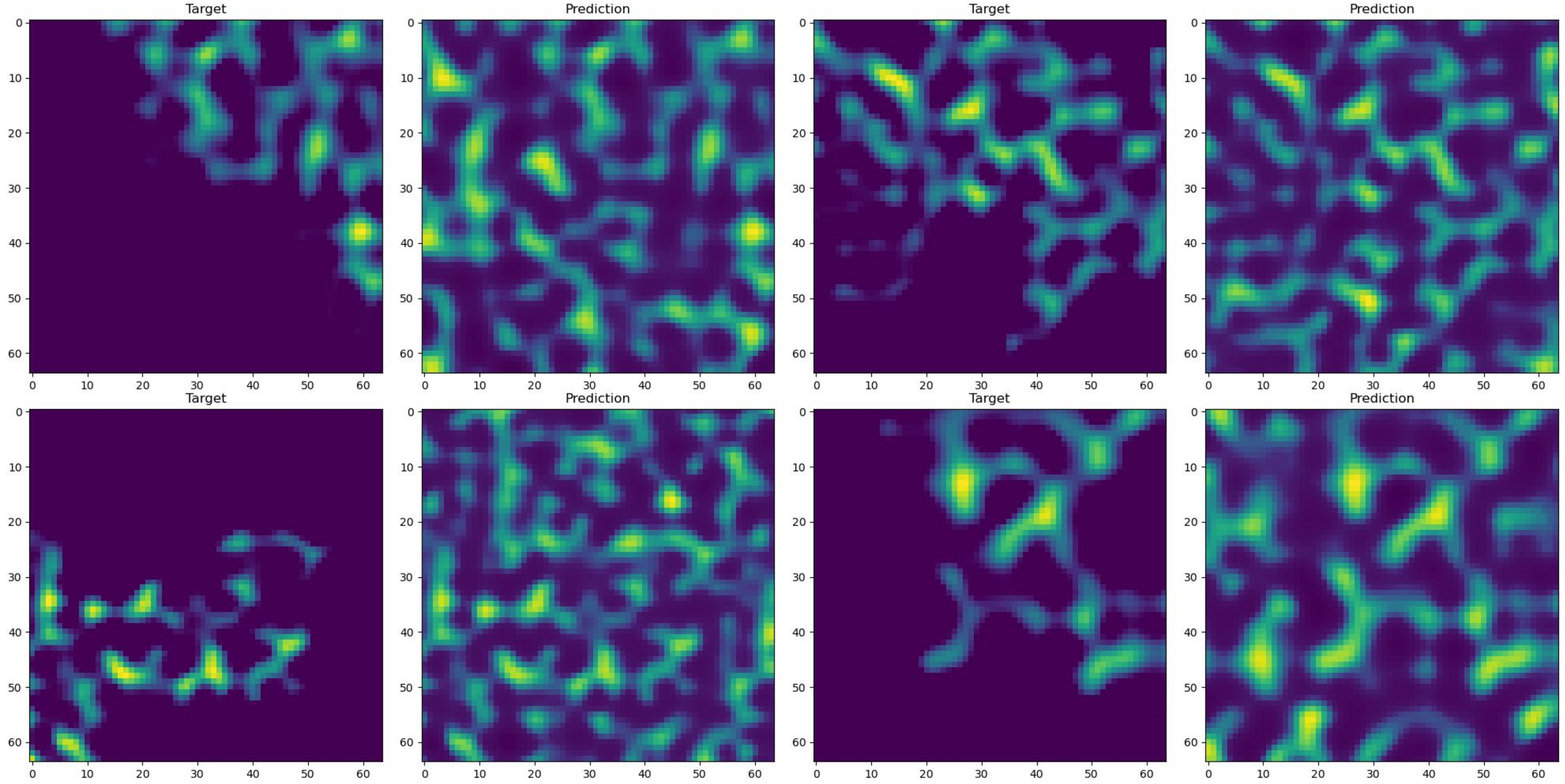
$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Different models: MLP, CNN, TempCNN

Description of models: MLP

Let's see if a basic MLP is suitable:

- Augmented data?
- 4 fully connected hidden layers, ReLU
- Predicts final timestep from the initial conditions only
    - Instead of needing one corner pixel with a huge weight, a channel was added that indexed which corner was the origin of the flood.


Target
Prediction
Target
Prediction
Target
Prediction
Target
Prediction

Description of models: CNN (1)

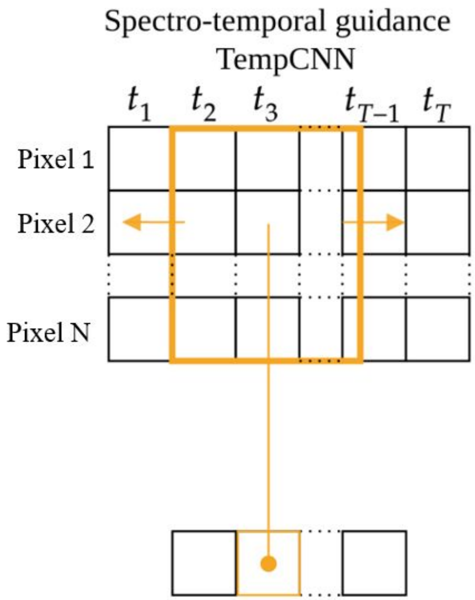
- Architecture:
  - Inputs 2 channels: topography and water depth (timestep t)
  - Outputs 1 channel: water depth (timestep t+1) → 1D convolution to transform final hidden layer to output
  - 4 hidden layers with size 64
  - … trainable parameters
- Don’t include data augmentation
- All predict the entire time series recursively!

Description of models: CNN (2)

- Basic CNN
- Custom loss function:
  - Physics-Informed Loss (PIL)
  - Includes current timestep in order to punish the model for having too much or not enough water in the domain
- Custom forward method:
  - Physics-Informed Difference (PID)
  - Uses additional parameter dV denoting maximum volume difference to punish the model for adding or removing too much water between time steps
- Custom loss function and forward method:
  - Physics-Informed Difference with Penalty in loss function for Incorrect Dry prediction (PIDPID)
  - Uses same parameter dV as PID and includes penalty in the loss function when the model false predicts a cell te be dry
  - False dry prediction more catastrophic than false wet prediction

Description of models: TempCNN

- Spectro-temporal U-net Based CNN
  - Regard time as the another axis of image (4096 x 97)
  - 97 timestep used one-time
  - Study the temporal evolution of each pixel
- Model architecture
  - Rectangle shape -> kernel size and striding should be chosen
  - 2 channels inputs: topography and water depth
  - 1 channel outputs: water level
  - Encoder-Decoder with hidden layers (2, 4, 8), (8, 4, 1) channels
- Defect
  - No spatial bias, every pixel is considered separately
  - Large training error at the early training stage
  - Error decreases slowly
- Extension to 3D

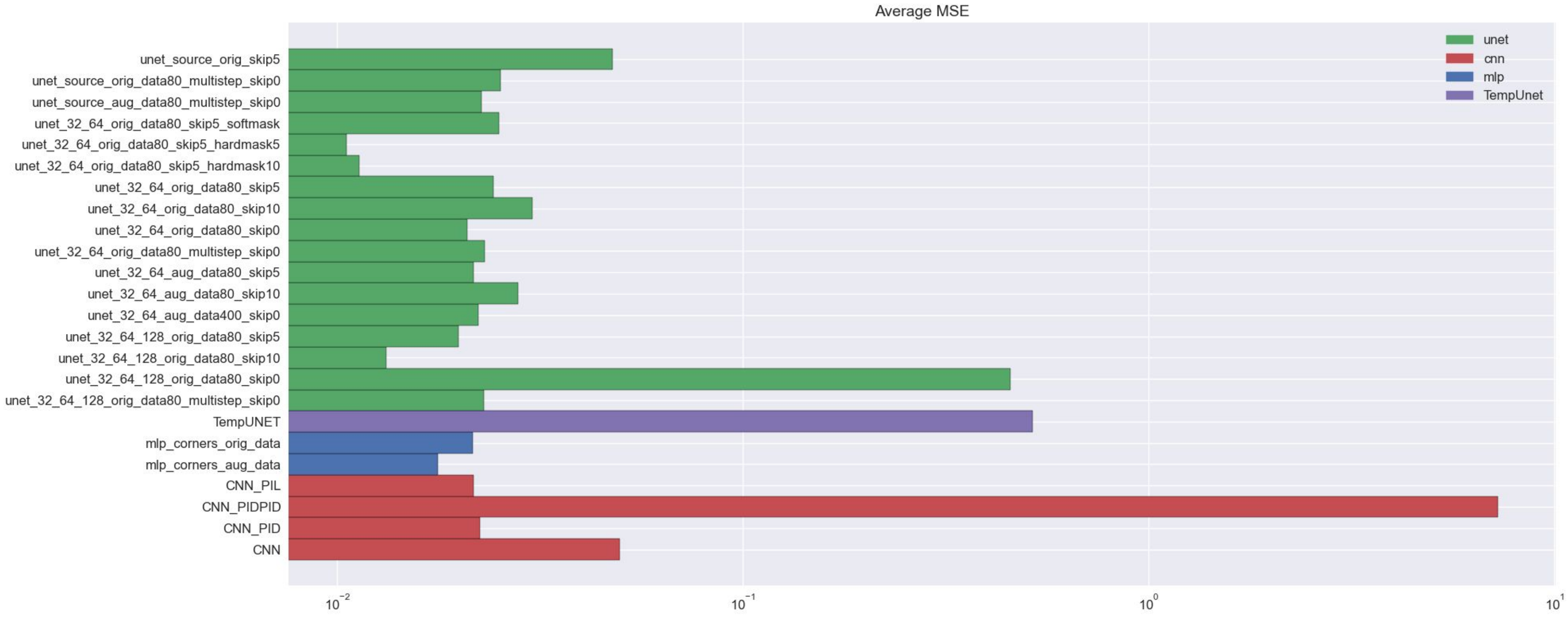


modified from Pelletier et al. 2019

Comparison models

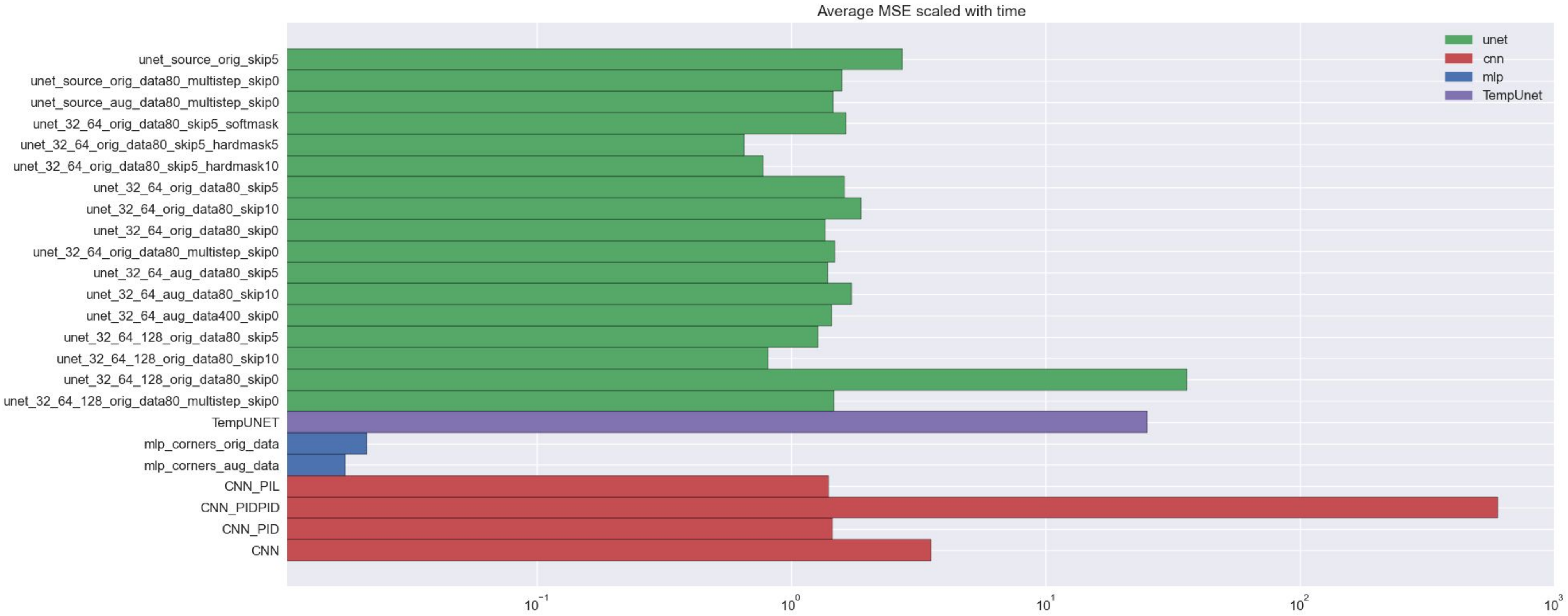
Performance of models: average MSE

-



Performance of models: average MSE scaled with time

- Scales the MSE linearly with time, in order to make predictions further away more 'important'

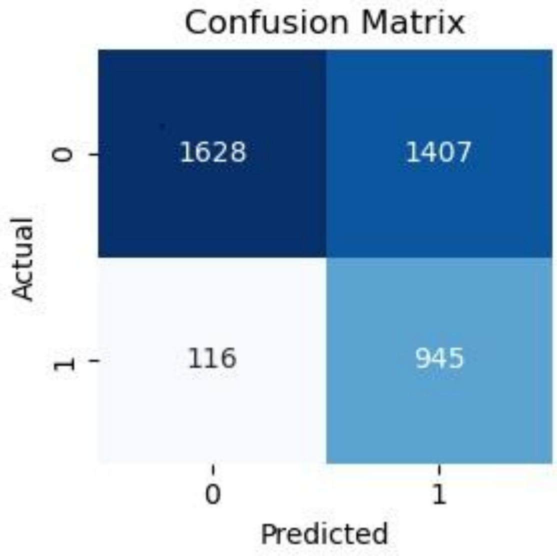


Confusion matrix

Performance of models: confusion matrix (1)

Example for a single run:

- Accuracy: 0.63
- Recall: 0.89
- Precision: 0.40
- F1 Score: 0.55



Reflection

Reflection

- We spent a long time working on RNNs, which didn’t end up working out
    - Maybe widen the scope? There were 4 of us, so maybe some of us could have worked on CNNs from the start.
- Make the MAIN branch a protected branch 😞
- Data handling through GitHub is not the most optimal
- The python module worked well!
- There was clear communication within the group