Representing unresolved processes in the modRSW model with a neural network

Raphael Kriegmair, Yvonne Ruckstuhl, Stephan Rasp, George Craig
Meteorological Institute Munich

Content

- Introduction / Problem / Thesis Objective
- Neural Network Basics
- Method
- Preliminary Results
- Long Term Model Behaviour

The Problem

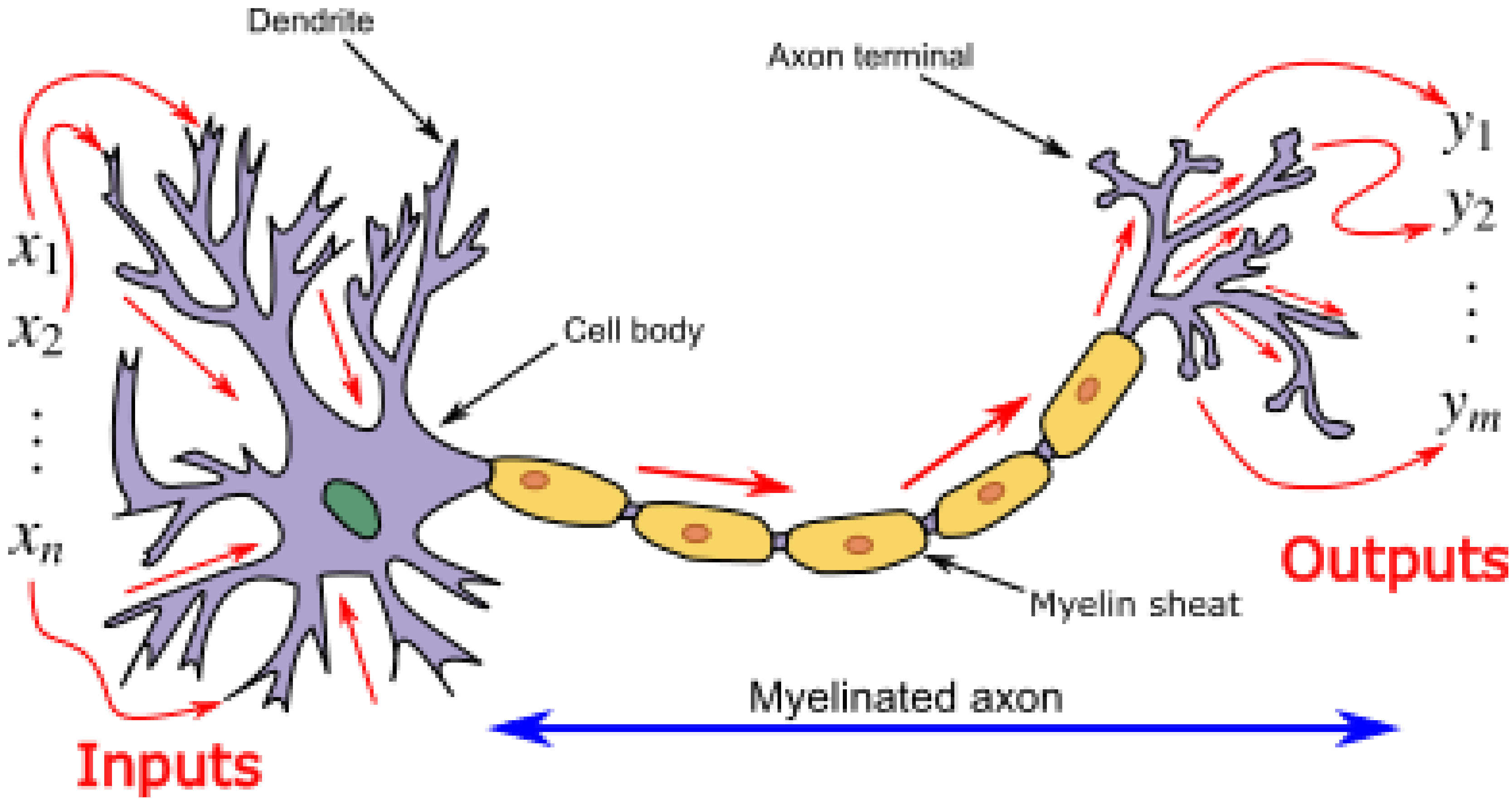
- Current Model Resolution limited by computational resources -> important processes not resolved
- Typical approaches: physical parametrizations, superparametrization, ... -> weaknesses
- Thesis Objective: Use neural network as subgrid scale model

Masters Thesis Objective

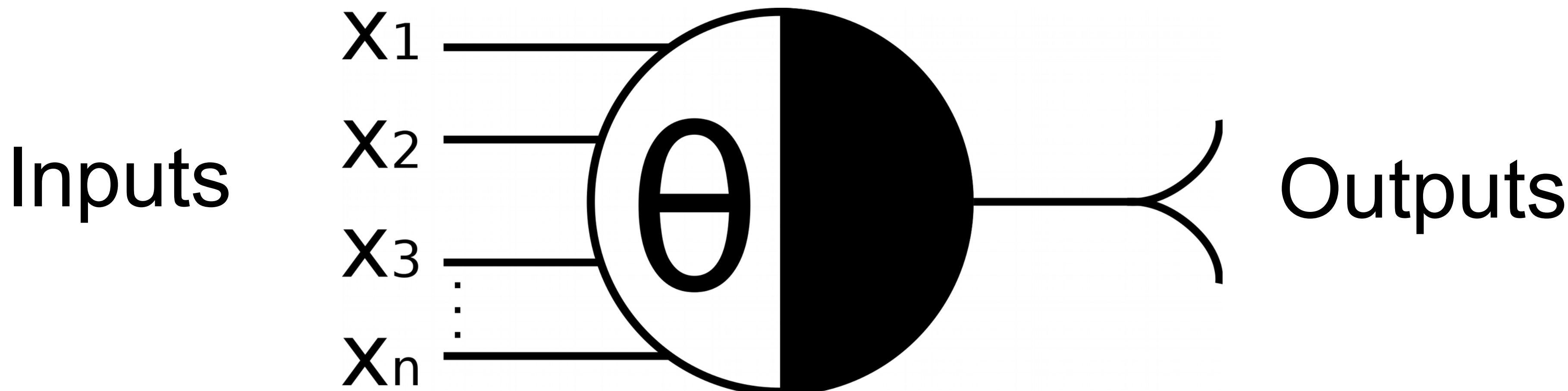
- Can Artificial Neural Networks (ANN) be employed as a subgrid-scale (SGS) model ?
- Case study using the modRSW model: idealized but physically relevant dynamics
- Desired Result: ANN, trained on high resolution data, improves accuracy of low resolution forecasts

What is an Artificial Neural Network?

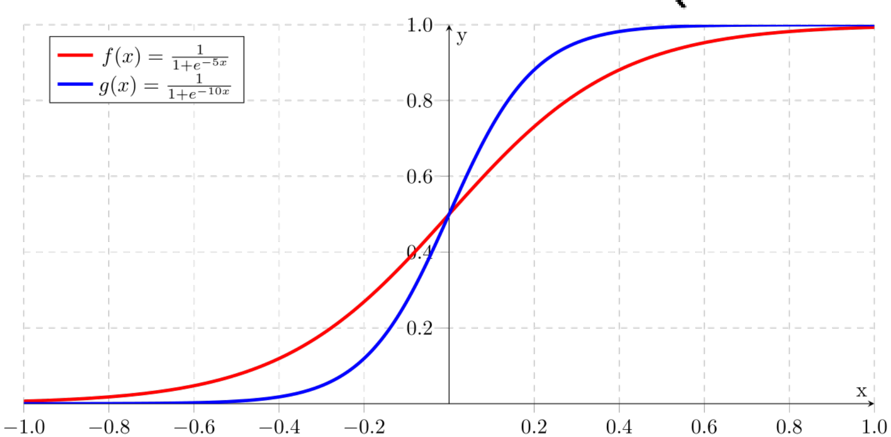
What is an Artificial **Neural** Network?

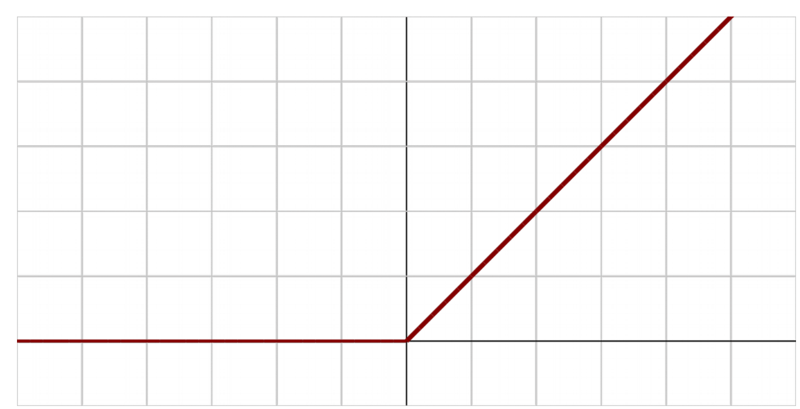


What is an **Artificial** Neural Network?

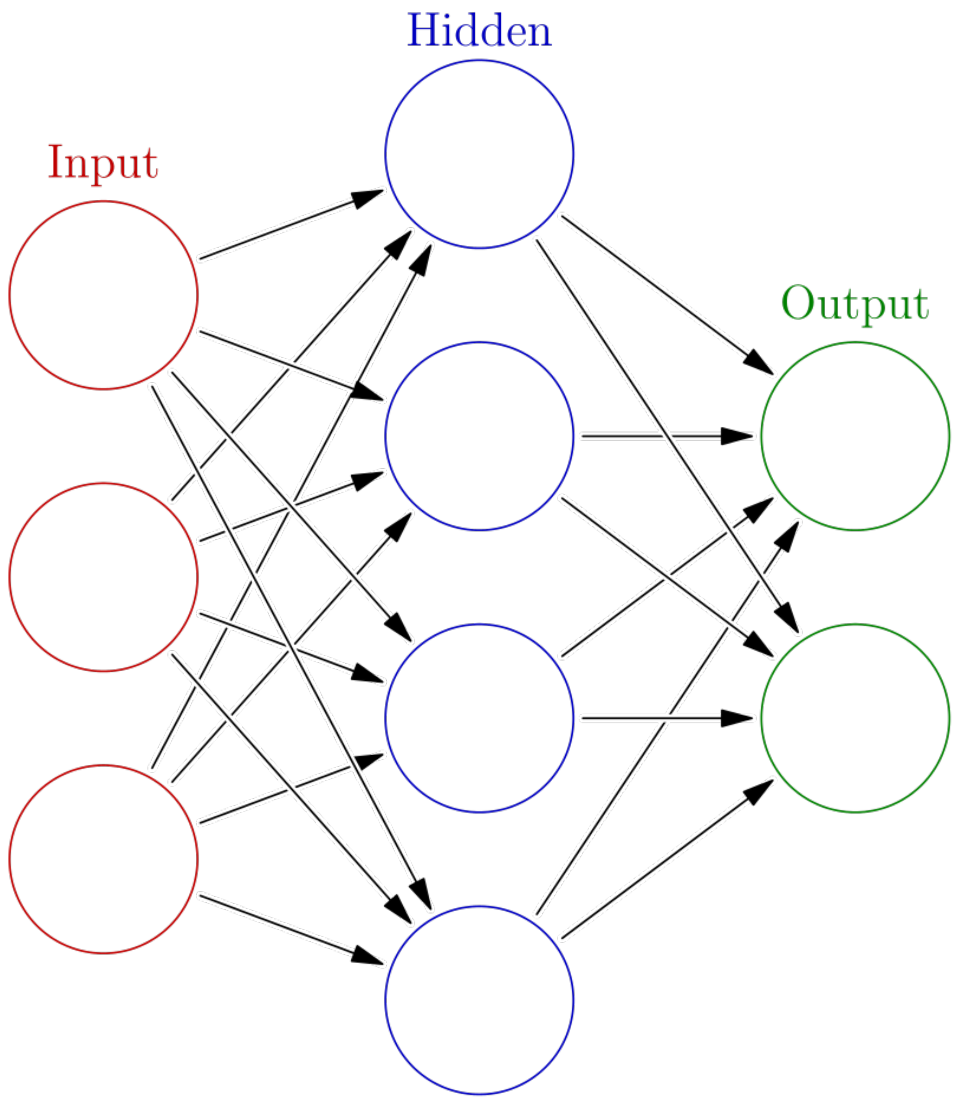


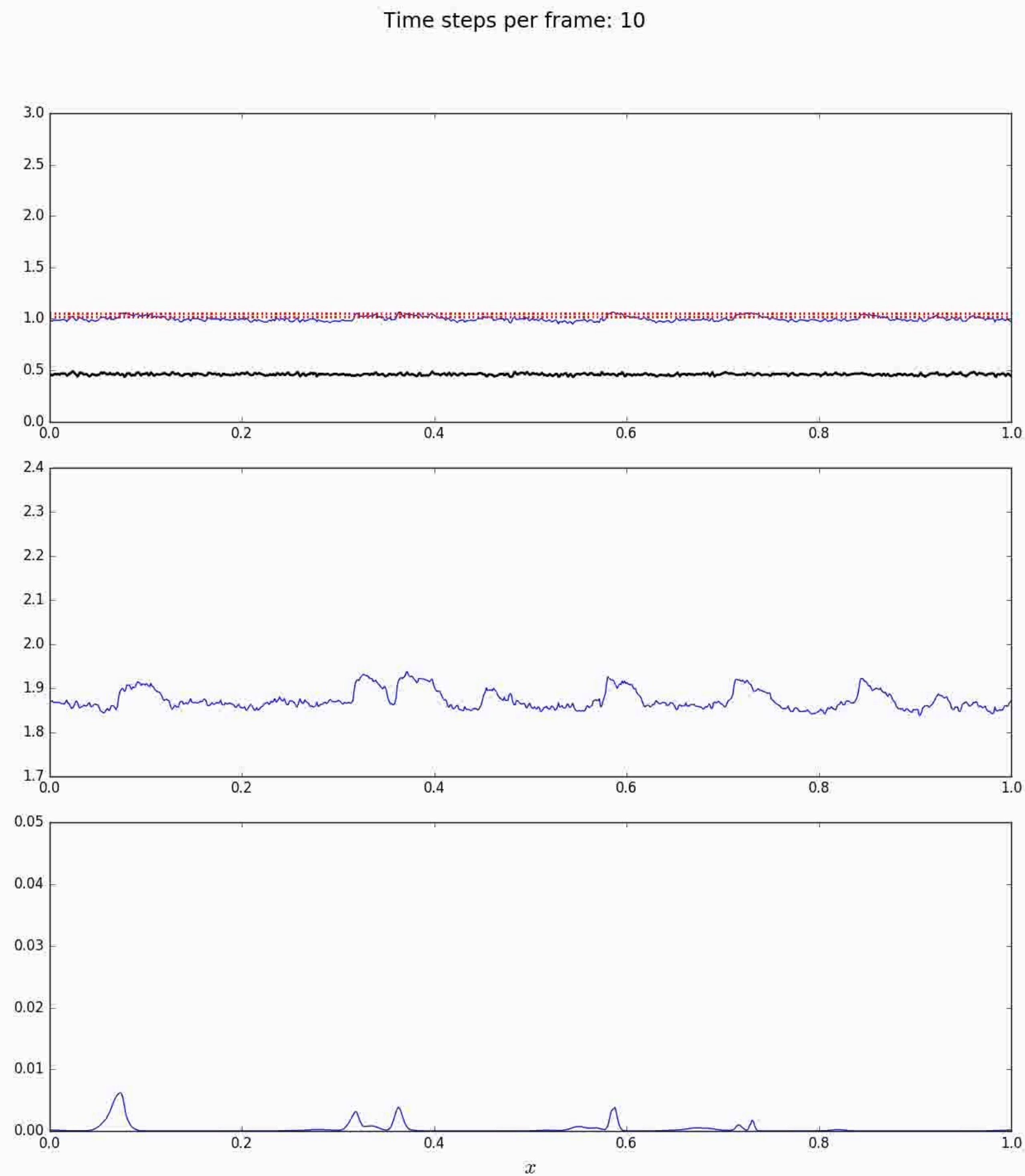
$$\theta\left(\sum_i w_i x_i + bias\right)$$



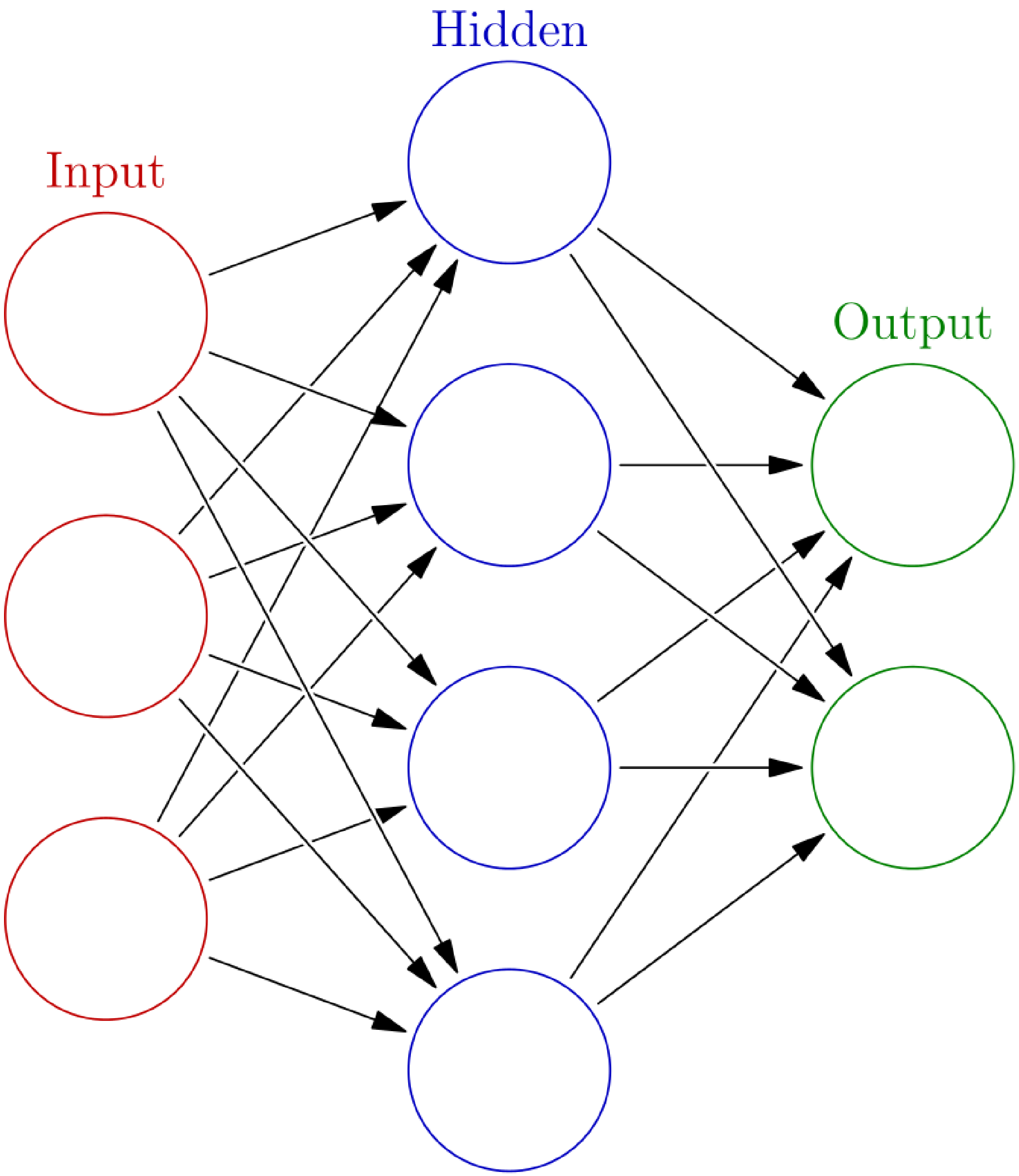


What is an Artificial Neural **Network**?





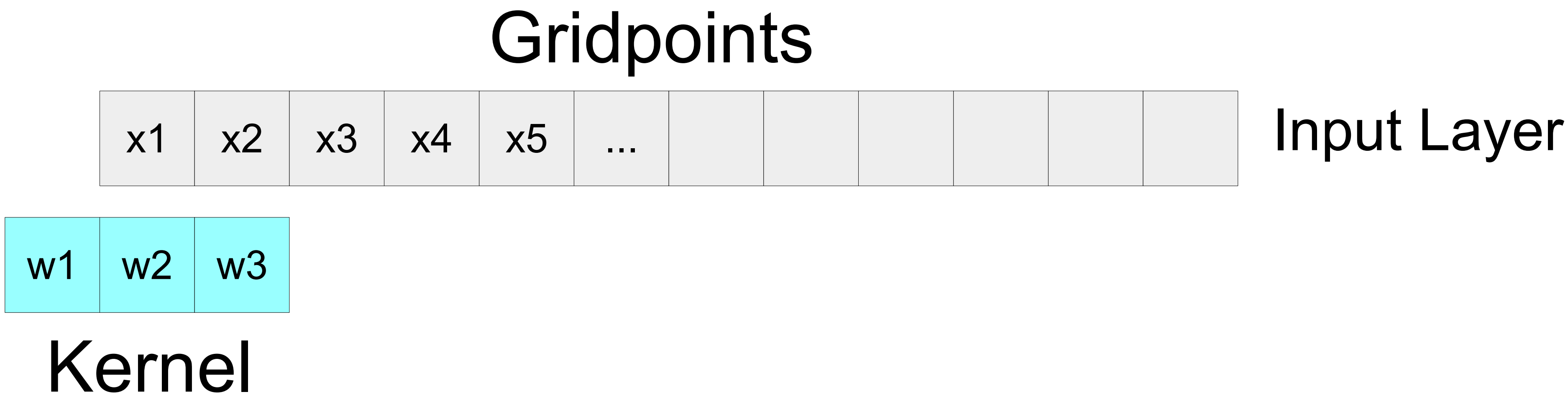
Time steps per frame: 10
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
2.4
2.3
2.2
2.1
2.0
1.9
1.8
1.7
0.05
0.04
0.03
0.02
0.01
0.00
x


Hidden
Input
Output

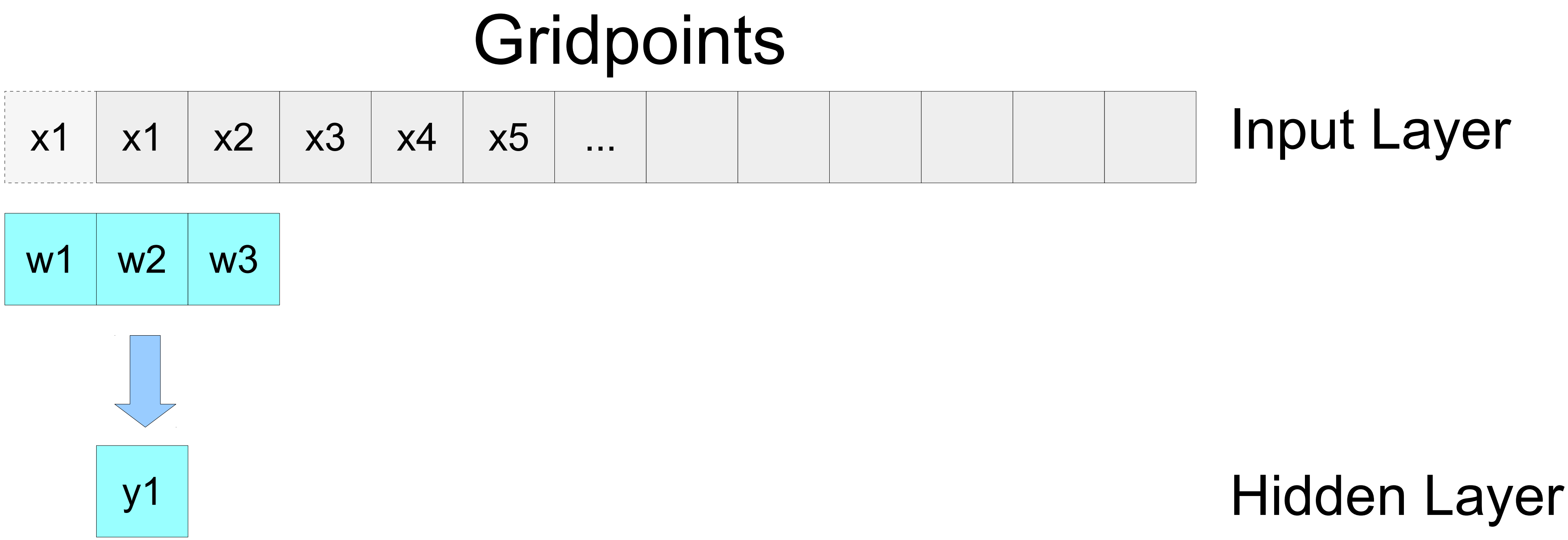
Convolutional Networks

- Localisation
- Translational invariance
- ... cheaper than fully connected networks

Convolutional Networks

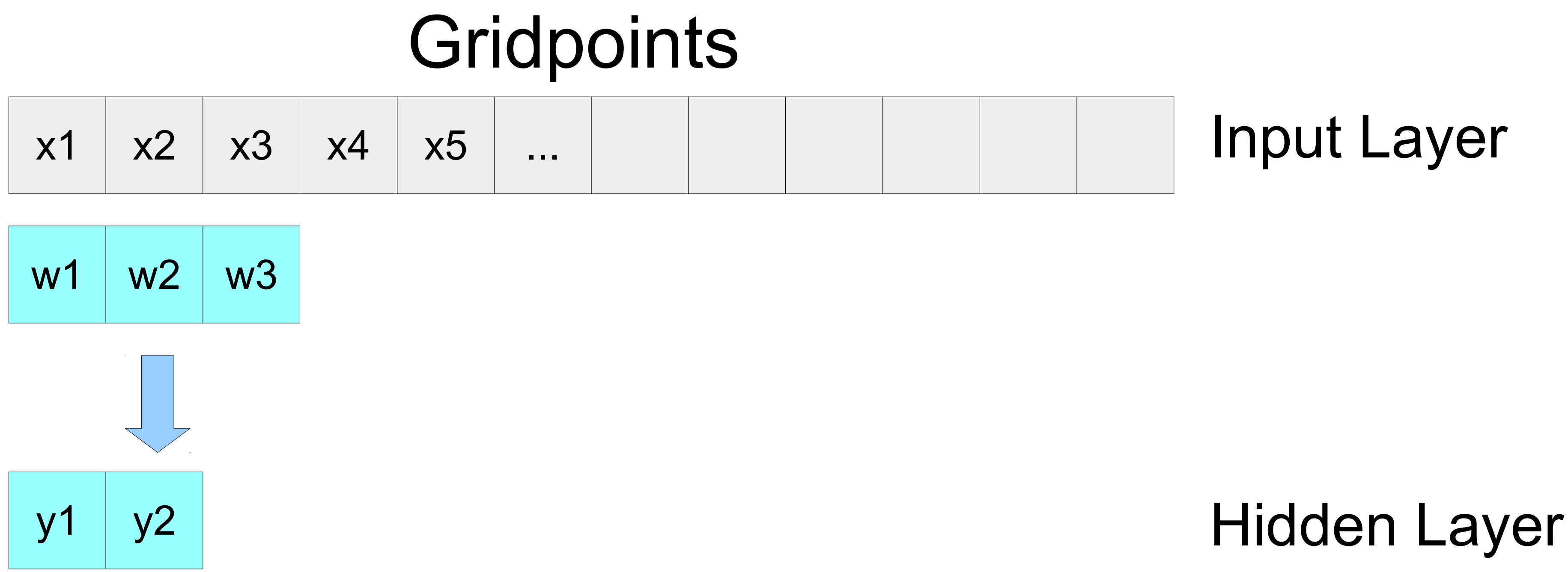


Convolutional Networks



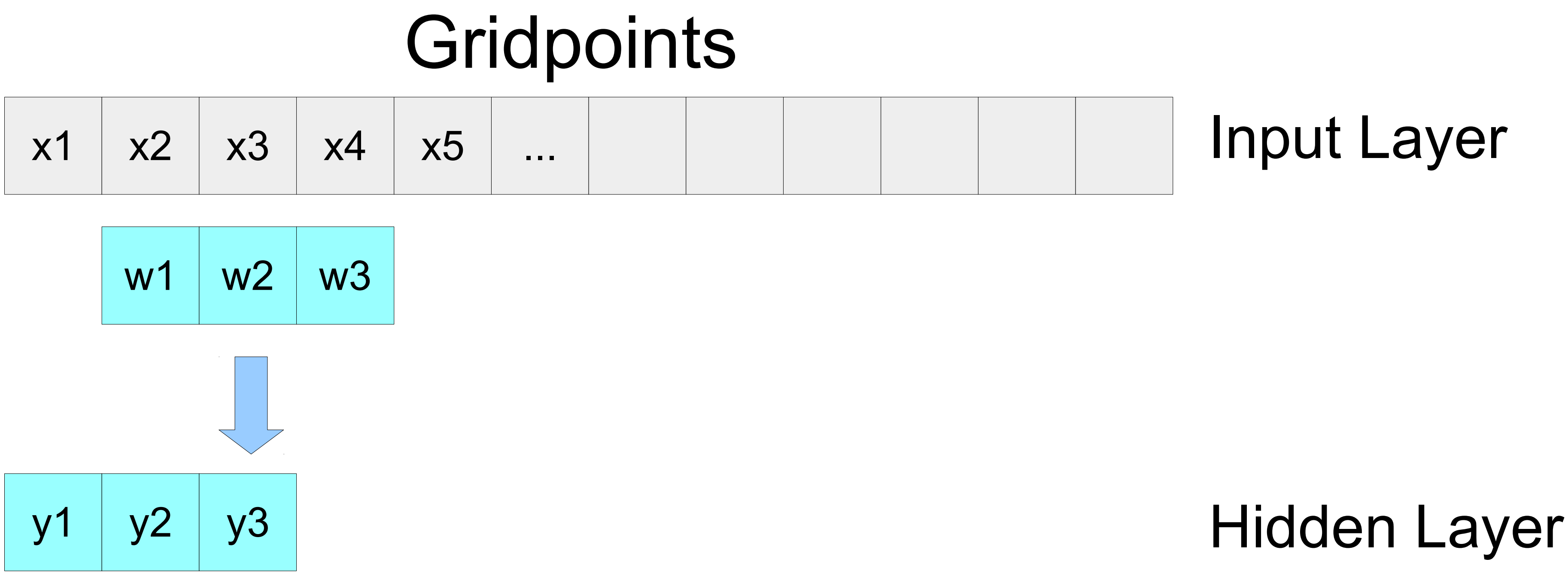
$$y1 = x1*w1 + x1*w2 + x2*w3 + bias$$

Convolutional Networks



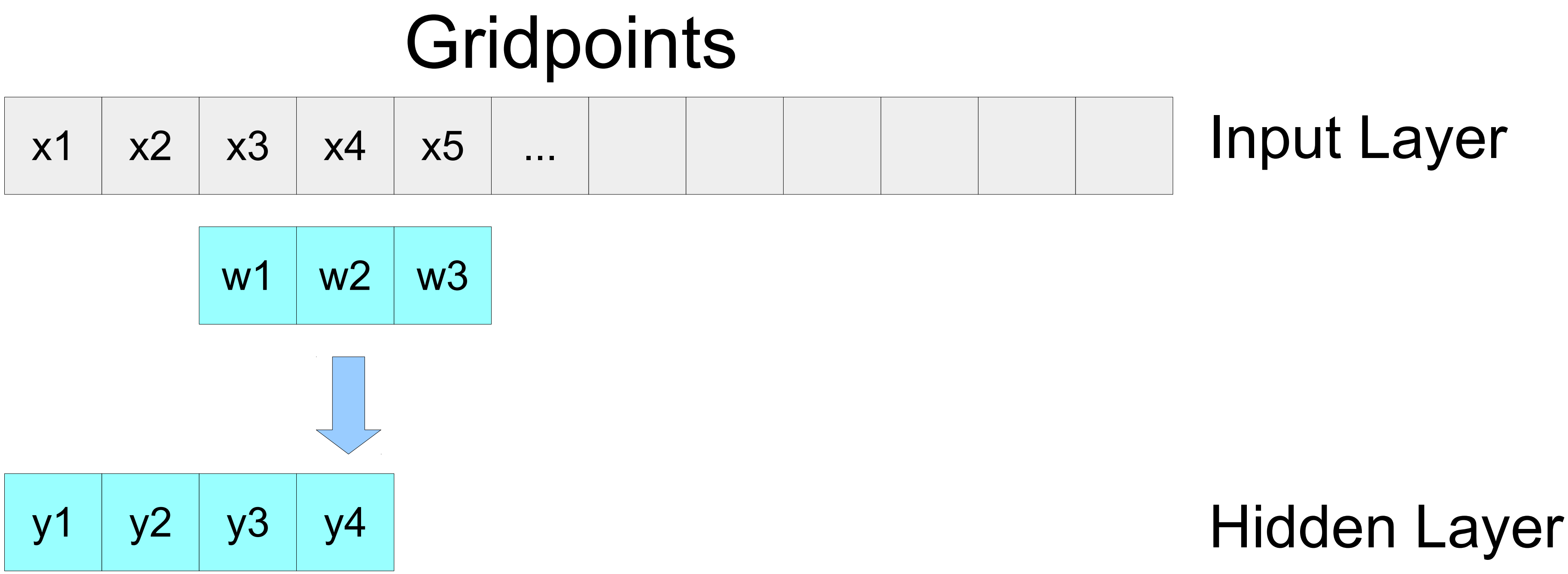
$$y2 = x1*w1 + x2*w2 + x3*w3 + bias$$

Convolutional Networks



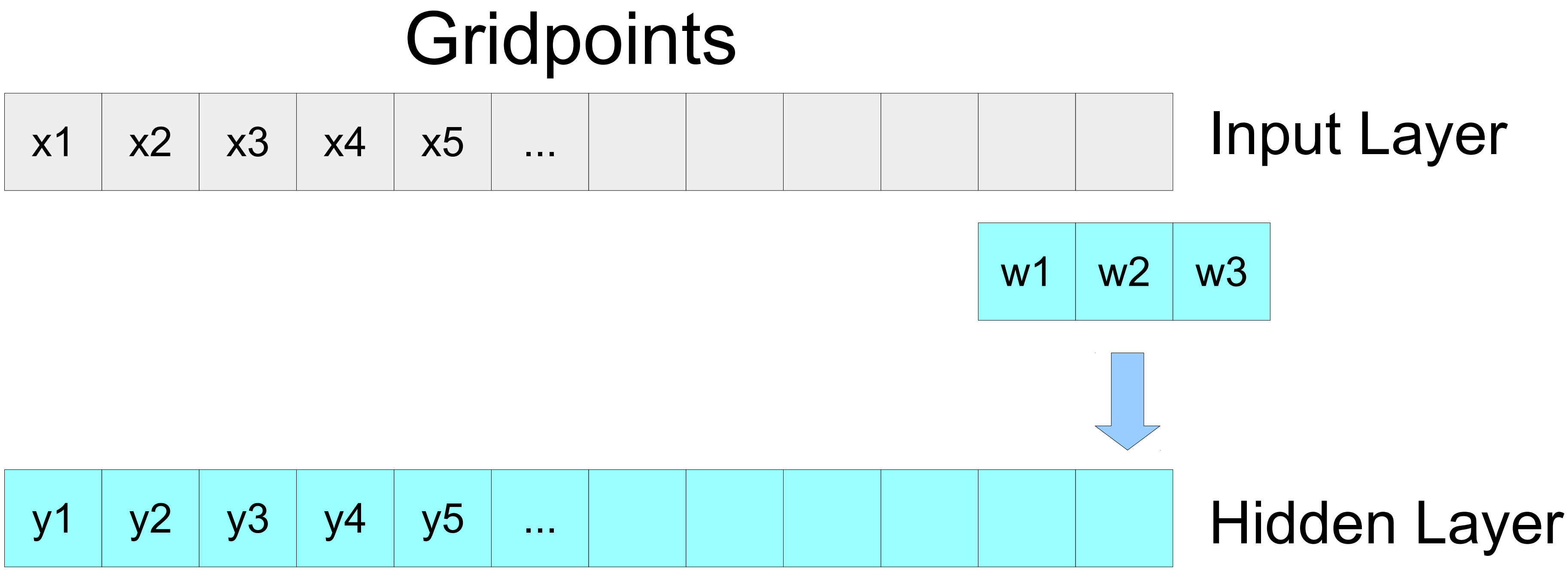
$$y3 = x2*w1 + x3*w2 + x4*w3 + bias$$

Convolutional Networks



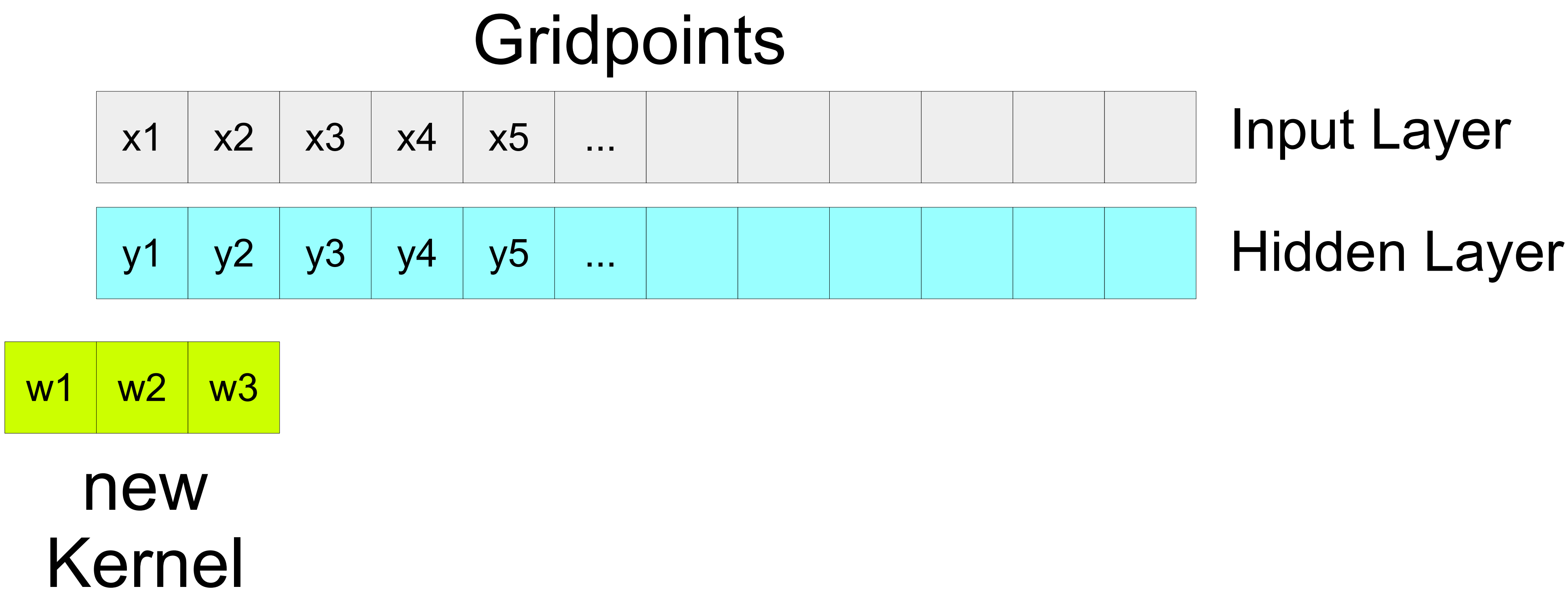
$$y4 = x3*w1 + x4*w2 + x5*w3 + bias$$

Convolutional Networks

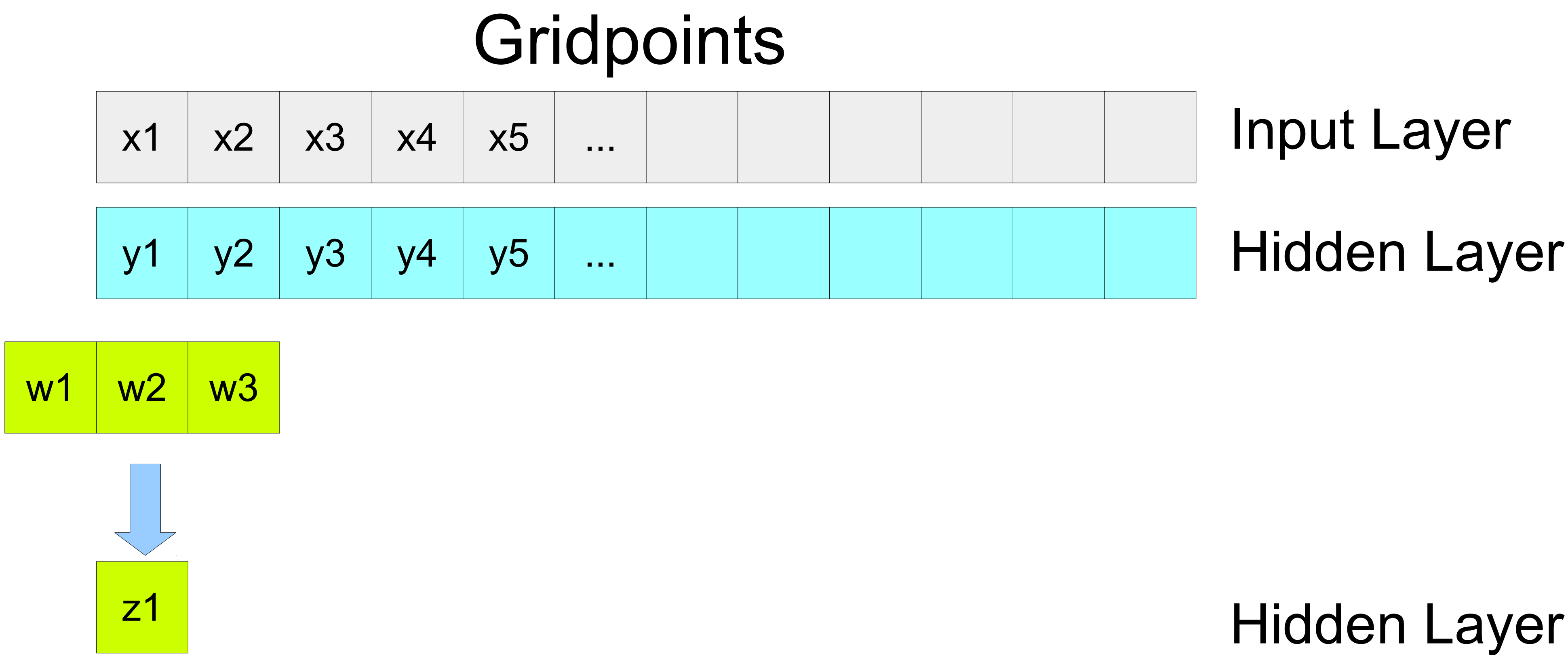


End of layer, repeat.

Convolutional Networks

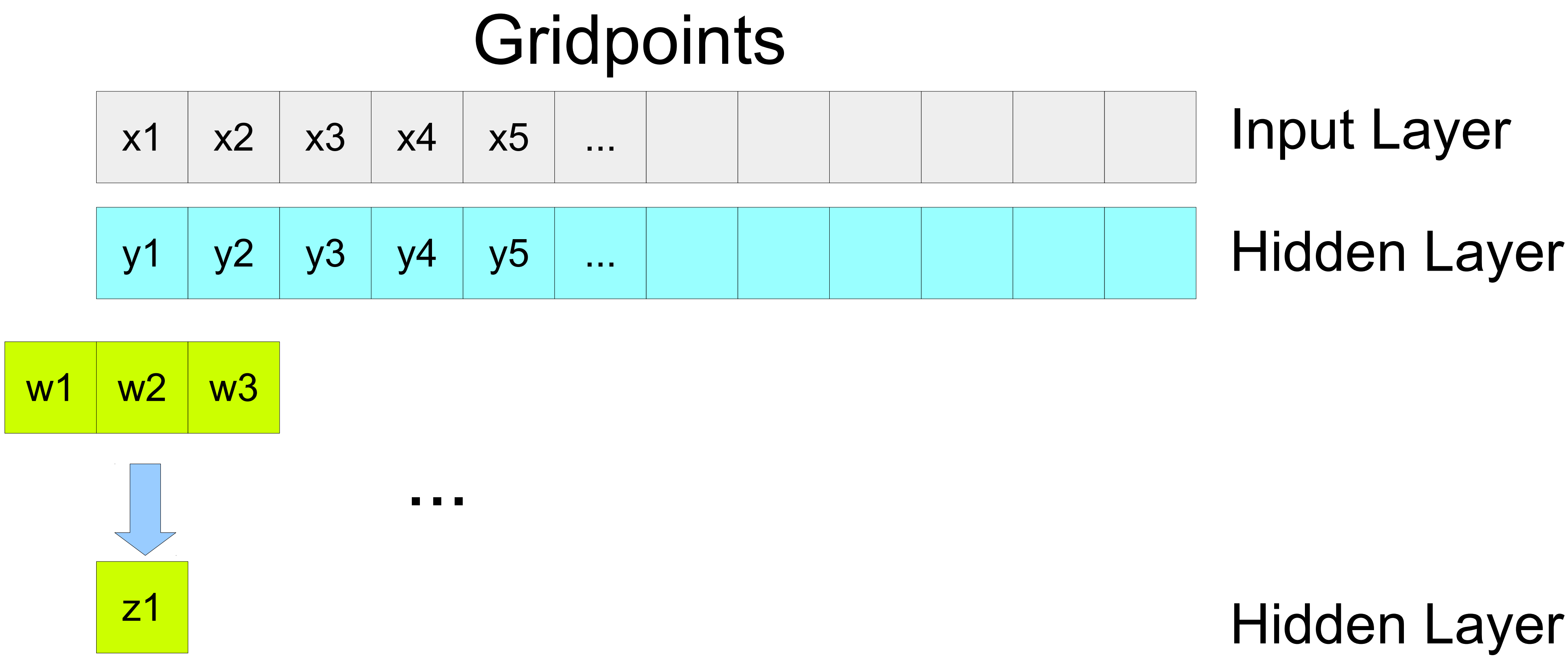


Convolutional Networks



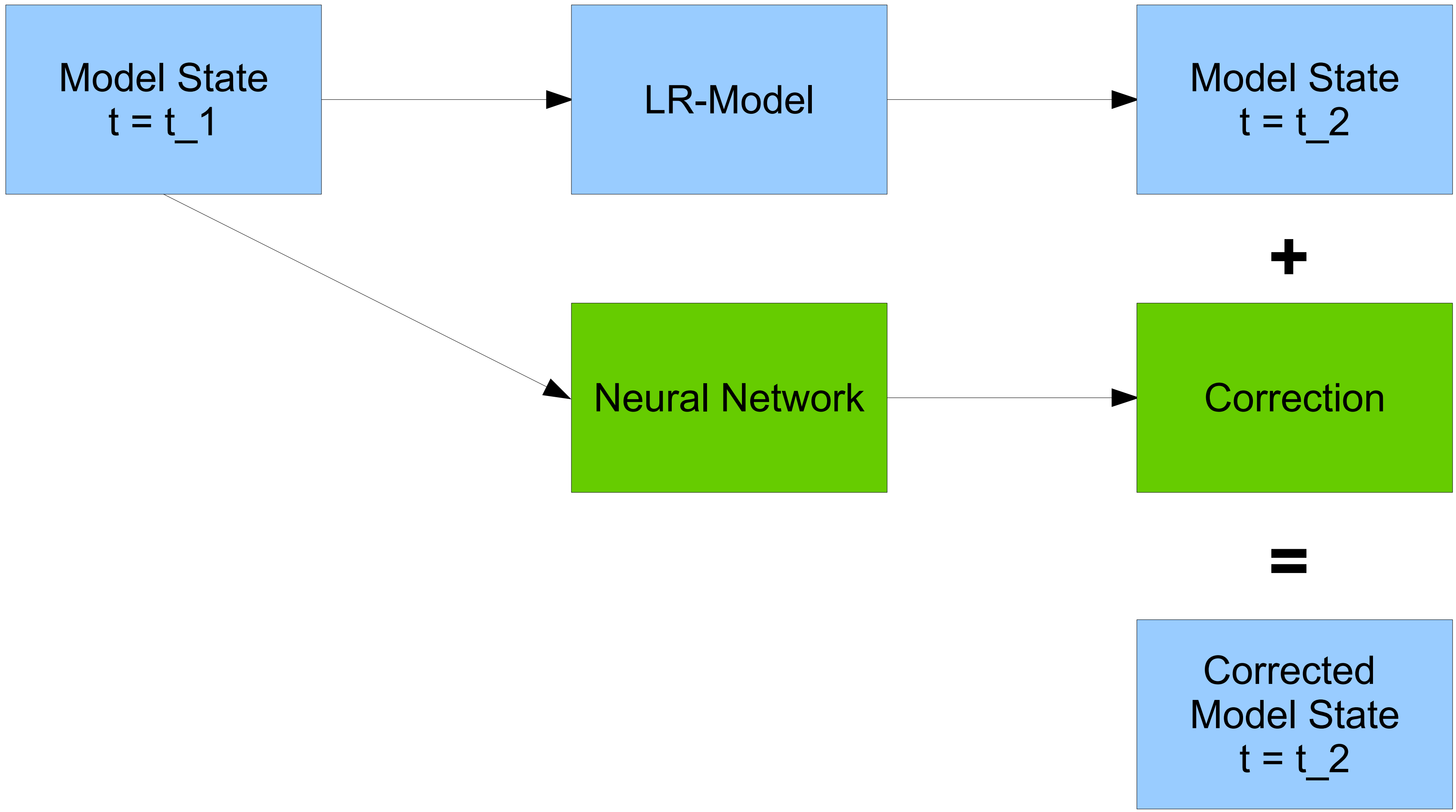
$$z1 = y1*w1 + y1*w2 + y2*w3 + bias$$

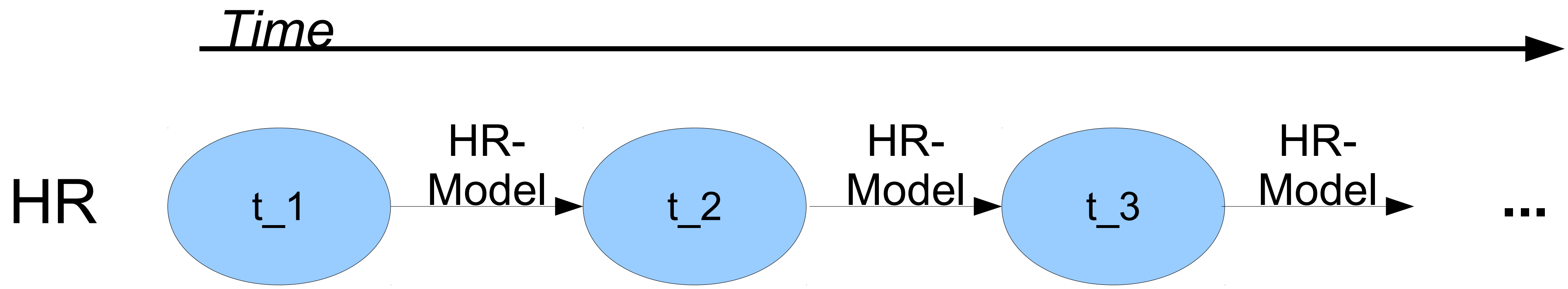
Convolutional Networks



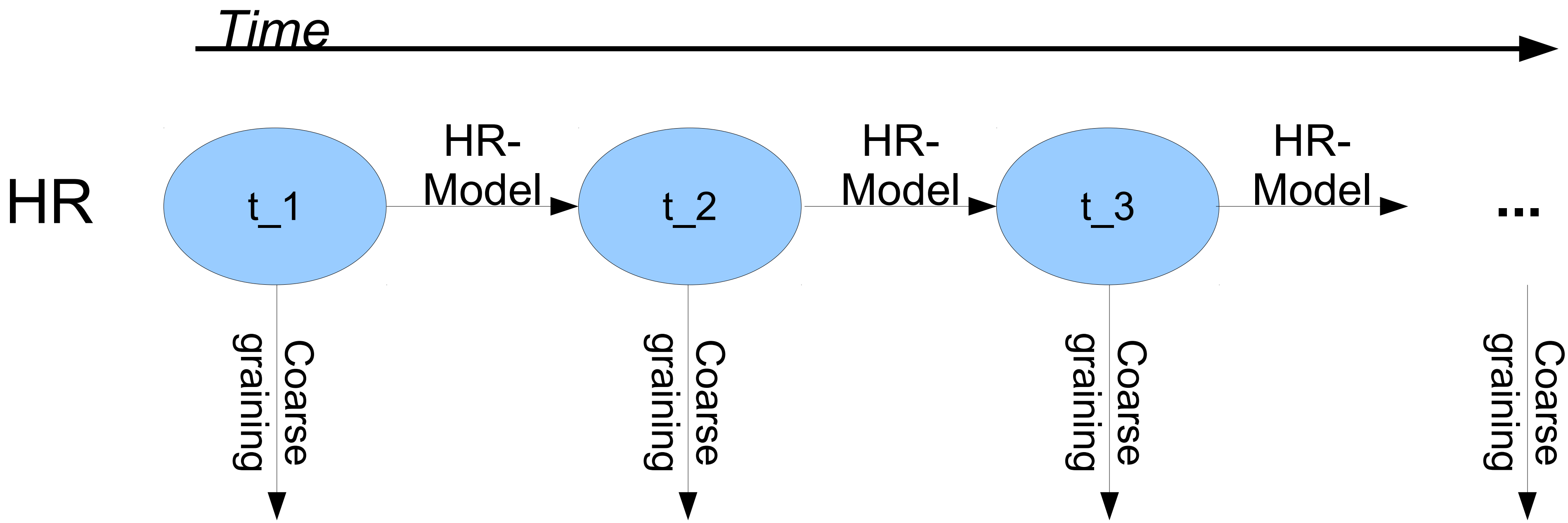
$$z1 = y1*w1 + y1*w2 + y2*w3 + bias$$

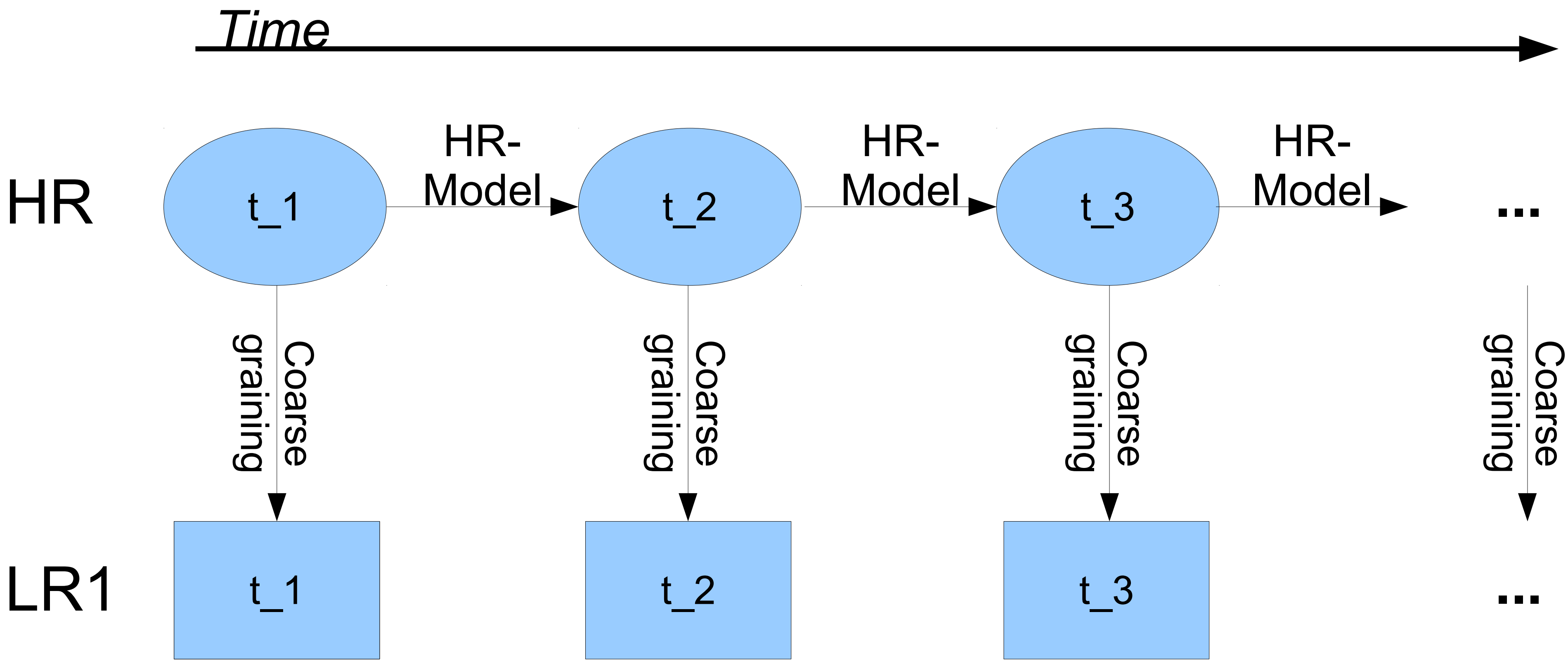
Concept


Model State
t = t_1
LR-Model
Model State
t = t_2
+
Neural Network
Correction
=
Corrected
Model State
t = t_2

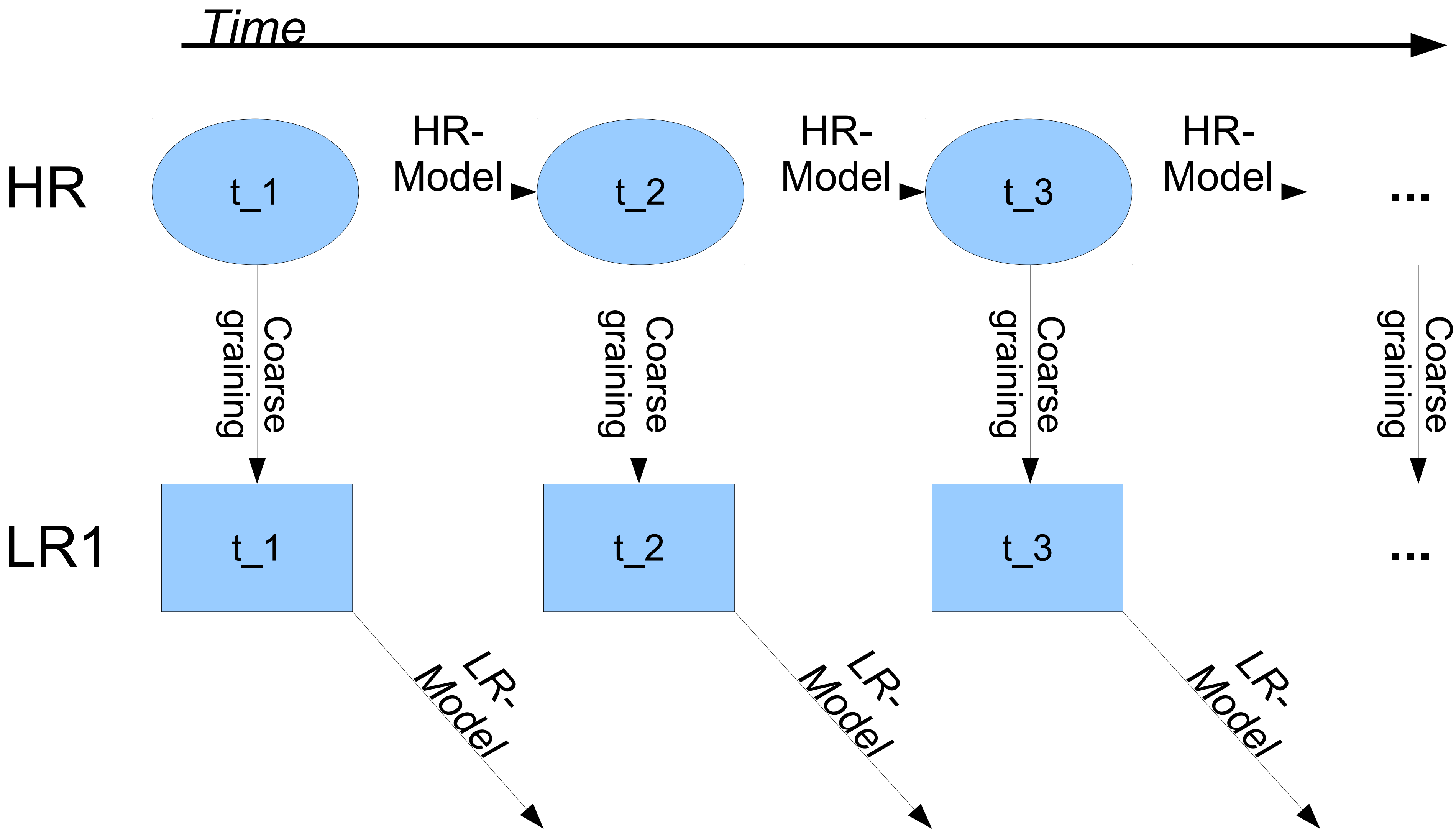


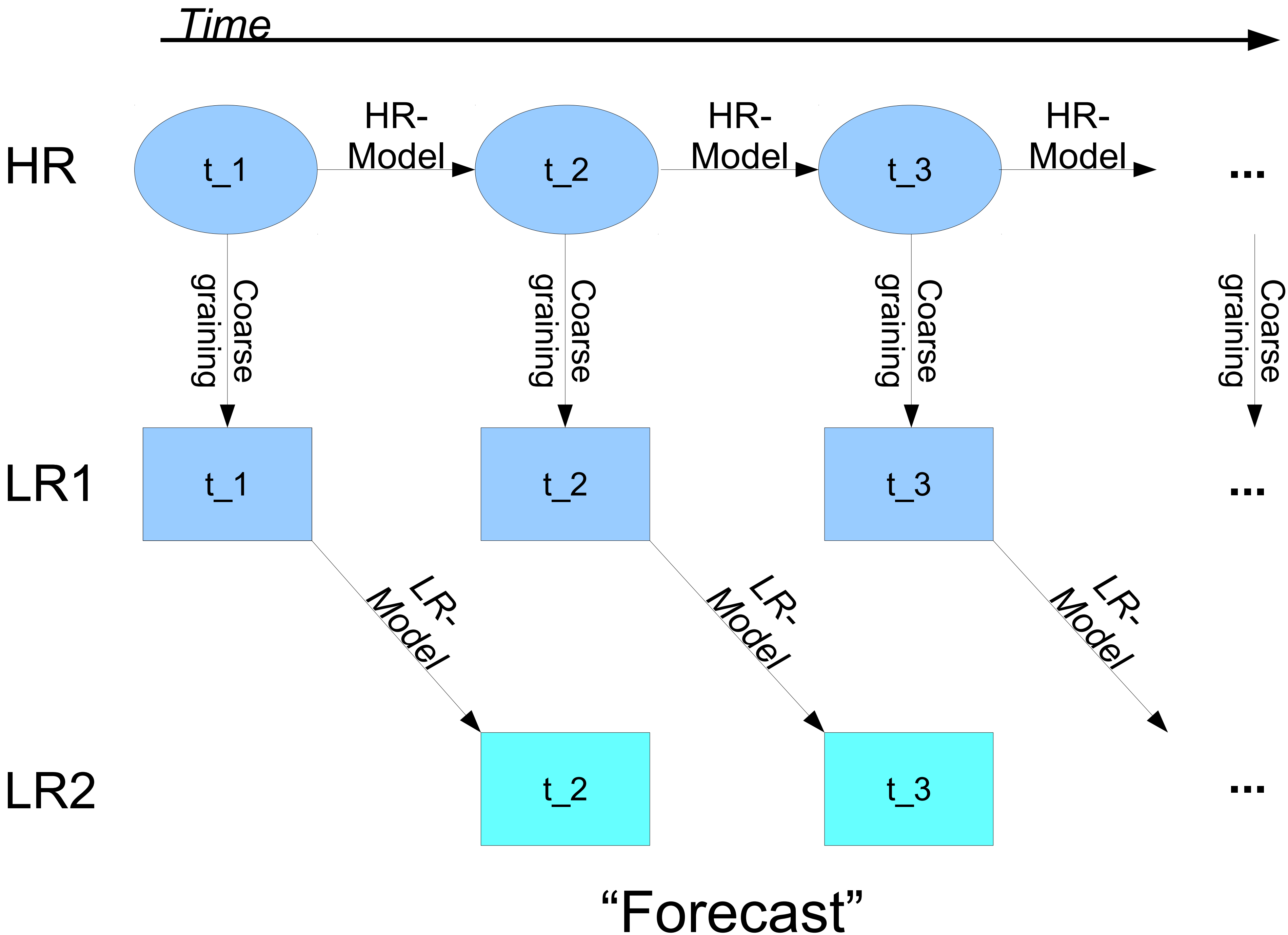
“Nature Run”

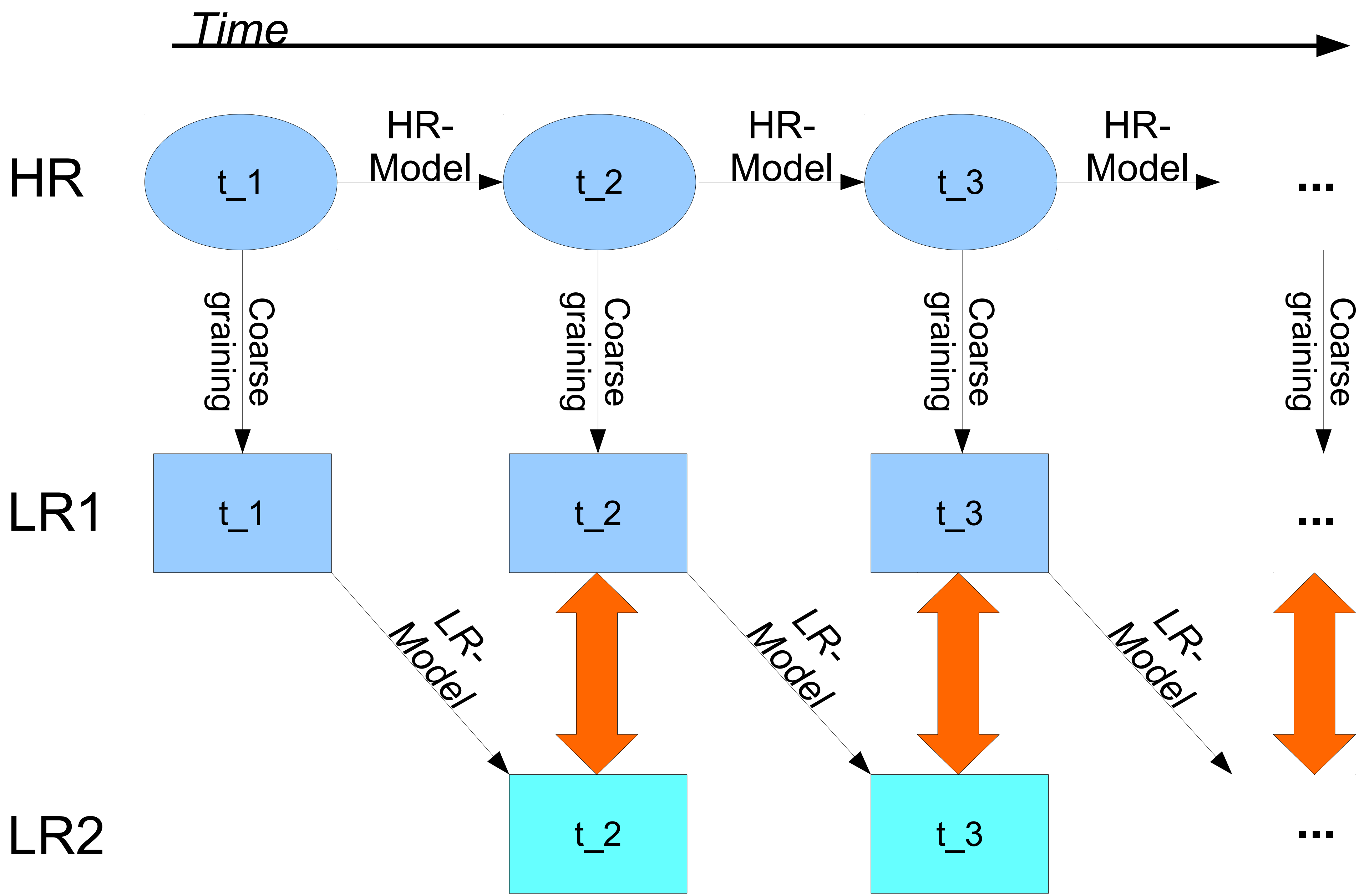

Time
HR
t_1
HR-
Model
t_2
HR-
Model
t_3
HR-
Model
...
Coarse graining
Coarse graining
Coarse graining
Coarse graining


Time
HR
t_1
HR-
Model
t_2
HR-
Model
t_3
HR-
Model
...
Coarse
graining
Coarse
graining
Coarse
graining
Coarse
graining
LR1
t_1
t_2
t_3
...

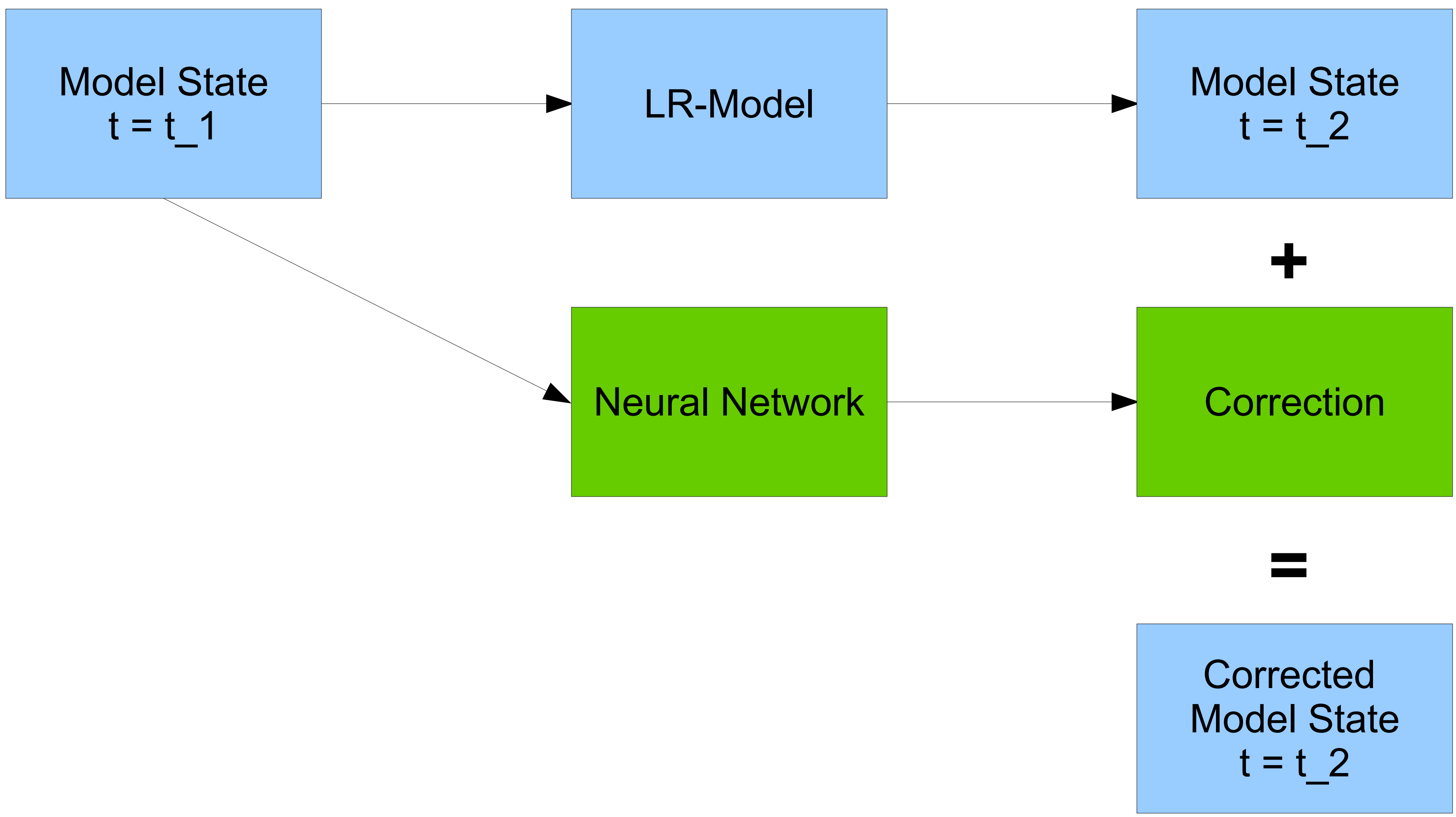
“Model Truth”

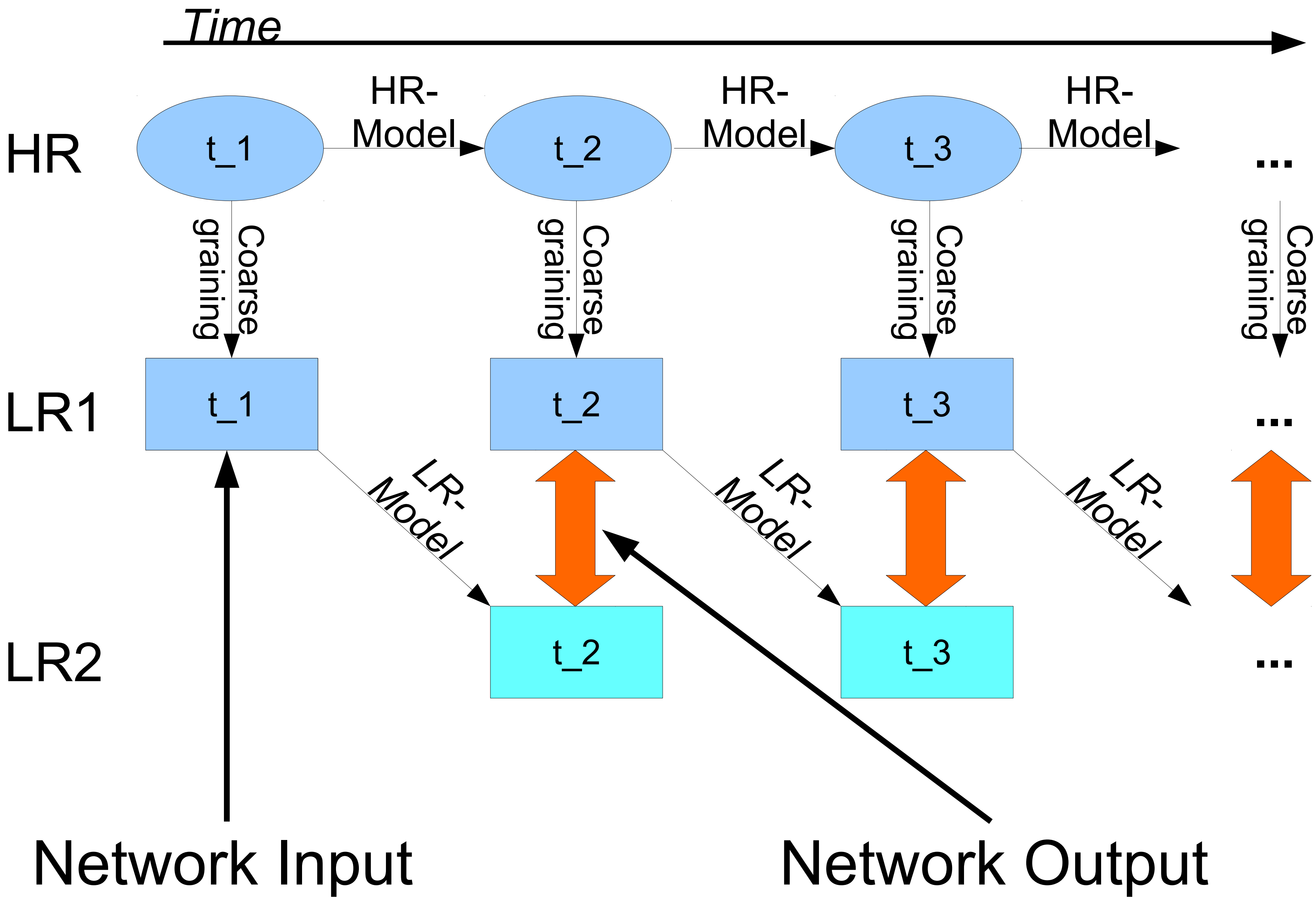

Time
HR
t_1
HR-
Model
t_2
HR-
Model
t_3
HR-
Model
...
Coarse
graining
Coarse
graining
Coarse
graining
Coarse
graining
LR1
t_1
t_2
t_3
...
LR-
Model
LR-
Model
LR-
Model


Time
HR
HR-Model
t_1
t_2
t_3
...
Coarse graining
LR1
LR-Model
LR2
"Forecast"

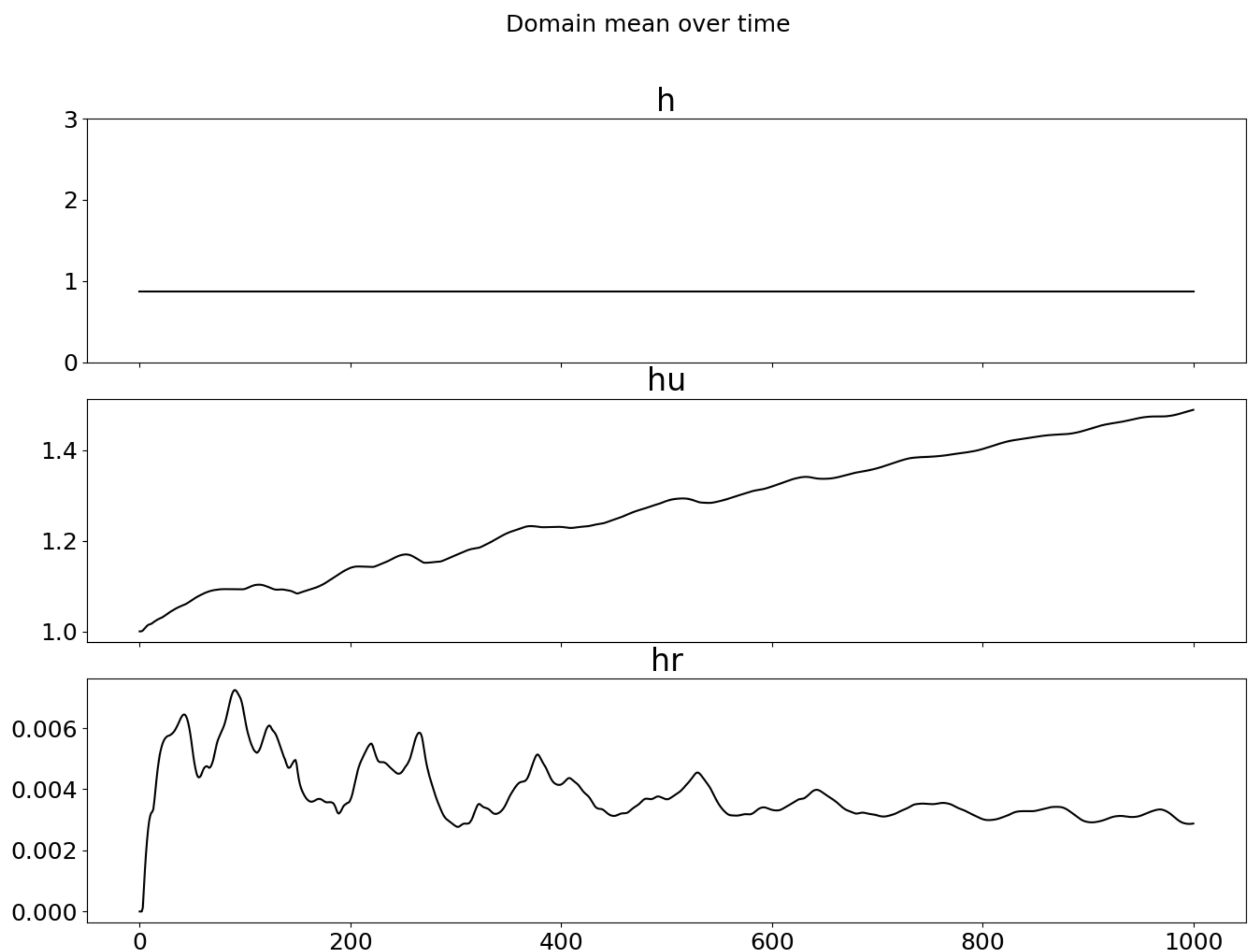

Time
HR
HR-
Model
HR-
Model
HR-
Model
t_1
t_2
t_3
...
Coarse
graining
Coarse
graining
Coarse
graining
Coarse
graining
LR1
t_1
t_2
t_3
...
LR-
Model
LR-
Model
LR-
Model
LR2
t_2
t_3
...

Concept


Model State
t = t_1
LR-Model
Model State
t = t_2
+
Neural Network
Correction
=
Corrected
Model State
t = t_2


Time
HR
HR-
Model
t_1
t_2
t_3
...
Coarse
graining
LR1
LR-
Model
LR2
Network Input
Network Output

Training Data Set


Domain mean over time
h
3
2
1
0
hu
1.4
1.2
1.0
hr
0.006
0.004
0.002
0.000
0
200
400
600
800
1000

Training Data Set

- Ensemble:
  - 20 Members (training/validation split: 15/5)
  - Perturbation: random Orography
- HR: 800 gridpoints, LR: 200 gridpoints
- t_measure = 0.001, Nmeas = 5000
- Only chaotic dynamics (see last part)
- Computational variables, orography added as 4th channel

Network Architecture

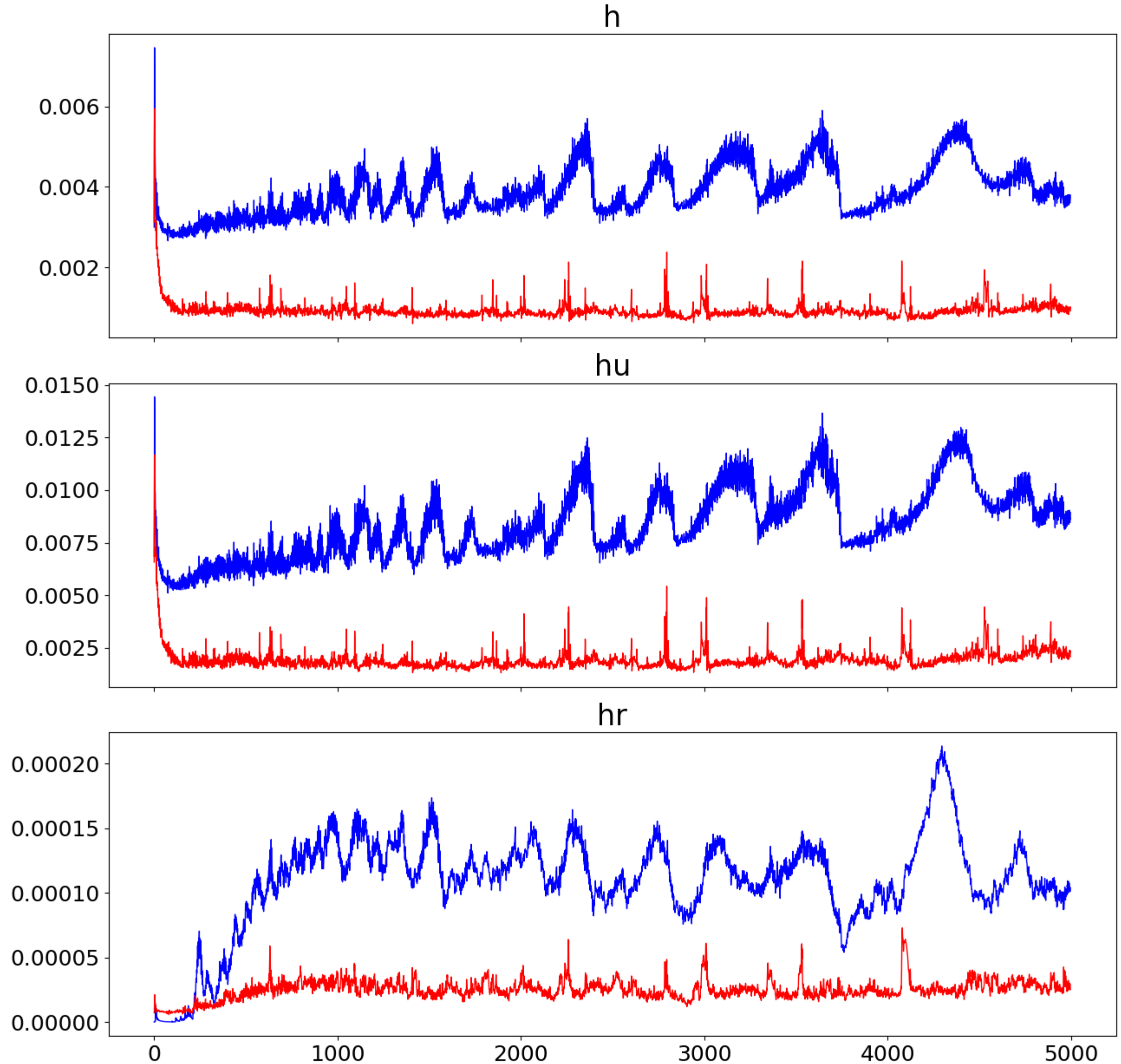
- 1-D Convolutional Network
- 5 hidden layers
- Activation: "relu"
- Kernel size: 10 gridpoints
- 40 Filters

Results

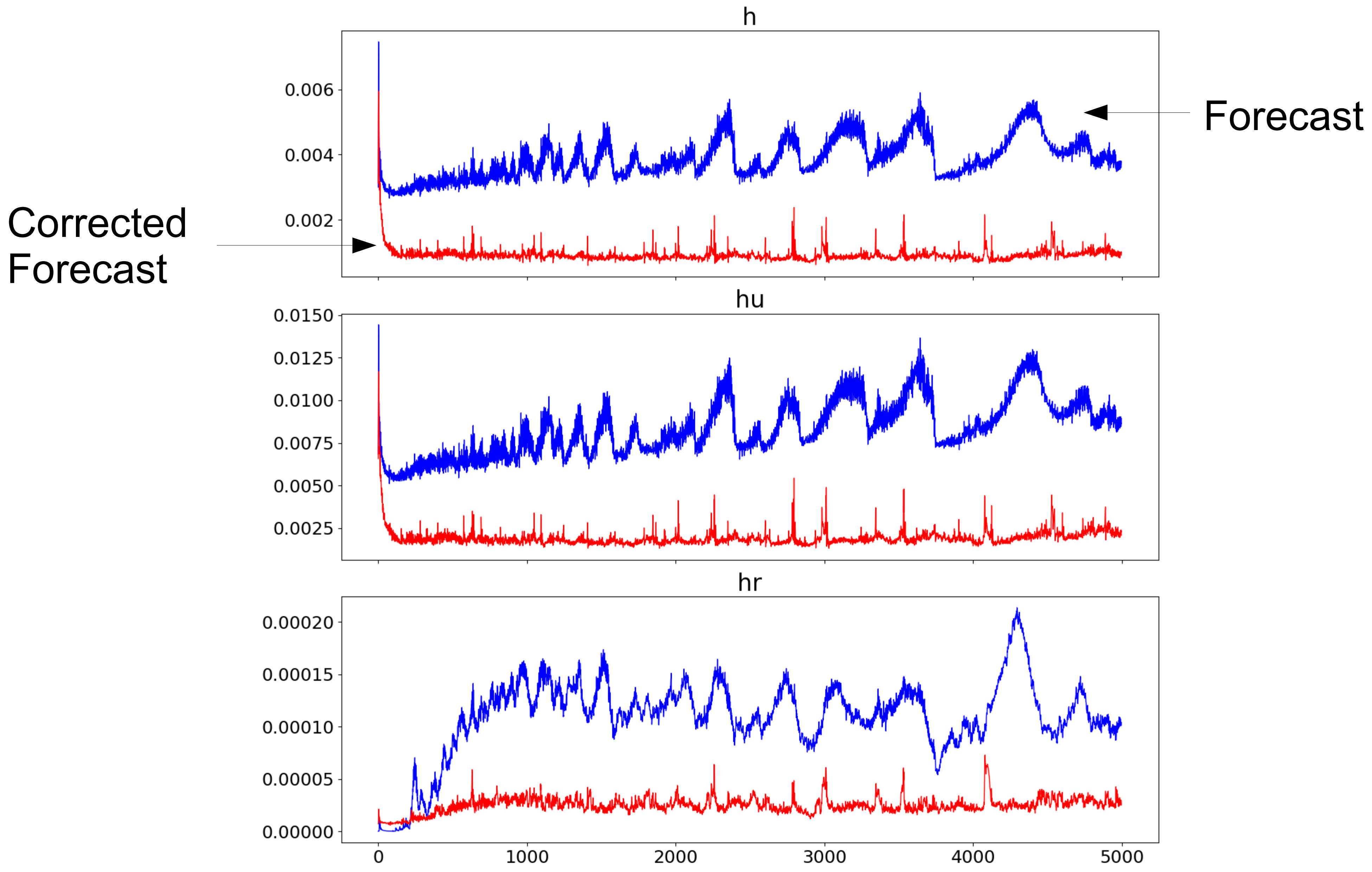
Results

- **Single step prediction**
  - RMSE(truth – forecast)
  - RMSE(truth – corrected forecast)

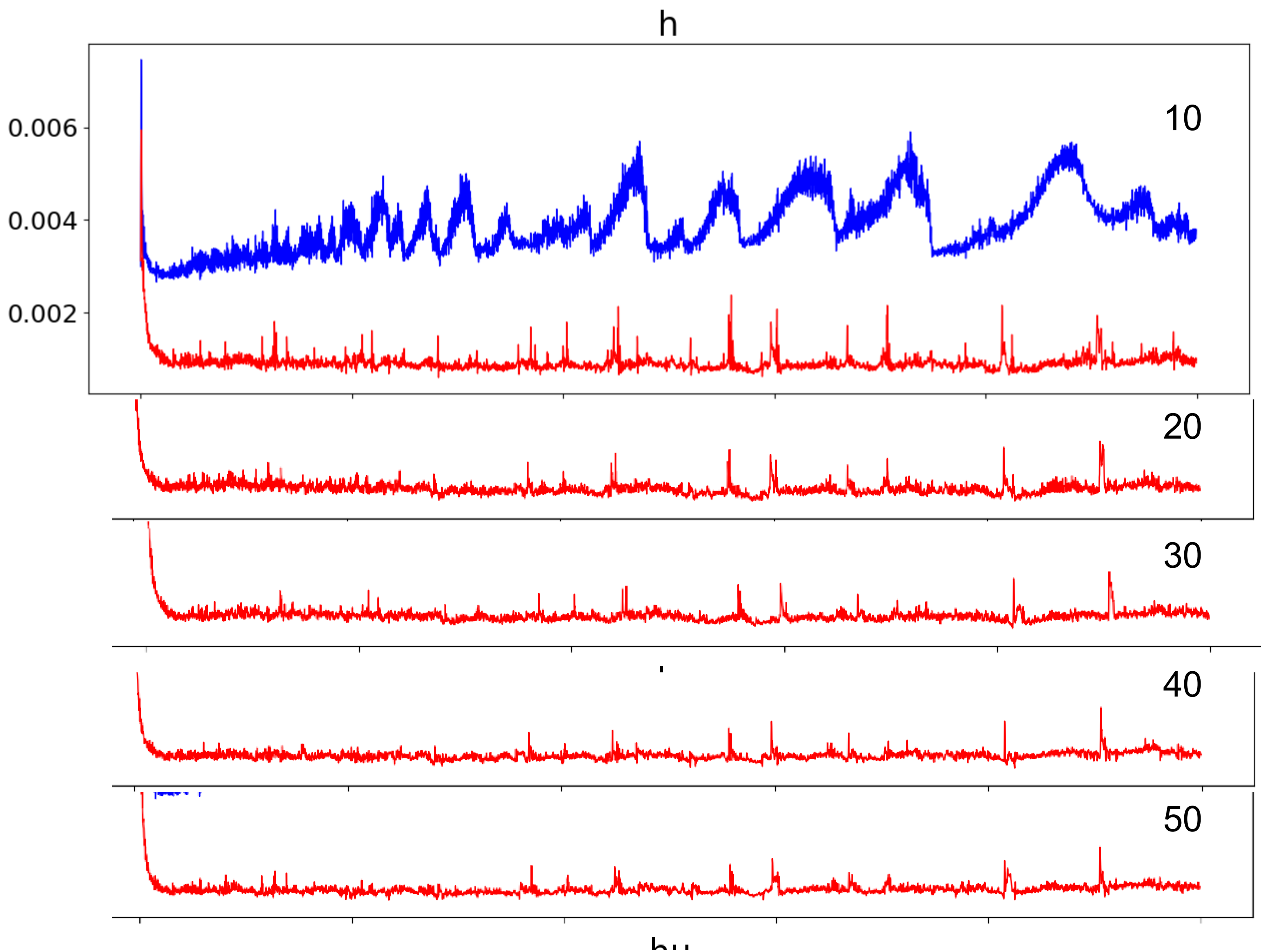
Single Step Prediction



Single Step Prediction

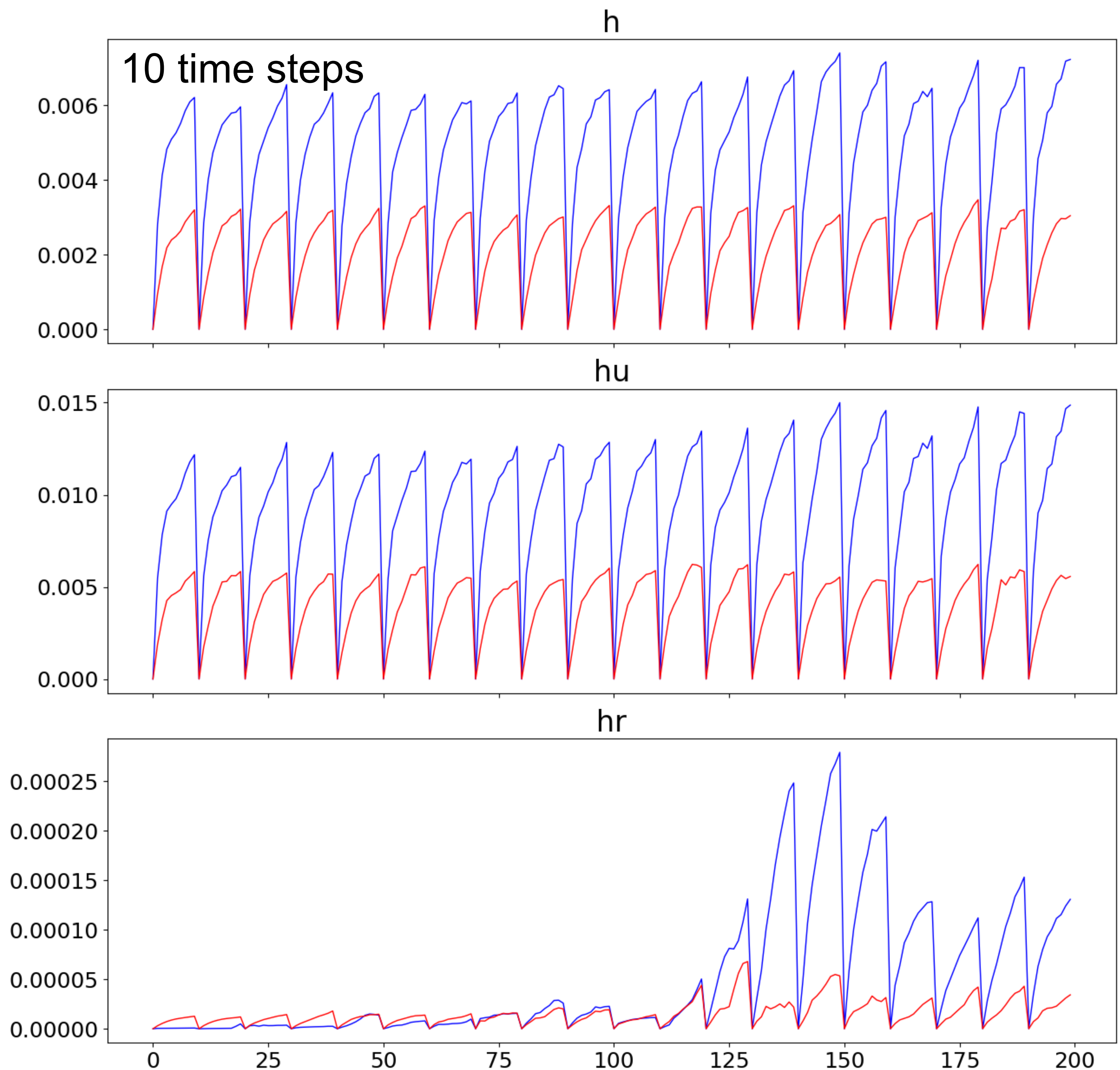


Increasing number of filters reduces “error noise“

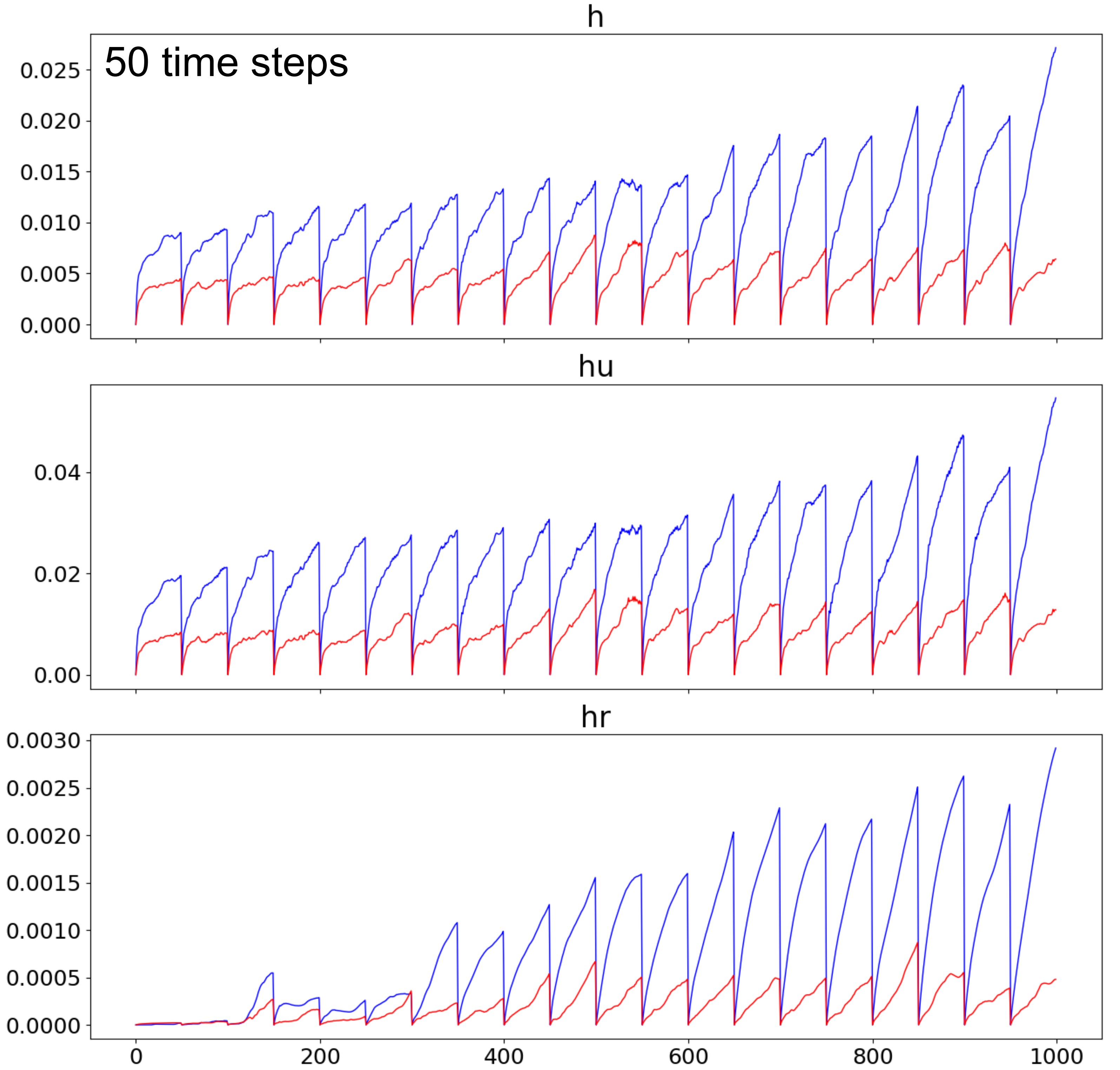


Results

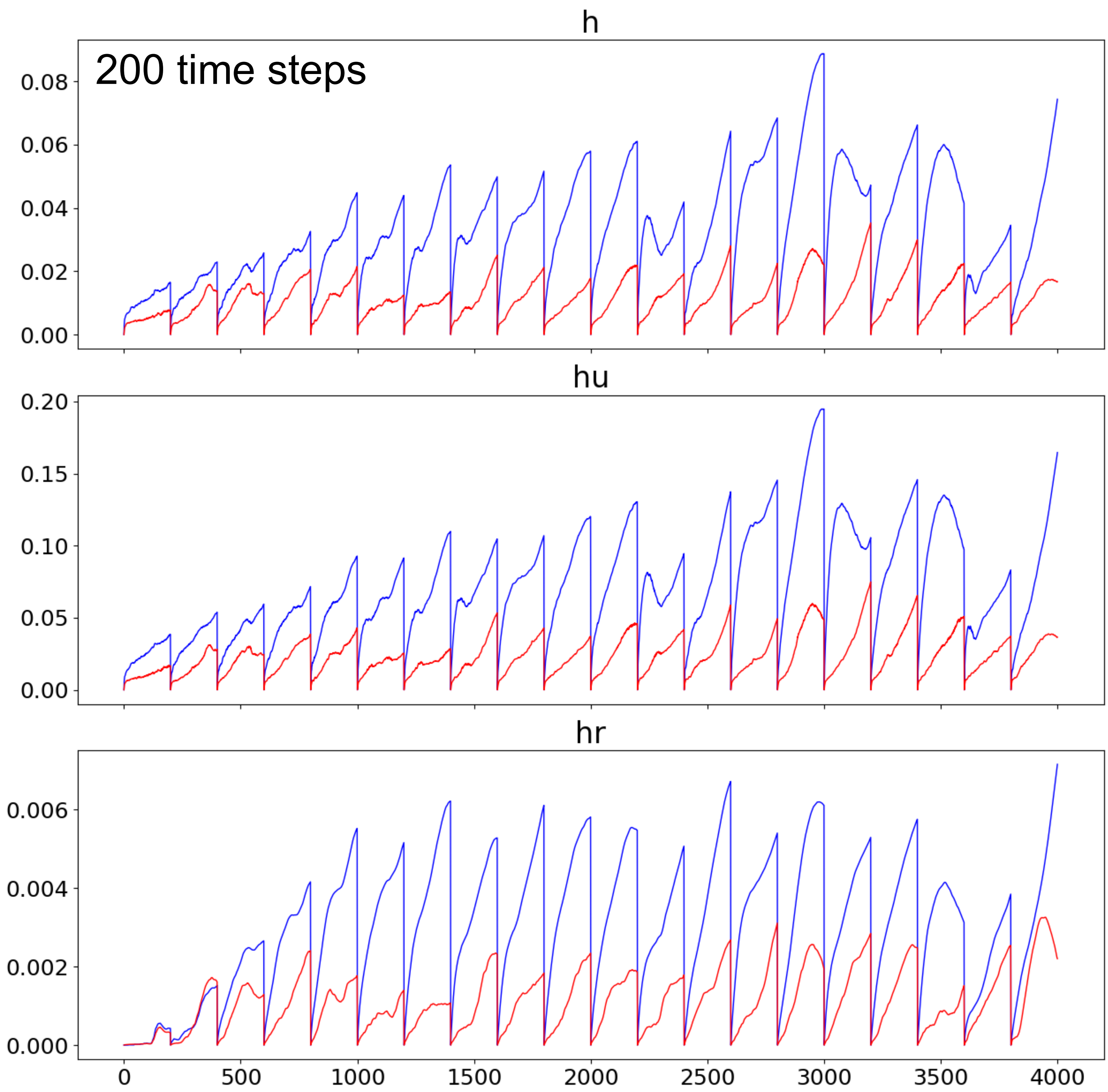
- Single step prediction
  - RMSE(truth – forecast)
  - RMSE(truth – corrected forecast)
- **Online version**
  - Different lead times

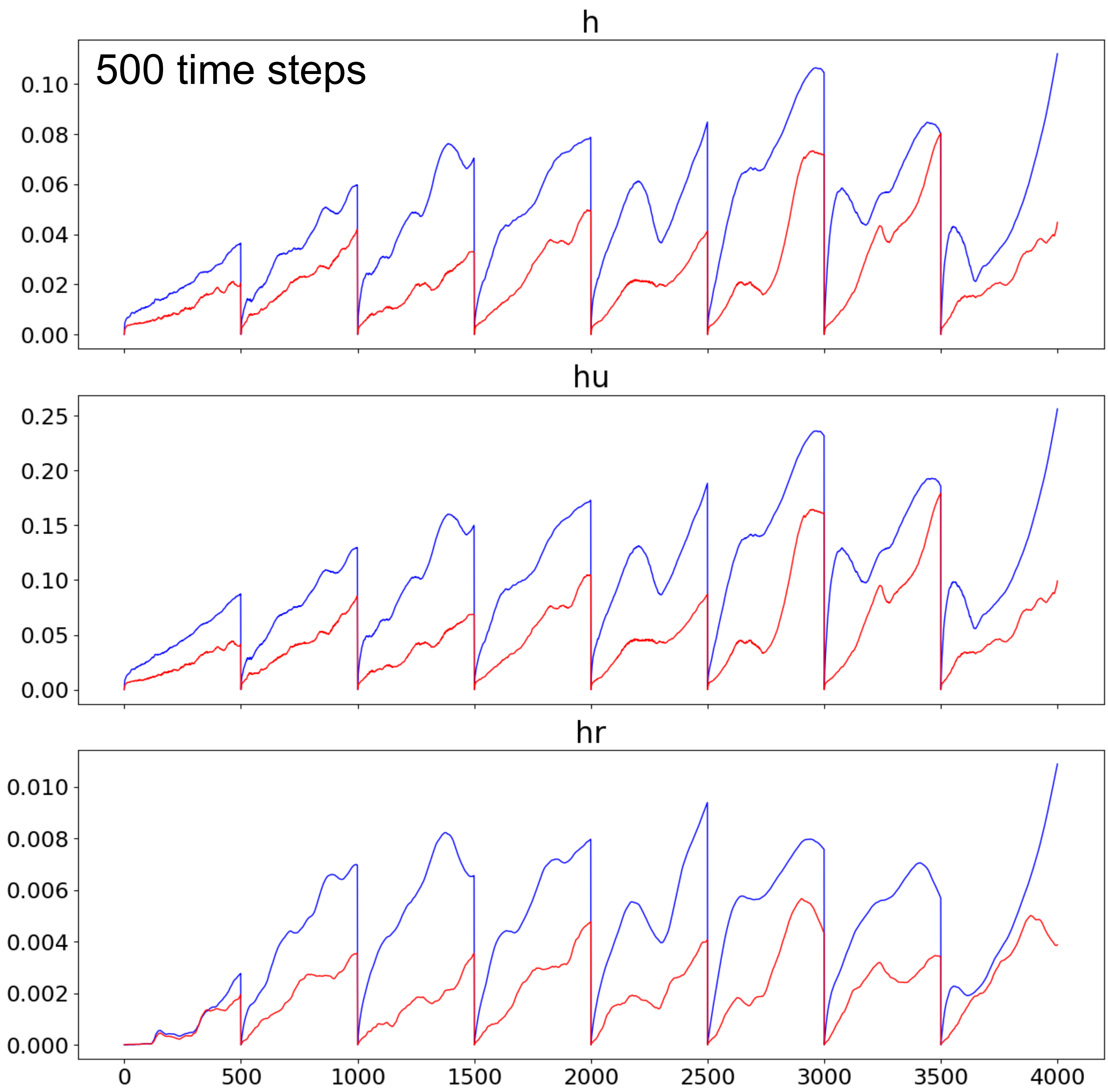


h
10 time steps
0.006
0.004
0.002
0.000
hu
0.015
0.010
0.005
0.000
hr
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0
25
50
75
100
125
150
175
200

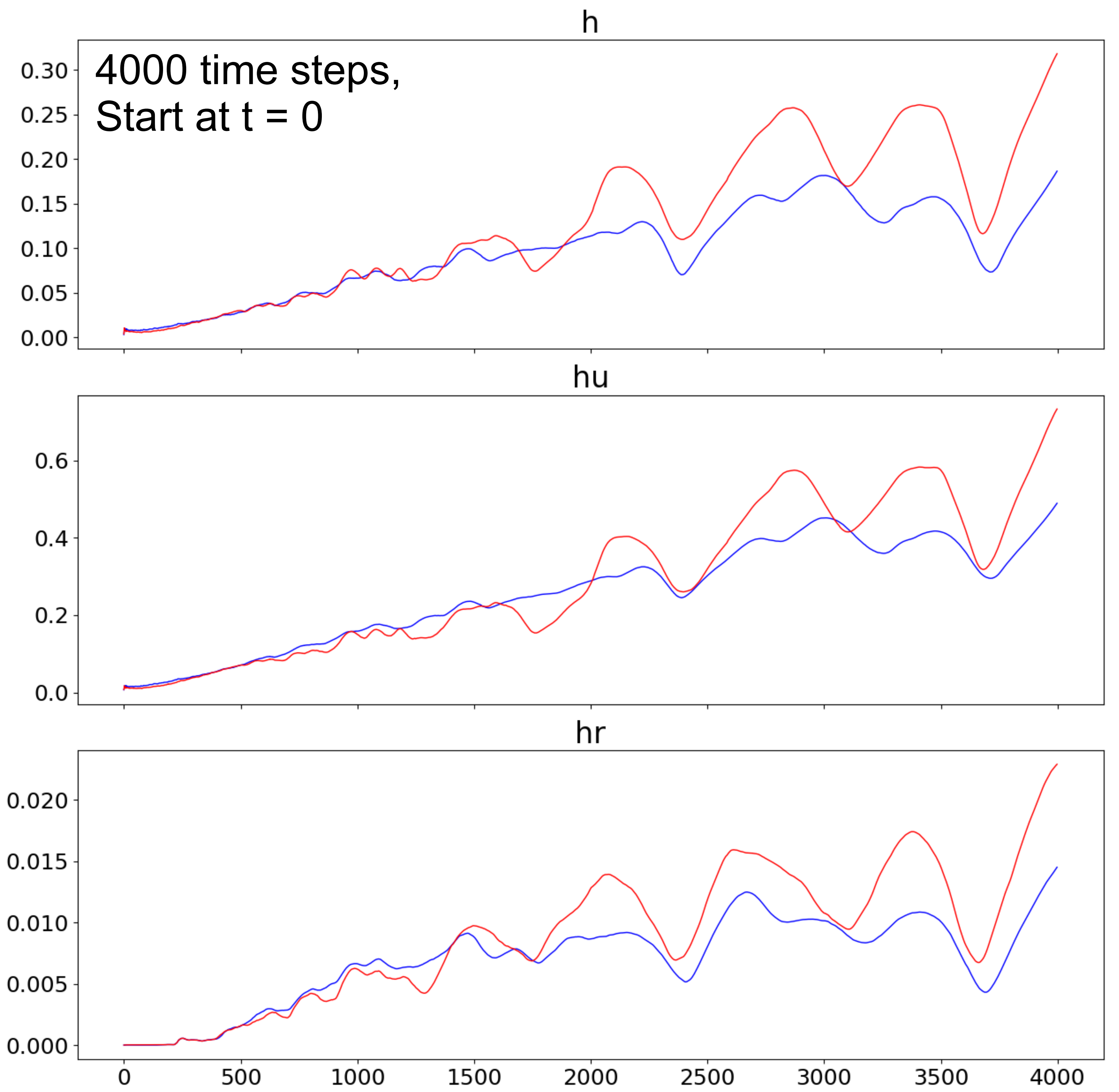


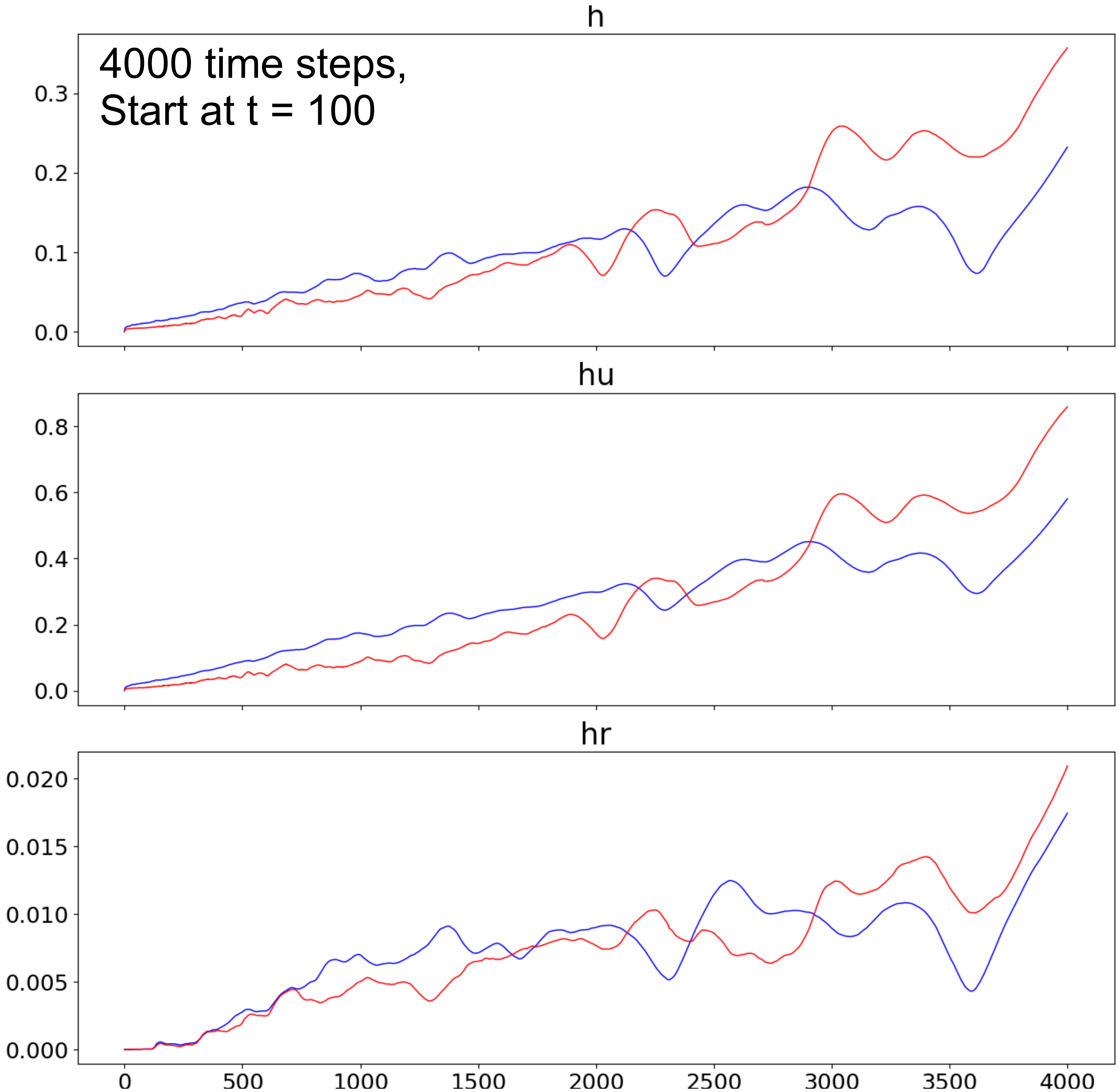
h
50 time steps
0.025
0.020
0.015
0.010
0.005
0.000
hu
0.04
0.02
0.00
hr
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
200
400
600
800
1000


h
200 time steps
0.00
0.02
0.04
0.06
0.08
hu
0.00
0.05
0.10
0.15
0.20
hr
0.000
0.002
0.004
0.006
0
500
1000
1500
2000
2500
3000
3500
4000



h
500 time steps
0.00
0.02
0.04
0.06
0.08
0.10
hu
0.00
0.05
0.10
0.15
0.20
0.25
hr
0.000
0.002
0.004
0.006
0.008
0.010
0
500
1000
1500
2000
2500
3000
3500
4000


h
4000 time steps,
Start at t = 0
0.30
0.25
0.20
0.15
0.10
0.05
0.00
hu
0.6
0.4
0.2
0.0
hr
0.020
0.015
0.010
0.005
0.000
0
500
1000
1500
2000
2500
3000
3500
4000

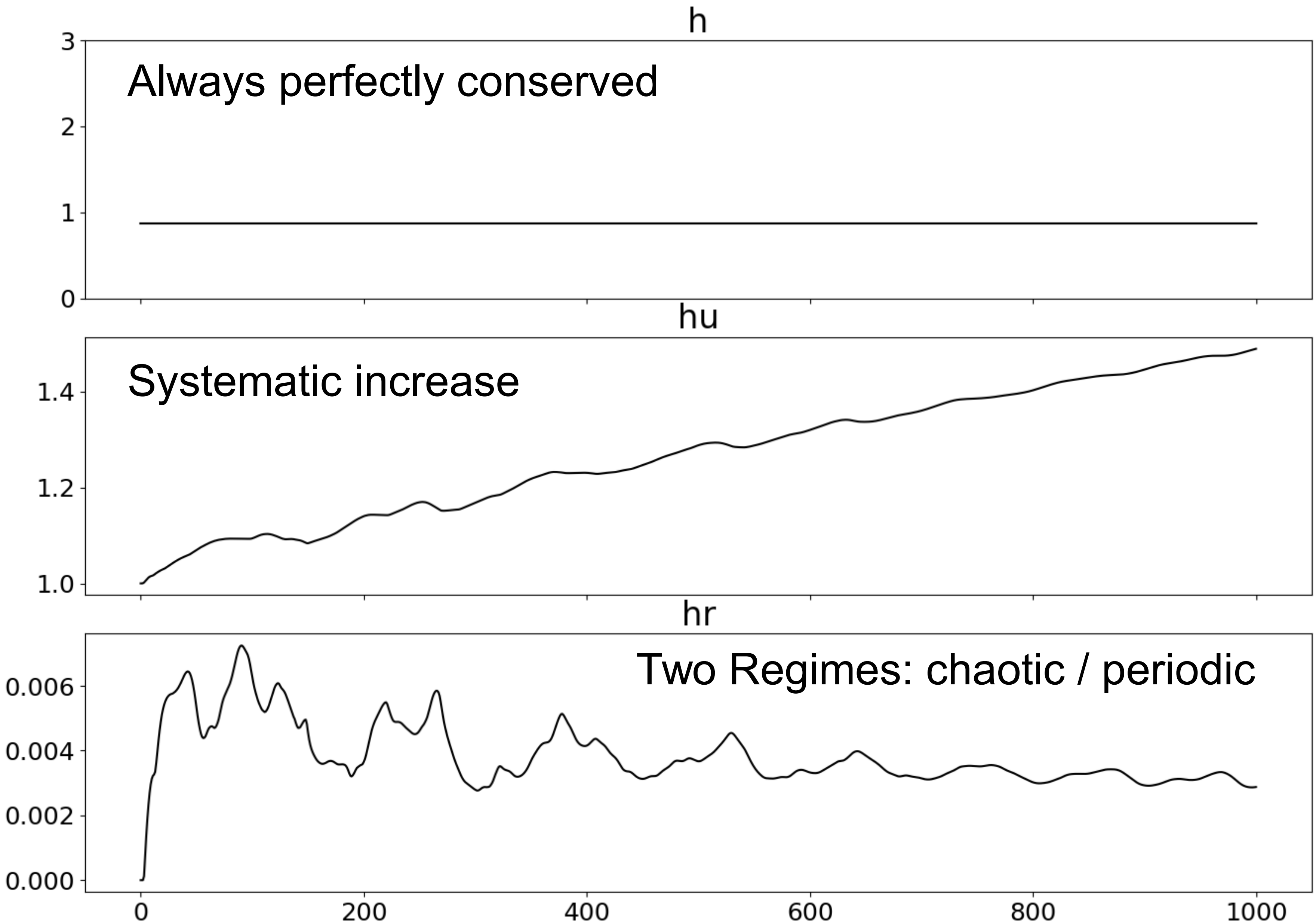

h
4000 time steps,
Start at t = 100
0.0
0.1
0.2
0.3
hu
0.0
0.2
0.4
0.6
0.8
hr
0.000
0.005
0.010
0.015
0.020
0
500
1000
1500
2000
2500
3000
3500
4000

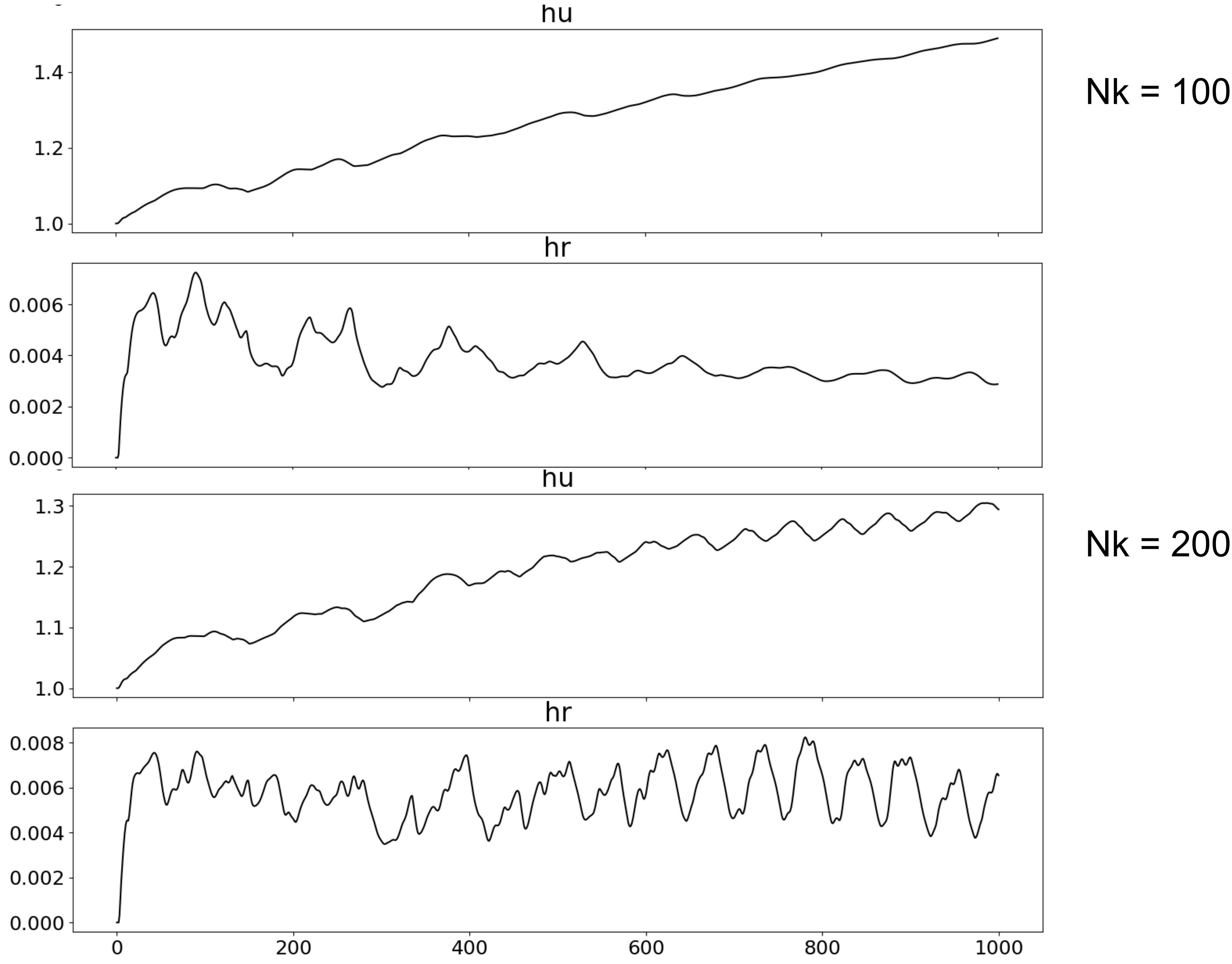
Preliminary Results

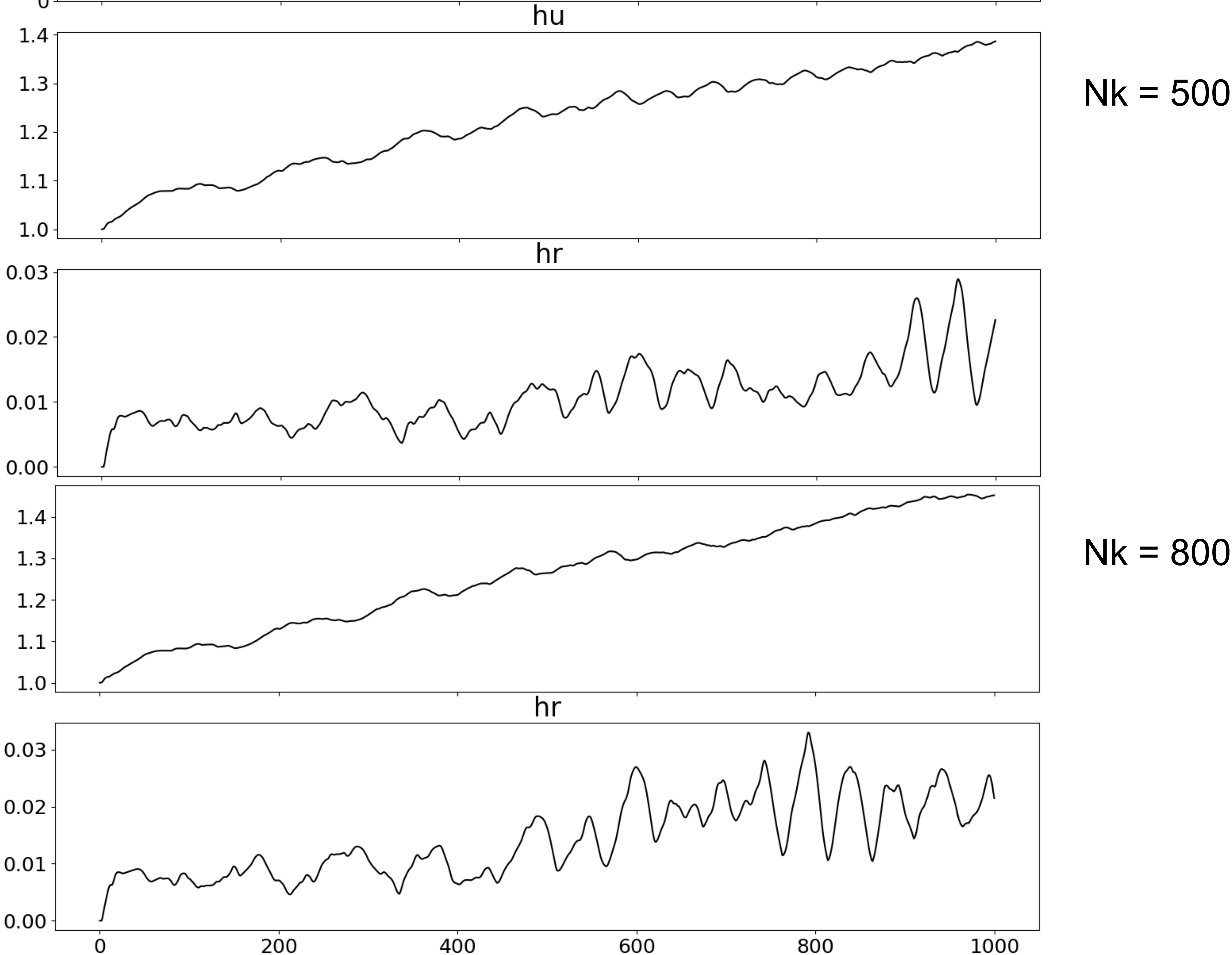
- Forecast improvement seems possible over extended lead times
- No feedback "disaster"
- Longer lead times -> greater improvement variability
  - Dependence on specific situation?
  - ...

ModRSW Model Behaviour: Long Term Simulations

Time Series of Domain Mean (Nk = 100)




hu
hr
hu
hr
Nk = 100
Nk = 200
1.4
1.2
1.0
0.006
0.004
0.002
0.000
1.3
1.2
1.1
1.0
0.008
0.006
0.004
0.002
0.000
0
200
400
600
800
1000



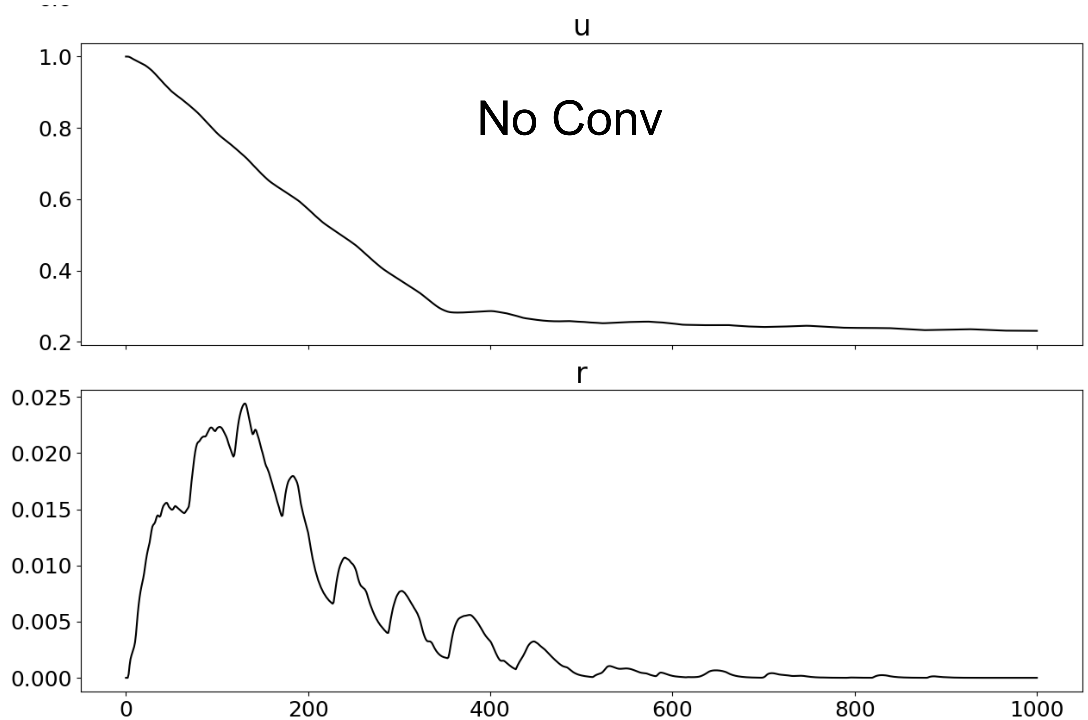
hu
1.4
1.3
1.2
1.1
1.0
hr
0.03
0.02
0.01
0.00
Nk = 500
1.4
1.3
1.2
1.1
1.0
hr
0.03
0.02
0.01
0.00
0
200
400
600
800
1000
Nk = 800

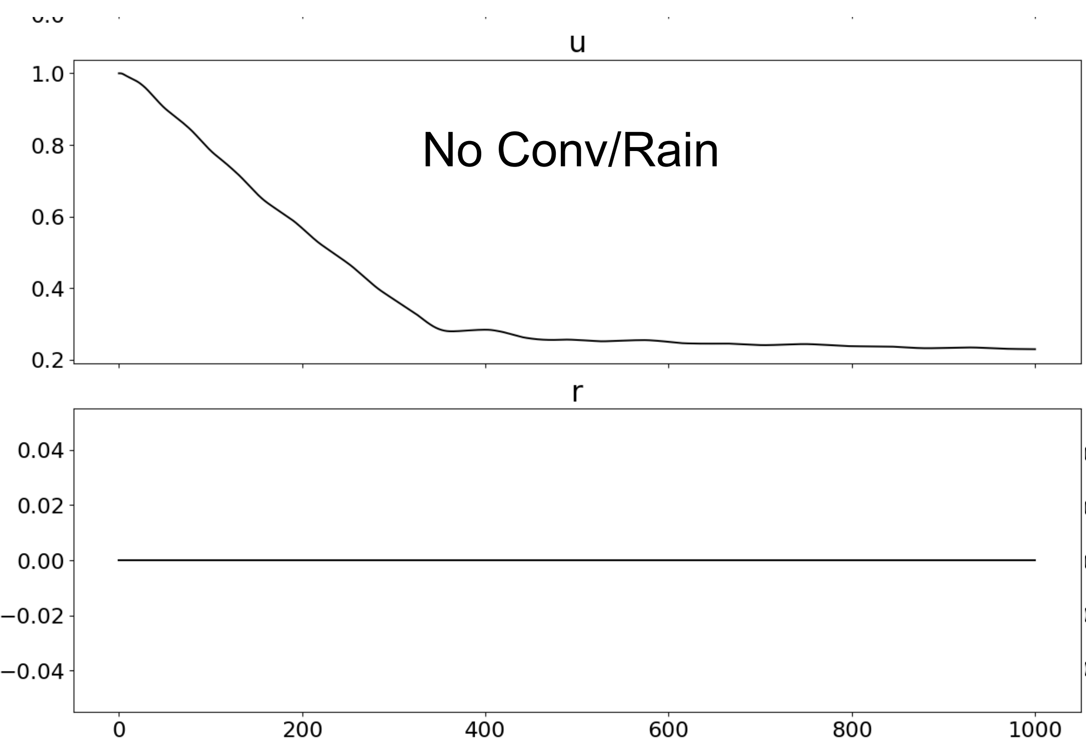
Thank you!

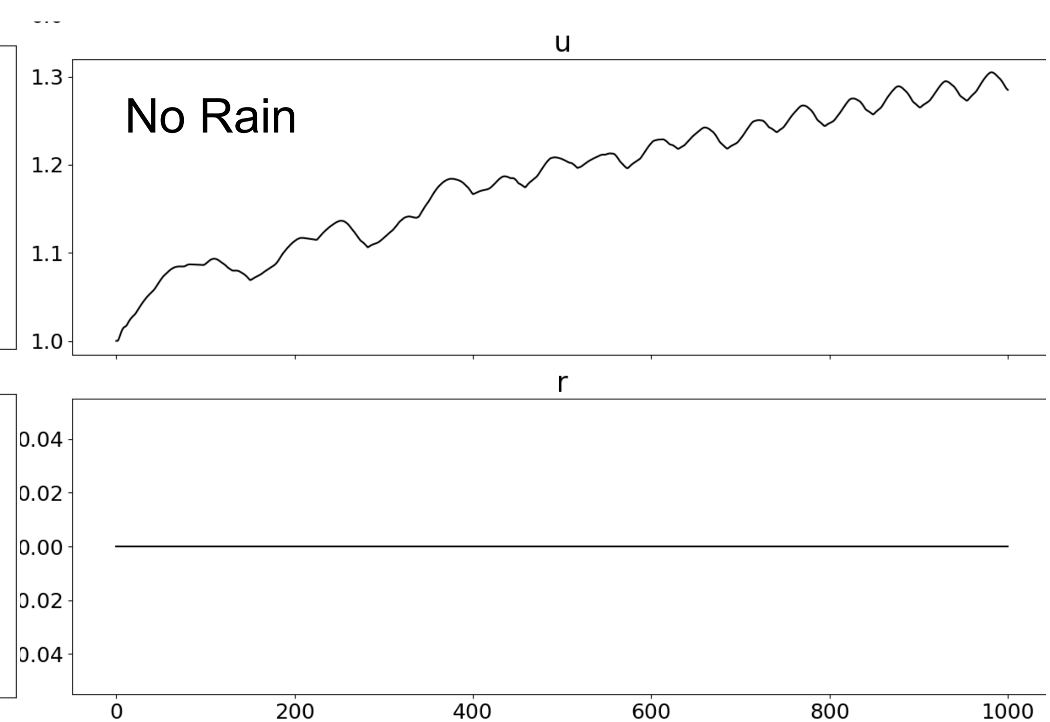
Turning off convection and/or rain

Time series of domain mean
Resolutions Nk = 200/500
dtmeasure = 0.01
ic = init_cond_topog_cos
H_c and/or H_r = 1000
All other parameters unchanged
Note: height is excluded in the plots, since it behaves as expected, i.e. is perfectly conserved

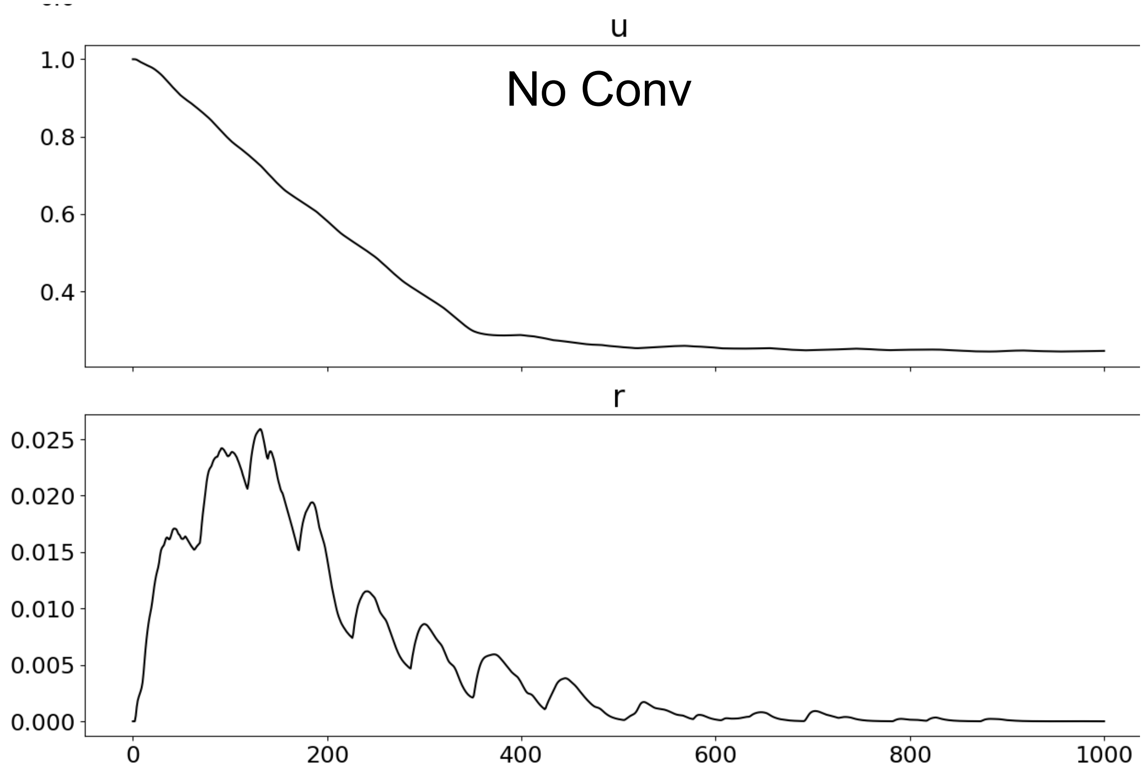
Nk = 200

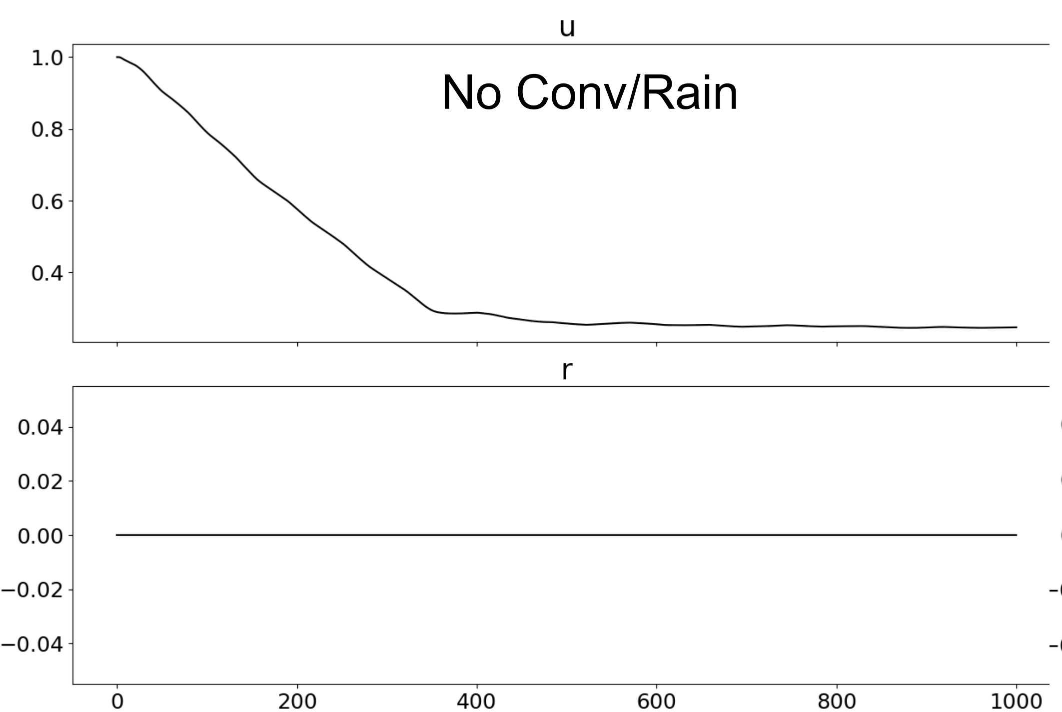

u
No Conv
1.0
0.8
0.6
0.4
0.2
r
0.025
0.020
0.015
0.010
0.005
0.000
0
200
400
600
800
1000

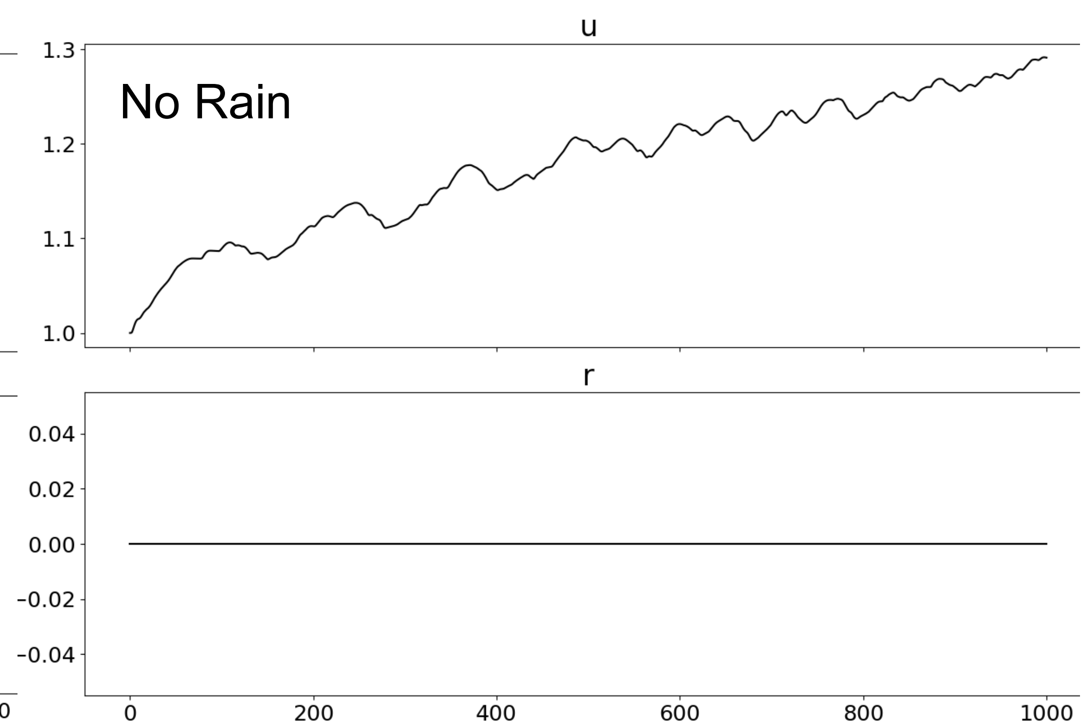

u
No Conv/Rain
1.0
0.8
0.6
0.4
0.2
r
0.04
0.02
0.00
−0.02
−0.04
0
200
400
600
800
1000


u
No Rain
1.3
1.2
1.1
1.0
r
0.04
0.02
0.00
0
200
400
600
800
1000

Nk = 500


u
No Conv
1.0
0.8
0.6
0.4
r
0.025
0.020
0.015
0.010
0.005
0.000
0
200
400
600
800
1000


u
No Conv/Rain
1.0
0.8
0.6
0.4
r
0.04
0.02
0.00
−0.02
−0.04
0
200
400
600
800
1000


u
No Rain
1.3
1.2
1.1
1.0
r
0.04
0.02
0.00
−0.02
−0.04
0
200
400
600
800
1000

Model dynamics

$$\partial_t h + \partial_x (hu) = 0, \qquad (2a)$$

$$\partial_t (hu) + \partial_x (hu^2 + P) + hc_0^2 \partial_x r - fhv = -Q\partial_x b, \qquad (2b)$$

$$\partial_t (hv) + \partial_x (huv) + fhu = 0, \qquad (2c)$$

$$\partial_t (hr) + \partial_x (hur) + h\widetilde{\beta}\partial_x u + \alpha hr = 0, \qquad (2d)$$

$$P(h; b) = \begin{cases} p(H_c - b), & \text{for } h + b > H_c, \\ p(h), & \text{otherwise,} \end{cases} \qquad (3a)$$

$$Q(h; b) = \begin{cases} p'(H_c - b), & \text{for } h + b > H_c, \\ p'(h), & \text{otherwise,} \end{cases} \qquad (3b)$$

$$p(h) = \tfrac{1}{2} g h^2$$

Model dynamics

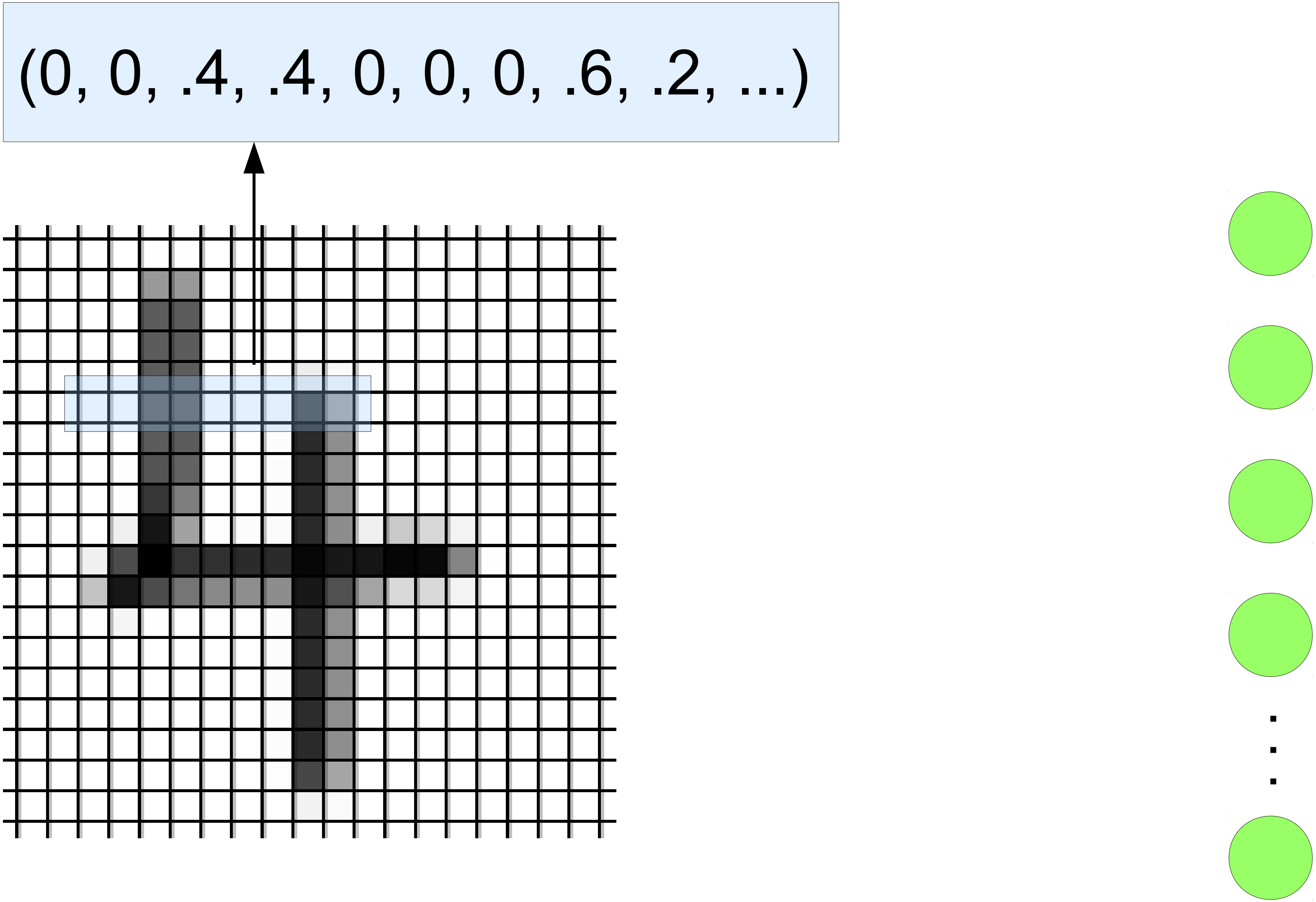
$$\partial_t h + \partial_x (hu) = 0, \quad (2a)$$

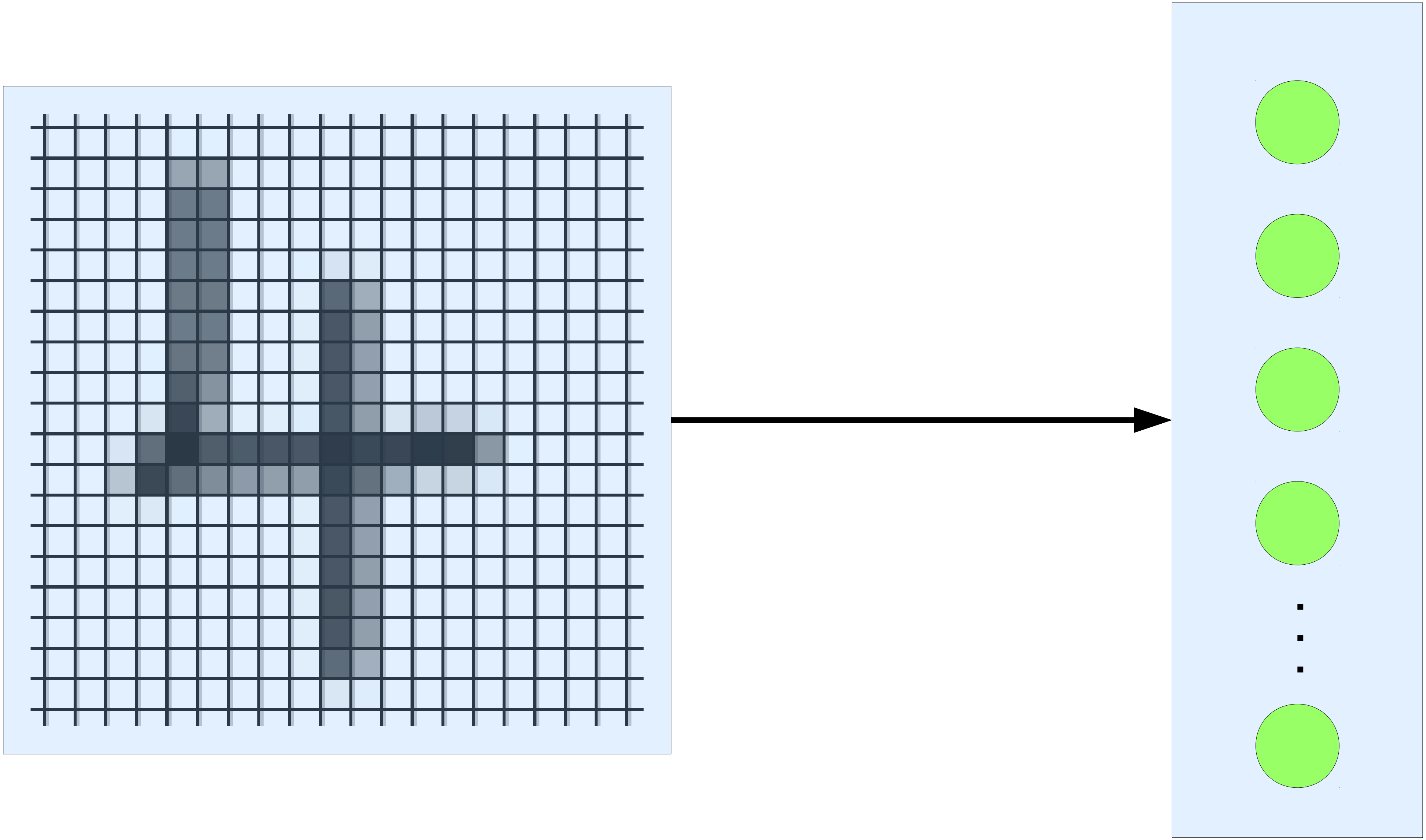
$$\partial_t (hu) + \partial_x (hu^2 + P) + hc_0^2 \partial_x r - fhv = -Q \partial_x b, \quad (2b)$$

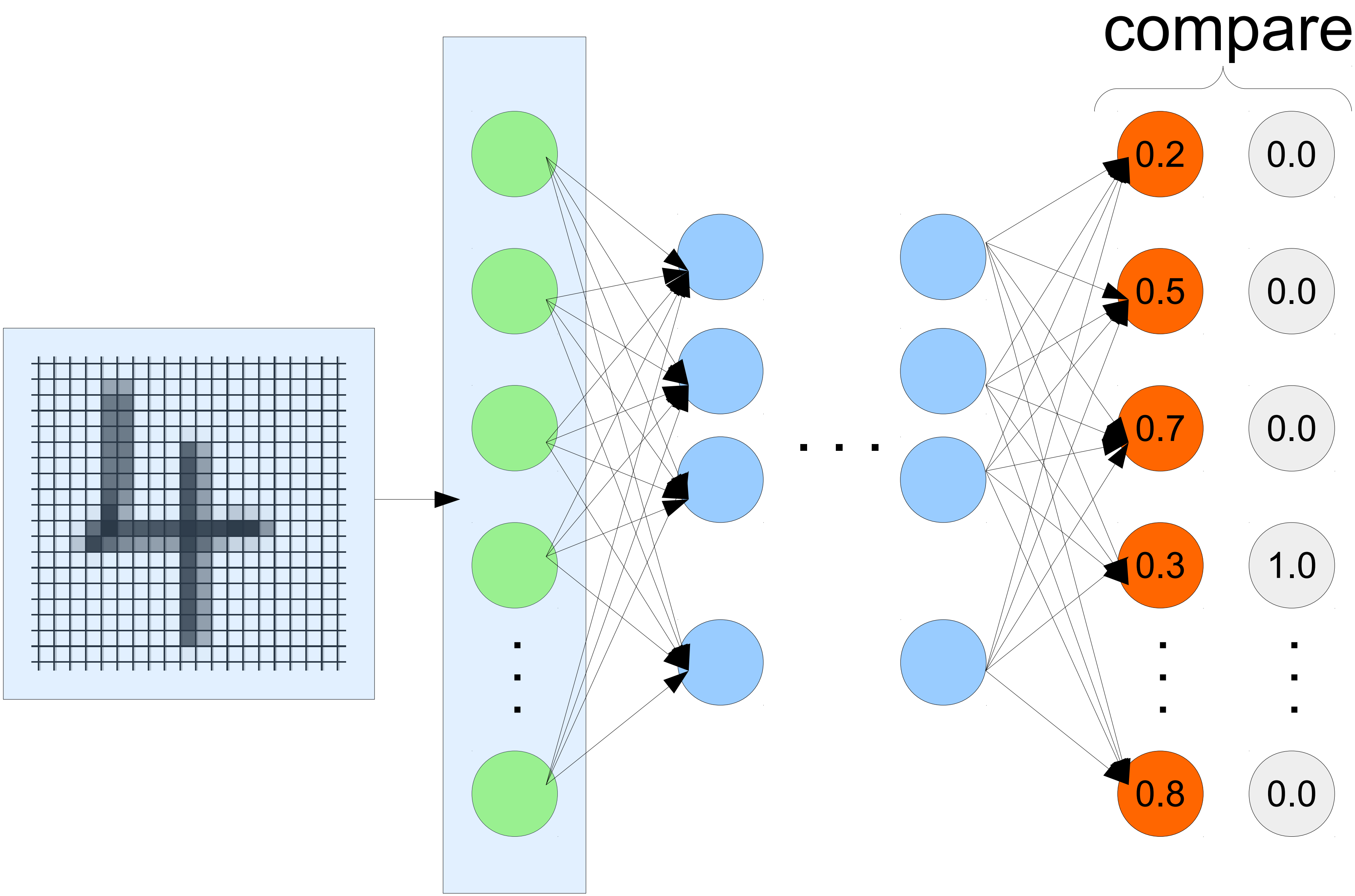
$$\partial_t (hv) + \partial_x (huv) + fhu = 0, \quad (2c)$$

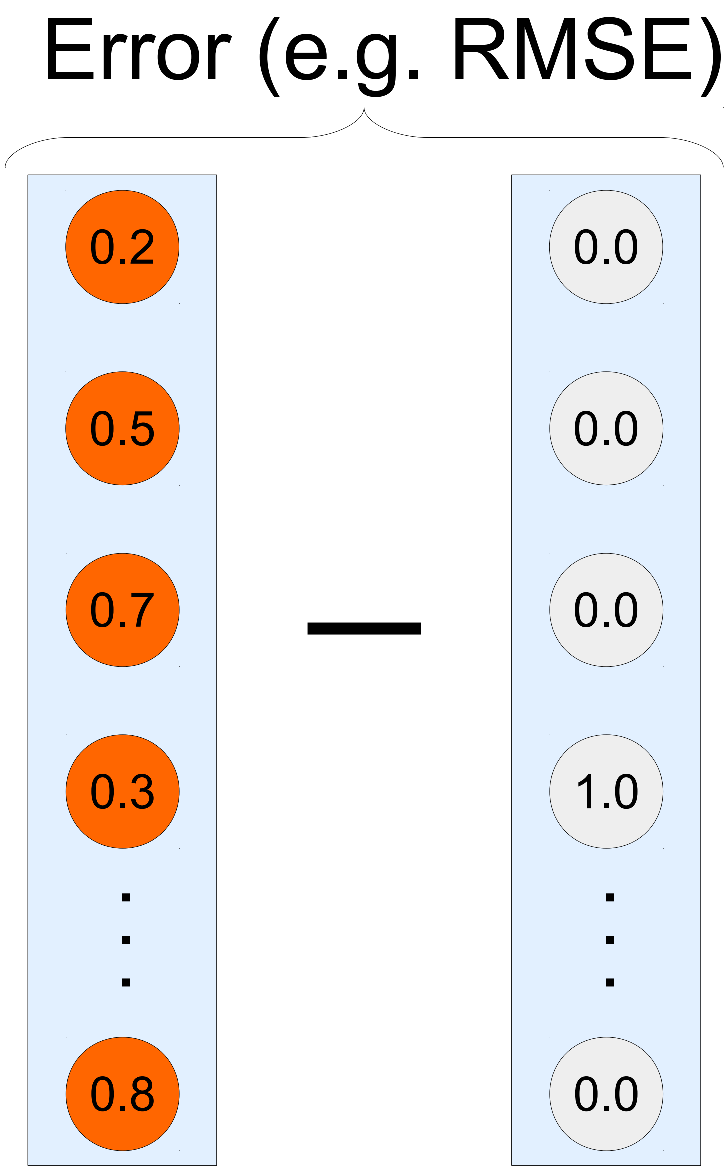
$$\partial_t (hr) + \partial_x (hur) + h\widetilde{\beta}\partial_x u + \alpha hr = 0, \quad (2d)$$

- No meridional velocity
- Prescribed topography
- Periodic boundary conditions
- Dynamic timestep
- Initial conditions:
  - h+b constant
  - hu = 1
  - hr = 0


(0, 0, .4, .4, 0, 0, 0, .6, .2, ...)




compare
0.2
0.0
0.5
0.0
0.7
0.0
0.3
1.0
0.8
0.0



- Let network process a number of different cases
- Define cost function that will depend on all weights and biases
- Compute its gradient
- Adjust weights and biases
- Repeat until minimum is reached