Physics-Constrained Deep Learning for Downscaling and Emulation

MAELSTROM dissemination workshop

Paula Harder,
PhD Candidate at RPTU

In collaboration with

Motivation

Accelerating climate modeling would be beneficial, we could

1. Increase both resolution of prediction and the time scale we are predicting for
2. Stay at same resolution while obtaining results faster, saving energy, making climate simulation more accessible

There are two common ways for machine learning (ML) to help

1. Downscaling: Increasing traditional models predictions resolution as a post-processing tool
2. Emulation: replacing expensive climate model parts with faster ML surrogates

Problem with deep learning approaches
Physical laws/constraints can be violated, e.g. negative masses predicted

Need for strategies for DL methods to obey those constraints

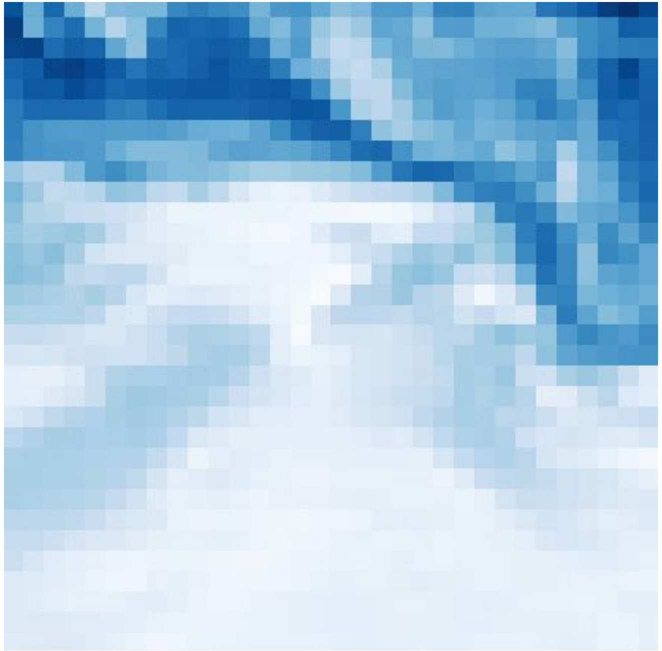
Physics-Constrained Deep Learning for Climate Downscaling

Paula Harder, Alex Hernandez-Garcia, Venkatesh Ramesh, Qidong Yang, Prasanna Sattigeri, Campbell Watson, Daniela Szwarcman, David Rolnick

Goal

Increasing climate data's resolution

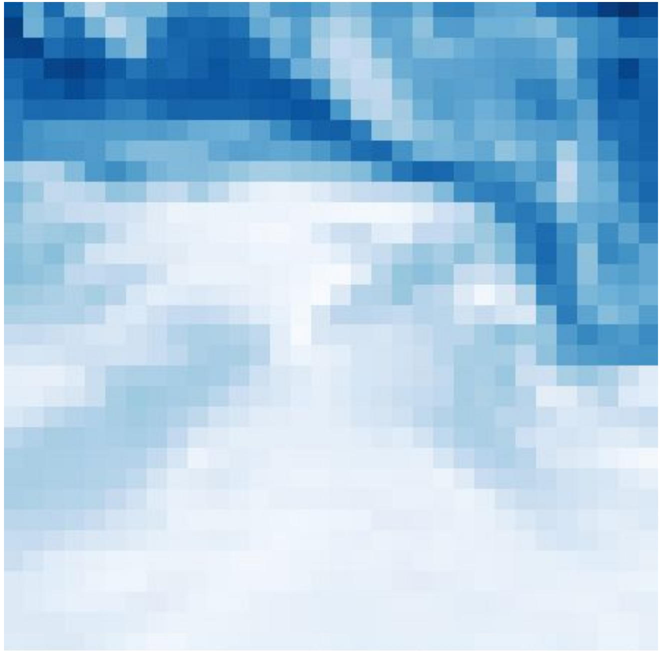
Low-resolution (LR) input



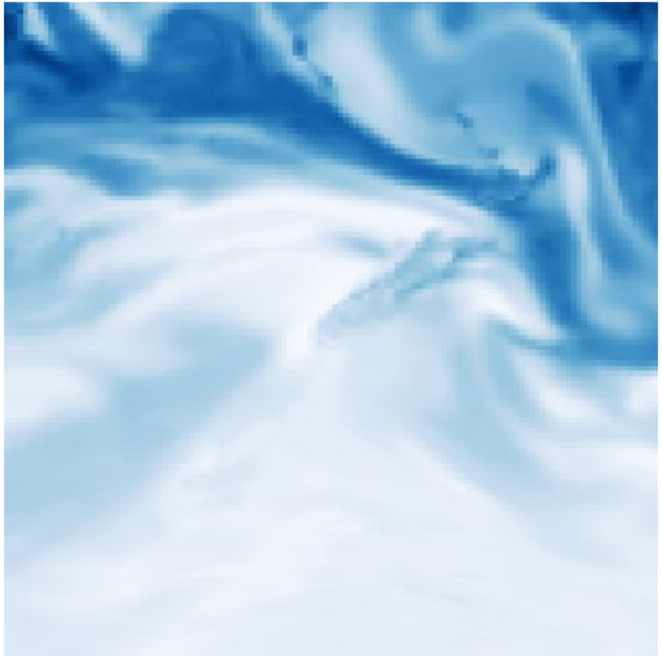
Goal

Increasing climate data's resolution

Low-resolution (LR) input



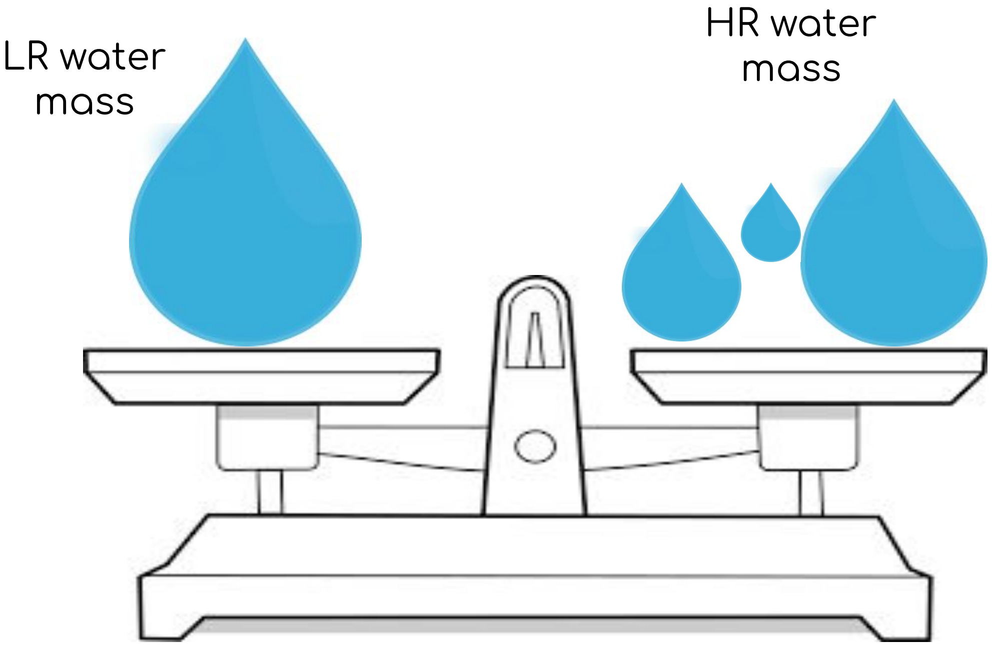
High-resolution (HR) target



Goal

Increasing climate data's resolution ...

while obeying laws of physics

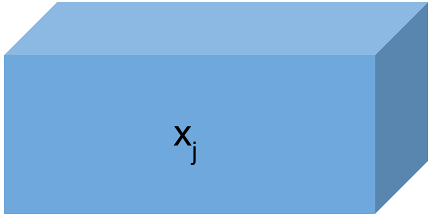


Enforcing Constraints for Downscaling

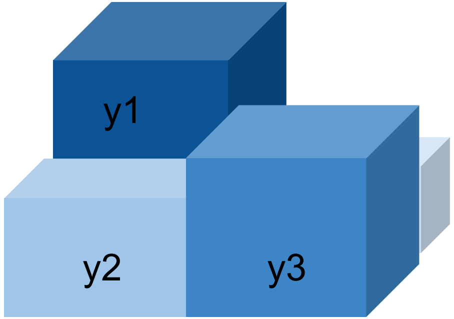
Physics constraints

Predicted quantity is water mass

Want to enforce conservation of mass between low-res input and super-res prediction



Low-res water mass

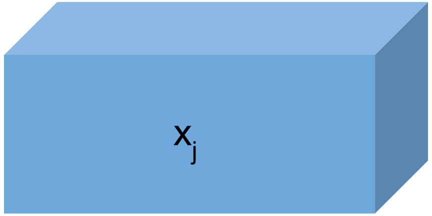


Super-res water mass

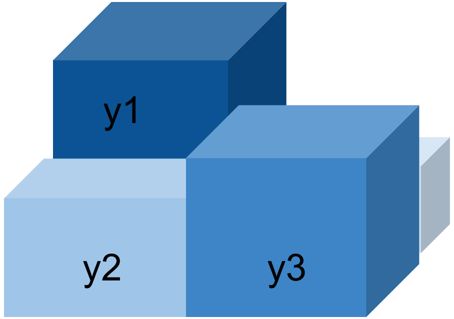
Physics constraints

Want to enforce mass conservation between low-res input and super-res prediction

$$\frac{1}{n}\sum_{i\in I_j} y_i - x_j = 0$$



Low-res water mass



Super-res water mass

Soft-constraining

Want to enforce mass conservation between low-res input and super-res prediction

$$\frac{1}{n}\sum_{i\in I_j} y_i - x_j = 0$$

First idea: Add regularization term to the loss function

$$\mathcal{L}^{(\mathrm{eq})}(y,\hat{y}) = \mathcal{L}(y,\hat{y}) + \mu \cdot \frac{1}{n_{\mathrm{in}}}\sum_{j=1}^{n_{\mathrm{in}}}\left(\frac{1}{n}\sum_{i\in I_j} y_i - x_j\right)^2$$

Soft-constraining

First idea: Add regularization term to the loss function

$$\mathcal{L}^{(\mathrm{eq})}(y,\hat{y}) = \mathcal{L}(y,\hat{y}) + \mu \cdot \frac{1}{n_{\mathrm{in}}} \sum_{j=1}^{n_{\mathrm{in}}} \left(\frac{1}{n} \sum_{i \in I_j} y_i - x_j\right)^2$$

Problems:

- No guarantee
- Need to optimize mu
- Can have accuracy-constraints trade-off

Hard constraining

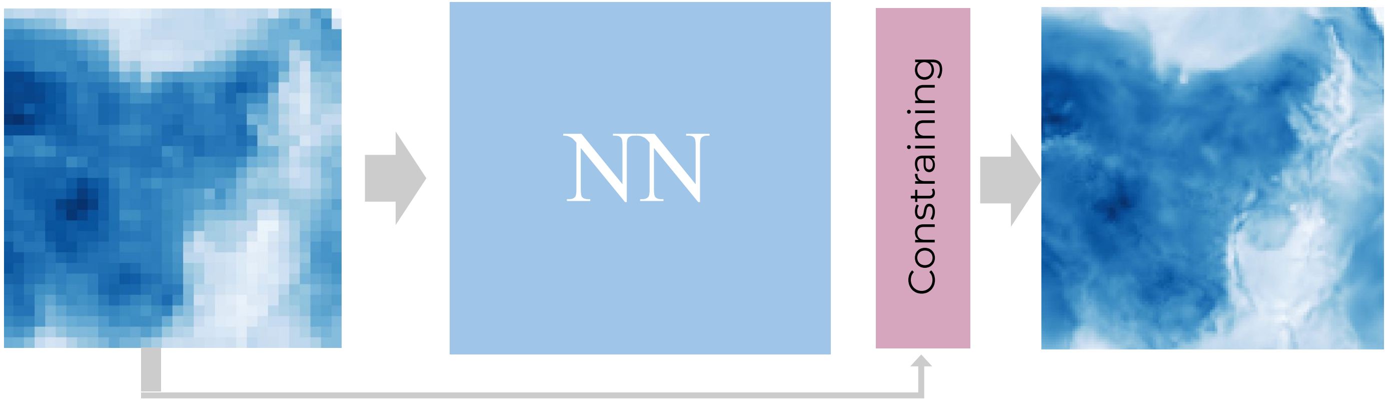
Want to enforce mass conservation/consistency between low-res input and super-res prediction

$$\frac{1}{n}\sum_{i\in I_j} y_i - x_j = 0$$

Hard constraining

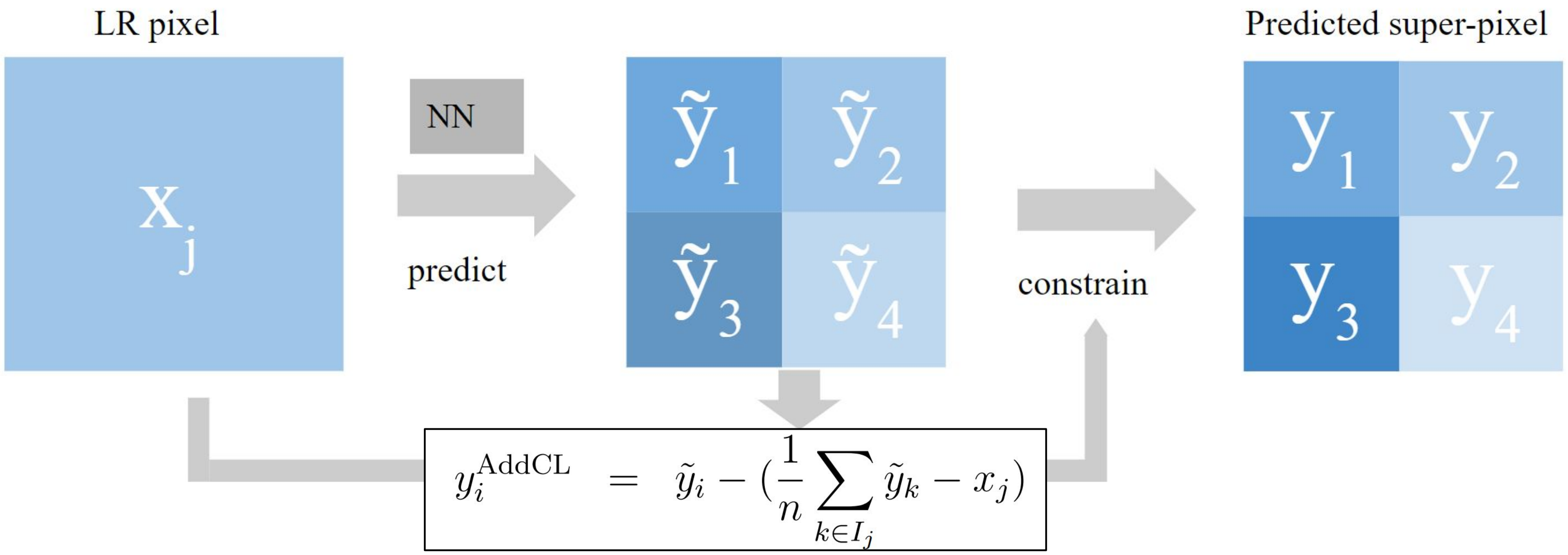
Want to enforce mass conservation between low-res input and super-res prediction

$$\frac{1}{n}\sum_{i\in I_j} y_i - x_j = 0$$



Additive Constraining (AddCL)

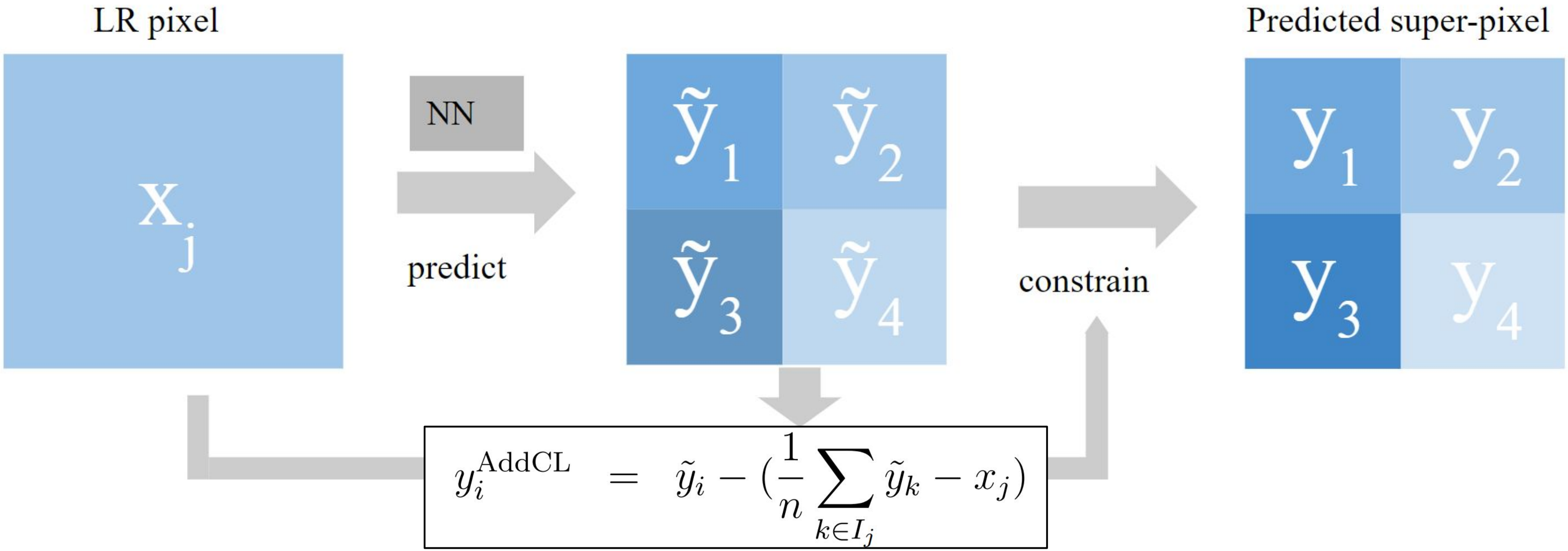
AddCL guarantees conservation of mass



Additive Constraining (AddCL)

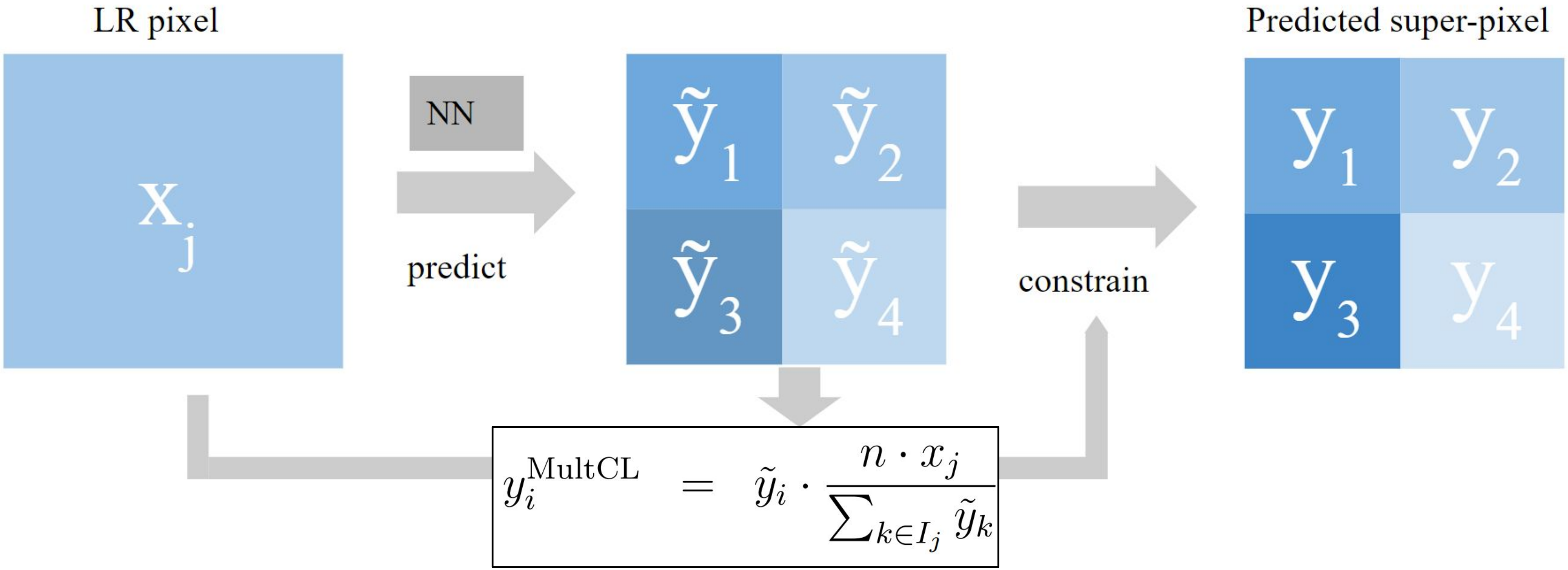
$$\frac{1}{n}\sum_{i\in I_j} y_i - x_j = 0$$

AddCL guarantees conservation of mass



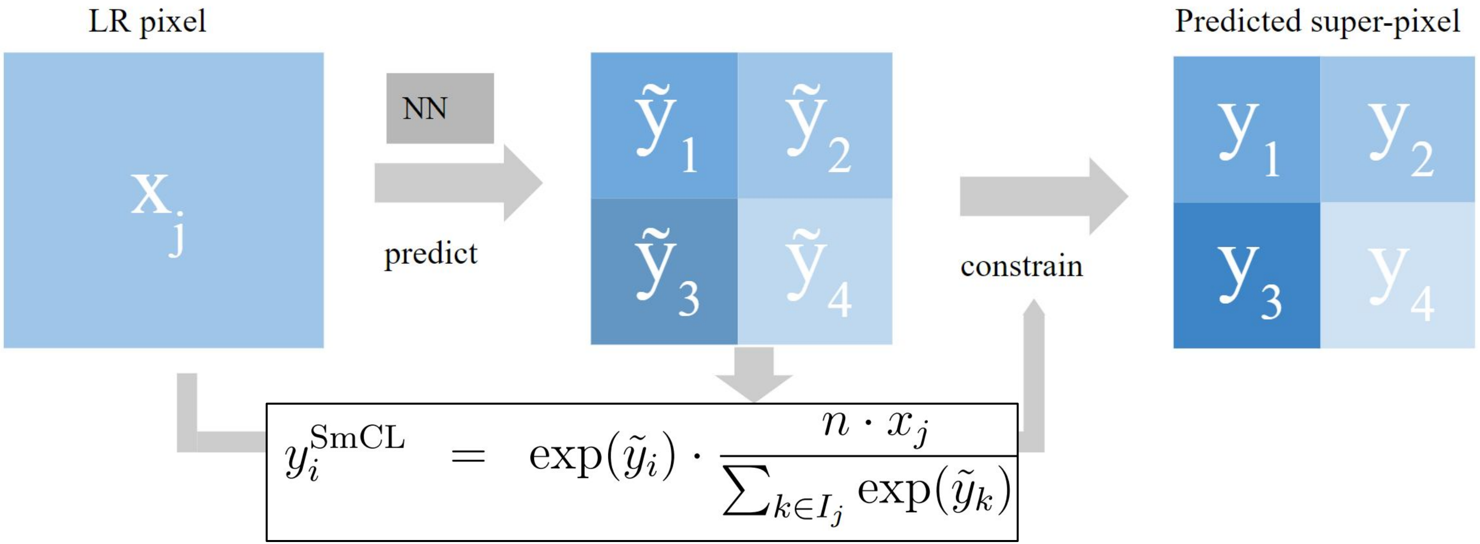
Multiplicative Constraining (MultCL)

MultCL guarantees conservation of mass



Softmax Constraining (SmCL)

SmCL guarantees conservation of mass and **positivity**



More general formulation

Generalized formulation

$$\sum_{i \in I_j^{(\mathrm{eq})}} \left(g_{i,j}^{(\mathrm{eq})}(y_i)\right) + h_j^{(\mathrm{eq})}(x) = 0$$

More general formulation

Generalized formulation

$$F^{(\mathrm{eq})}(g^{(\mathrm{eq})}(y)) + h^{(\mathrm{eq})}(x) = 0$$

$$\mathcal{L}^{(\mathrm{eq})}(y, \hat{y}) = \mathcal{L}(y, \hat{y}) + \mu \cdot ||F^{(\mathrm{eq})}(g^{(\mathrm{eq})}(y)) + h^{(\mathrm{eq})}(x)||$$

$$y^{\mathrm{AddCL}} := g^{-1}(\tilde{y} - a \cdot (F(\tilde{y}) + h(x))$$

Data

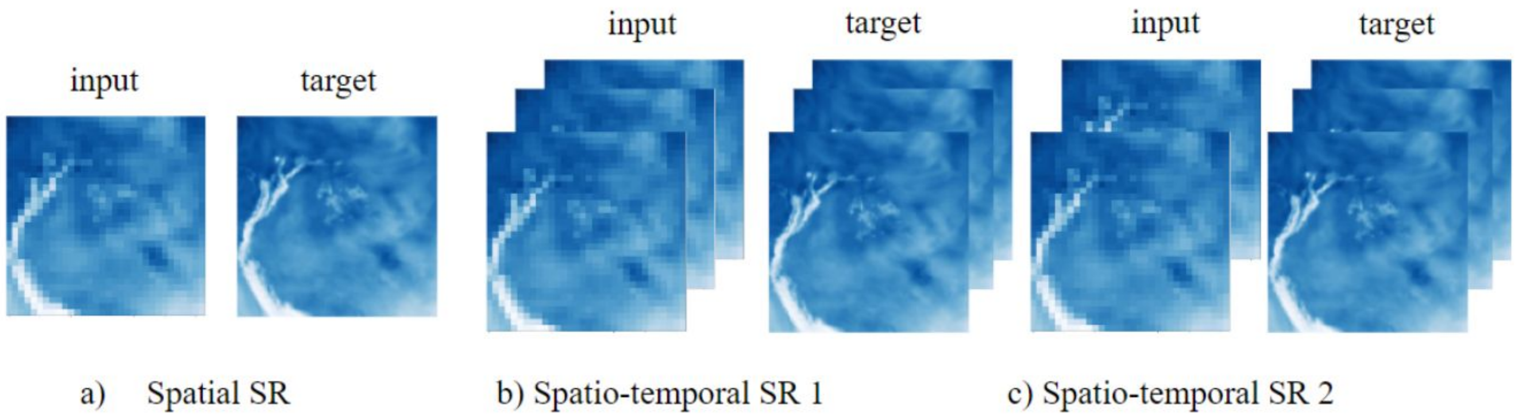
Data - overview

ERA5 (ECMWF reanalysis data) - Water content, synthetic low-res

- Different upsampling factors, multi- and single-time-step data

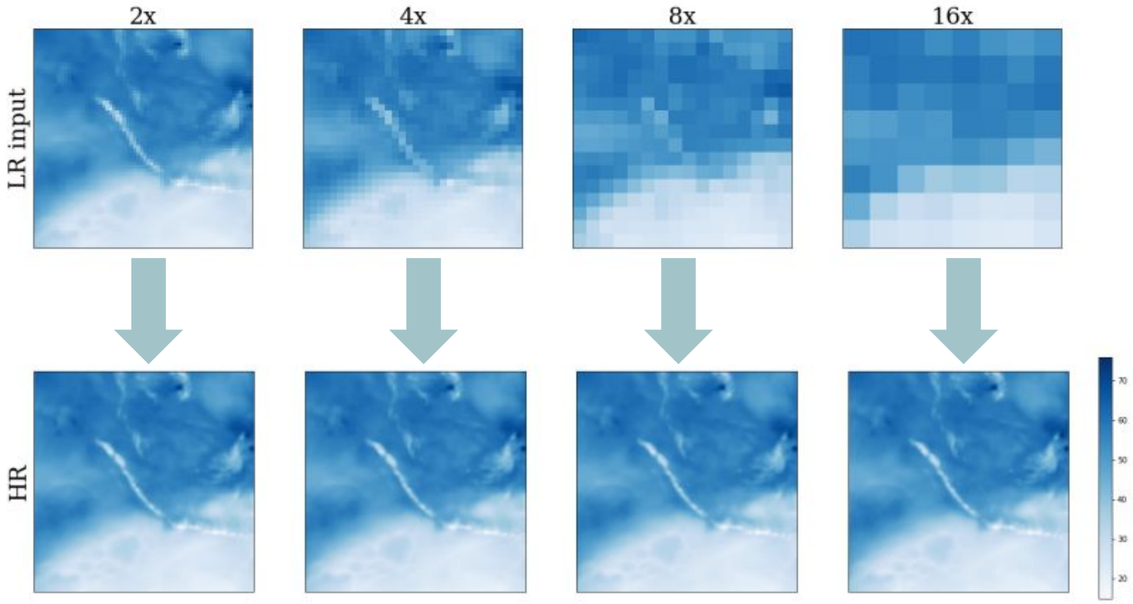
WRF (Weather and Research Forecast) - Temperature, two different simulations

Data - ERA5



a) One time step is super-resolved spatially at once
b) 3 time-steps are super-resolved simultaneously
c) Super-resolve both spatially and temporally (frame interpolation)

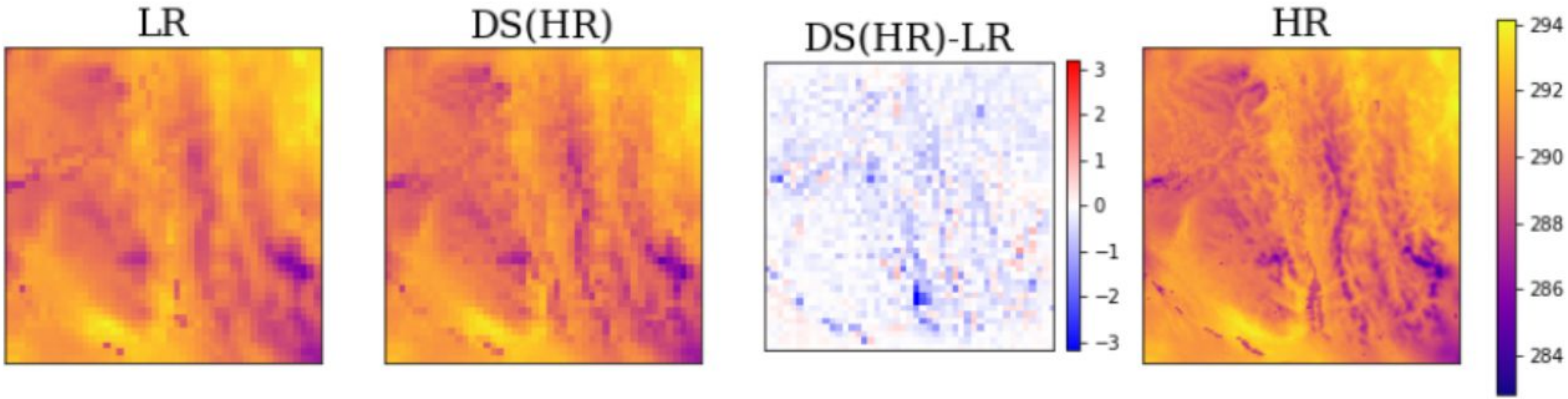
Data - ERA5 Different Upsampling Factors



RPTU

Data - WRF

- Operational weather forecast
- Lake George in New York
- Hourly
- 2017-2020
- LR not created by downsampling HR, but different simulation!
- HR: 3 km resolution, LR 9 km resolution



Experiments

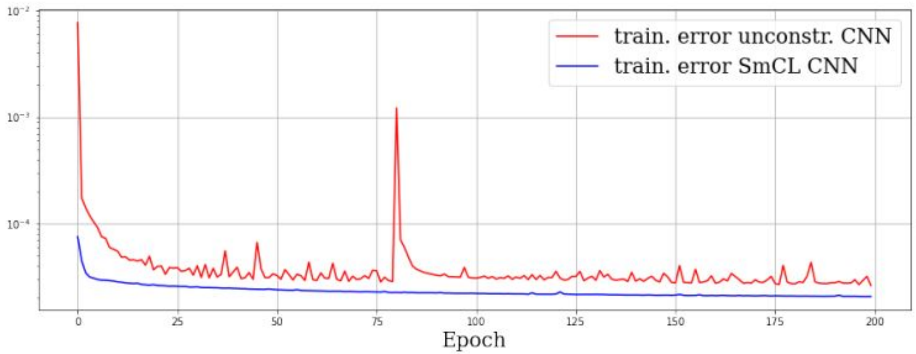
Architectures

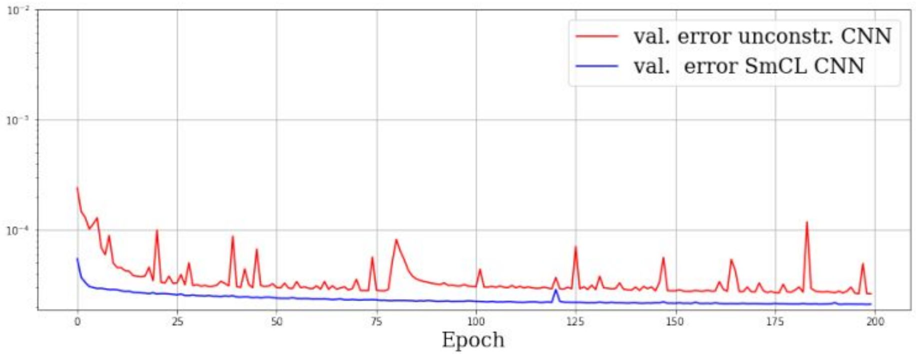
We look at different neural architectures

- Convolutional neural network (CNN)
- Generative adversarial neural network (GAN)
- ConvRNN (mix of CNN and recurrent NN)
- FlowConvRNN (mix of CNN/RNN/optical flow)
- New work: Fourier Neural Operator (FNO) for arbitrary resolution downscaling

Results

Results - Loss Curve

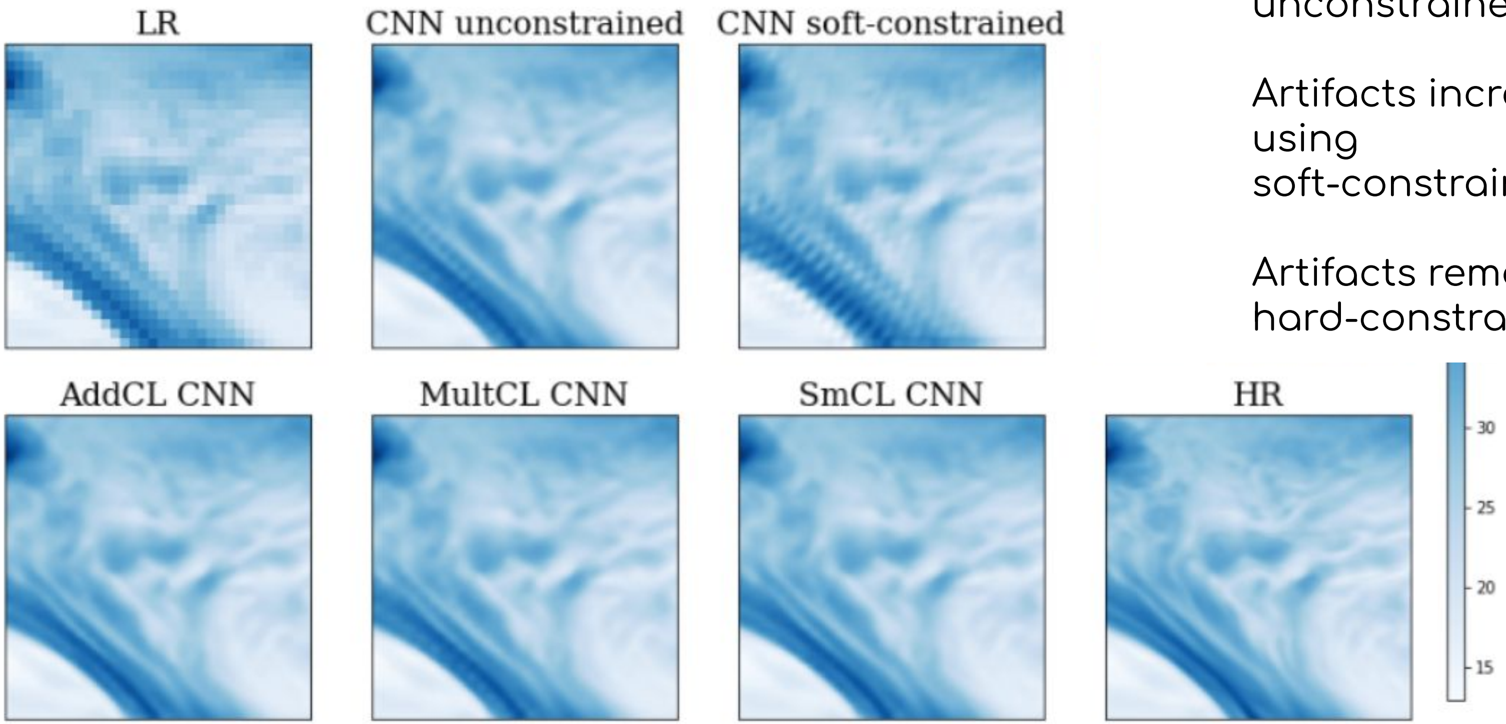




Constraining makes
learning curve smoother!

RPTU

Results - CNN water content 4x



Visual artifacts in the unconstrained CNN

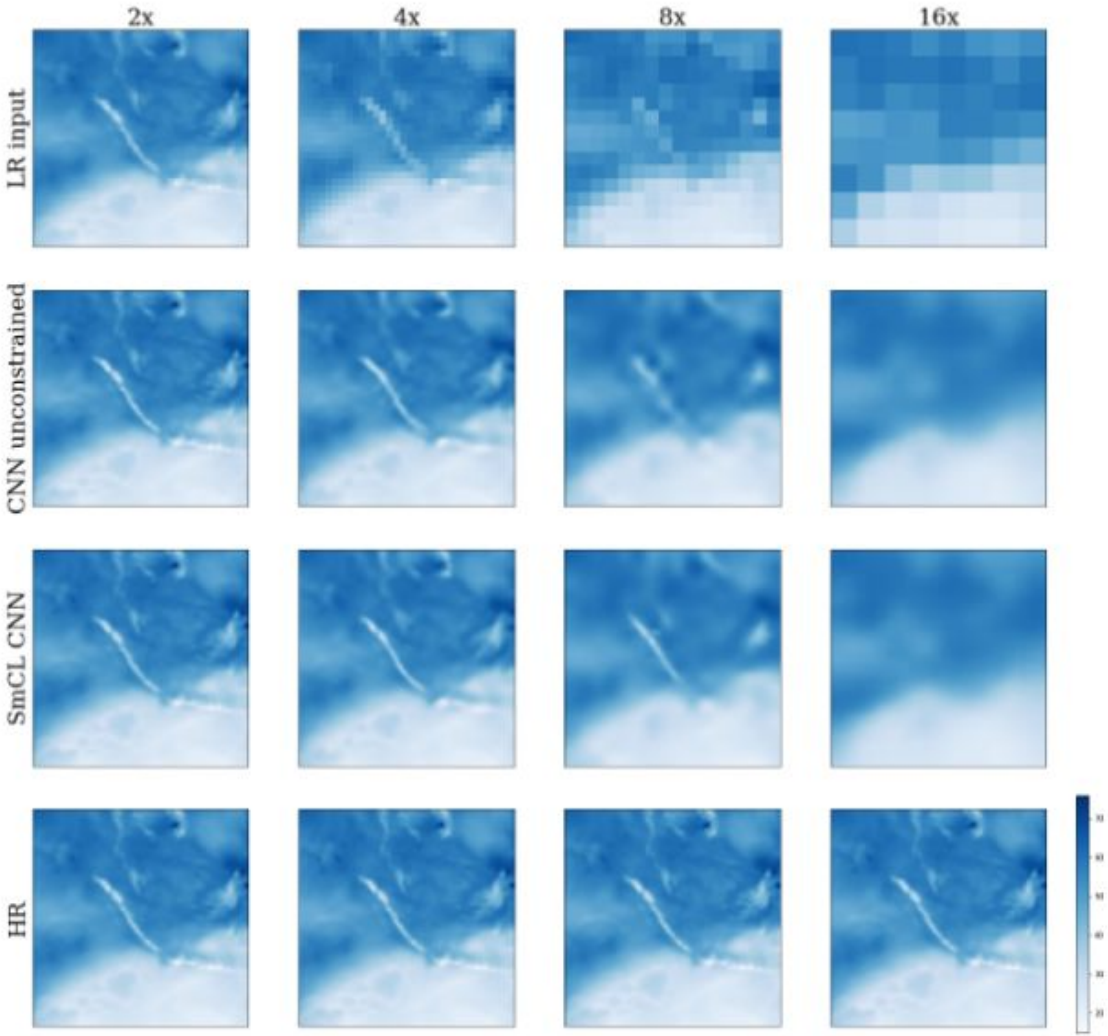
Artifacts increased using soft-constraining

Artifacts removed with hard-constraining

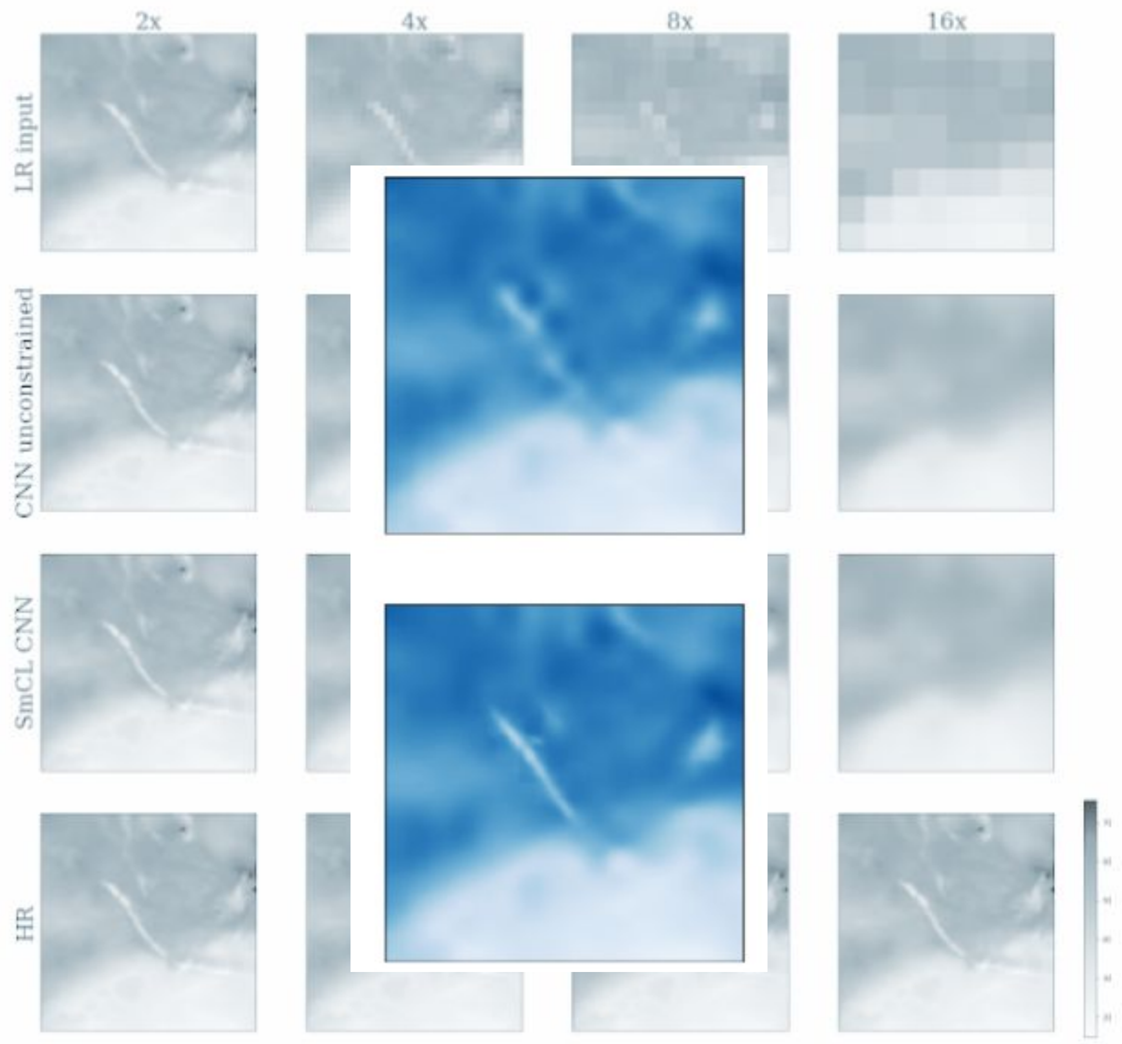
Results - CNN water content 4x

Constraining	unconstrained	soft constrained	AddCL	MultCL	SmCL
RMSE	0.657	0.801	**0.580**	0.606	0.582
Mass violation	0.058	0.023	**0.000**	**0.000**	**0.000**
#neg pixels per Mil pixels	396	95,300	234	**0**	**0**

Results - different upsampling factors



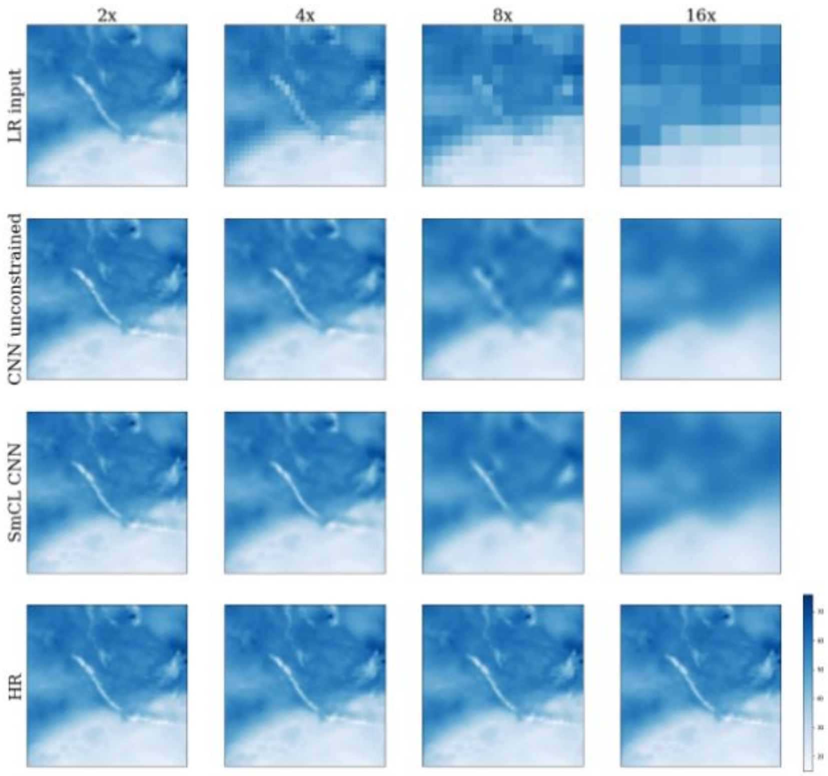
Results - different upsampling factors



Results - different upsampling factors

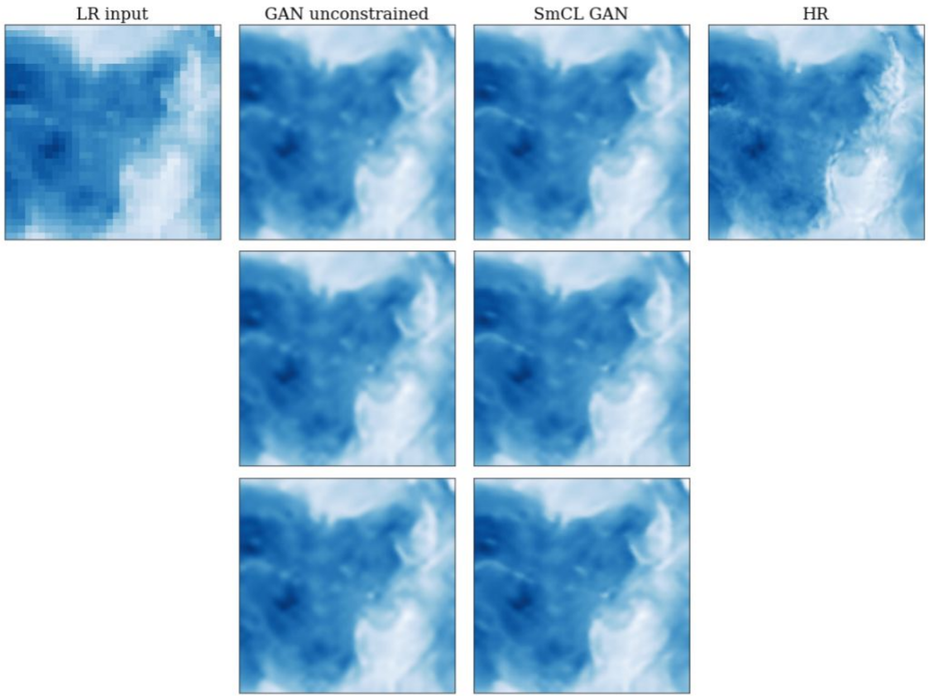
Factor	unconstrained	Hard-constrained (SmCL)
2 x	0.251	**0.215**
4 x	0.657	**0.582**
8 x	1.358	**1.268**
16 x	2.450	**2.368**

RMSE shown in table



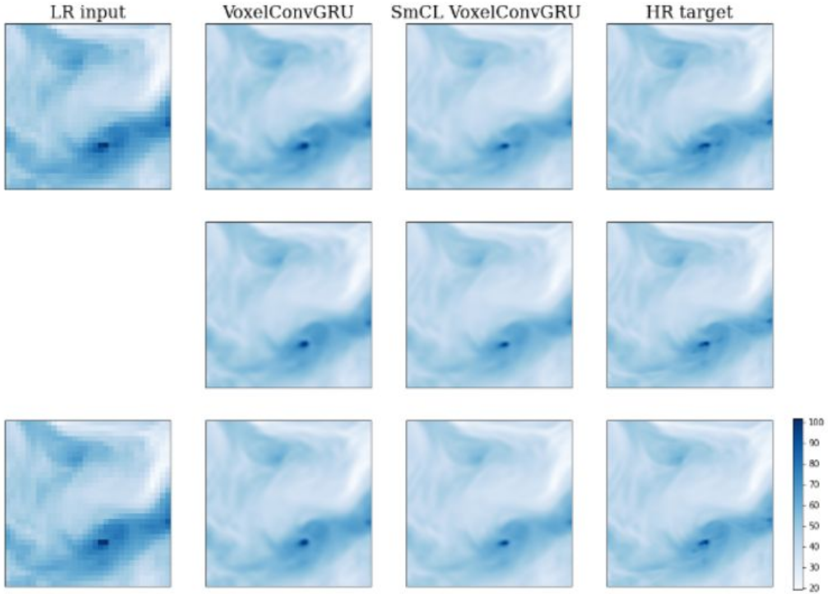
Results - GAN water content 4x

Model	unconstrained	Hard-constrained (SmCL)
RMSE	0.628	**0.603**
MAE	0.313	**0.310**
SSIM	99.44	**99.46**

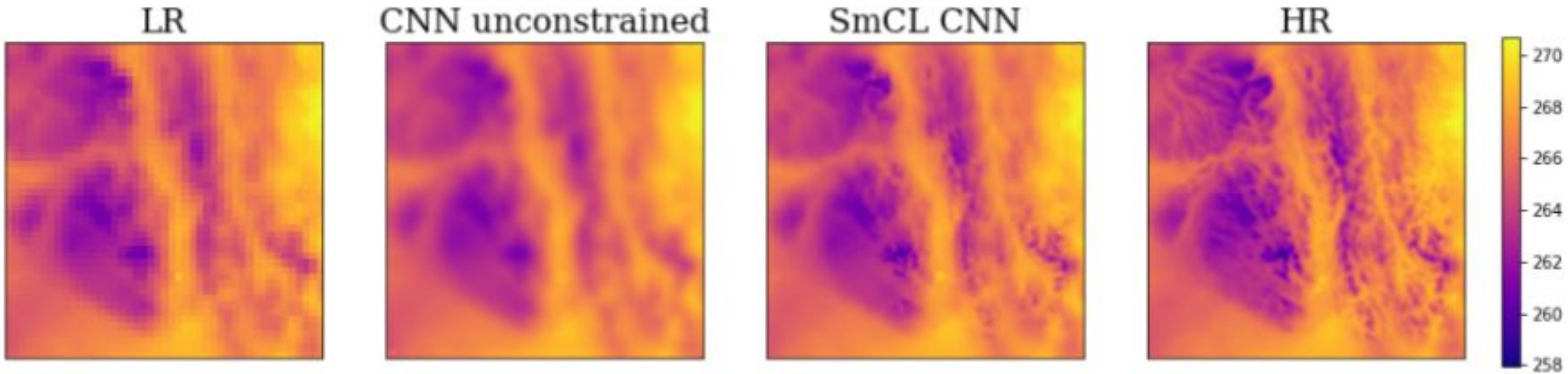


Results - Spatial-temporal super-resolution

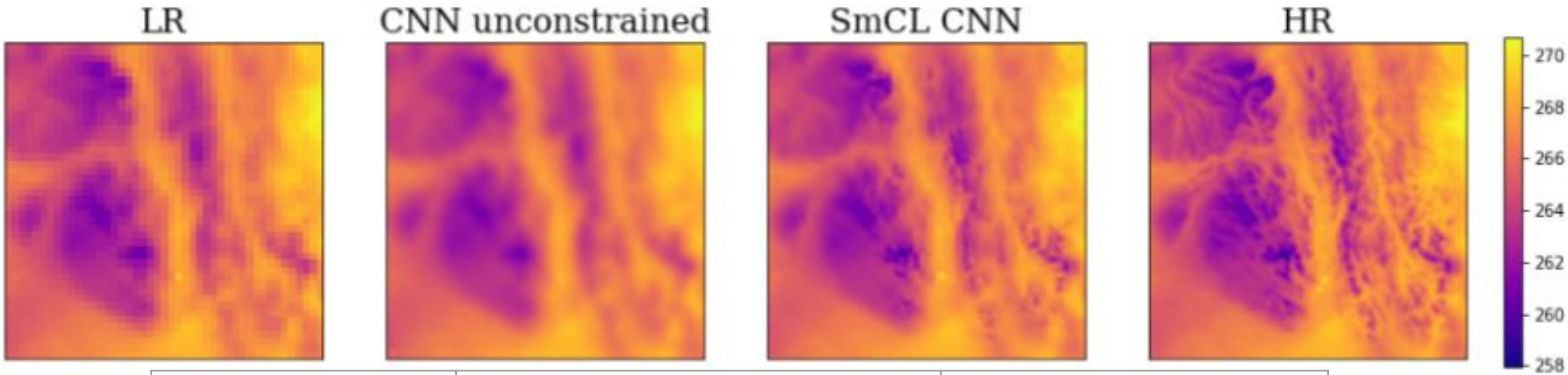
Model	unconstrained	Hard-constrained (SmCL)
RMSE	0.673	**0.514**
MAE	0.352	**0.276**
SSIM	99.40	**99.62**



Results - WRF data



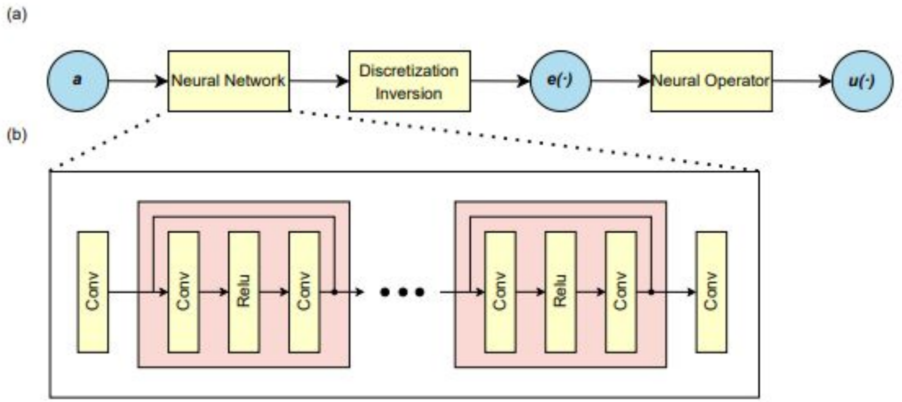
Results - WRF data

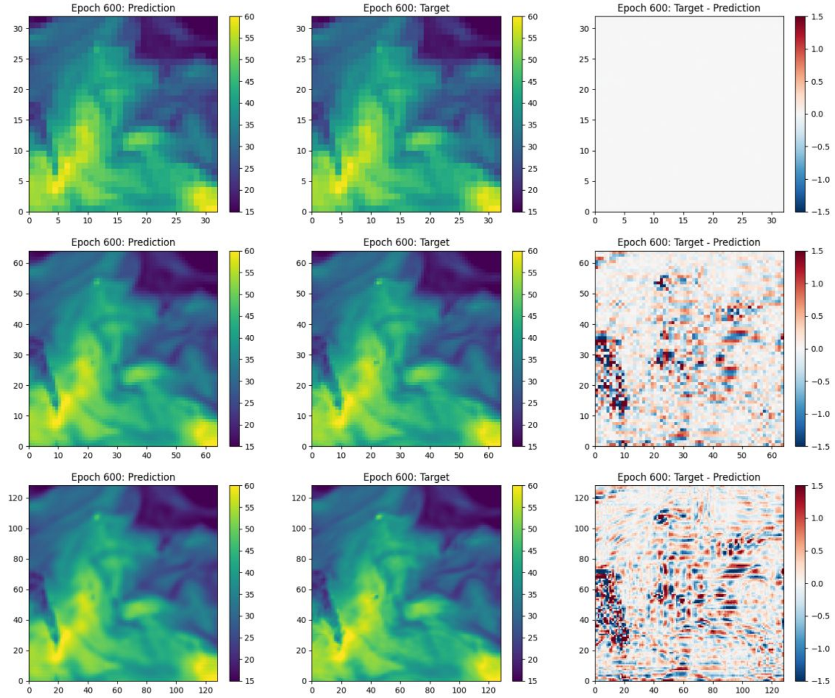


Model	unconstrained	Hard-constrained (SmCL)
RMSE	0.952	**0.950**
MAE	0.618	**0.592**
SSIM	94.92	**95.24**

More Constrained Downscaling Work

Fourier Neural Operator for Arbitrary Resolution Downscaling



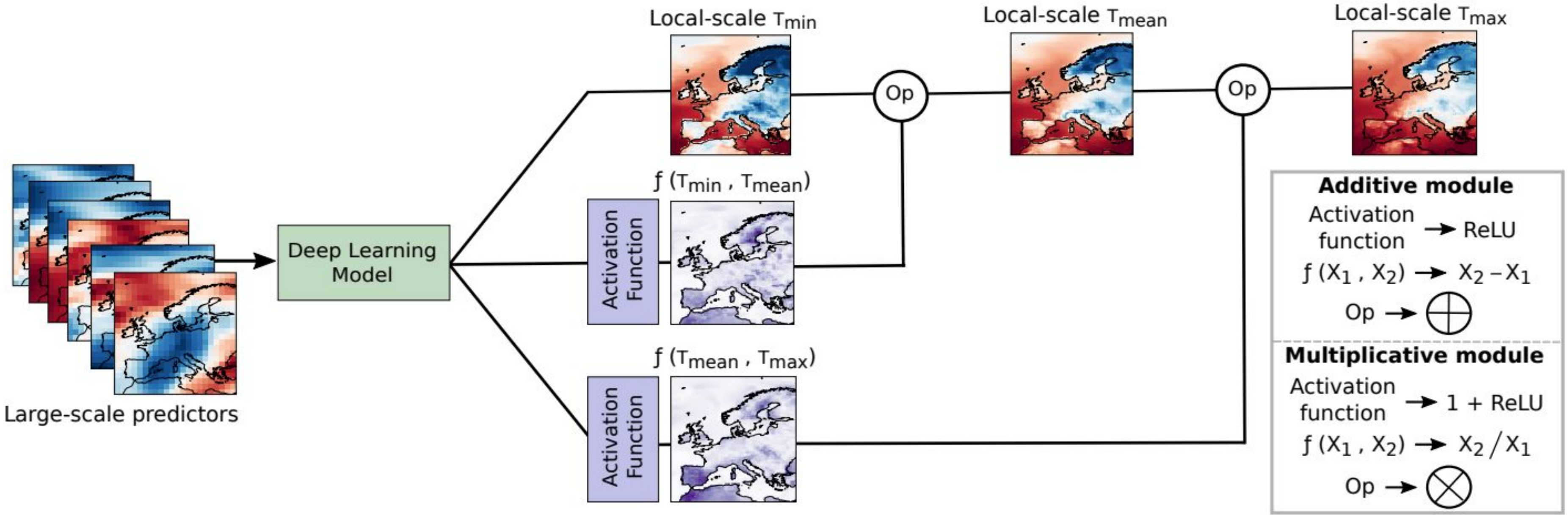


Can train on eg 2x downscaling and apply for 4x

Learns a mapping in function space

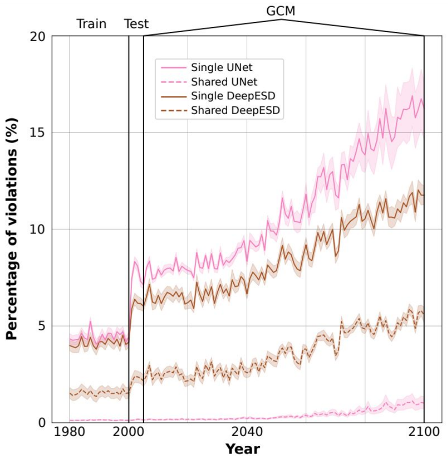
Reference: Yang et al, Fourier Neural Operators for Arbitrary Resolution Climate Data Downscaling, https://arxiv.org/pdf/2305.14452.pdf

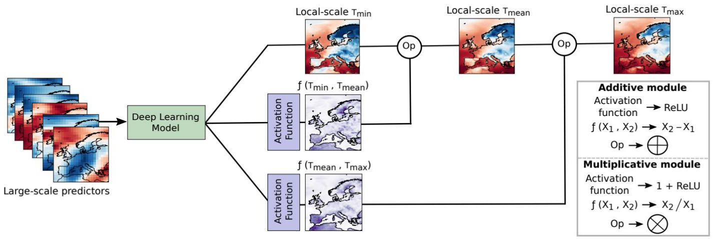
Multi-variable constrained downscaling



Gonzalez-Abad et al, Multi-variable Hard Physical Constraints for Climate Model Downscaling, https://arxiv.org/pdf/2308.01868.pdf

Multi-variable constrained downscaling





- Enforces constraints in a perfect prognosis setup
- Tmax >= Tmin
- Violations increase with time

Gonzalez-Abad et al, Multi-variable Hard Physical Constraints for Climate Model Downscaling, https://arxiv.org/pdf/2308.01868.pdf

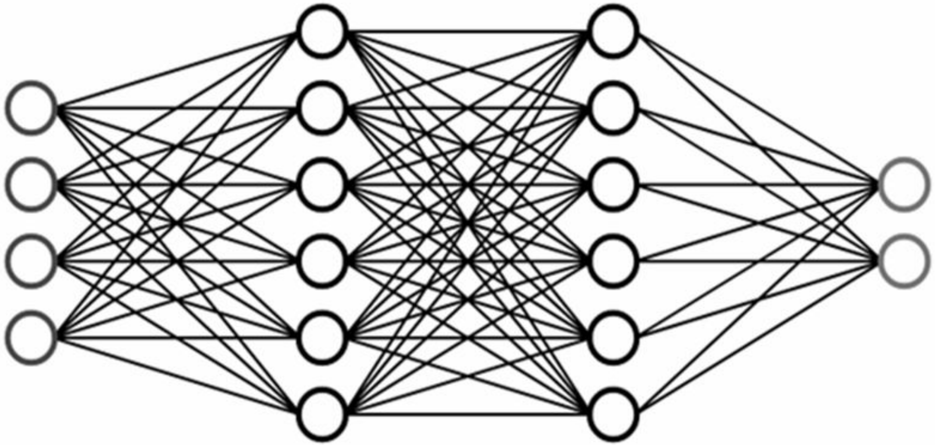
Physics-Constrained Deep Learning for Aerosol Microphysics Emulation

Paula Harder, Duncan Watson-Parris, Philip Stier, Nico Gauger, Janis Keuper, Phillip Weiss

Network architecture

Neural network with 2 hidden layers, 256 nodes per hidden layer, ReLU activation

Inputs:
Temperature, RH, pressure, etc
Aerosol masses
Aerosol number



Outputs:
Change in aerosol masses
Change in aerosol numbers
Water content

Idea: Replace M7 with NN

M7: An efficient size-resolved aerosol microphyiscs module for large-scale aerosol transport models, Vignatti et al. 2004


Constraining

Our Constraints

Mass conservation

Let S= {SO4,DU,OC,BC} be the set of aerosol species. For every s∈S let I_s be the indices of our output y corresponding to value of that species.

Mass conservation is given by

$$\sum_{i \in I_s} y_i = 0$$

Soft constraining: Add loss term $\mathcal{L}^{(\mathrm{eq})}(y, \hat{y}) = \mathcal{L}(y, \hat{y}) + \mu \cdot \frac{1}{n_s} \sum_{s \in S} a_s (\sum_{i \in I_s} y_i)^2$

Hard constraining (completion): Choose $i_c \in I_s$ and set $y_{i_c}^{\mathrm{CompL}} = -\sum_{k \in I_s \setminus \{i_c\}} y_k$

Our Constraints

Mass positivity

All predicted masses have to be positive. For our input x (masses at time 0) and our output y (change in mass), the constraint is

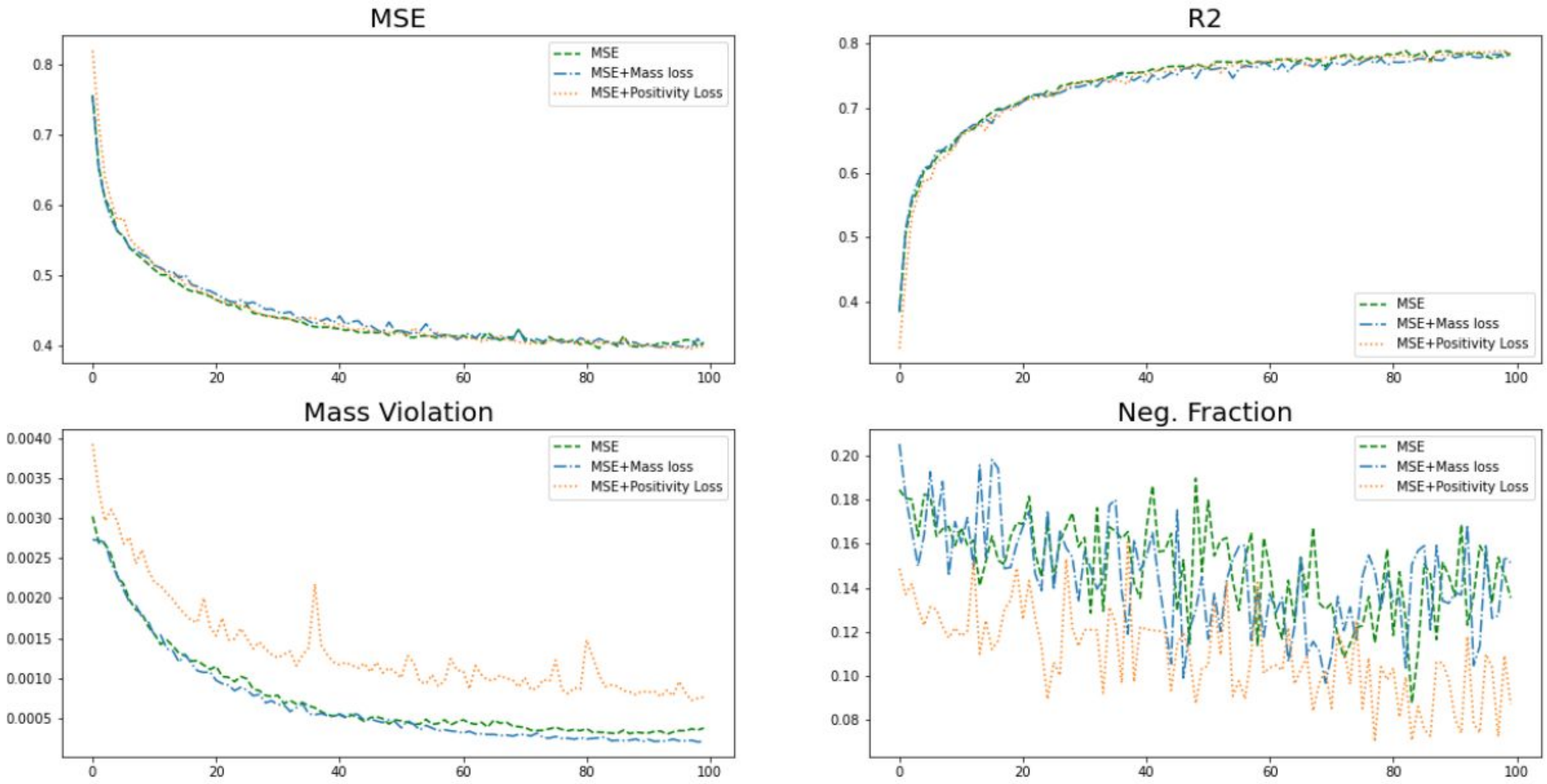
$$y_i + x_i \geq 0$$

Soft constraining: Add loss term $\mathcal{L}(y, \hat{y}) + \gamma \cdot (\frac{1}{n_p} \sum_{i=1}^{n_p} b_i \mathrm{ReLU}(-(y_i + x_i))^2$

Hard constraining: Add correction layer $y^{\mathrm{corr}} = \mathrm{ReLU}(\tilde{y} + x) - x$

Results

Results - Loss Curve



Results - Mass conservation

Soft constraining with mass loss term decreases mass violation of each aerosol species, but accuracy is decreased too

Hard constraining with completion procedure guarantees mass conservation, but also decreases accuracy

Model	Base NN	NN + mass loss (soft-constr.)	NN + completion (hard-constr.)
R^2	**0.763**	0.730	0.738
Overall Mass Violation	0.00037	0.00014	**0**

Results - Positivity

Soft constraining with positivity loss decreases negative fraction and negative mean, but also accuracy

Hard constraining with correction procedure guarantees no negative values and also increases accuracy

Model	NN Base	NN + Positivity Loss (soft-constr.)	NN + Correction (hard-constr.)
R^2	0.763	0.709	**0.771**
Negative Fraction	0.134	0.0894	**0**
Negative Mean	0.00151	0.000081	**0**

Offline Results - Runtimes

Model	M7	NN pure GPU	NN CPU-GPU-CPU	NN CPU
time (s)	5.781	0.000517	0.0897	2.042
speed-up	-	11181.8	64.4	2.80

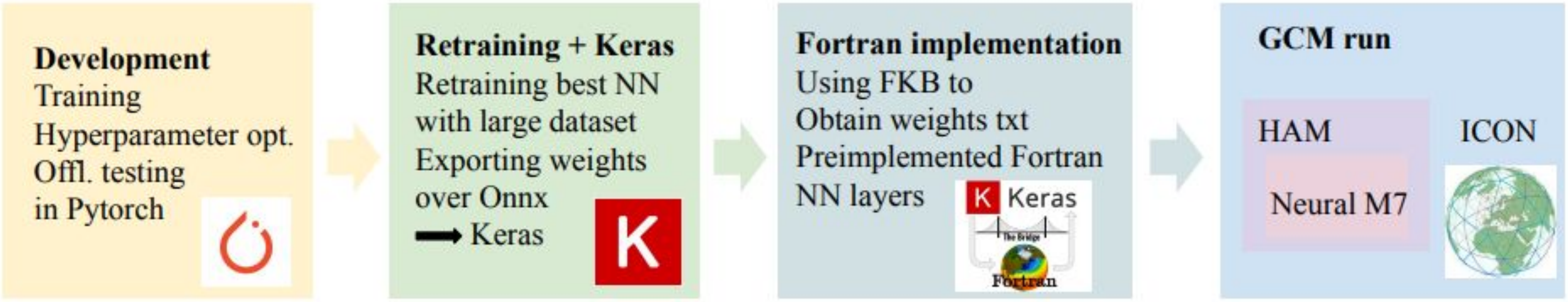
Comparing NN in PyTorch vs. orig. M7 in Fortran

Whole point of NN is to be faster

– > Very large speed-up using a pure GPU setting
– > Significant speed-up transferring data from CPU and back
– > Small speed-up in a single CPU setup

Integration in ICON

Using Fortran-Keras-Bridge (FKB) [1] library to integrate NN in ICON-HAM



[1] https://github.com/scientific-computing/FKB

Integration in ICON

Simulation easily unstable, if out-of-distribution samples appear - > important to include samples from all year, all times of day, all areas, all vertical levels in training data

First simulation with baseline NN now are running stably for couple month

Things that helped for stable run

- Retrain with all data (including training,validation and testing) to achieve stable runs
- Bigger NN

Take away

Simple hard-constraints can be incorporated into NNs to ensure some constraints, e.g. conservation of mass

Hard-constraints can improve ML performance for e.g. for downscaling

Soft-constraints didn't work well in our cases, usually trade-off between accuracy and constraint satisfaction

Work in progress & Outlook

- Constraining Methodologies
  - Application cases with non-linear constraints
  - Dealing with non-linear constraints
  - Improving soft-constraining through scheduling
  - Combining correction and completion
- Downscaling
  - Application to new architectures, such as Transformers, Normalizing Flows and Diffusion-based models
  - Application in a two-step approach, including bias correction
- Aerosol Emulation
  - Pruning methods to make NN emulator faster & smaller
  - Integration into GPU ICON version

Thanks for your attention!