T. HOEFLER

ML for High-Performance Climate: Massive Data Post Processing, Extreme Compression, and Earth Virtualization Engines

with contributions by B. Stevens, O. Fuhrer, L. Huang, P. Groenquist, P. Dueben, F. Scheidl, and the whole SPCL team, CSCS, ECMWF, the EVE Summit Attendees, industry friends, and many others

Keynote at ECMWF's HPC Meeting, Bologna, Oct. 2023

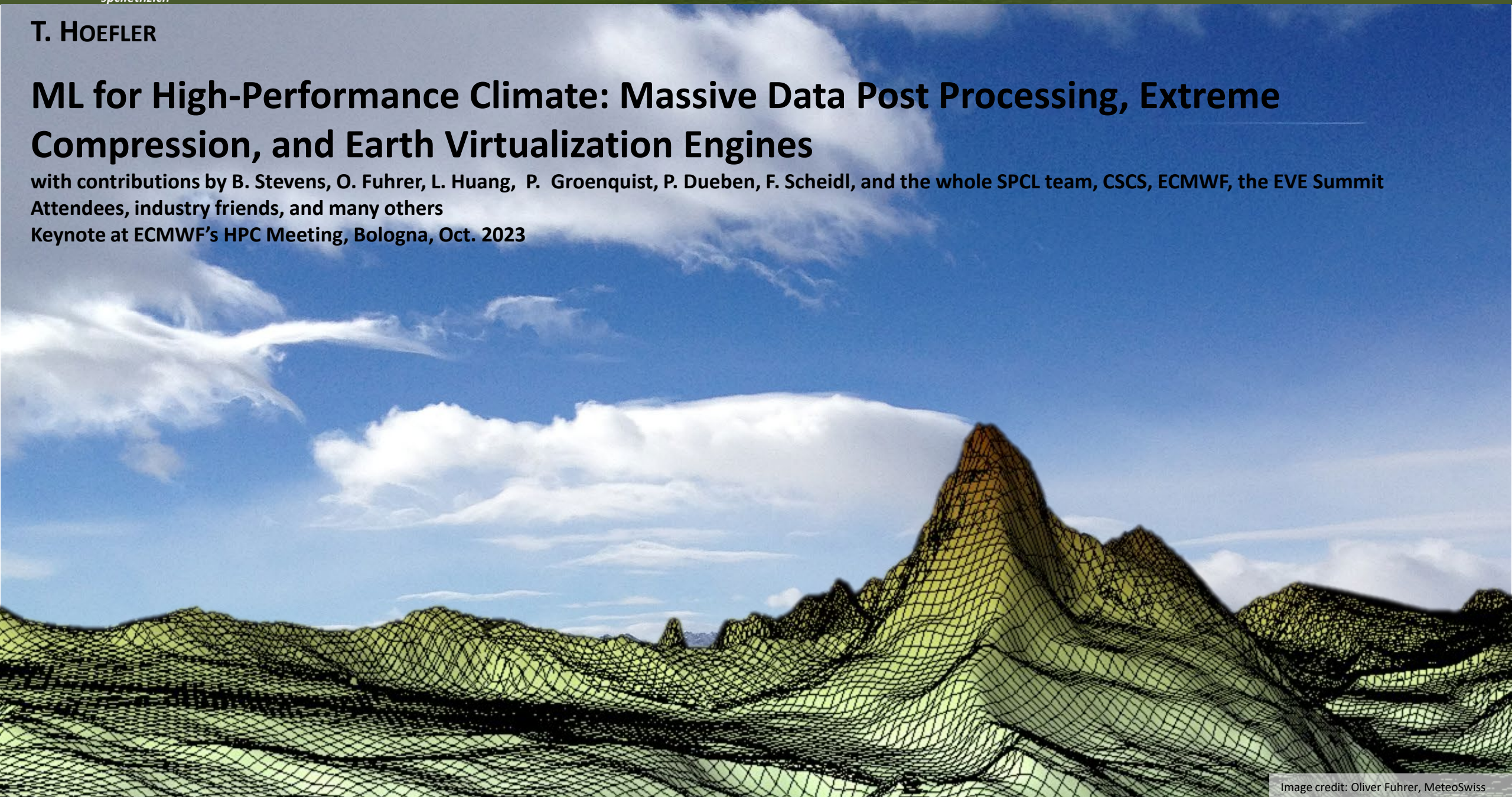


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

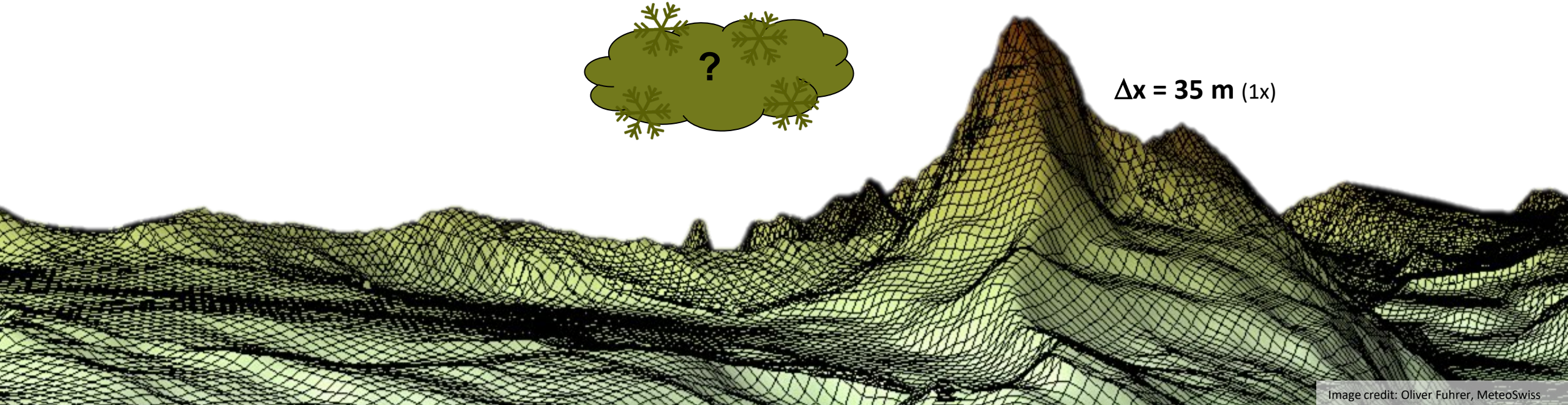


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

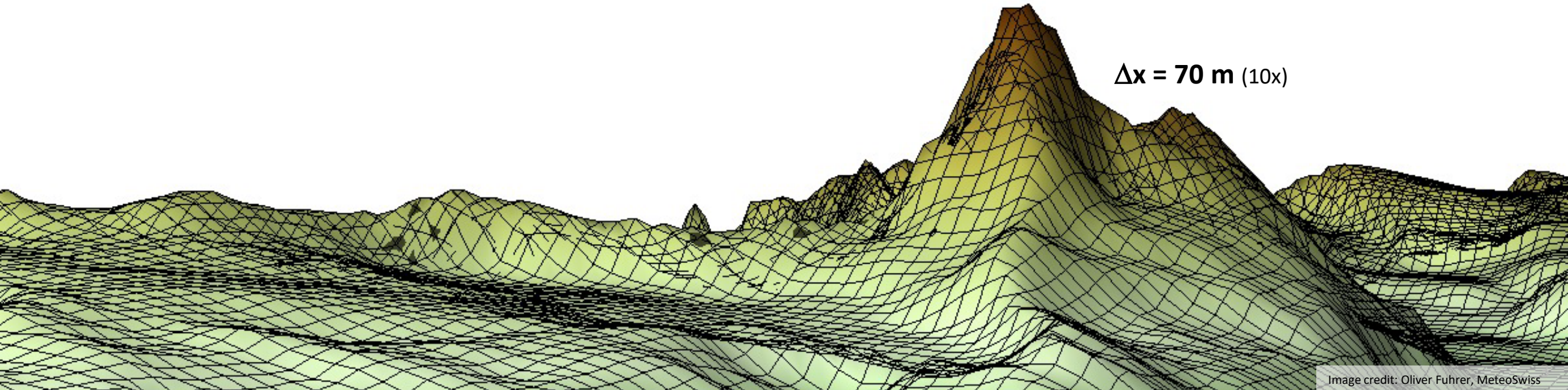


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

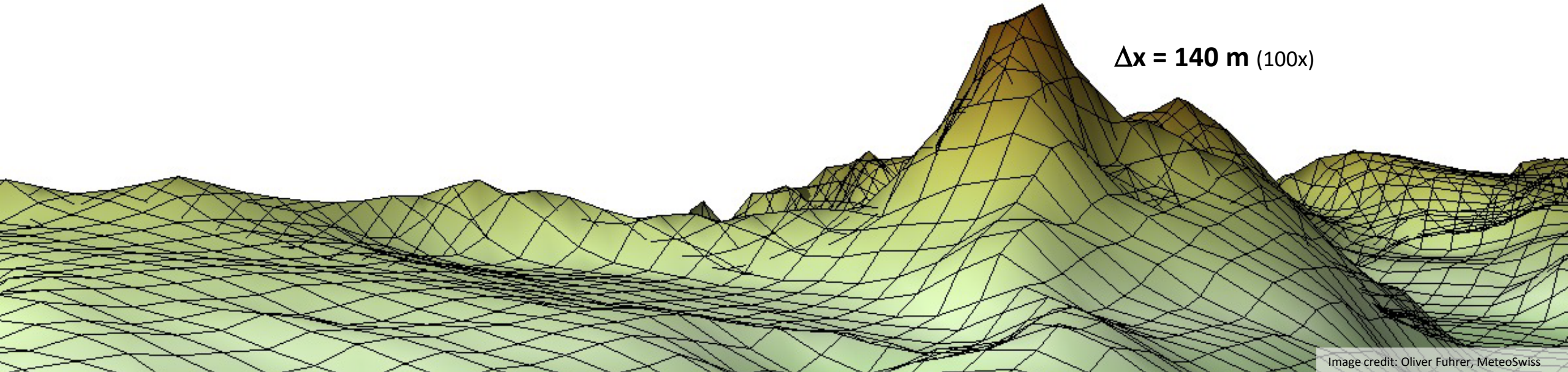


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

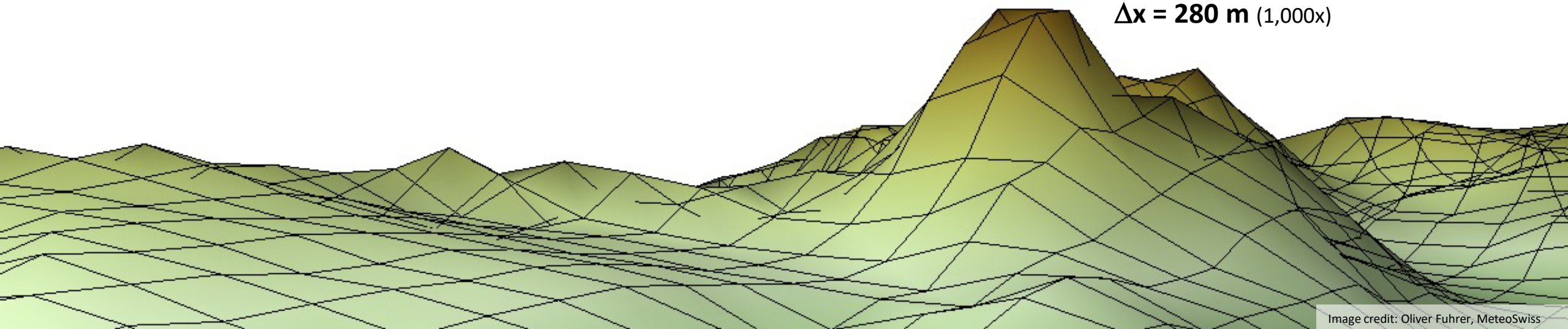


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

Δx = 550 m (10,000x)

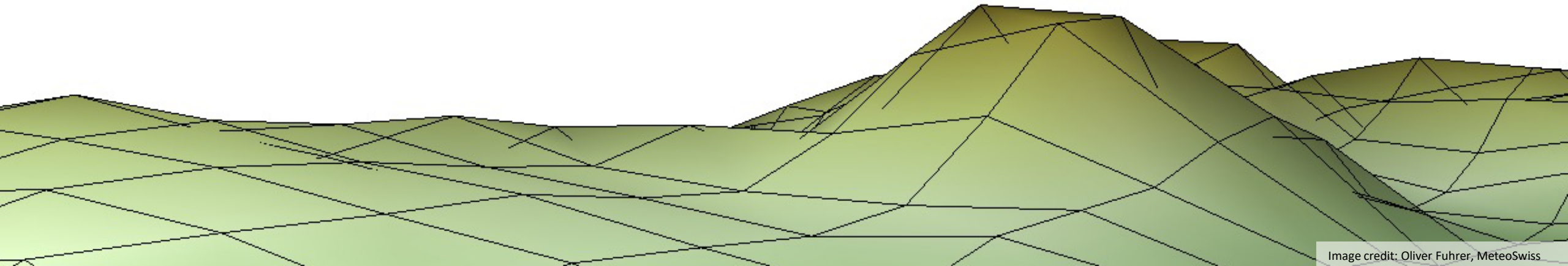


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

Operational model of MeteoSwiss today!

Δx = 1100 m (100,000x)

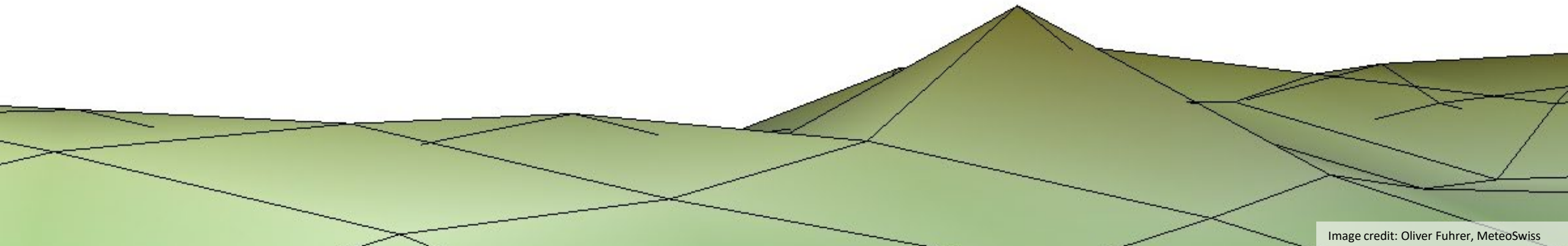


Image credit: Oliver Fuhrer, MeteoSwiss

Intel says Moore's Law is still alive and well. Nvidia says it's ended.

PUBLISHED TUE, SEP 27 2022·3:26 PM EDT

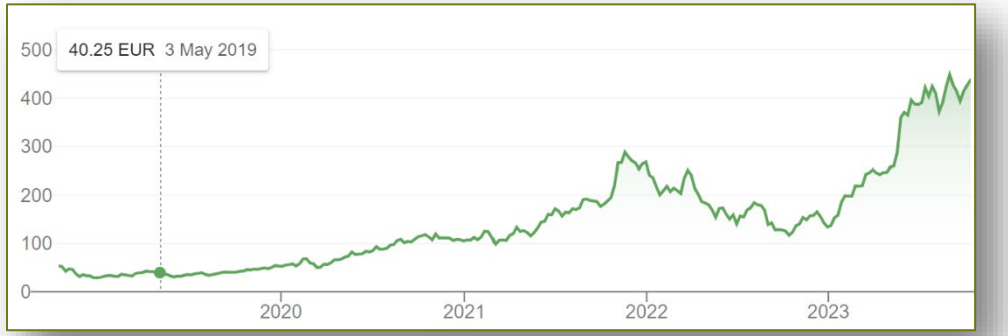
Kif Leswing
@KIFLESWING

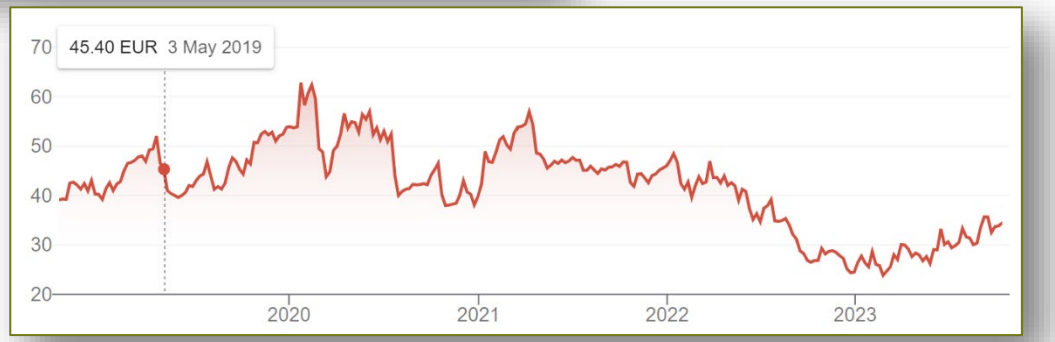
CNBC

WATCH LIVE

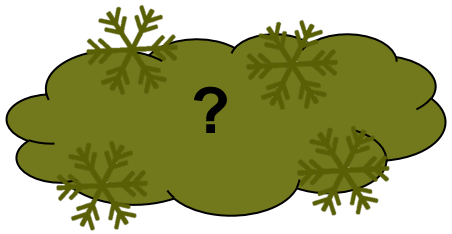
KEY POINTS

- Intel CEO Pat Gelsinger said on Tuesday at a company launch event that Moore's Law is "alive and well."
- Nvidia CEO Jensen Huang said last week Moore's Law has ended.
- Intel has committed to continue manufacturing some of its chips, while Nvidia relies entirely on third-party foundries for its production.





A factor **2x** in resolution roughly corresponds to a factor **10x** compute



Operational model of MeteoSwiss before 2016!

Δx = 2200 m (1,000,000x)

We're a factor of **100,000** away!

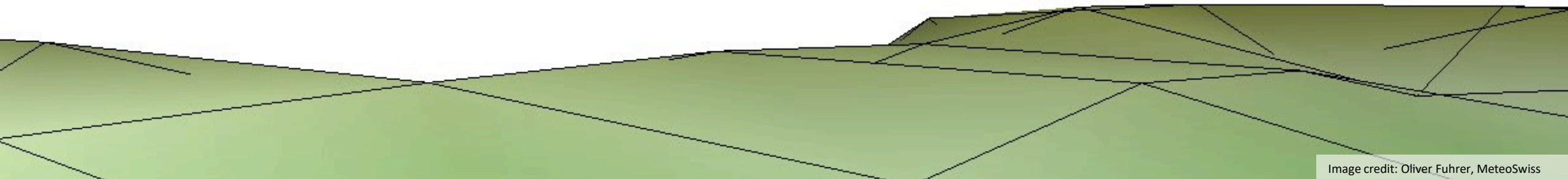
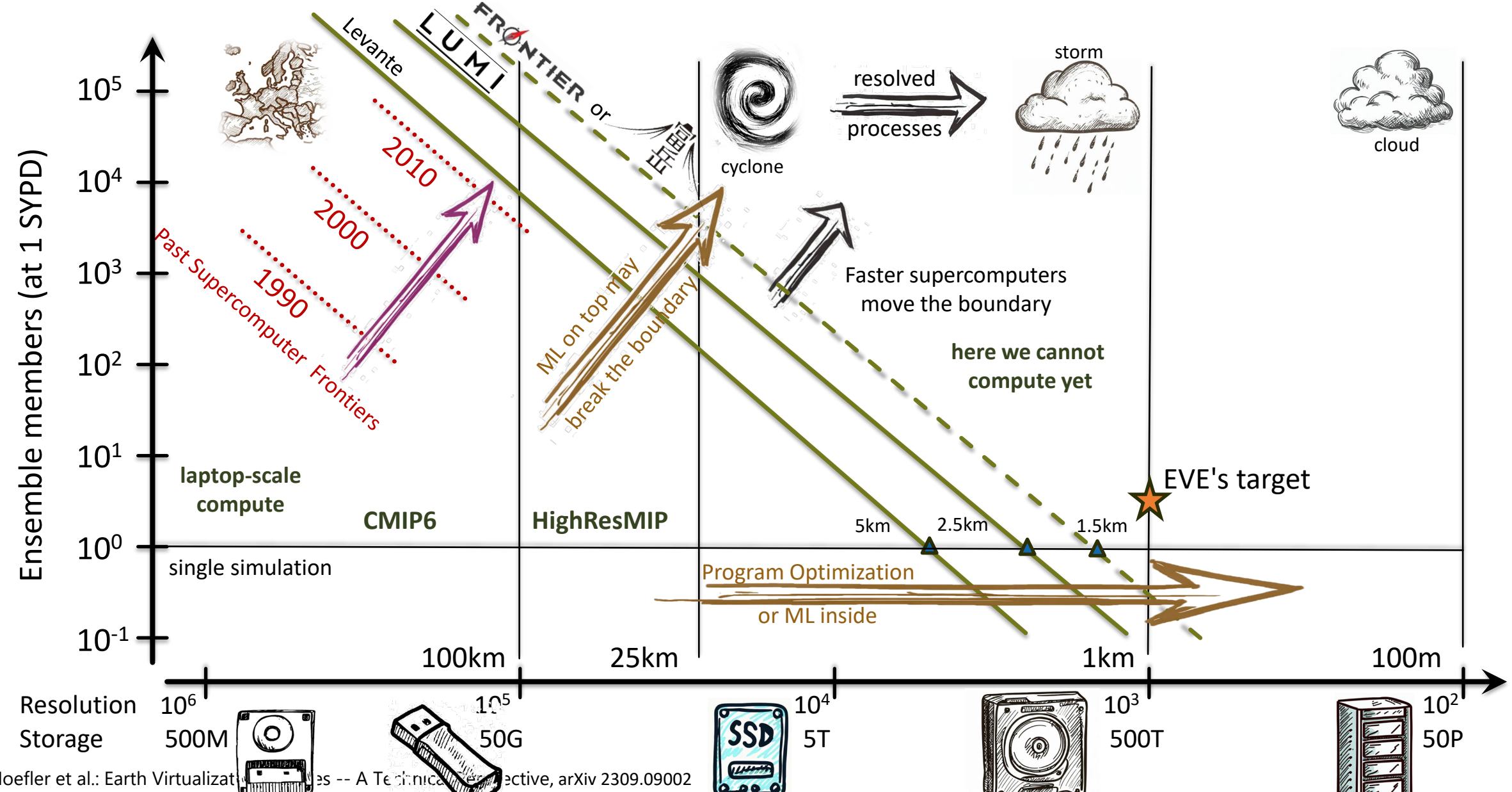
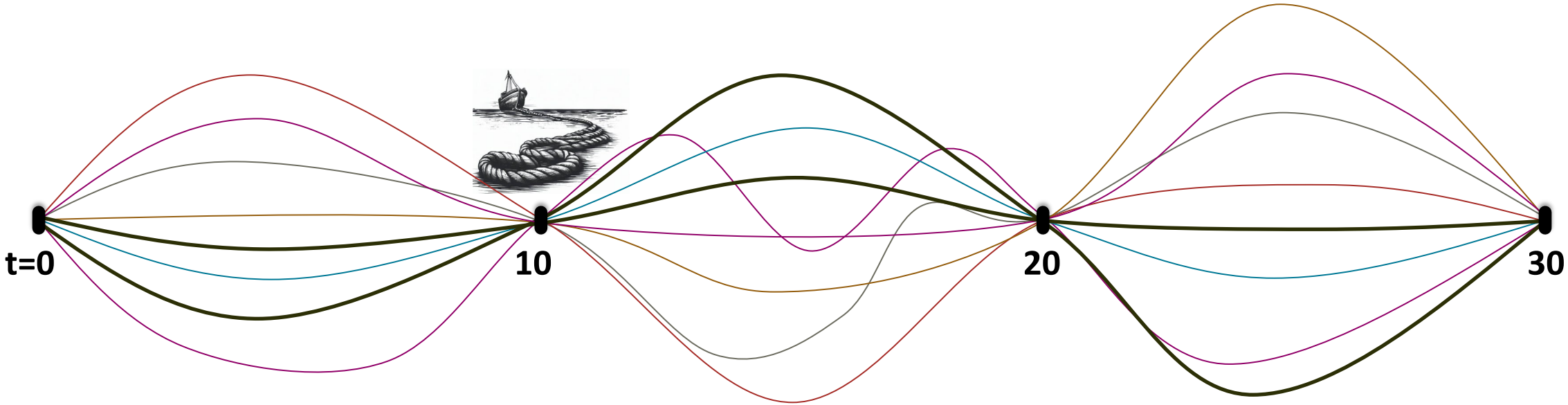


Image credit: Oliver Fuhrer, MeteoSwiss

Climate prediction is even more demanding (decades, physics, storage, …)



From a single to many trajectories / ensemble members



A Changing High-P

Bloomberg Subscribe

Technology | AI

ChatGPT to Fuel $1.3 Trillion AI Market by 2032, New Report Says

- Bloomberg Intelligence expects generative AI market to soar
- Amazon, Microsoft, Google and Nvidia seen as biggest winners

By Jake Rudnitsky
June 1, 2023 at 3:00 PM GMT+2
Updated on June 1, 2023 at 5:50 PM GMT+2

Brian Lamacraft
Chat GTP-4 Could Pass the Bar Exam
How Our Technology Evolves FAST
Source: https://medium.com/

AI chatbot's MBA ex
business schools
ChatGPT earned a solid grade and out

Microsoft invests $1 billion in OpenAI to pursue holy grail of artificial intelligence
Building artificial general intelligence is OpenAI's ambitious goal

A.I. SUPERCOMPUTER
10k GPUs

reddit

A robot may __ injure a
to harm.

	0.74	not 1.00
	0.28	sometimes 0.00
always	0.07	always 0.00

layer-wise weight update

Deep Learning Drives Future Computing Architectures!

Small datatypes
(int + fp – 4, 8, 16 bits)
Example: fp8 is 33x faster than fp64

Matrix and vector ops
(tensor cores and vector units)
Example: NVIDIA TCs: 8.3x over CUDA cores

(Structured) Sparsity
(in tensor cores and vector units)
Example: NVIDIA TCs: 2:4 sparsity

T. Ben-Nun, TH: Demystifying parallel and distributed deep learning: An in-depth concurrency analysis, ACM Computing Surveys (CSUR), 2019

A multi billion dollar (hardware) industry

krtkl
Cambricon 寒武纪科技
Tachyum
adapteva
antmicro
SiFive
Tenstorrent
GRAPHCORE
BITMAIN
cerebras
another brain
thinci
KONIKU
GREENWAVE SYSTEMS
PEZY Computing
Kneron
KNOWM
LIGHTMATTER
LIGHTELLIGENCE
AMD
HUAWEI
d-Matrix
Google
NVIDIA
Qualcomm
IBM


intel Nervana
SambaNova SYSTEMS
MYTHIC
TERADEEP
KALRAY
Think Force
NOVUMIND
AMPERE
Esperanto TECHNOLOGIES
GREENWAVES TECHNOLOGIES
WAVE COMPUTING
gyrfalcon technology
ChipIntelli
DEEPVISION
groq
DEEPSCALE
brainchip The Neuromorphic Computing Company

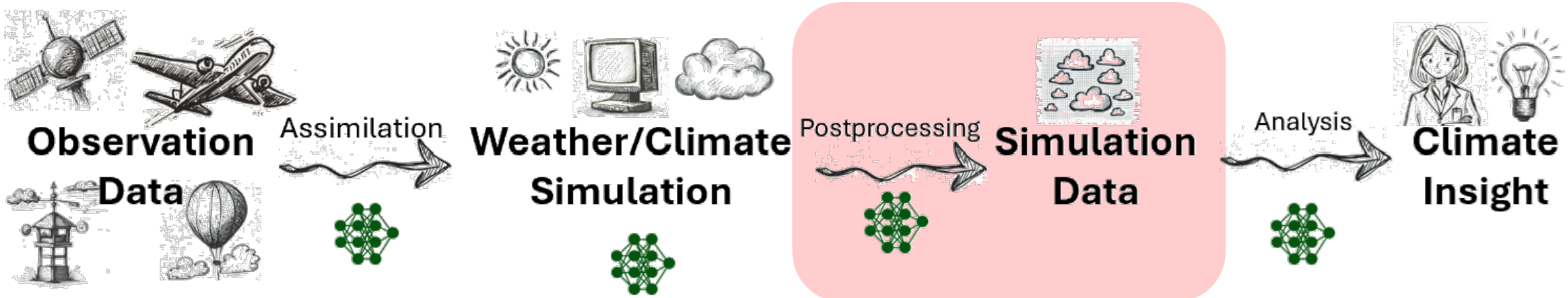
Embracing the future of accelerated computation

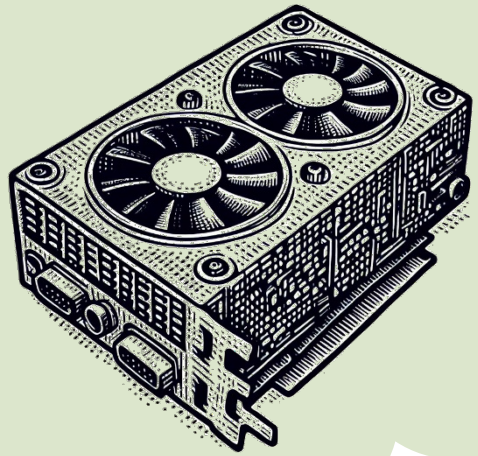
Accelerate Simulations on ML/AI Hardware

- Systems challenges:
  - Enable efficient GPU support
  - Optimize for data movement
- Algorithm challenges:
  - Make simulations look like low-precision matrix multiplication!

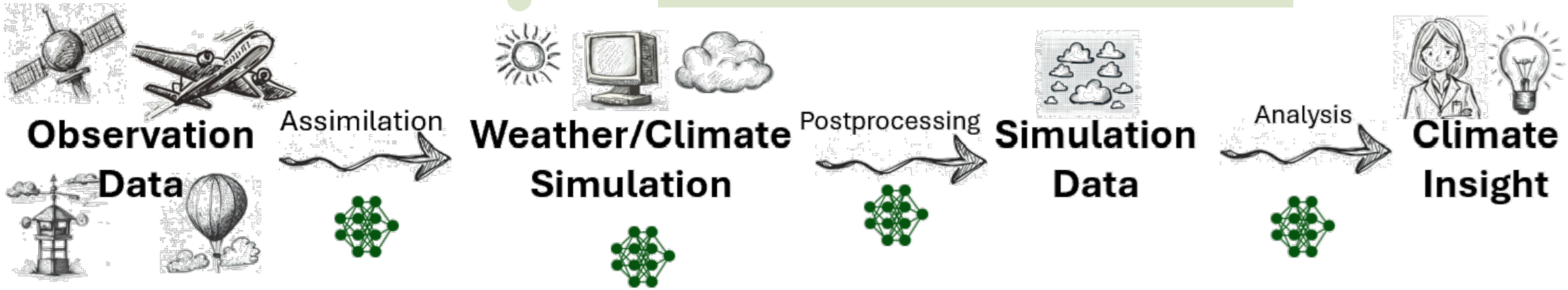
Use ML/AI Techniques ("ML inside/on top")

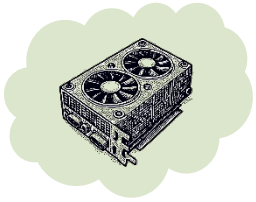
- Use ML models to replace simulations completely
  - E.g., GraphCast, FourCastNet, PanGu, FuXi, …
- Use ML to replace parts of the workflow
  - E.g., physics parametrization in simulations, data post processing, analyses, …





```
!$ACC DATA &
!$ACC PRESENT(density1,energy1) &
!$ACC PRESENT(vol_flux_x,vol_flux_y,volume,mass_flux_x,mass_flux_y,vertexdx,vertexdy) &
!$ACC PRESENT(pre_vol,post_vol,ener_flux)

!$ACC KERNELS

  IF(dir.EQ.g_xdir) THEN

    IF(sweep_number.EQ.1)THEN

!$ACC LOOP INDEPENDENT
      DO k=y_min-2,y_max+2
!$ACC LOOP INDEPENDENT
        DO j=x_min-2,x_max+2
          pre_vol(j,k)=volume(j,k)+(vol_flux_x(j+1,k  )-vol_flux_x(j,k)+vol_flux_y(j  ,k+1)-vol_flux_y(j,k))
          post_vol(j,k)=pre_vol(j,k)-(vol_flux_x(j+1,k  )-vol_flux_x(j,k))
        ENDDO
      ENDDO
    ELSE
!$ACC LOOP INDEPENDENT
      DO k=y_min-2,y_max+2
!$ACC LOOP INDEPENDENT
        DO j=x_min-2,x_max+2
          pre_vol(j,k)=volume(j,k)+vol_flux_x(j+1,k)-vol_flux_x(j,k)
          post_vol(j,k)=volume(j,k)
        ENDDO
      ENDDO

    ENDIF
```





```
!$ACC DATA &
   !$ACC COPY(chunk%tiles(1)%field%density0)   &
   !$ACC COPY(chunk%tiles(1)%field%density1)   &
   !$ACC COPY(chunk%tiles(1)%field%energy0)    &
   !$ACC COPY(chunk%tiles(1)%field%energy1)    &
   !$ACC COPY(chunk%tiles(1)%field%pressure)   &
   !$ACC COPY(chunk%tiles(1)%field%soundspeed) &
   !$ACC COPY(chunk%tiles(1)%field%viscosity)  &
   !$ACC COPY(chunk%tiles(1)%field%xvel0)      &
   !$ACC COPY(chunk%tiles(1)%field%yvel0)      &
   !$ACC COPY(chunk%tiles(1)%field%xvel1)      &
   !$ACC COPY(chunk%tiles(1)%field%yvel1)      &
   !$ACC COPY(chunk%tiles(1)%field%vol_flux_x) &
   !$ACC COPY(chunk%tiles(1)%field%vol_flux_y) &
   !$ACC COPY(chunk%tiles(1)%field%mass_flux_x)&
   !$ACC COPY(chunk%tiles(1)%field%mass_flux_y)&
   !$ACC COPY(chunk%tiles(1)%field%volume)     &
   !$ACC COPY(chunk%tiles(1)%field%work_array1)&
   !$ACC COPY(chunk%tiles(1)%field%work_array2)&
   !$ACC COPY(chunk%tiles(1)%field%work_array3)&
   !$ACC COPY(chunk%tiles(1)%field%work_array4)&
   !$ACC COPY(chunk%tiles(1)%field%work_array5)&
   !$ACC COPY(chunk%tiles(1)%field%work_array6)&
   !$ACC COPY(chunk%tiles(1)%field%work_array7)&
   !$ACC COPY(chunk%tiles(1)%field%cellx)      &
   !$ACC COPY(chunk%tiles(1)%field%celly)      &
   !$ACC COPY(chunk%tiles(1)%field%celldx)     &
   !$ACC COPY(chunk%tiles(1)%field%celldy)     &
   !$ACC COPY(chunk%tiles(1)%field%vertexx)    &
   !$ACC COPY(chunk%tiles(1)%field%vertexdx)   &
   !$ACC COPY(chunk%tiles(1)%field%vertexy)    &
   !$ACC COPY(chunk%tiles(1)%field%vertexdy)   &
   !$ACC COPY(chunk%tiles(1)%field%xarea)      &
   !$ACC COPY(chunk%tiles(1)%field%yarea)      &
   !$ACC COPY(chunk%left_snd_buffer)    &
   !$ACC COPY(chunk%left_rcv_buffer)    &
   !$ACC COPY(chunk%right_snd_buffer)   &
   !$ACC COPY(chunk%right_rcv_buffer)   &
   !$ACC COPY(chunk%bottom_snd_buffer)  &
   !$ACC COPY(chunk%bottom_rcv_buffer)  &
   !$ACC COPY(chunk%top_snd_buffer)     &
   !$ACC COPY(chunk%top_rcv_buffer)
```

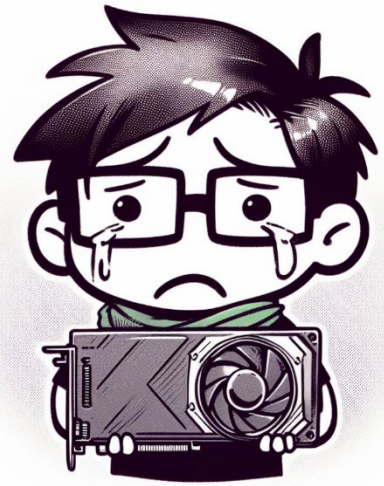
Sloccount *f90: 6,440

,mass_flux_x,mass_flux_y,vertexdx,vertexdy) &

!$ACC: 833 (13%)

1,k
+1,l

l,k)



Heitlager et al.: A Practical Model for Measuring Maintainability

ISO 9126 maintainability	source code properties: volume	complexity per unit	duplication	unit size	unit testing
analysability	x		x	x	x
changeability		x	x		
stability					x
testability		x		x	x

ICON: about 1% OpenMP + 3% OpenACC

Performance Metaprogramming for Optimization and Performance Portability

10s of SLOC

Domain Scientist

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$

NumPy DSLs TensorFlow PyTorch MATLAB

Applied Scientist

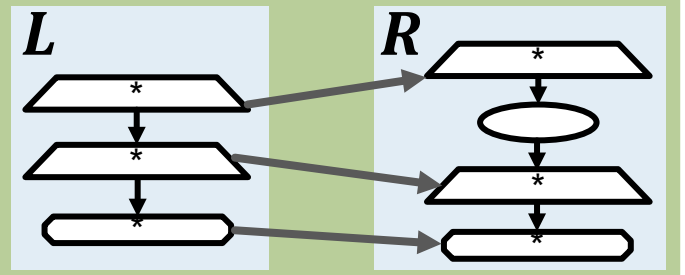
translate DSL into parametric program graph IR

IR Builder API
Multi-Level
Library Nodes

Performance Engineer

100s of reusable SLOC

Parametric Program Graphs (IR)



Graph Transformations (API, Interactive)

Transformed Dataflow

Performance Results

1000s of auto-generated SLOC

Specialized Code Generation

Runtime

CPU Code

GPU Code

FPGA Code

C++ code generation/runtime

Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs, SC19

Why not Good old Fortran? Some say its syntax is not so bad …

```
subroutine q_j_stencil(is,ie,js,je,npz,x_area_flux,area_with_x_flux,q,area,fx1,fx2,q_j)
  integer, intent(in):: is, ie, js, je, npz
  real, intent(in):: x_area_flux(is:ie+1, js:je, npz)
  real, intent(in):: area_with_x_flux(is:ie, js:je, npz)
  real, intent(in):: q(isd:ied, js:je, npz)
  real, intent(in):: area(is:ie, js:je)
  real, intent(inout):: fx1(is:ie+1, js:je, npz)
  real, intent(in):: fx2(is:ie+1, js:je, npz)
  real, intent(out):: q_j(is:ie, js:je, npz)
  integer:: i, j, k
  do k = 1, npz
    do j = js, je
      do i = is, ie+1
        fx1(i,j,k) = x_area_flux(i,j,k)*fx2(i,j,k)
      enddo
      do i = is, ie
        area_with_x_flux(i,j,k) = area(i, j)+x_area_flux(i,j,k)-x_area_flux(i+1,j,k)
      enddo
      do i = is, ie
        q_j(i,j,k) = (q(i,j,k)*area(i,j)+fx1(i,j,k)-fx1(i+1,j,k))/area_with_x_flux(i,j,k)
      enddo
    enddo
  enddo
end subroutine q_j_stencil
```

Domain size embedded to computation

Loop order is fixed

Schedule (fusion, recomputation) is fixed

Memory layout is fixed

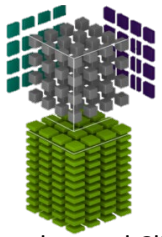
Hardcoded tiling strategies, rank distribution

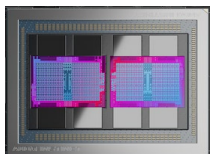
…

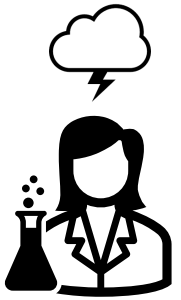
Hardware details fixed

+

A weakening ecosystem







Ben-Nun et al.: Productive Performance Engineering for Weather and Climate Modeling with Python, SC22

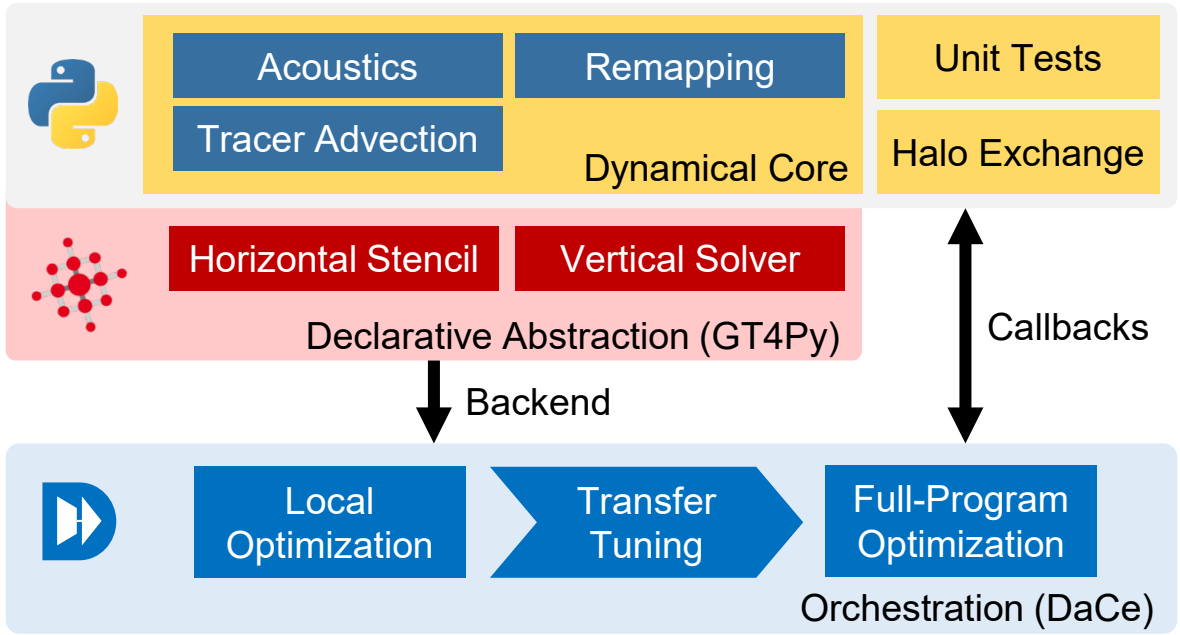
The Pace Project

https://github.com/ai2cm/pace

```
Usage: python -m pace.driver.run [OPTIONS] CONFIG_PATH

  Run the driver.

  CONFIG_PATH is the path to a DriverConfig yaml file.

Options:
  ...
```

- **FV3 reimagined in Python**
  - Goal: Atmospheric model that can run at scale on modern supercomputers
  - No FORTRAN involved – move to 21st century programming + devops + package management (with similar syntax!)
- **Full dynamical core: 12,450 Python LoC across 36 modules**

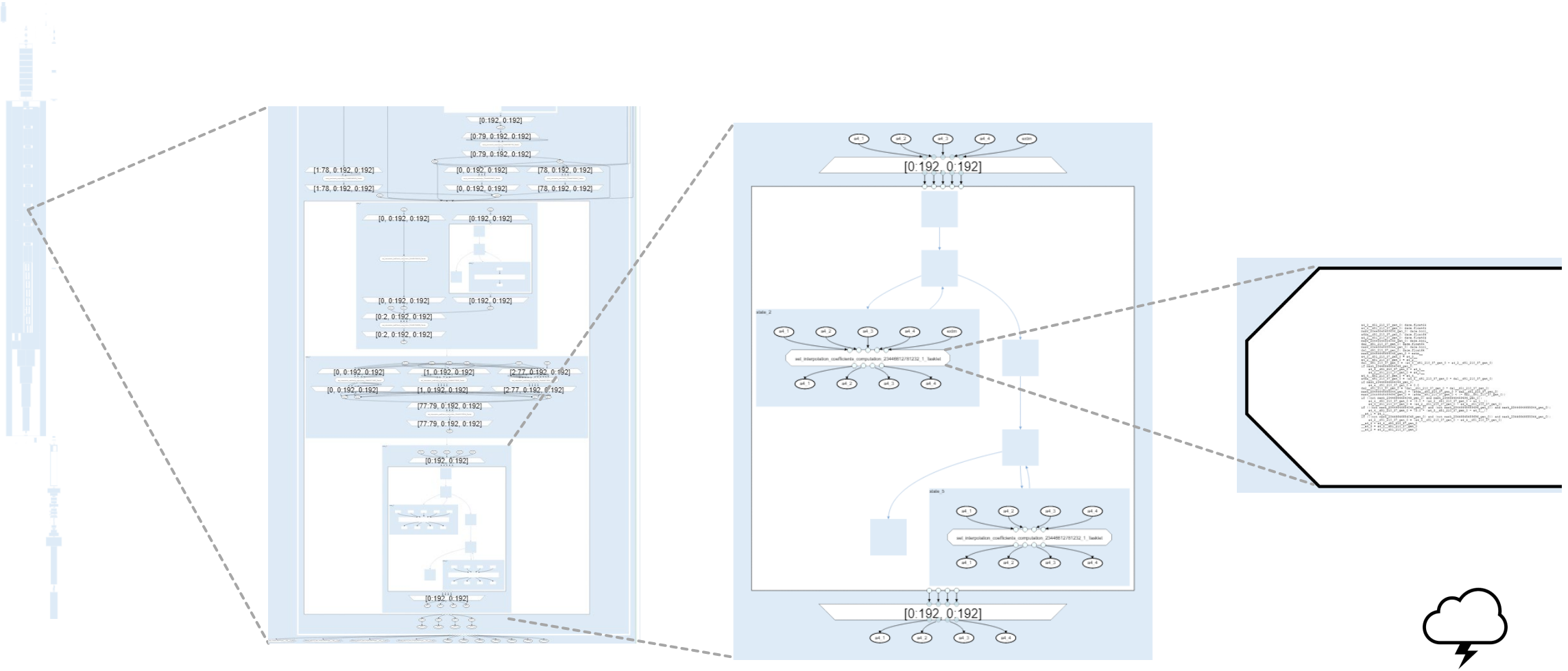
  vs. 29,458 in the baseline implementation

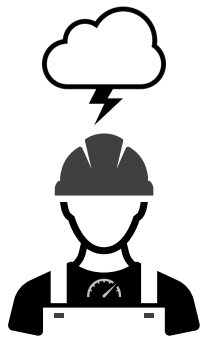


Building around GridTools/Gt4Py

J. Dahm et al., "Pace v0.1: A Python-based Performance-Portable Implementation of the FV3 Dynamical Core". EGUSphere'22

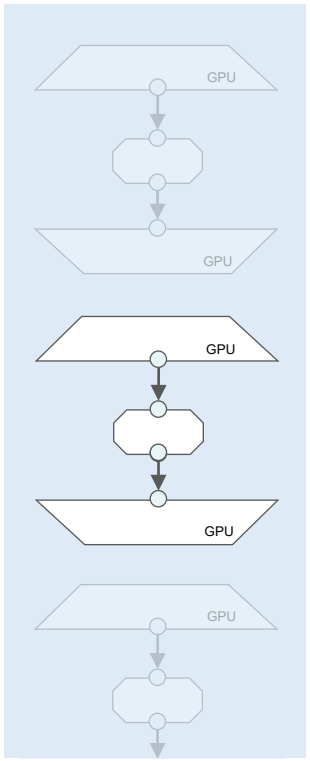
Pace in DaCe for Performance Metaprogramming – 12k SLOC Python





Ben-Nun et al.: Productive Performance Engineering for Weather and Climate Modeling with Python, SC22

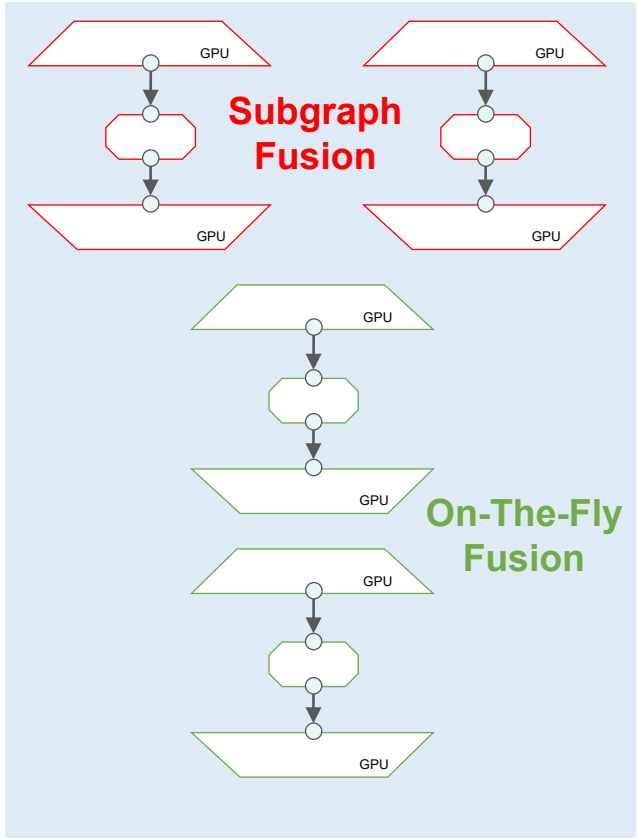
Initial Heuristics → Module-Based Autotuning → Transfer-Tune to Full Application



Exhaustive tuning on graph cutouts

```
[
 {copy_corners_y_nord: 5},
 ...
 {compute_y_flux: 2,
  final_fluxes: 1}
]
```

Store top-k patterns



Test and apply on full program

2:42 hours on Piz Daint

8:24 hours total

Without transfer tuning:

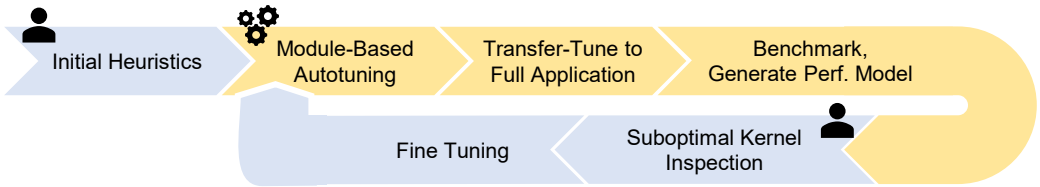
≥30,302,185 configurations

With transfer tuning:

603

Truemper et al.: Performance Embeddings: A Similarity-based Approach to Automatic Performance Optimization, ACM ICS'23, arXiv, 2303.08142

Evaluated Systems



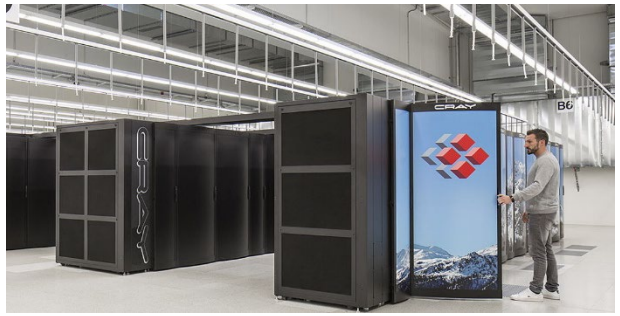


Photo courtesy of the Swiss National Supercomputing Centre

Photo courtesy of Forschungszentrum Jülich

Piz Daint:

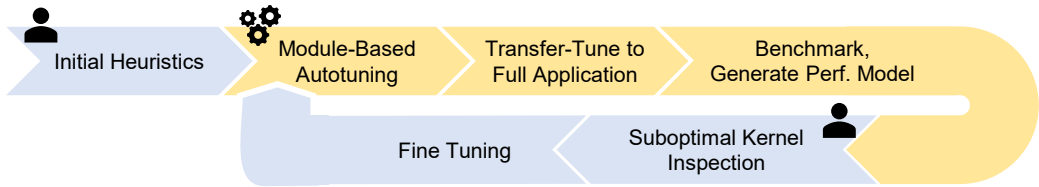
- GPU: 1 x NVIDIA Tesla P100 / Node
- CPU: Intel Xeon E5-2690 v3 (12 cores)

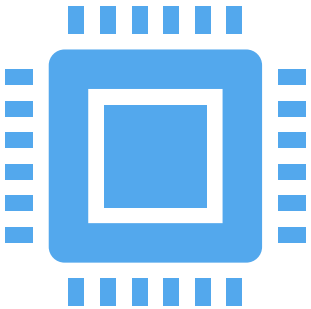
JUWELS Booster:

- GPU: 4 x NVIDIA Tesla A100 / Node
- CPU: AMD EPYC 7402 (2 sockets, 24 cores)

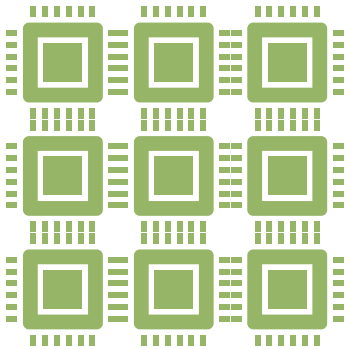
Domain size: 192x192x80

Memory Bounds





43.77 GB/s



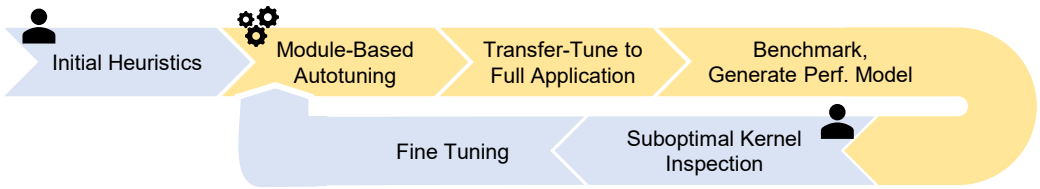
501.1 GB/s

Potential Speedup $\leq$ **11.45x**

Ben-Nun et al.: Productive Performance Engineering for Weather and Climate Modeling with Python, SC22

Representative Vertical Solver

Riemann Solver (`riem_solver_c`)



Semi-implicit solver for nonhydrostatic terms of vertical velocity and pressure perturbation

	FORTRAN		GT4Py+DaCe		
Domain Size (relative size)	Time [ms]	Scaling	Time [ms]	Scaling	Speedup
128×128×80 (1x)	12.27	—	1.85	—	6.63×
192×192×80 (2.25x)	27.94	2.28	3.86	2.08	7.25×
256×256×80 (4x)	52.40	4.27	6.96	3.76	7.53×
384×384×80 (9x)	121.80	9.92	15.31	8.26	7.96×

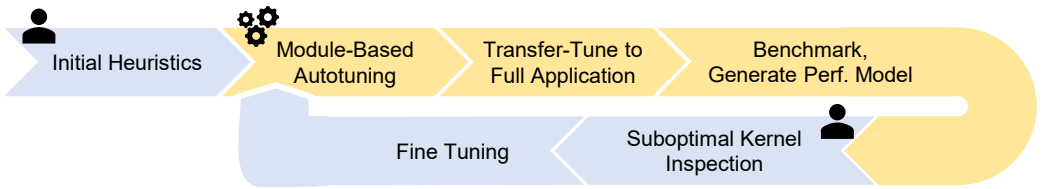
CPU cache runs out, data layout not ideal

Not enough parallelism

Ben-Nun et al.: Productive Performance Engineering for Weather and Climate Modeling with Python, SC22

Representative Horizontal Stencil

Finite Volume Transport (`fv_tp_2d`)



FORTRAN runs on a **single slice**, GT4Py/DaCe runs on entire 3D domain

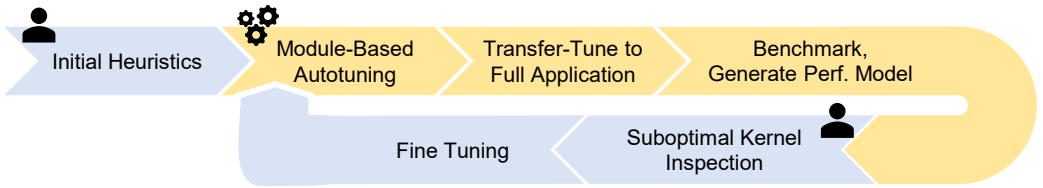
	FORTRAN		GT4Py+DaCe		
Domain Size (relative size)	Time [ms]	Scaling	Time [ms]	Scaling	Speedup
128×128×80 (1x)	3.41	—	1.81	—	1.88×
192×192×80 (2.25x)	12.31	3.61	3.41	1.88	3.61×
256×256×80 (4x)	35.79	10.49	5.67	3.13	6.31×
384×384×80 (9x)	106.66	31.27	13.10	7.23	8.14×

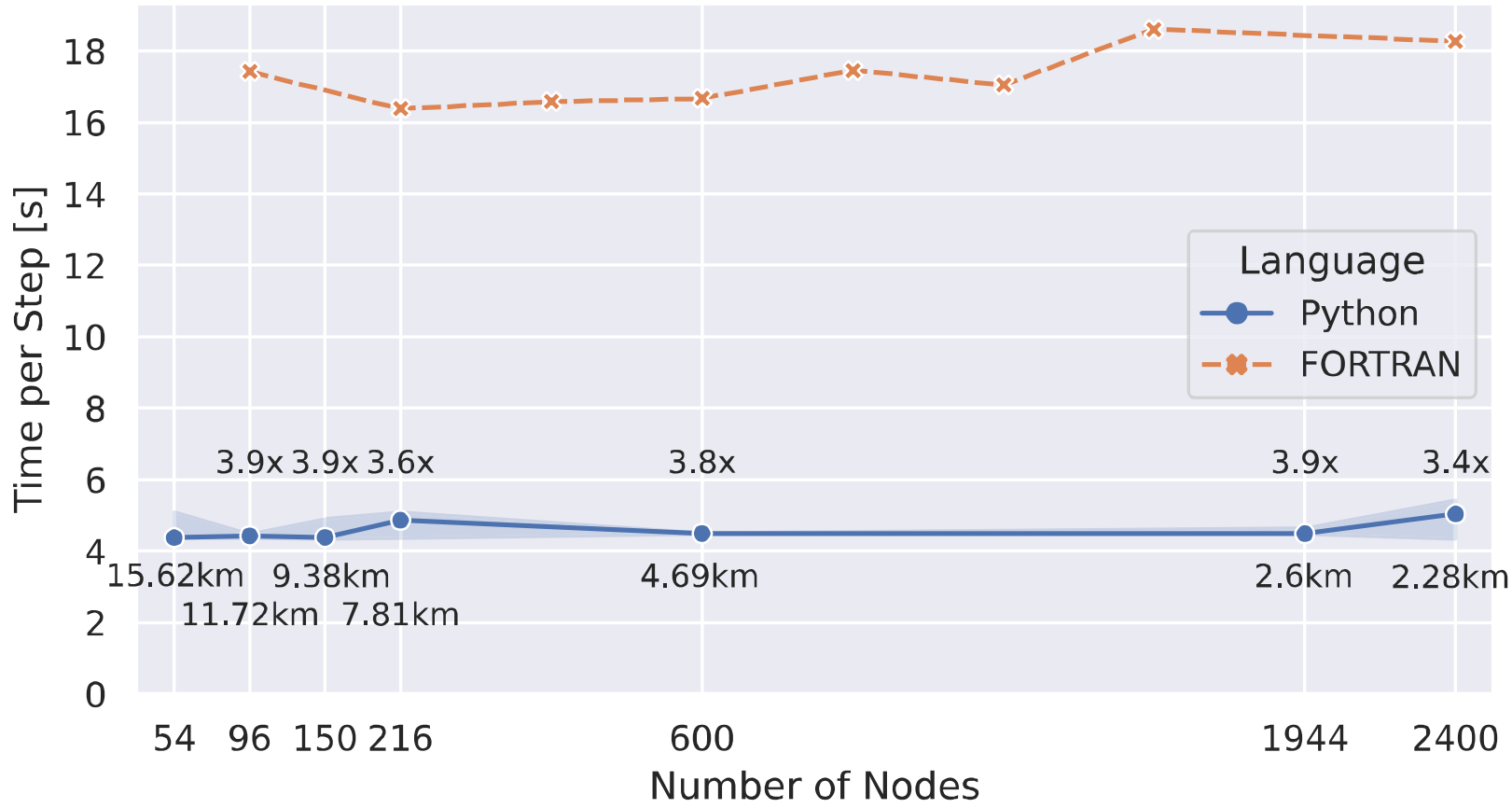
0.13% of load/stores are L3 misses

Closing gap to ideal memory bandwidth factor

Ben-Nun et al.: Productive Performance Engineering for Weather and Climate Modeling with Python, SC22

Weak Scaling





Simulation throughput of **0.12 SYPD** at 2.6 km grid spacing

Ben-Nun et al.: Productive Performance Engineering for Weather and Climate Modeling with Python, SC22

https://github.com/ai2cm/pace
https://github.com/GridTools/gt4py
https://github.com/spcl/dace

That's all nice but do we really want to rewrite all codes?

6	10	4	3.92 – 8.48x	0
weeks of work	optimization revisions	performance engineers	speedup vs. production FORTRAN	model changes

youtube.com/@spcl

Ben-Nun et al.: Productive Performance Engineering for Weather and Climate Modeling with Python, SC22

Ok, a second look at Fortran

RAJA GridTools PyTorch kokkos TensorFlow ...

- Lots of DSLs for productivity & portability
- I somewhat see those as an attempt to uplevel performance libraries
  - BLAS, LINPACK, LAPACK, etc.

The core of many scientific problems is manipulation of arrays with scientific formulas

Still doubts about future, ecosystem, and productivity … let's be pragmatic …

- Invented by Turing Awardee John Backus in 1957 (at IBM)
  - At the time, people were skeptical that it would beat assembly
  - Gave rise to the first optimizing compiler!

(Tiobe index 2023)

14	20		MATLAB	0.86%	+0.12%
15	24		Scratch	0.79%	+0.13%
16	11		R	0.76%	-0.79%
17	14		Swift	0.72%	-0.28%
18	15		Ruby	0.66%	-0.22%
19	28		Rust	0.63%	+0.18%
20	31		Fortran	0.59%	+0.24%

TIOBE- The programming language popularity index has come out with the updated list of the top coding languages for April month. While the programming language C has attained the first position, Fortran made a comeback to the top-20 list .

1612 Views 7 Apr 2021, 12:12 PM
Sukriti Yaduwanshi

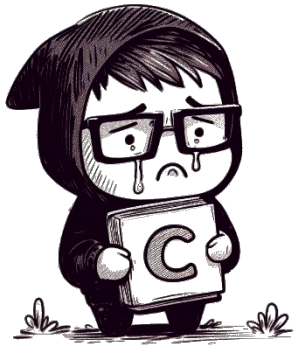
The programming language ratings for April 2021 are out from TIOBE. The Popularity index, which is updated once a month, rates the coding languages based on the number of skilled engineers world-wide, courses, and third party vendors. Data from various search engines like Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube etc. is used to calculate the ratings.

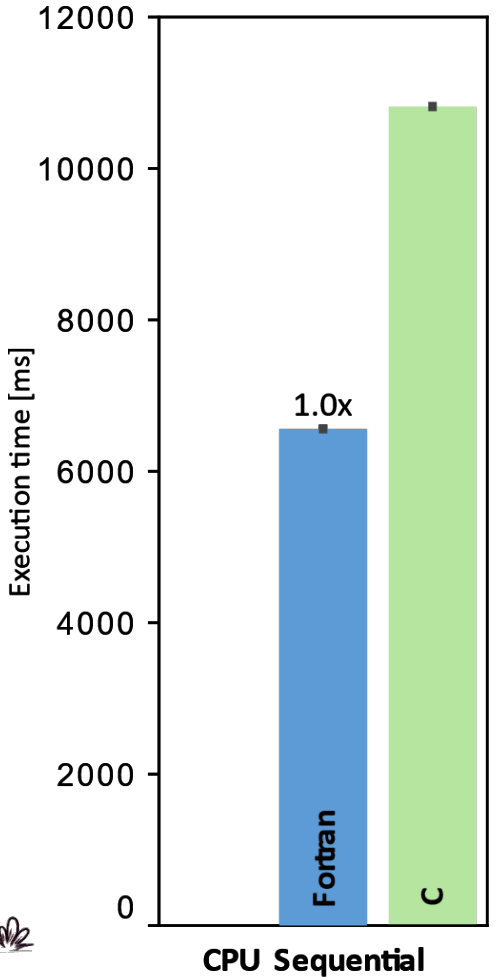
Let's get to some real example … ECMWF's CLOUDSC

```
 9
10  SUBROUTINE CLOUDSC &
11   !---input
12   & (KIDIA,    KFDIA,    KLON,    KLEV, &
13   & PTSPHY,&
14   & PT, PQ, tendency_cml,tendency_tmp,tendency_loc, &
15   & PVFA, PVFL, PVFI, PDYNA, PDYNL, PDYNI, &
16   & PHRSW,    PHRLW,&
17   & PVERVEL,  PAP,      PAPH,&
18   & PLSM,     LDCUM,    KTYPE, &
19   & PLU,      PLUDE,    PSNDE,    PMFU,     PMFD,&
20   !---prognostic fields
21   & PA,&
22   & PCLV,  &
23   & PSUPSAT,&
24  !-- arrays for aerosol-cloud interactions
25  !!! & PQAER,    KAER, &
26   & PLCRIT_AER,PICRIT_AER,&
27   & PRE_ICE,&
28   & PCCN,     PNICE,&
29   !---diagnostic output
30   & PCOVPTOT, PRAINFRAC_TOPRFZ,&
31   !---resulting fluxes
32   & PFSQLF,   PFSQIF ,  PFCQNNG,  PFCQLNG,&
33   & PFSQRF,   PFSQSF ,  PFCQRNG,  PFCQSNG,&
34   & PFSQLTUR, PFSQITUR , &
35   & PFPLSL,   PFPLSN,   PFHPSL,   PFHPSN, KFLDX, &
36   & YDCST, YDTHF, YDECLDP)
```

… variable setup/initialization until line 500 ;-)

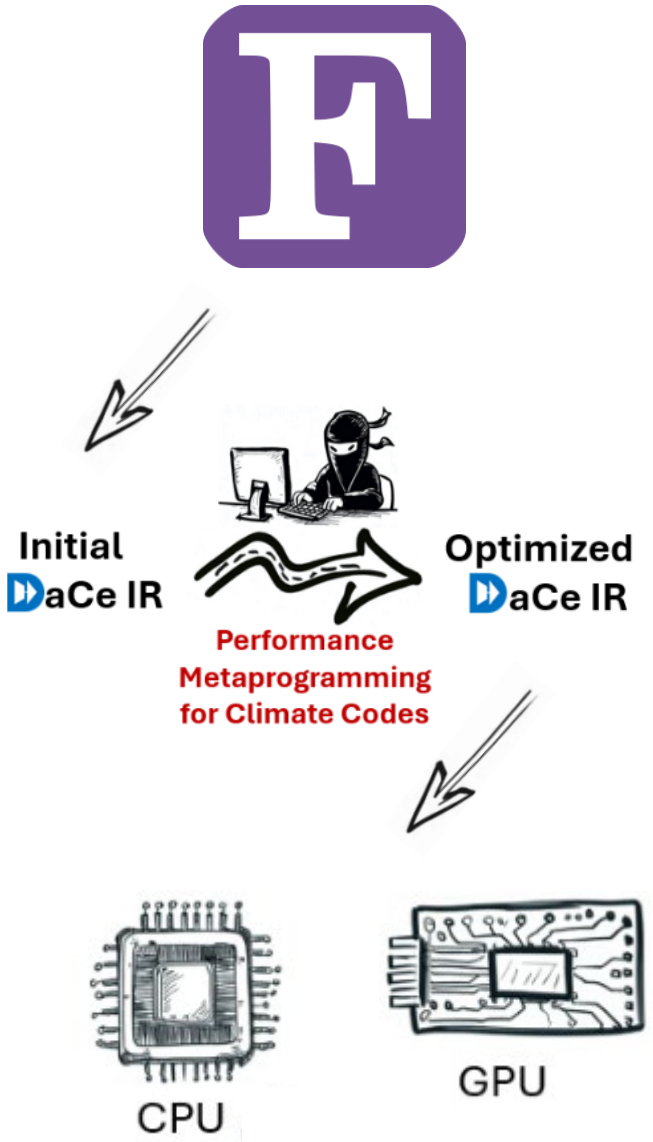
- **Cloud Microphysics of IFS**
  - Resolve sub-grid features
  - Original 2,525 SLOC of Fortran 95
- **Rewritten for performance portability benchmarking (optimization took months!)**
  - 2,635 SLOC C
  - 2,610 SLOC C++/CUDA

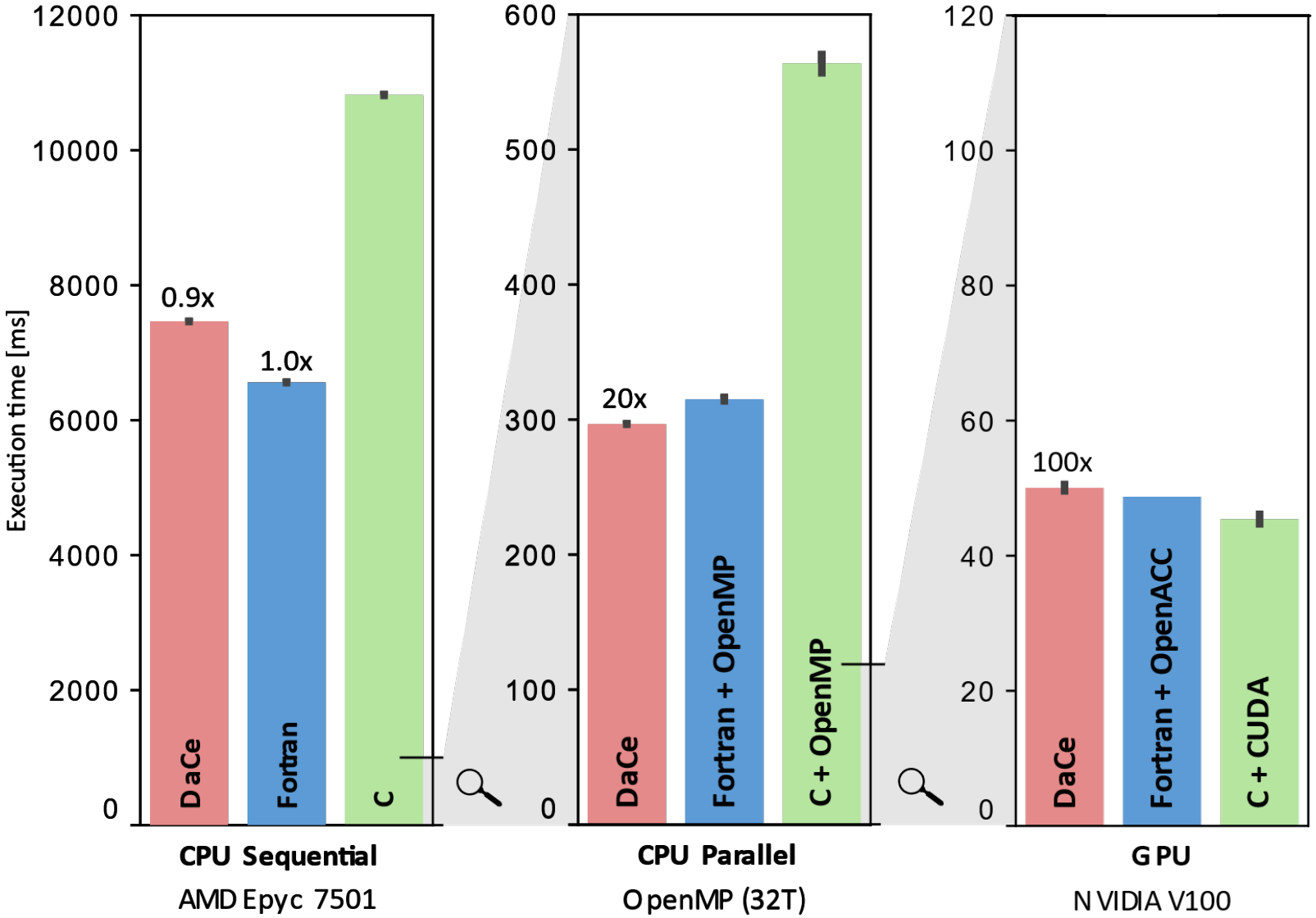




https://github.com/ecmwf-ifs/dwarf-p-cloudsc

Performance Metaprogramming – from the unchanged CLOUDSC Fortran code!





Embracing the future of accelerated computation

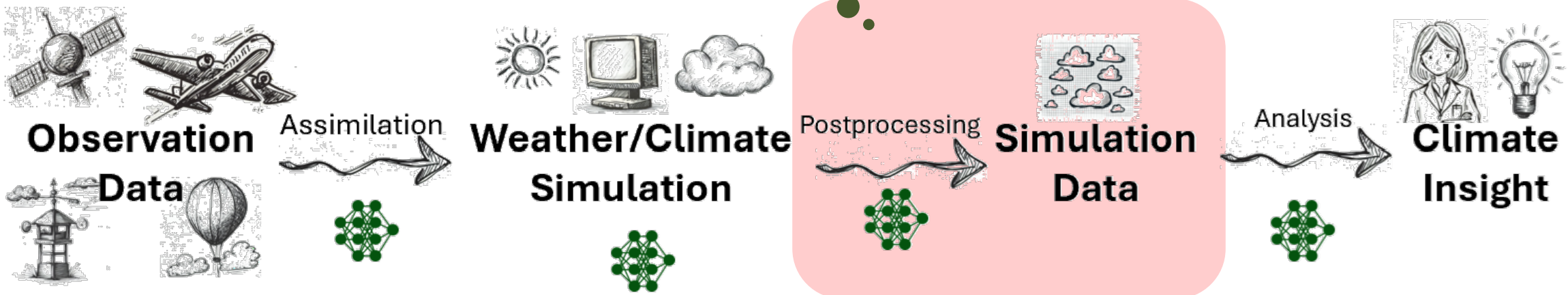
Accelerate Simulations on ML/AI Hardware

- Systems challenges:
  - Enable efficient GPU support
  - Optimize for data movement
- Algorithm challenges:
  - Make simulations look like low-precisi
    matrix multiplication!

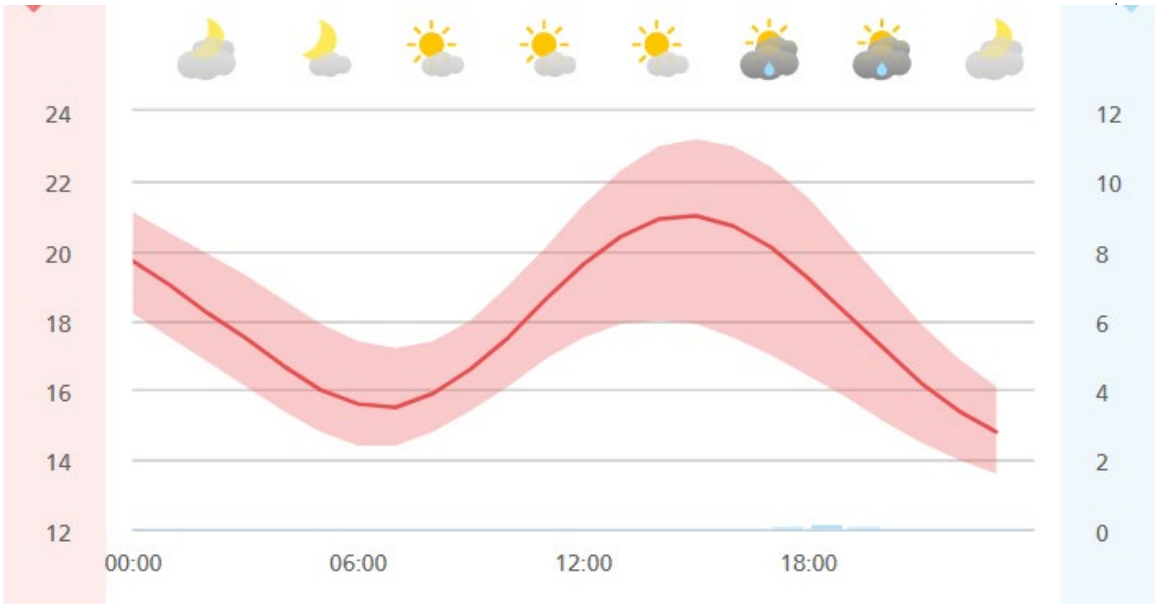
Use ML/AI Techniques

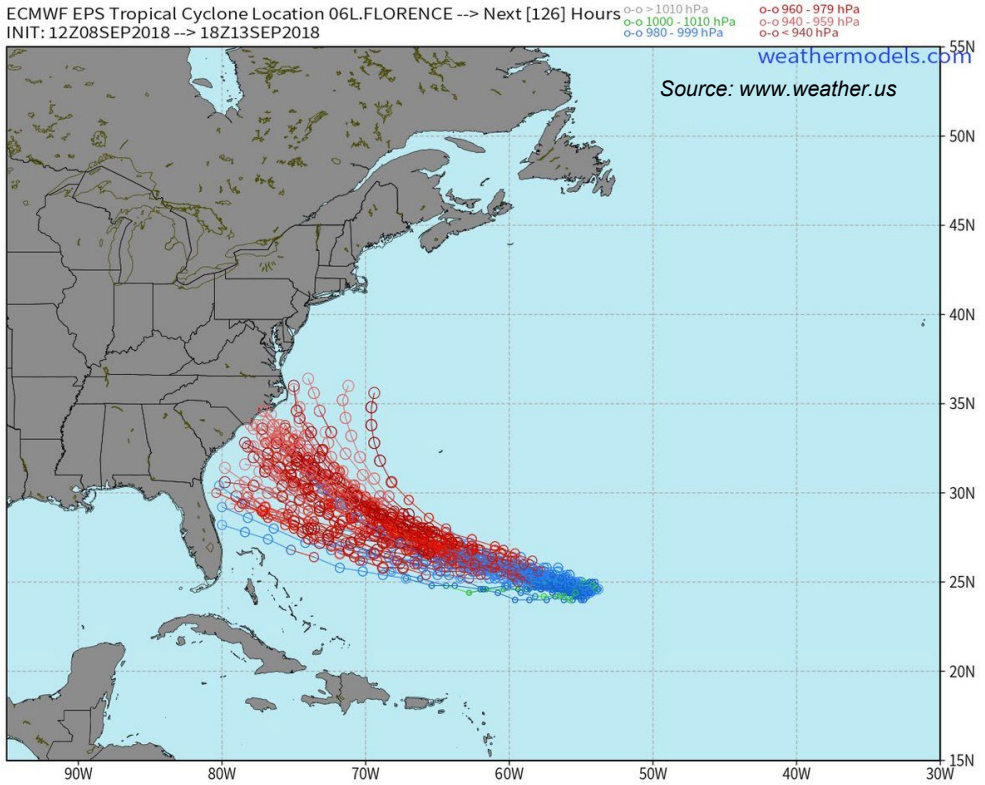
- Use ML models to replace simulations completely
  - E.g., GraphCast, FourCastNet, PanGu, FuXi, …
- Use ML to replace parts of the workflow
  - E.g., physics parametrization in simulations, data post processing, analyses, …

"ML on top"

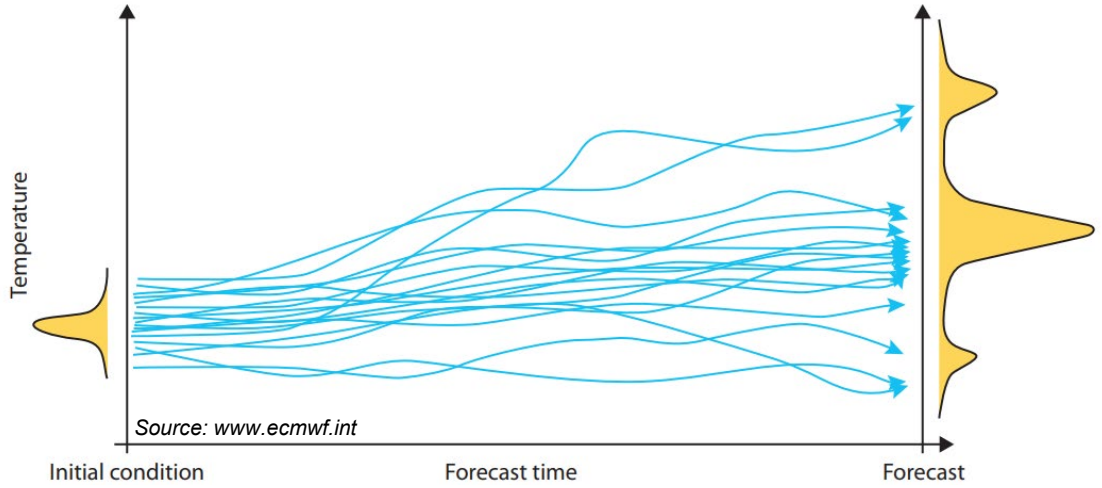


Uncertainty in forecasting





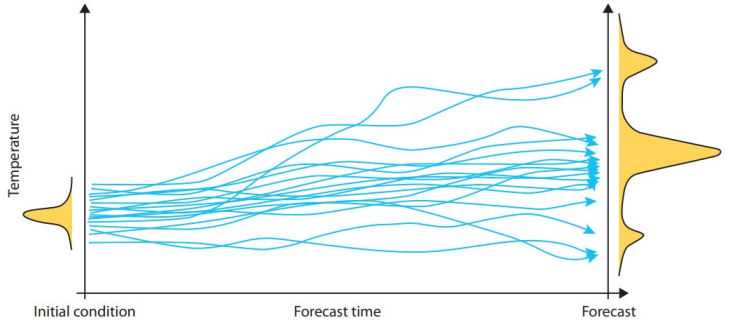
- **Weather is a chaotic system**
  - Minor perturbations affect the outcome the further into the future we predict
- **Solution: Ensemble Prediction Systems – predict weather as a probability distribution**
  - Approximated by partial differential equations with perturbed inputs



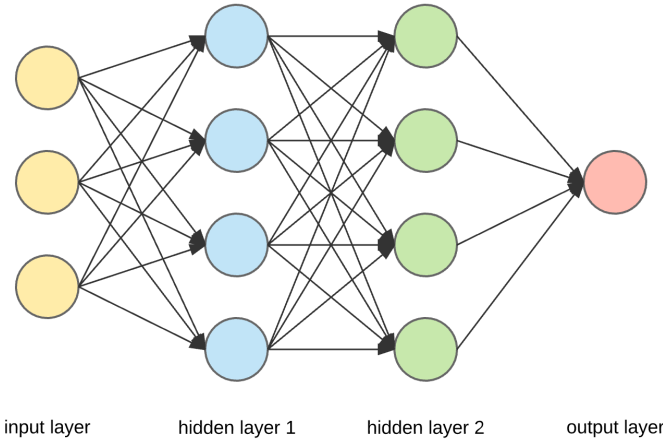
Ensemble Prediction System at ECMWF

- **Initial condition uncertainties result from data assimilation of sensor data**

- **51 ensemble members, 1 control (deterministic), 50 perturbed (stochastic)**
  - Approximate the highest likely trajectory from output distribution D
  - Lower resolution (9km vs. 18km) in order to fit compute budget
    mostly an economic argument

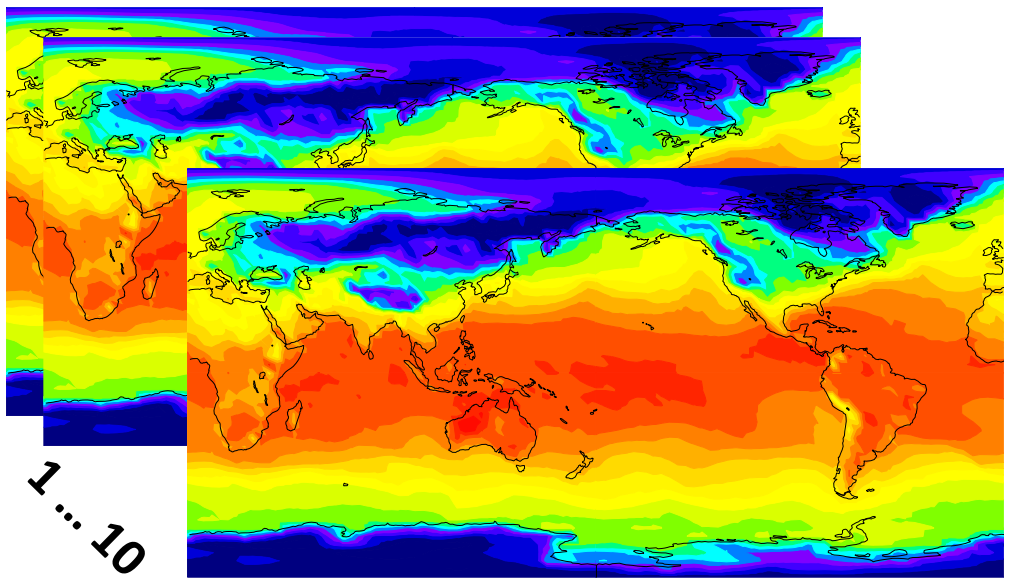


- **Next step in the economic argument:**
  - Could the number of ensemble members be reduced without sacrificing accuracy?
  - Idea I: predict mean and standard deviation (StdDev) of D from a smaller ensemble
    This may allow us to increase resolution at equal cost – better predictions
  - Can we improve prediction quality by learning from ground truth observations?
  - Idea II: learn (local) model bias from observations
    This may allow us to increase accuracy – better predictions



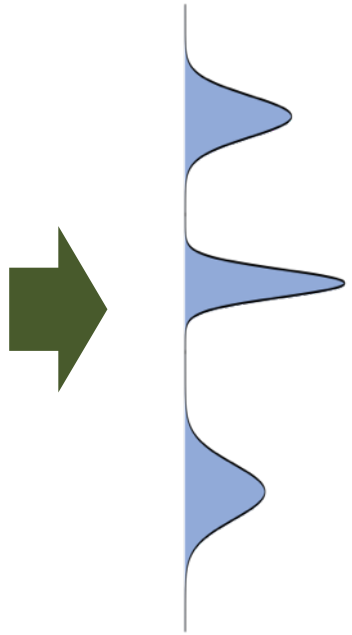
ENS-10: A Dataset For Post-Processing Ensemble Weather Forecasts

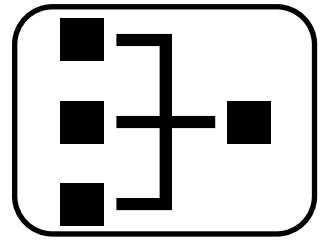
Ashkboos et al., NeurIPS'22



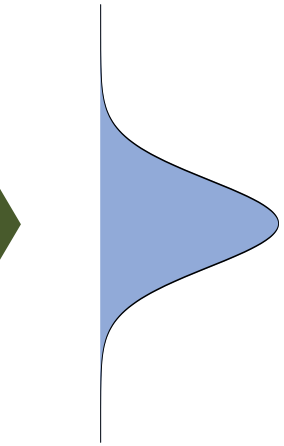
ENS-10

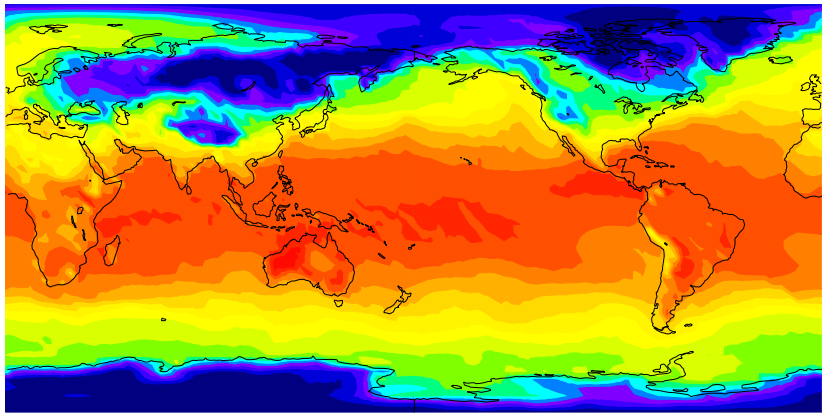
(biased distribution)



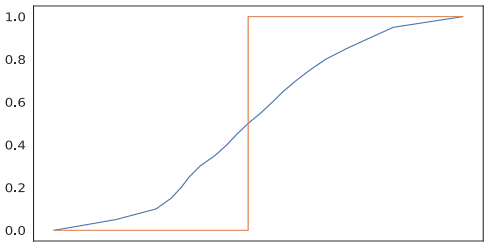


(corrected distribution)





ERA5
(ground truth)



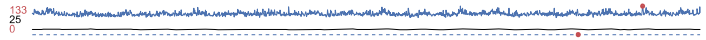
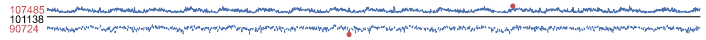
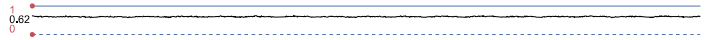
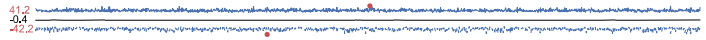
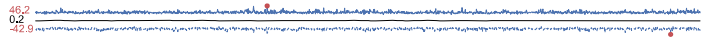
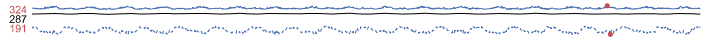
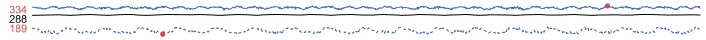
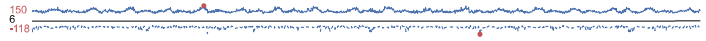
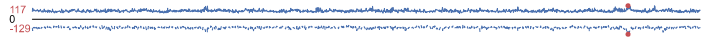
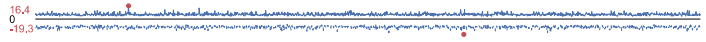
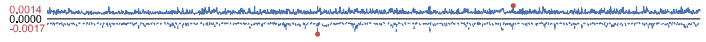
Score

S. Ashkboos et al.: ENS-10: A Dataset For Post-Processing Ensemble Weather Forecasts, NeurIPS'22, arXiv

ENS-10: Variables

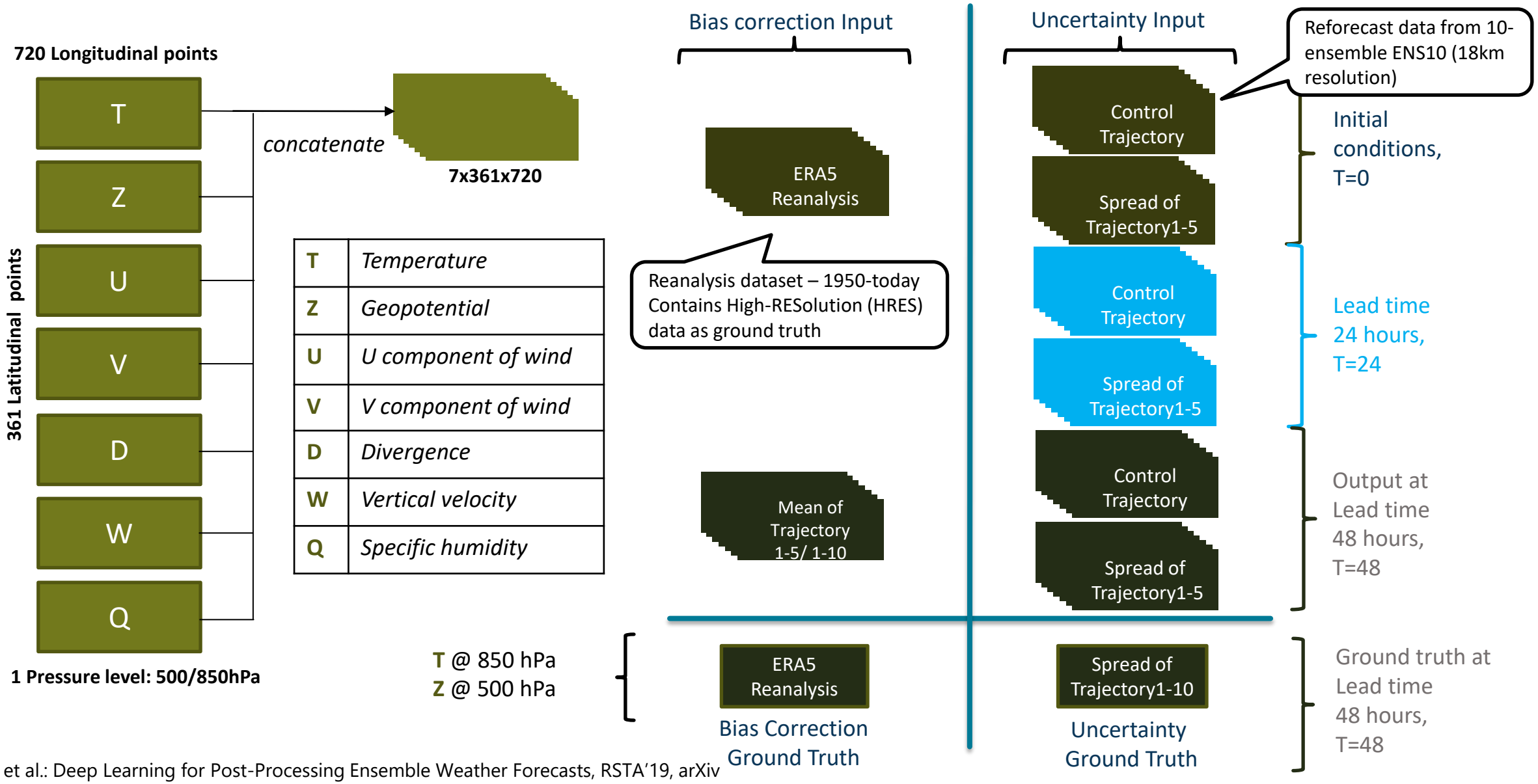
ENS-10 dataset is designed for post-processing ensemble weather forecasts
- produced by the ECMWF production re-forecast model (IFS).

- 10 ensemble members
- 20 years of data (1998–2017), ~3 TB
- 2 forecasts per week, lead times 0, 24, and 48 h
- 11 surface variables
- 7 vertical variables on 11 pressure levels
- 0.5° spatial resolution (higher than previous datasets)

Name	Unit	48 h forecast min, mean, max (1998–2015)
Surface:		
Sea surface temperature	K	
Total column water	kg/m^2	
Total column water vapor	kg/m^2	
Convective precipitation	m	
Mean sea level pressure	Pa	
Total cloud cover	(0–1)	
10 m U wind component	m/s	
10 m V wind component	m/s	
2 m temperature	K	
Total precipitation	m	
Skin temperature at surface	K	
Pressure levels (10, 50, 100, 200, 300, 400, 500, 700, 850, 925 and 1000 hPa):		
U wind component	m/s	
V wind component	m/s	
Geopotential	m^2/s^2	
Temperature	K	
Specific humidity	$kg\,kg^{-1}$	
Vertical velocity	Pa/s	
Divergence	1/s	

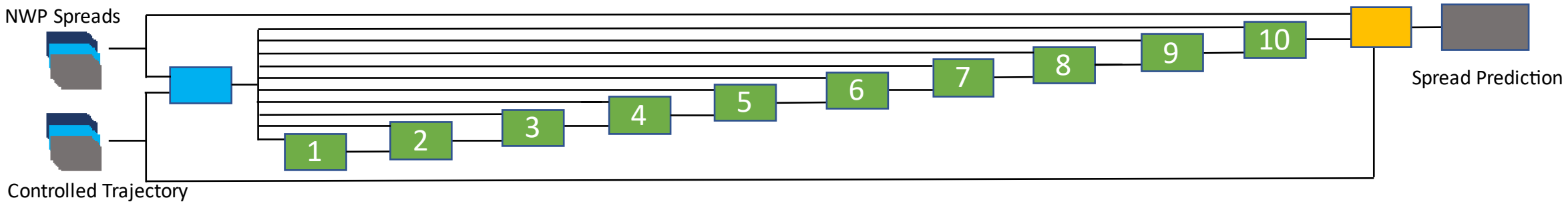
S. Ashkboos et al.: ENS-10: A Dataset For Post-Processing Ensemble Weather Forecasts, NeurIPS'22, arXiv

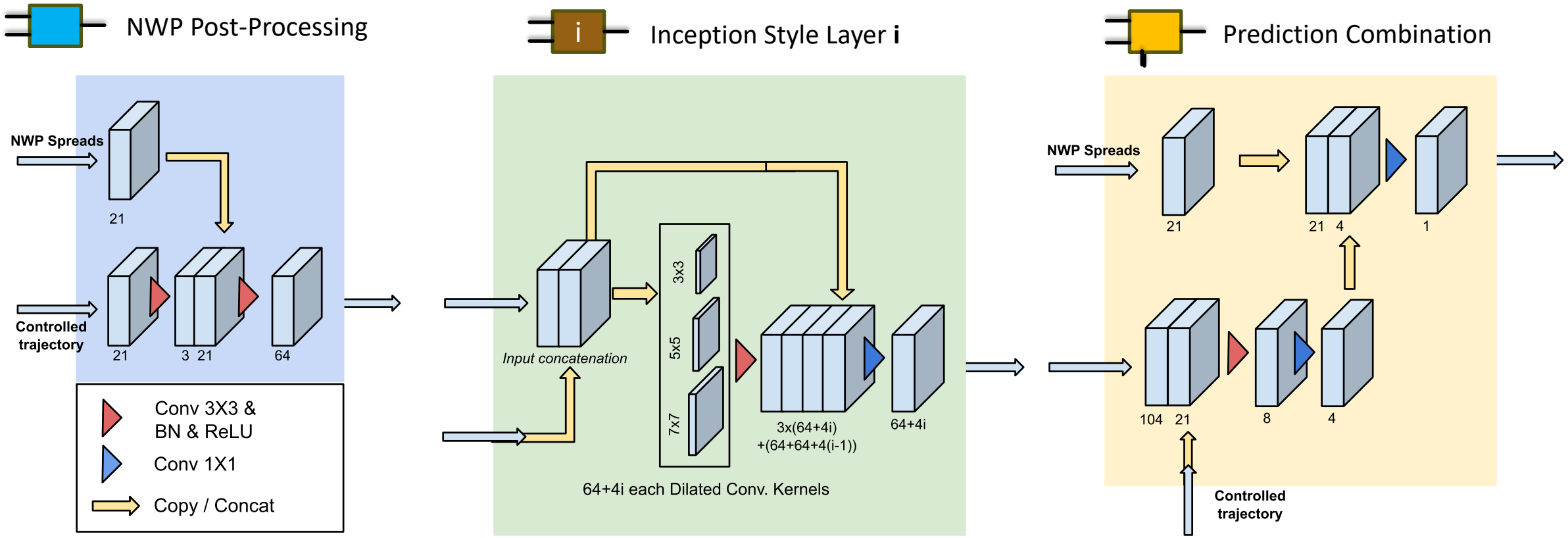
Learning Setup With 7 Variables



P. Gronquist et al.: Deep Learning for Post-Processing Ensemble Weather Forecasts, RSTA'19, arXiv

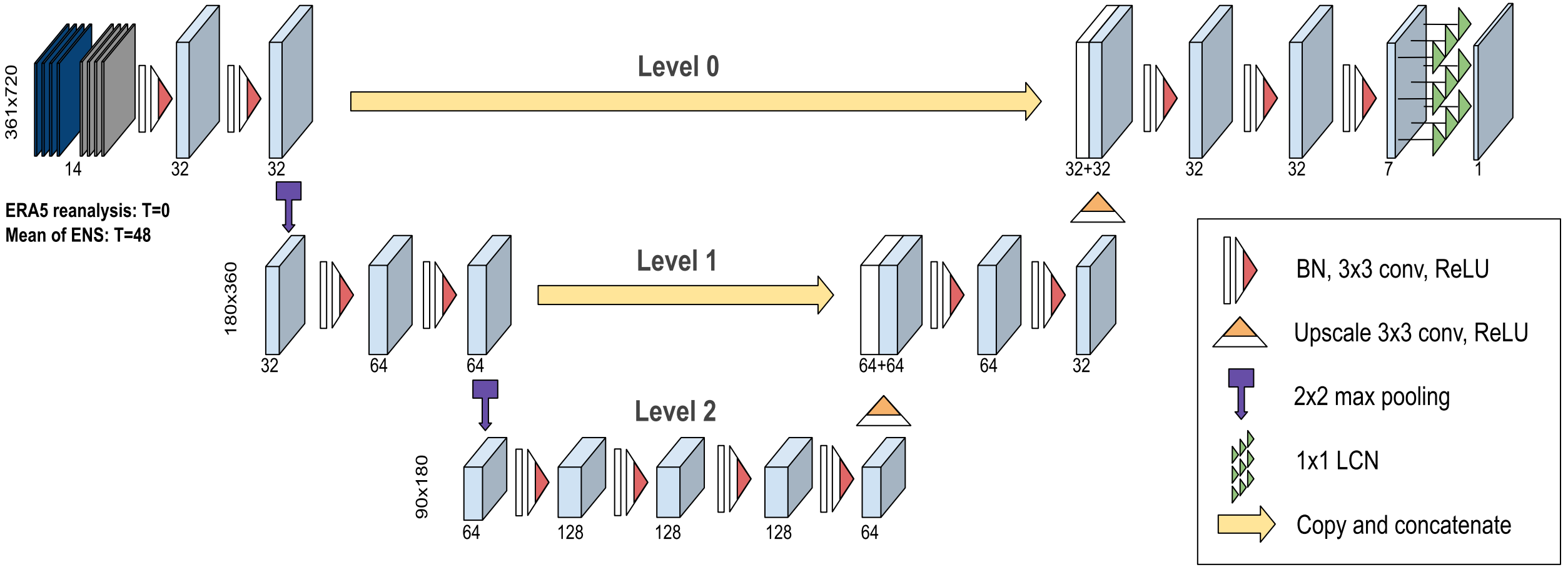
Uncertainty Quantification Network (based on ResNet)





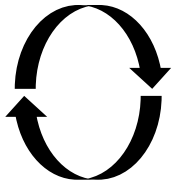
P. Gronquist et al.: Deep Learning for Post-Processing Ensemble Weather Forecasts, RSTA'19, arXiv

Bias Correction Network (based on 3D-Unet + LCN)



P. Gronquist et al.: Deep Learning for Post-Processing Ensemble Weather Forecasts, RSTA'19, arXiv

Training: Setup



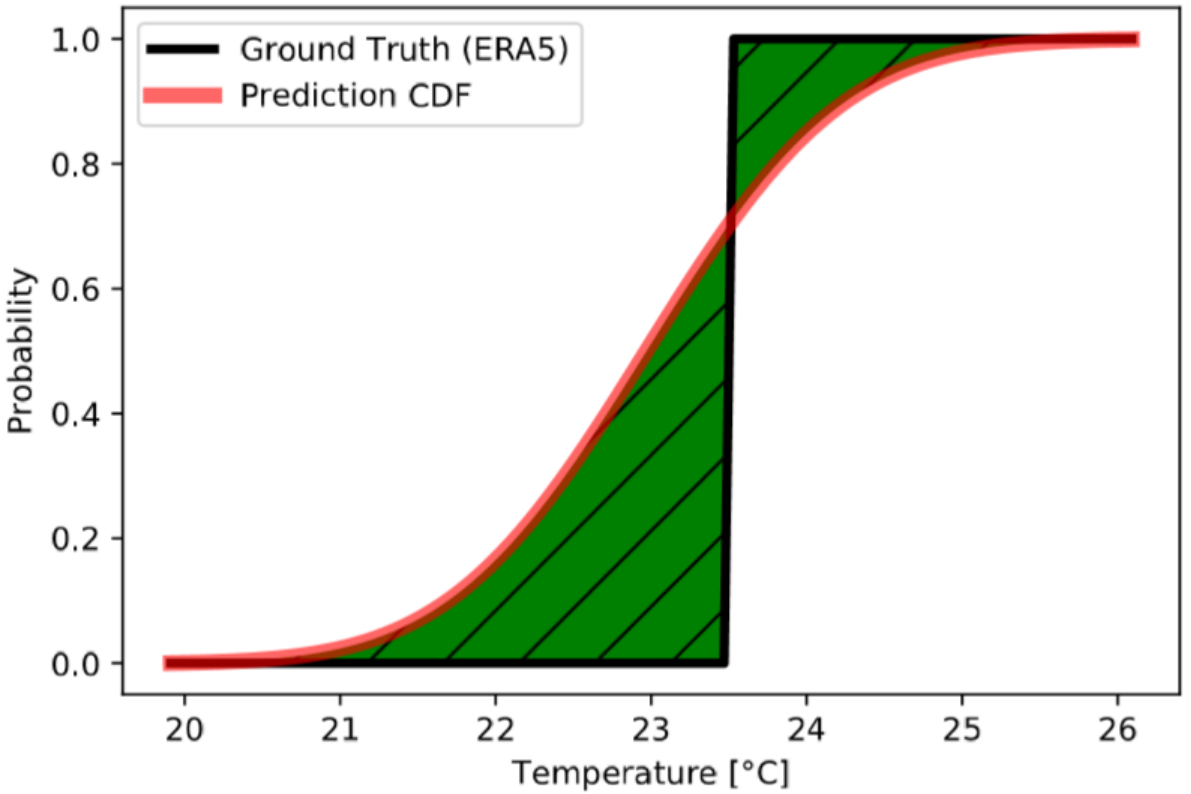
- **Framework: TensorFlow**
  - Default Adam optimizer
  - NVIDIA V100

    Four hours for training

    1/3rd second for inference
  - Batch size 2

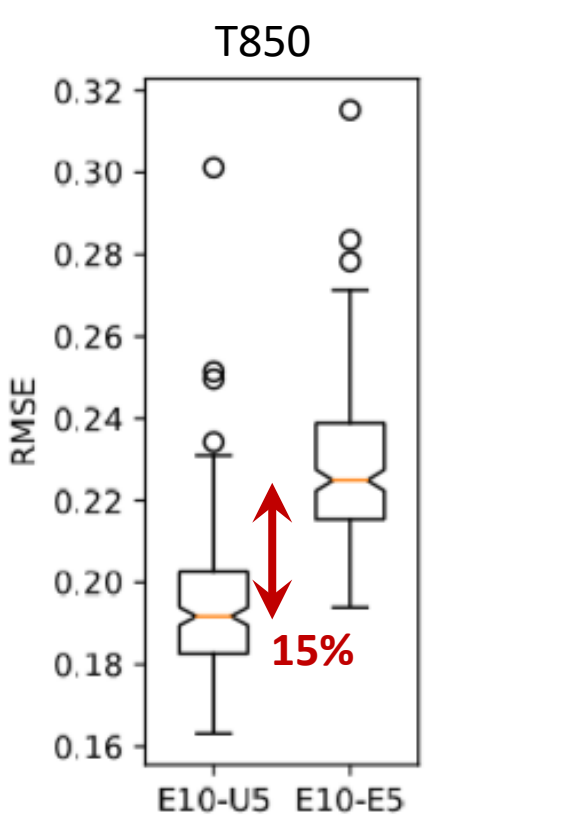
- **Training Loss: MSE**
  - Evaluation on RMSE

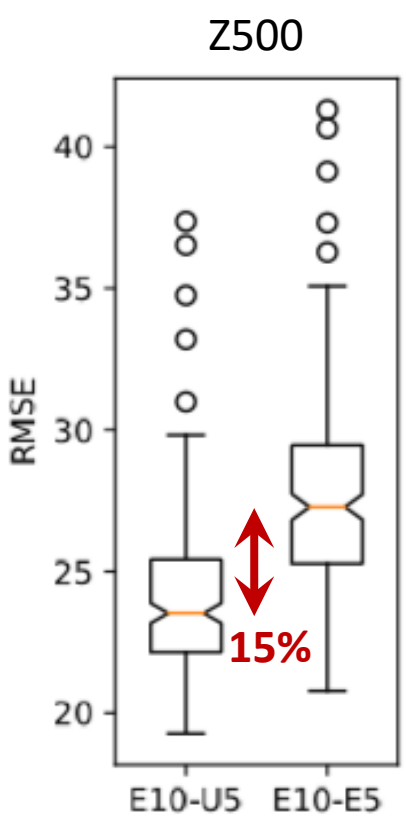
- **Combined training of both models**
  - Loss function $\mathrm{CRPS}(F, y) = \int_{-\infty}^{\infty} [F(x) - \mathbf{1}_{x>y}]^2 dx$

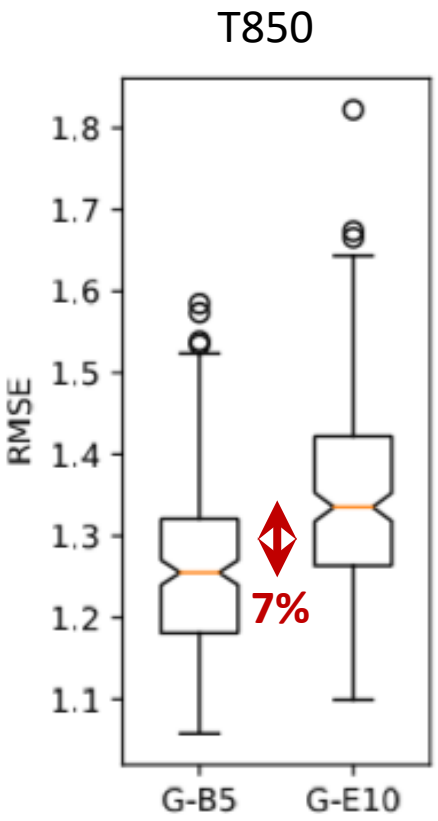


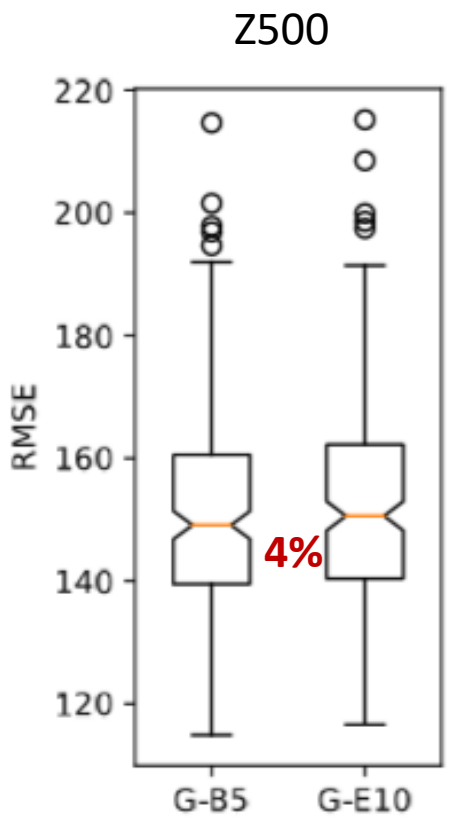
P. Gronquist et al.: Deep Learning for Post-Processing Ensemble Weather Forecasts, RSTA'19, arXiv

Global RMSE results









10 ensembles vs. UQ with 5 ensembles

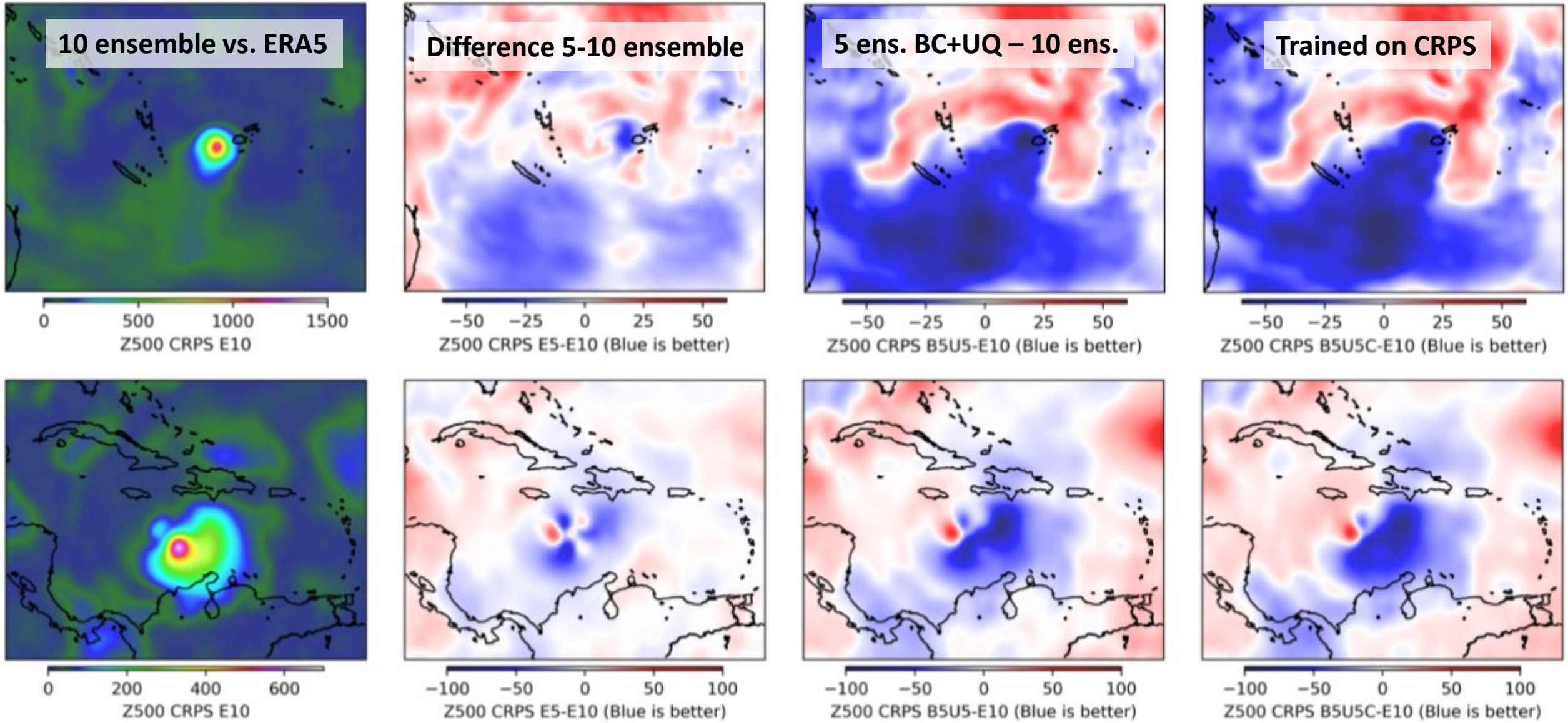
10 ensembles vs. 5 ensembles

ERA5 (ground truth) vs. BC with 5 trajectories

ERA5 (ground truth) vs. 10 trajectories

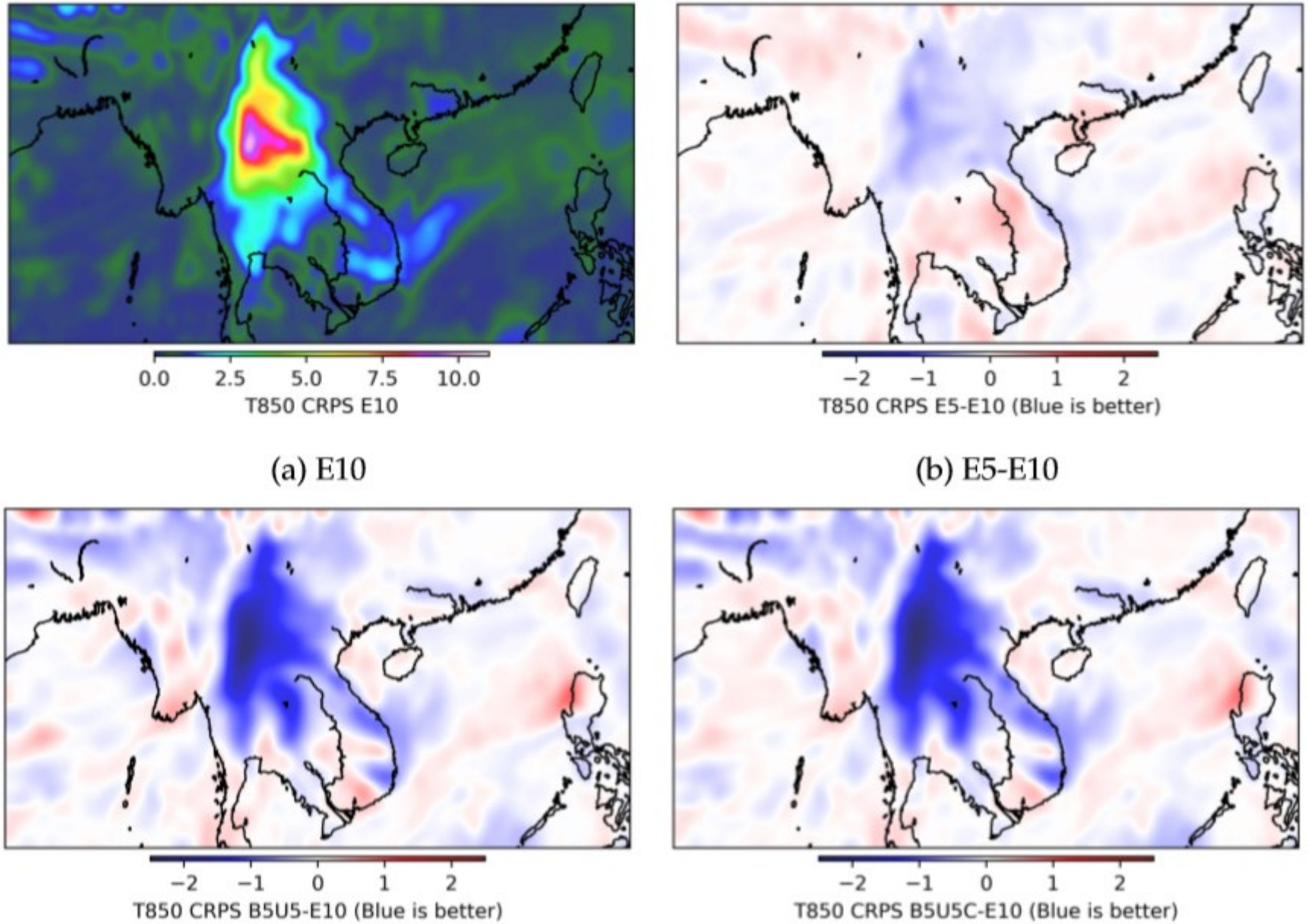
P. Gronquist et al.: Deep Learning for Post-Processing Ensemble Weather Forecasts, RSTA'19, arXiv

Extreme event: Tropical Cyclone Winston & Hurricane Matthews



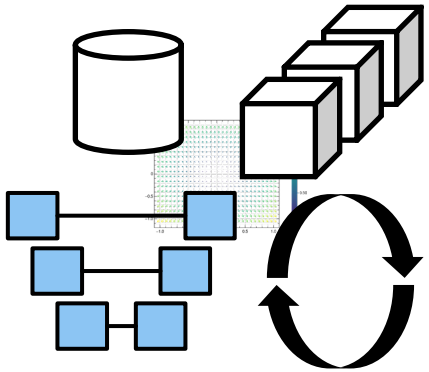
P. Gronquist et al.: Deep Learning for Post-Processing Ensemble Weather Forecasts, RSTA'19, arXiv

Cold wave over Asia



P. Gronquist et al.: Deep Learning for Post-Processing Ensemble Weather Forecasts, RSTA'19, arXiv

Intermediate Conclusions (Preliminary Study)

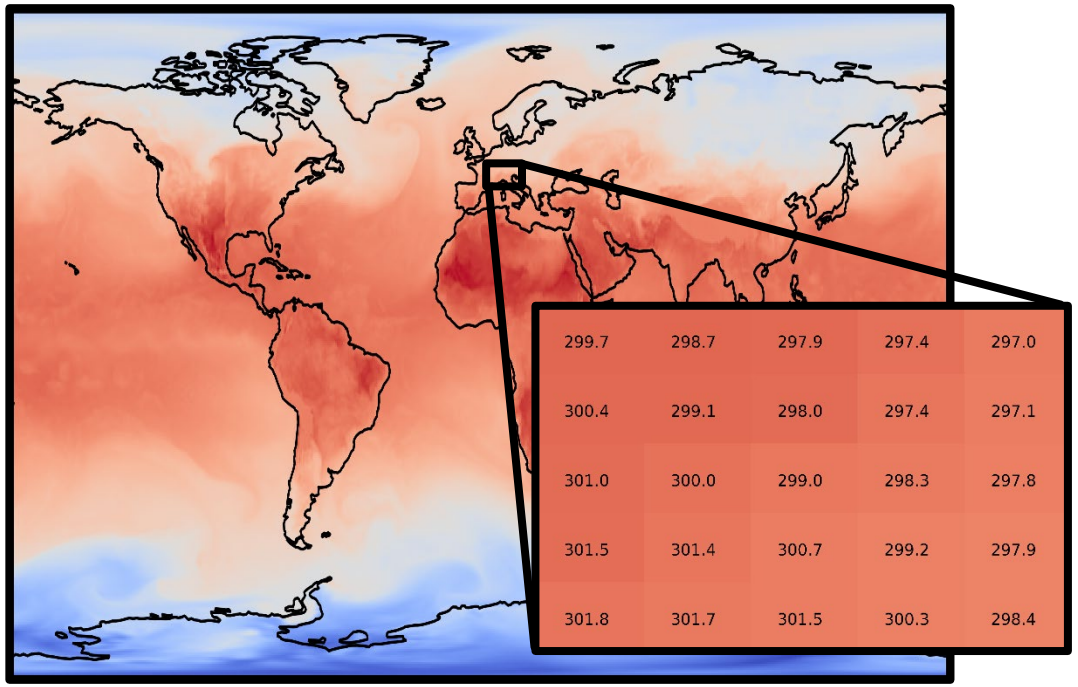


- **Simple Deep Learning can be used to accelerate forecast pipelines**
  - Take advantage of industry efforts to tune hardware and tool-chains
  - An informed approach is **necessary** to ensure improved results
- **Using Encoder-Decoder networks for predicting mean and StdDev in ensemble systems yields higher accuracy than using small ensemble statistics**
  - Fewer than half of the ensemble members are necessary
  - Accuracy improved with custom operators
- **Promising for increasing performance in large-scale settings**
  - Needs further investigation!
  - Join us/try yourself: https://github.com/spcl/deep-weather
- **Future directions:**
  - Larger datasets
  - Custom neural architectures for unstructured grids
  - Integrate into dace tool-chain for further optimization

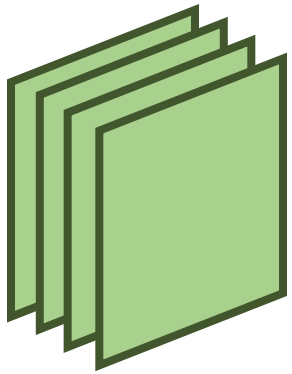
P. Gronquist et al.: Deep Learning for Post-Processing Ensemble Weather Forecasts, RSTA'19, arXiv

Compressing Weather and Climate Data into Neural Networks

Langwen Huang, Torsten Hoefler, ICLR'23 Top 5% (Oral)



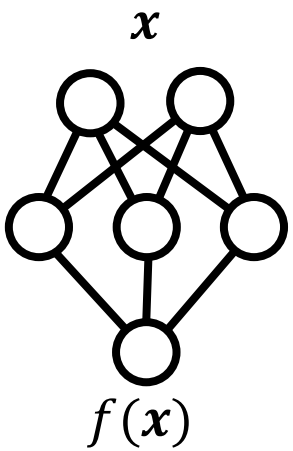
Multidimensional Data



Compress/Train

300 x – 3,000 x
15.6GB → 13.8MB

Neural Representation



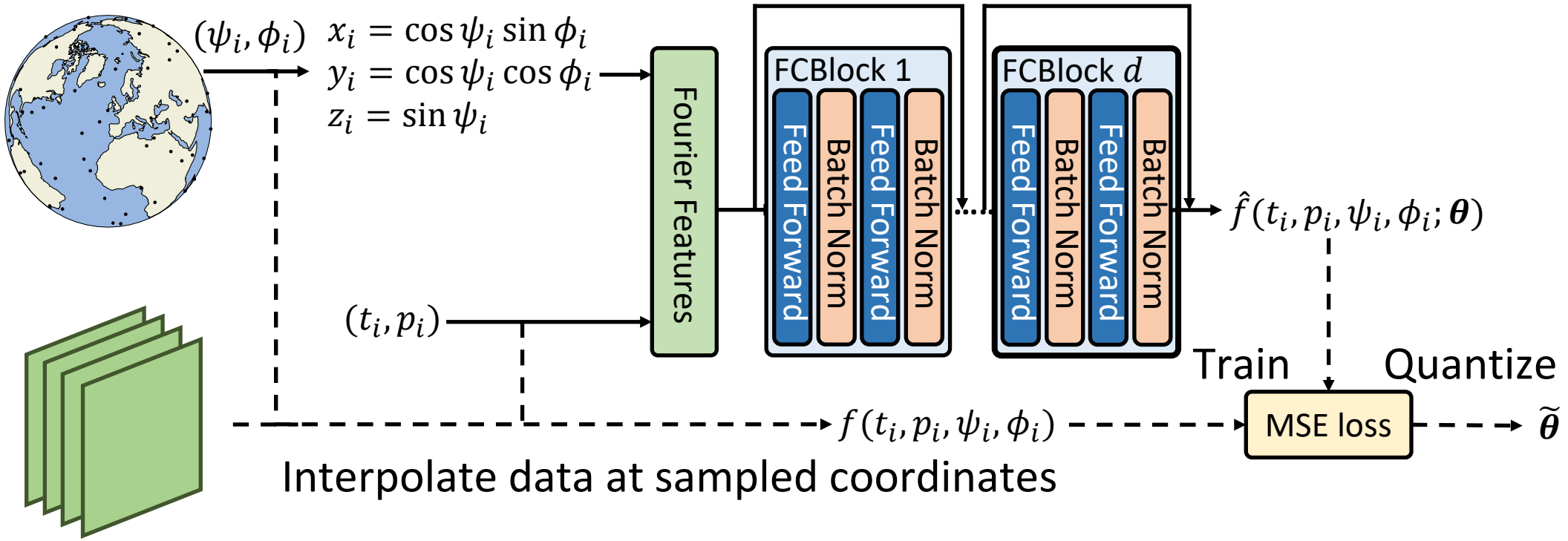
Properties of Weather Data

- Continuous and smooth
- Random/strided access is preferred
- Stratified
- Defined on a sphere

Method	Compression Ratio	Comp. Speed	Decompression Continuous Access	Decompression Random Access
ZFP [1]	< 10 x	▶▶▶	▶▶▶	▶▶▶
TTHRESH [2]	< 300 x	▶▶	▶▶▶	▶▶
SZ3 [3]	< 400 x	▶	▶▶▶	▶▶
NN (Ours)	**300 x – 3,000 x**	▶	▶▶▶	▶▶▶

Neural Network Structure

Compression / Training

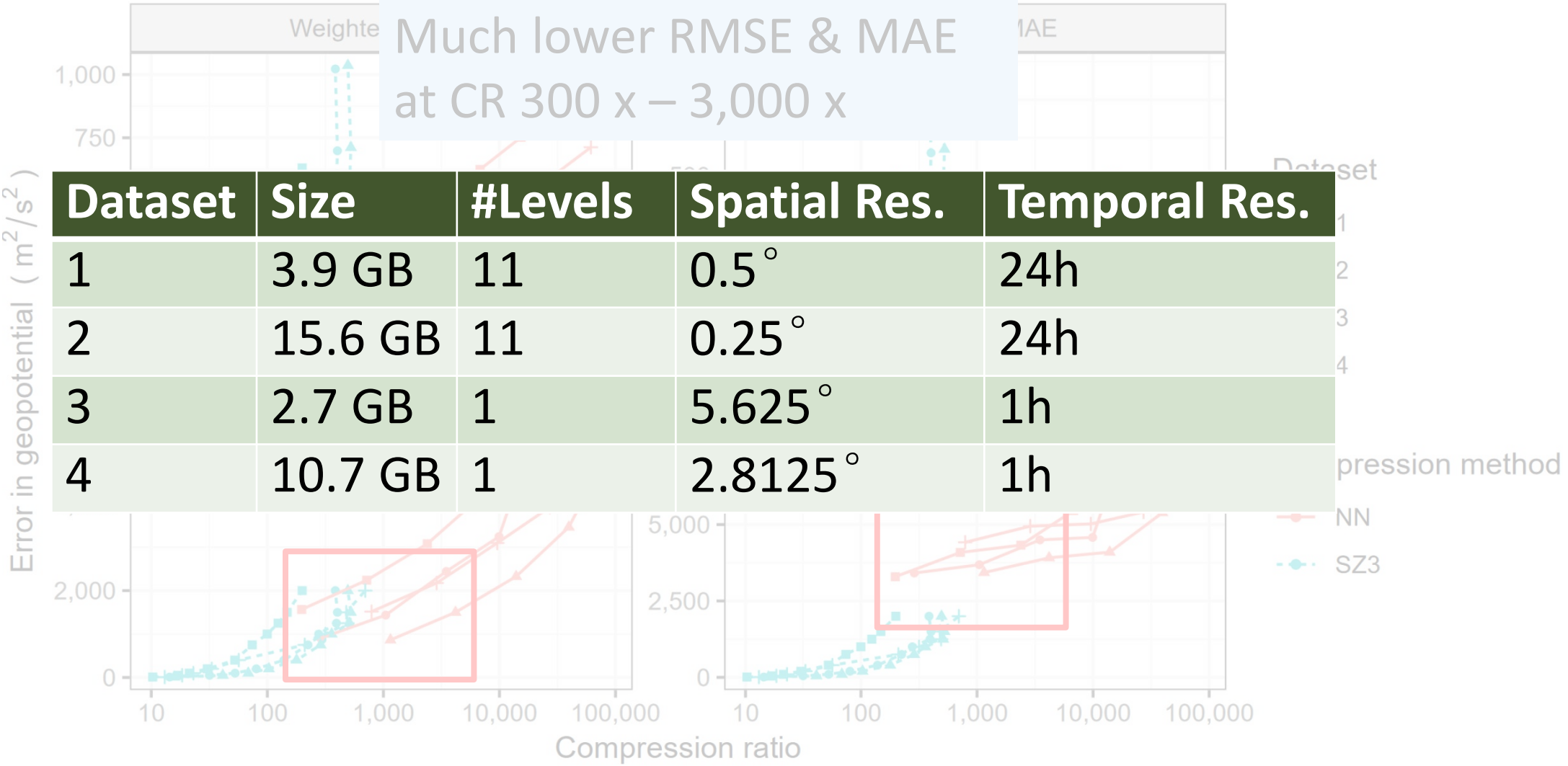


Huang, Hoefler: Compressing multidimensional weather and climate data into neural networks, ICLR'23, arXiv

Compression: Error Evaluation

Much lower RMSE & MAE
at CR 300 x – 3,000 x

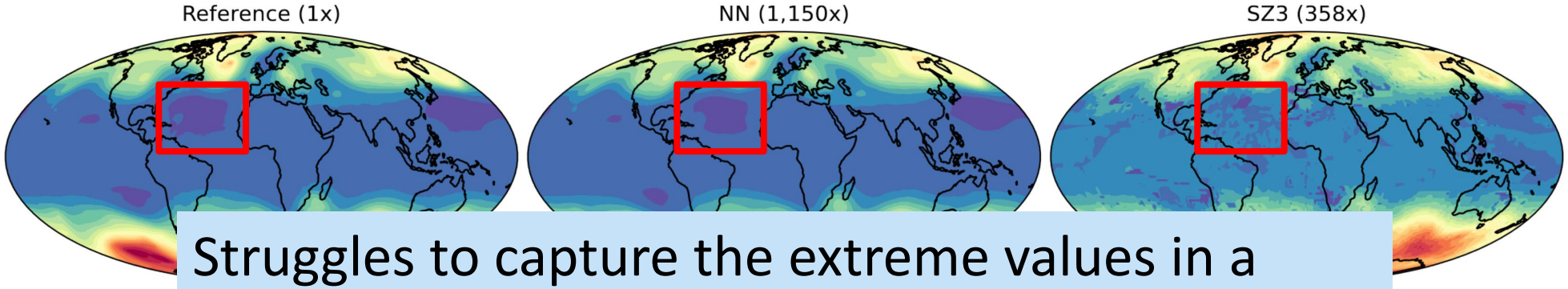
Dataset	Size	#Levels	Spatial Res.	Temporal Res.
1	3.9 GB	11	0.5°	24h
2	15.6 GB	11	0.25°	24h
3	2.7 GB	1	5.625°	1h
4	10.7 GB	1	2.8125°	1h



Huang, Hoefler: Compressing multidimensional weather and climate data into neural networks, ICLR'23, arXiv

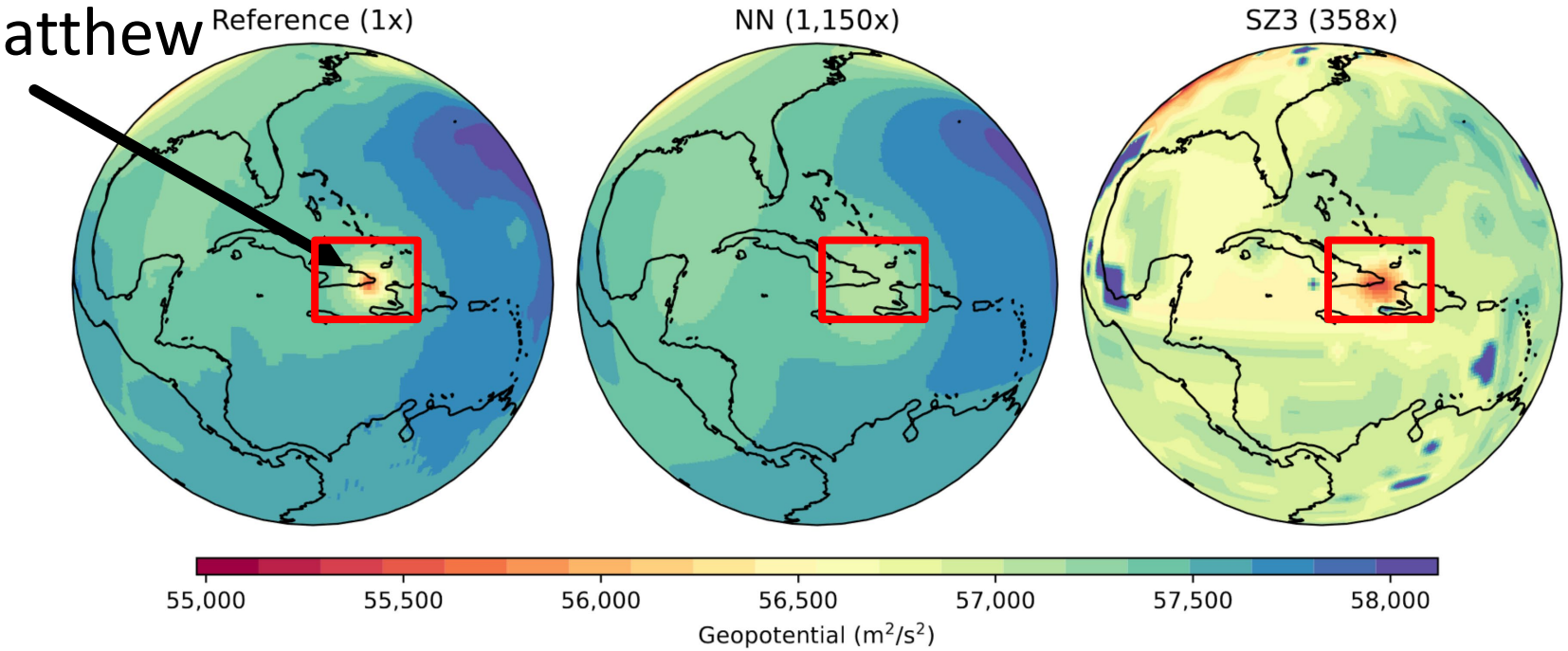
Compression: Case Study

Geopotential at 500hPa, 2016 Oct 5th



Struggles to capture the extreme values in a small area like a hurricane center

Hurricane Matthew

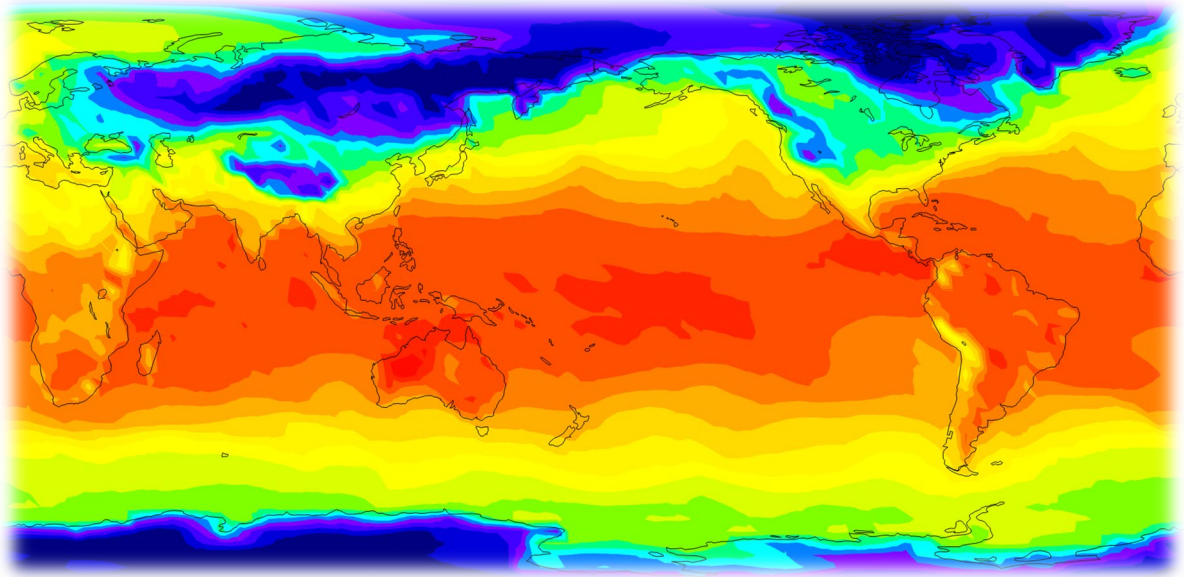


Huang, Hoefler: Compressing multidimensional weather and climate data into neural networks, ICLR'23, arXiv

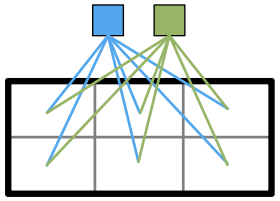
Bonus: Data Has Spatial Structure – Spatial Mixture of Experts

Weather and Climate grids have spatial structure!

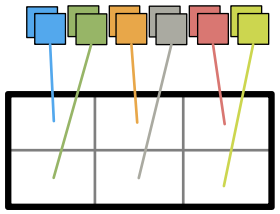
Locality matters!



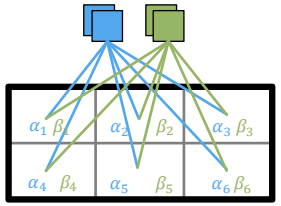
Convolution



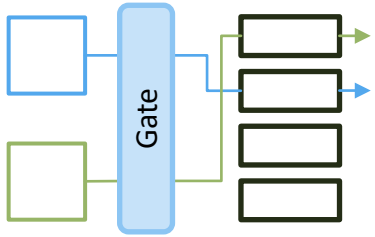
Locally-connected



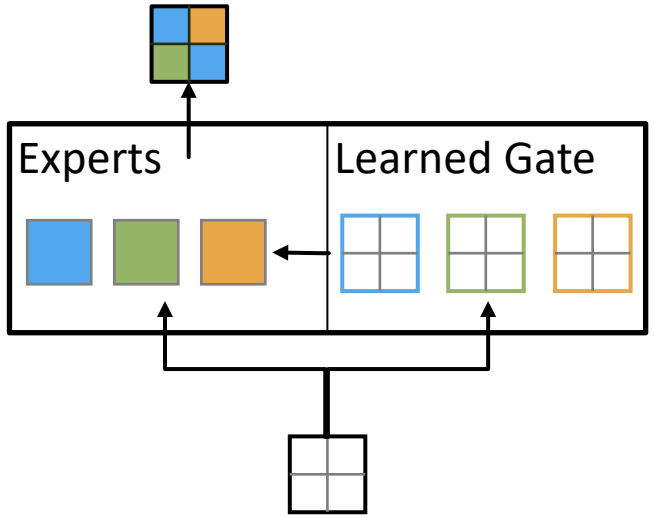
Low-rank locally-connected



Mixture-of-Experts



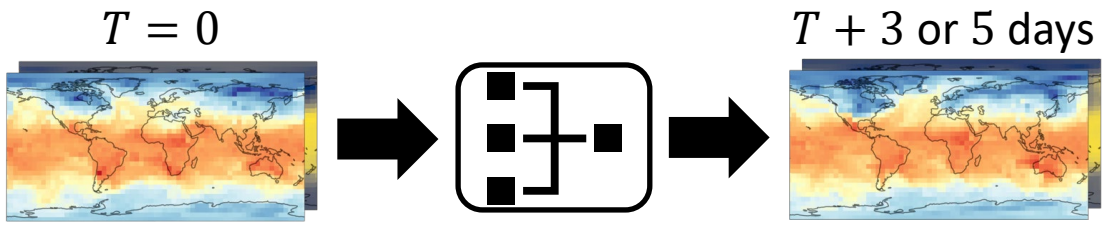
Spatial Mixture-of-Experts



Dryden, Hoefler: Spatial Mixture-of-Experts, NeurIPS'22, arXiv

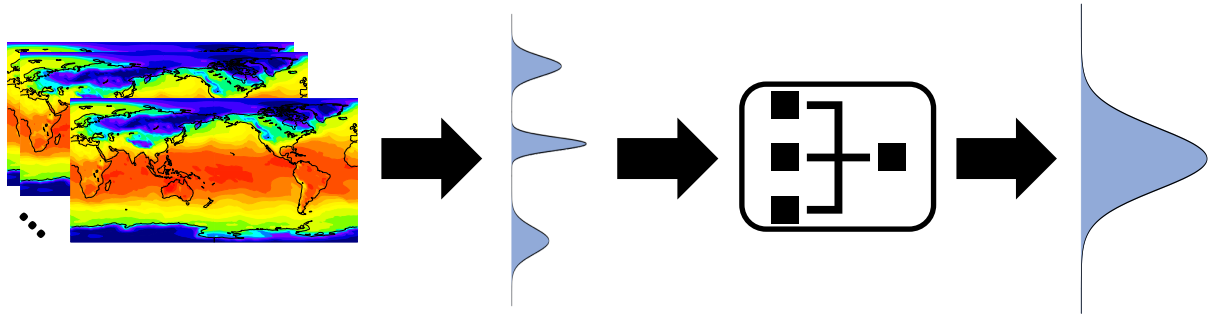
Spatial Mixture of Experts for Weather Prediction

Medium-range weather prediction WeatherBench



Model	Z500 [m^2s^{-2}]		T850 [K]	
	3 days	5 days	3 days	5 days
Rasp & Thuerey	$316^{\pm 2.4}$	$563^{\pm 3.1}$	$1.80^{\pm 0.02}$	$2.84^{\pm 0.03}$
➥2× wide	$310^{\pm 2.0}$	$555^{\pm 2.8}$	$1.76^{\pm 0.03}$	$2.78^{\pm 0.01}$
LRLCN	$290^{\pm 1.4}$	$549^{\pm 1.9}$	$1.73^{\pm 0.03}$	$2.79^{\pm 0.01}$
ViT	$438^{\pm 2.8}$	$638^{\pm 3.1}$	$2.24^{\pm 0.04}$	$2.88^{\pm 0.03}$
SMoE	**$270^{\pm 2.0}$**	**$525^{\pm 2.0}$**	**$1.66^{\pm 0.02}$**	**$2.60^{\pm 0.01}$**

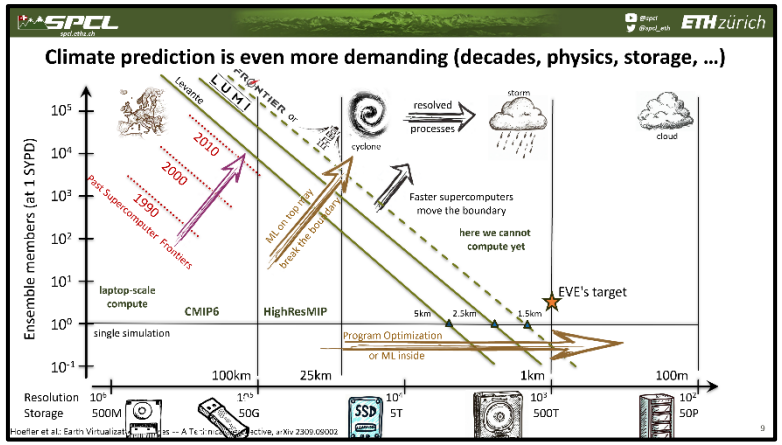
Ensemble post-processing ENS-10

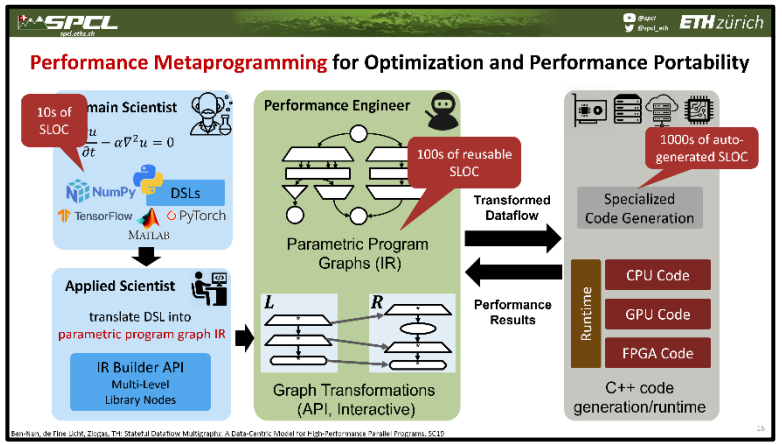


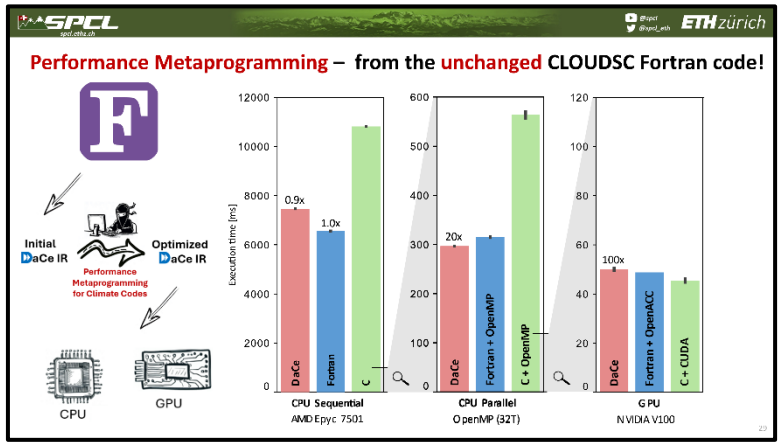
	Model	Z500 [m^2s^{-2}]		T850 [K]		T2M [K]	
		5 ens	10 ens	5 ens	10 ens	5 ens	10 ens
CRPS	EMOS	$79.12^{\pm 0.12}$	$78.80^{\pm 0.21}$	$0.721^{\pm 0.01}$	$0.706^{\pm 0.04}$	$0.720^{\pm 0.00}$	$0.711^{\pm 0.03}$
	U-Net	$76.54^{\pm 0.20}$	$76.18^{\pm 0.12}$	$0.685^{\pm 0.00}$	$0.670^{\pm 0.01}$	$0.657^{\pm 0.01}$	$0.644^{\pm 0.01}$
	SMoE	**$68.94^{\pm 0.14}$**	**$67.43^{\pm 0.12}$**	**$0.612^{\pm 0.01}$**	**$0.590^{\pm 0.02}$**	**$0.601^{\pm 0.02}$**	**$0.594^{\pm 0.02}$**
EECRPS	EMOS	$29.21^{\pm 0.18}$	$29.02^{\pm 0.13}$	$0.247^{\pm 0.00}$	$0.245^{\pm 0.02}$	$0.244^{\pm 0.00}$	$0.241^{\pm 0.02}$
	U-Net	$27.78^{\pm 0.11}$	$27.55^{\pm 0.19}$	$0.230^{\pm 0.01}$	$0.229^{\pm 0.01}$	$0.225^{\pm 0.00}$	$0.220^{\pm 0.01}$
	SMoE	**$23.79^{\pm 0.20}$**	**$23.10^{\pm 0.16}$**	**$0.207^{\pm 0.03}$**	**$0.197^{\pm 0.03}$**	**$0.199^{\pm 0.01}$**	**$0.190^{\pm 0.02}$**

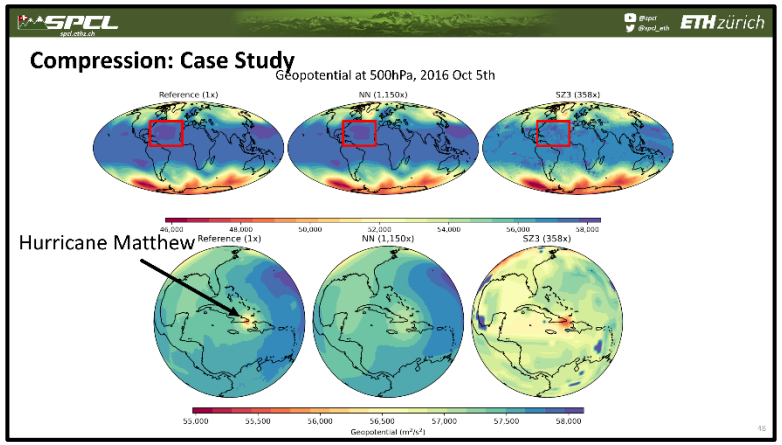
Dryden, Hoefler: Spatial Mixture-of-Experts, NeurIPS'22, arXiv

Summary









More of SPCL's research:

youtube.com/@spcl — 150+ Talks

twitter.com/spcl_eth — 1.2K+ Followers

github.com/spcl — 2K+ Stars

… or spcl.ethz.ch

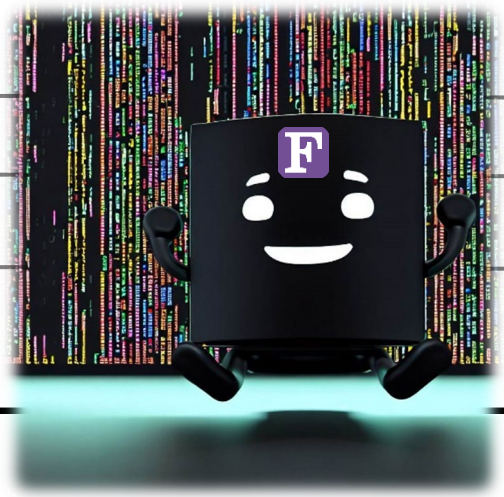
Join us! We're looking for PhD students, postdocs, and academic visitors in Zurich!

http://spcl.inf.ethz.ch/Jobs/

A first simple loop from CLOUDSC*

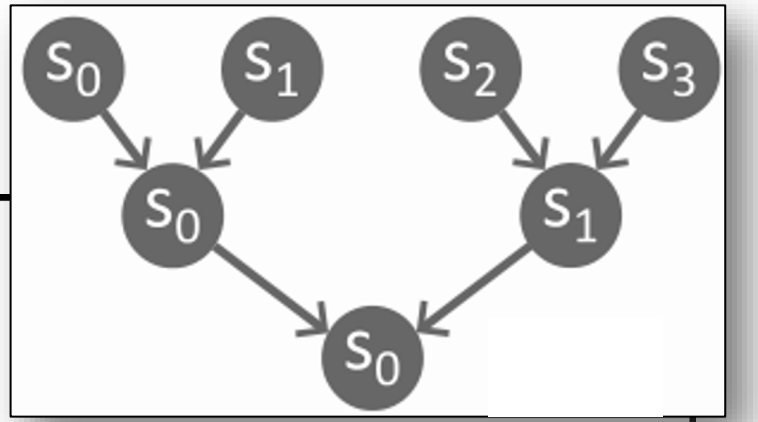
Data Parallelism	✓ ..... do JK=1,KLEV ✓ ..... do JL=1,KFDIA ZQSM(JL,JK)=ZQSM(JL,JK)/(1.0-RE*ZQSM(JL,JK)) enddo enddo
Work	KLEV * KFDIA
Depth	**1**
Average Parallelism	KLEV * KFDIA

Fully data parallel



* examples are simplified for presentation purposes

A second more complex loop from CLOUDSC



Data Parallelism	✗ ..... do JN=1, NSTEP-1 ✓ ..... do JL=1, KFDIA ZQXN(JL, NSTEP) = ZQXN(JL, NSTEP)+ZQXN(JL, JN) enddo enddo	(array) accumulation prevents parallelization ☹
Work	(NSTEP-1) * KFDIA	(NSTEP-1) * KFDIA
Depth	(NSTEP-1) * KFDIA	$\log_2$(NSTEP-1)
Average Parallelism	1	(NSTEP-1) * KFDIA / $\log_2$(NSTEP-1)

Now multiple realistic CLOUDSC loops

Data Parallelism

L1

```
X  do JM=1,4
X  do JK=1,KLEV
X  do JL=1,KFDIA
1    if ZQX(JL,JK,JM)<RLMIN) then
2      ZQADJ=ZQX(JL,JK,JM)*ZQTMST
3      tend_q(JL,JK)=tend_q(JL,JK)+ZQADJ
4      tend_T(JL,JK)=tend_T(JL,JK)-RAL*ZQADJ
5      ZQX(JL,JK,JM)=0.0
```

reuse of temporary variable prevents parallelization

L2

```
✓  do JK=1,KLEV
✓  do JL=1,KFDIA
6    ZQSM(JL,JK)=ZQSM(JL,JK)/(1.0-RE*ZQSM(JL,
```

L3

```
✓  do JK=1,KLEV
✓  do JL=1,KFDIA
7    ZA(JL,JK)=MAX(0.0,MIN(1.0,ZA(JL,JK)))
8    ZLI(JL,JK)=ZQX(JL,JK,1)+ZQX(JL,JK,2)
9    if (ZLI(JL,JK)>RLMIN) then
10     ZLFRAC(JL,JK)=ZQX(JL,JK,1)/ZLI(JL,JK)
     else
11     ZLFRAC(JL,JK)=0.0
```

Order Constraints

Happens-before

- 1 → 2 Control
- 2 → 3 RAW
- 2 → 4 RAW
- 2 → 5 WAR

No order constraint: L2 || L1

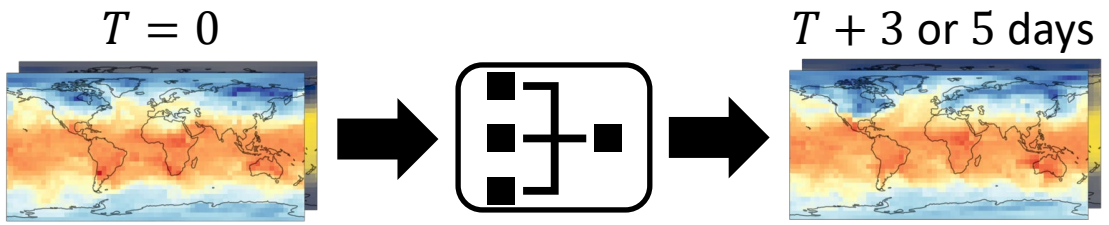
- 5 → 8 RAW
- 8 → 9 RAW
- 9 → 10 Control
- 9 → 11 Control

L3 → L1

	L1	L2	L3	L1	L2	L3
Work	4 * KLEV * KFDIA * (1+4)	KLEV * KFDIA	KLEV * KFDIA * 4	KLEV * KFDIA * 25		
Depth	log2(4) * 1 *1 * (1+2)	1	1 * 1 * (2+1)	8		
Average Parallelism	KLEV * KFDIA * 10/3	KLEV * KFDIA	KLEV * KFDIA * 4/3	KLEV * KFDIA * 25/8		

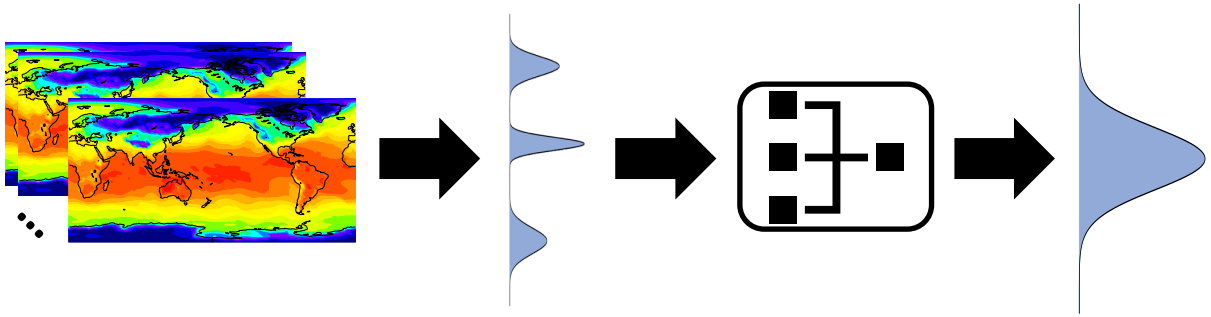
Weather

Medium-range weather prediction WeatherBench



Model	Z500 [m^2s^{-2}]		T850 [K]	
	3 days	5 days	3 days	5 days
Rasp & Thuerey	$316^{\pm 2.4}$	$563^{\pm 3.1}$	$1.80^{\pm 0.02}$	$2.84^{\pm 0.03}$
➥2× wide	$310^{\pm 2.0}$	$555^{\pm 2.8}$	$1.76^{\pm 0.03}$	$2.78^{\pm 0.01}$
LRLCN	$290^{\pm 1.4}$	$549^{\pm 1.9}$	$1.73^{\pm 0.03}$	$2.79^{\pm 0.01}$
ViT	$438^{\pm 2.8}$	$638^{\pm 3.1}$	$2.24^{\pm 0.04}$	$2.88^{\pm 0.03}$
SMoE	**$270^{\pm 2.0}$**	**$525^{\pm 2.0}$**	**$1.66^{\pm 0.02}$**	**$2.60^{\pm 0.01}$**

Ensemble post-processing ENS-10



	Model	Z500 [m^2s^{-2}]		T850 [K]		T2M [K]	
		5 ens	10 ens	5 ens	10 ens	5 ens	10 ens
CRPS	EMOS	$79.12^{\pm 0.12}$	$78.80^{\pm 0.21}$	$0.721^{\pm 0.01}$	$0.706^{\pm 0.04}$	$0.720^{\pm 0.00}$	$0.711^{\pm 0.03}$
	U-Net	$76.54^{\pm 0.20}$	$76.18^{\pm 0.12}$	$0.685^{\pm 0.00}$	$0.670^{\pm 0.01}$	$0.657^{\pm 0.01}$	$0.644^{\pm 0.01}$
	SMoE	**$68.94^{\pm 0.14}$**	**$67.43^{\pm 0.12}$**	**$0.612^{\pm 0.01}$**	**$0.590^{\pm 0.02}$**	**$0.601^{\pm 0.02}$**	**$0.594^{\pm 0.02}$**
EECRPS	EMOS	$29.21^{\pm 0.18}$	$29.02^{\pm 0.13}$	$0.247^{\pm 0.00}$	$0.245^{\pm 0.02}$	$0.244^{\pm 0.00}$	$0.241^{\pm 0.02}$
	U-Net	$27.78^{\pm 0.11}$	$27.55^{\pm 0.19}$	$0.230^{\pm 0.01}$	$0.229^{\pm 0.01}$	$0.225^{\pm 0.00}$	$0.220^{\pm 0.01}$
	SMoE	**$23.79^{\pm 0.20}$**	**$23.10^{\pm 0.16}$**	**$0.207^{\pm 0.03}$**	**$0.197^{\pm 0.03}$**	**$0.199^{\pm 0.01}$**	**$0.190^{\pm 0.02}$**

[Rasp et al., 2020; Rasp & Thuerey, 2021; Ashkboos et al., 2022; Elsayed et al., 2020; Dosovitskiy et al., 2021]