Emulating Atmospheric Transport to estimate GHG emissions using machine learning model

- Nikhil Dadheech, Tai-long He, Alexander J. Turner

Nikhil Dadheech
Third Year PhD student
Advisor: Dr. Alexander Turner
University of Washington
nd349@uw.edu

UNIVERSITY *of* WASHINGTON

This work is funded by
NASA FINESST Award #80NSSC22K1557,
NASA ECF Grant #80NSSC21K1808
Environmental Defense Fund
Integral Environmental Big Data Fund (UW)

How do we compute emissions?

How do we compute emissions?

Bottom Up

How do we compute emissions?

Bottom Up

Average emission -> x

How do we compute emissions?

Bottom Up

Average emission -> x

Total units-> N

How do we compute emissions?

Bottom Up

Average emission -> x

Total units-> N

Emission budget from industries: ~Nx

How do we compute emissions?

Bottom Up

Average emission -> x

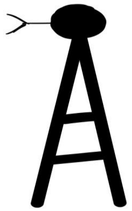
Total units-> N

Emission budget from industries: ~Nx

Top Down

Measurement (**y**)

Emission (**x**)



How do we compute emissions?

Bottom Up

Average emission -> x

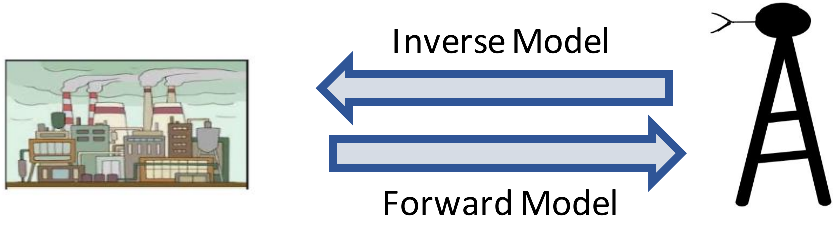
Total units-> N

Emission budget from industries: ~Nx

Top Down

Measurement (**y**)

Emission (**x**)



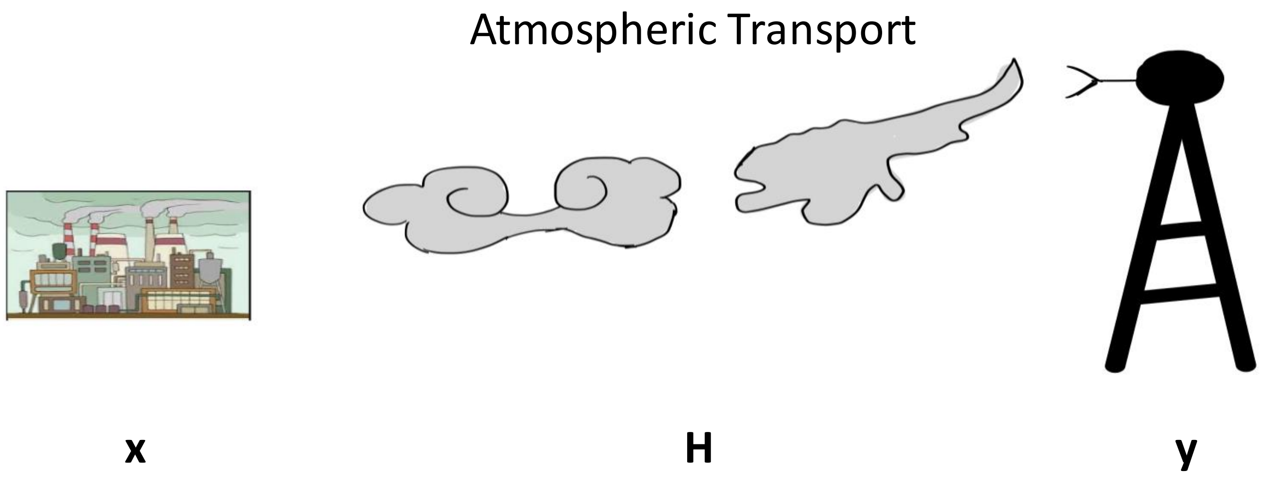
Trace back the emissions based on given measurements

Estimating emissions using top-down approach

Emissions

Observations $\Leftarrow$ $\mathbf{y = Hx + \in}$ $\Rightarrow$ Error

H is the relationship between observations and emissions



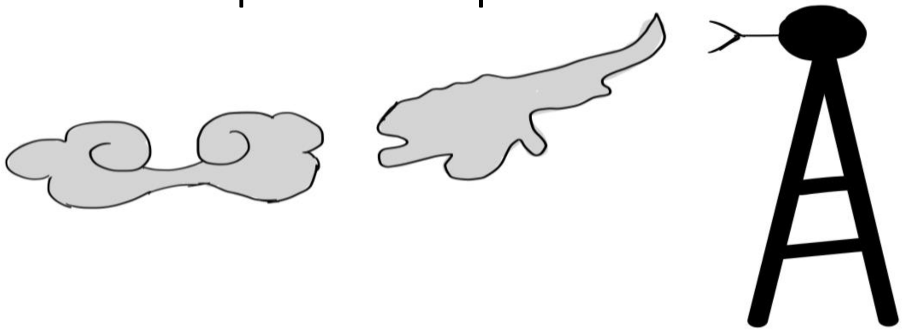
Estimating emissions using top-down approach

Emissions

Observations $\Longleftarrow$ $\mathbf{y = Hx + \in}$ $\Longrightarrow$ Error

H is the relationship between observations and emissions

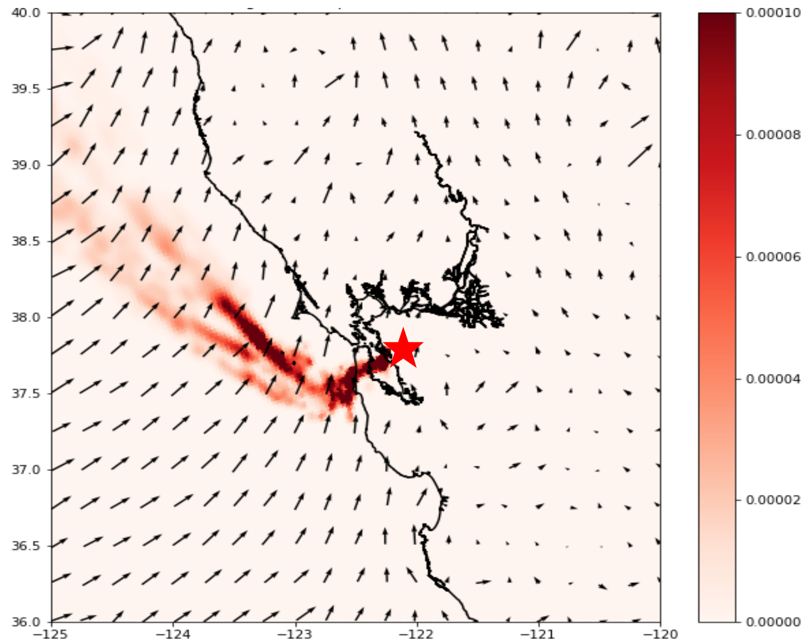
Atmospheric Transport



x

H

y



Estimating emissions using top-down approach

Emissions

Observations $\Leftarrow$ $\mathbf{y} = \mathbf{Hx} + \epsilon$ $\Rightarrow$ Error

H is the relationship between observations and emissions

Cost Function $$J(\mathbf{x}) = \frac{1}{2}(\mathbf{y}-\mathbf{Hx})^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{Hx}) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_a)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}_a)$$

Posterior Solution $$\hat{x} = \mathbf{x}_a + (\mathbf{HB})^T(\mathbf{HBH}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{Hx}_a)$$

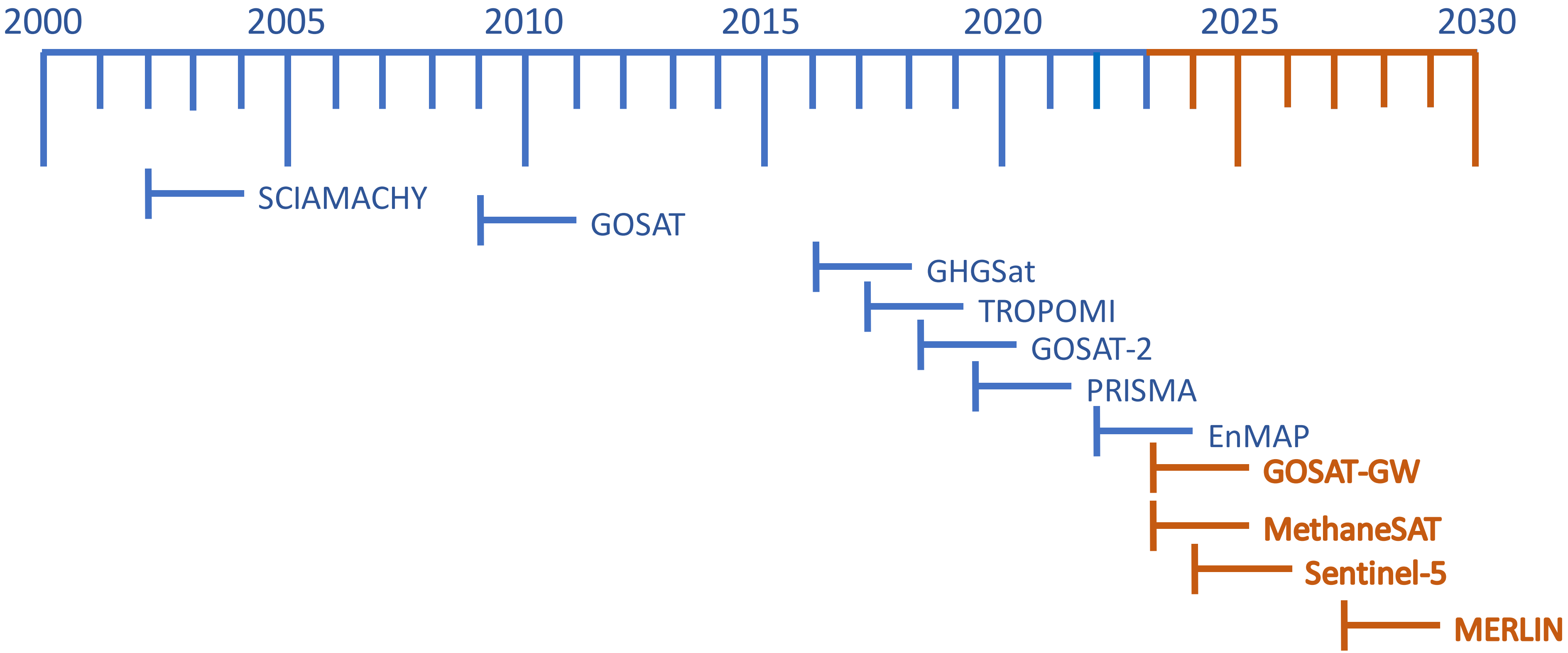
$\mathbf{x}_a$: Prior estimate

R: Observational covariance matrix

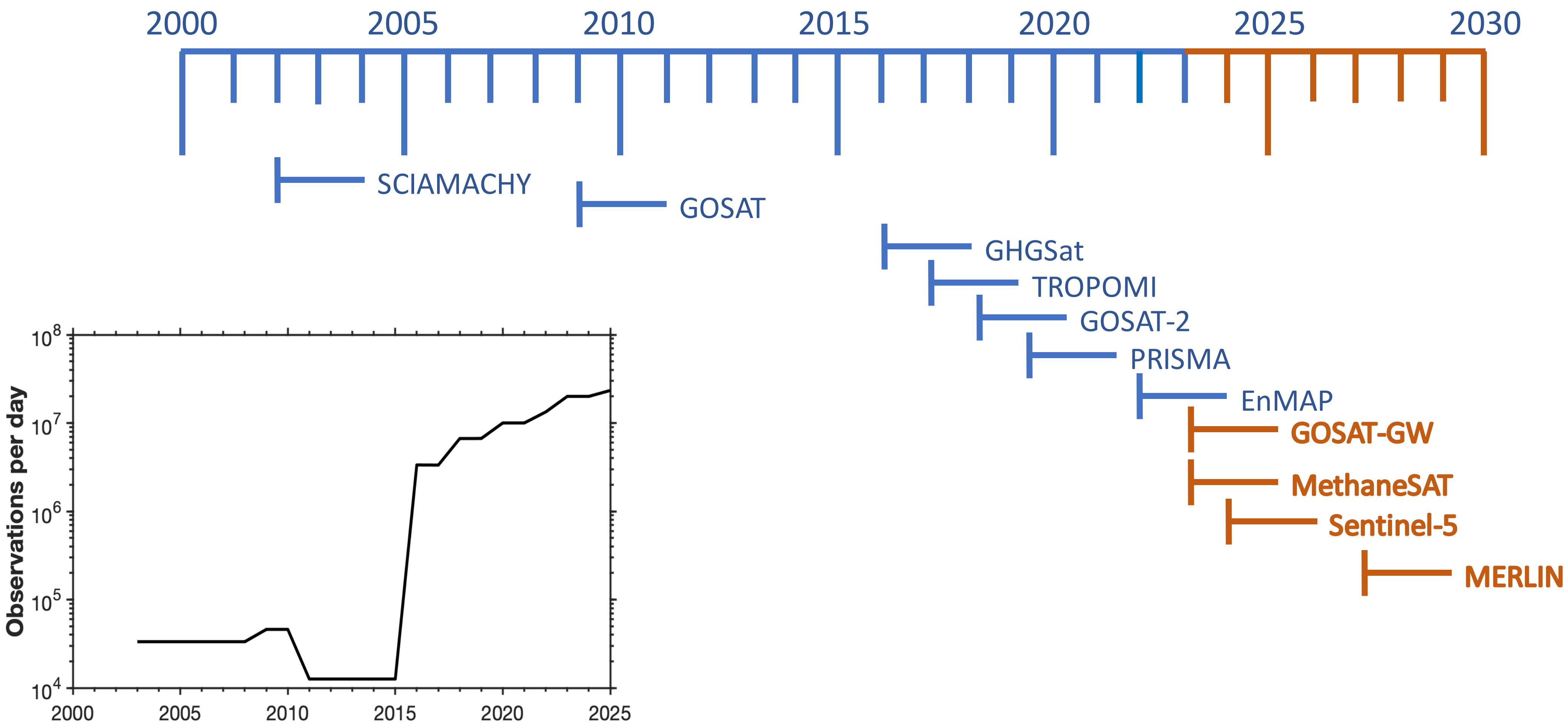
B: Prior covariance matrix

High-resolution data is required to study point sources and methane plumes

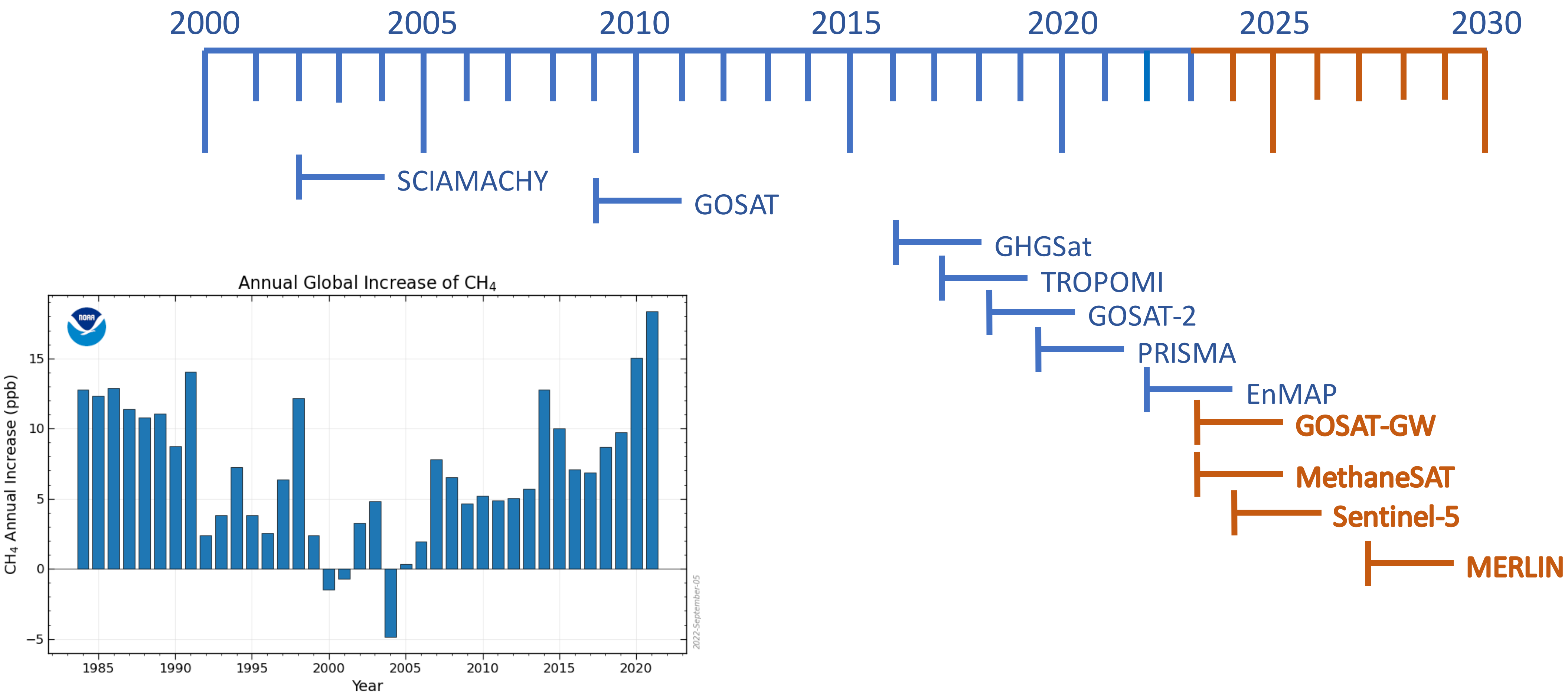
Greenhouse Gas Observing Systems



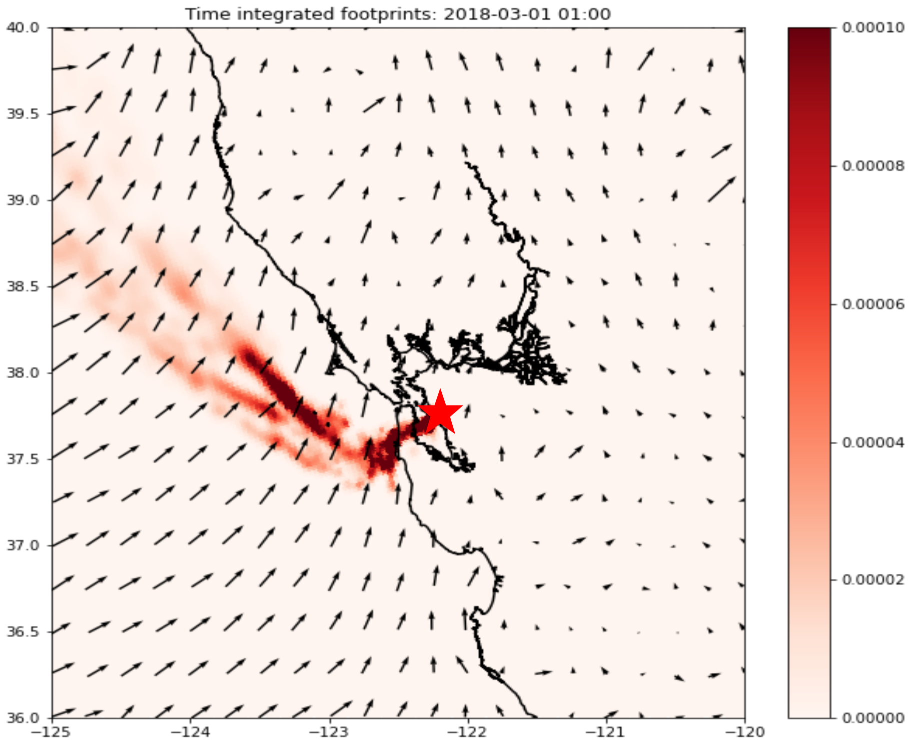
Greenhouse Gas Observing Systems



Greenhouse Gas Observing Systems



Computational and storage complexities with footprints



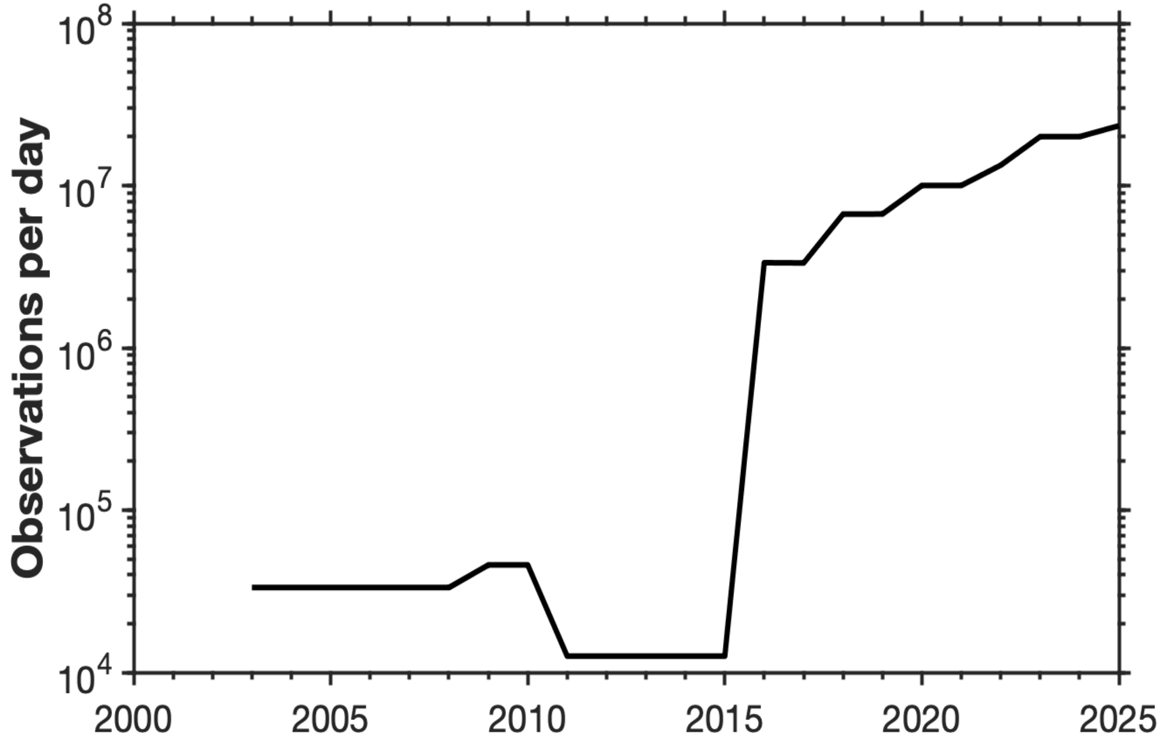
Footprint in Bay Area (generated with WRF-STILT)

$$y = \mathbf{H}x + \epsilon$$

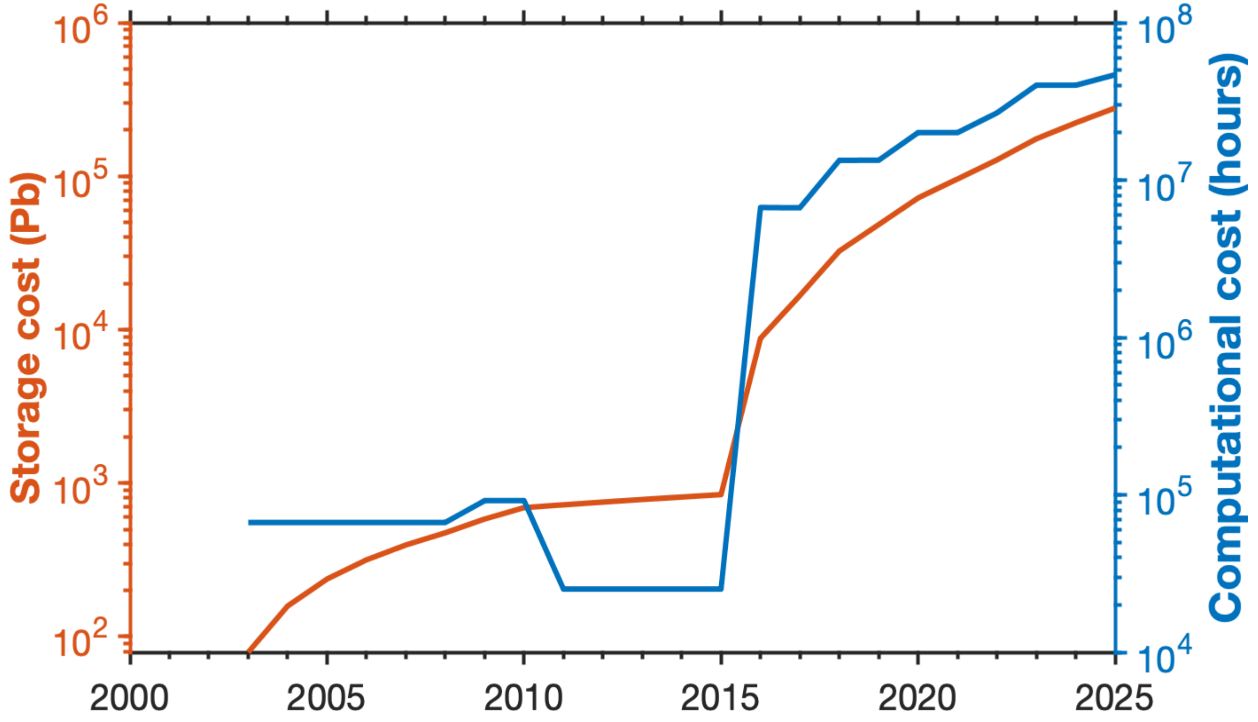
- For every measurement, we need a footprint
- These footprints look spatially similar

Atmospheric transport models become computationally expensive as the number of measurements increases

Computational and storage complexities with footprints

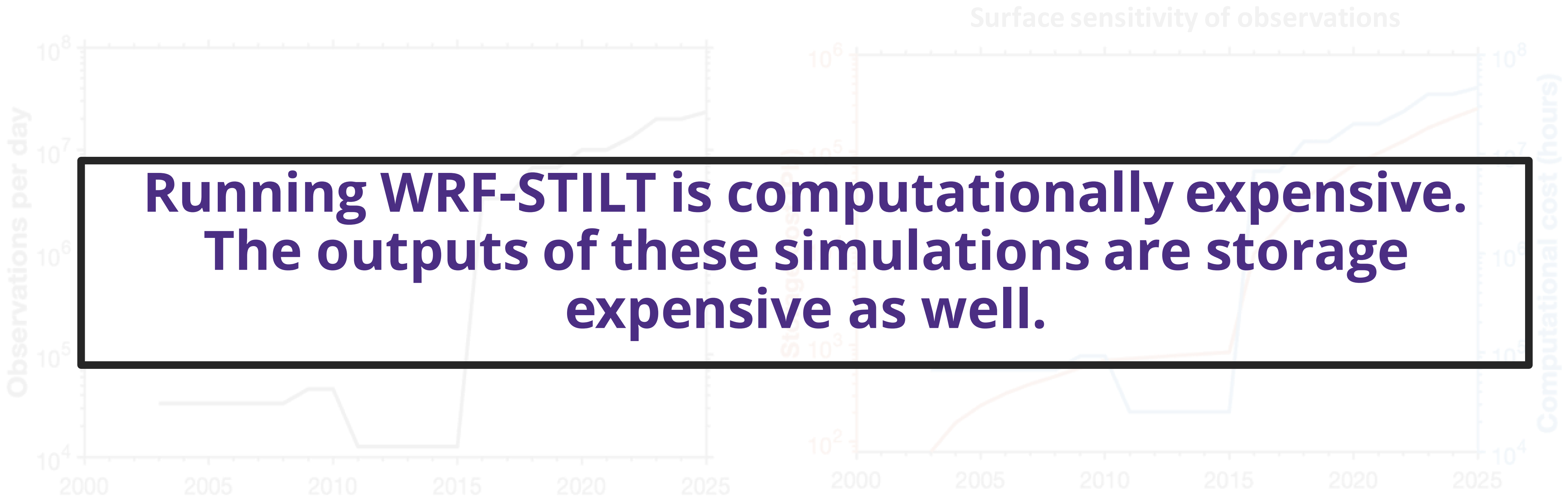


Surface sensitivity of observations



Assuming a computational time of 2 hours for each STILT simulation and 6.5 MB of storage space

Computational and storage complexities with footprints



Running WRF-STILT is computationally expensive. The outputs of these simulations are storage expensive as well.

Goal of this work

> **Developing an efficient method to compute source-receptor relationship using machine learning emulator (FootNet)**

> **Estimating GHG emission fluxes by emulating atmospheric transport using FootNet**

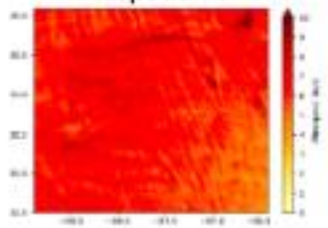
Can machine learning help?

Tai-long He, postdoc at Turner's group

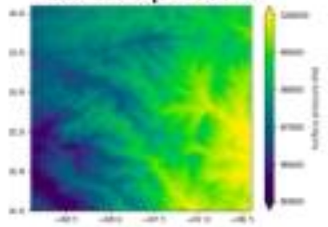
Can machine learning help?

Inputs

Windspeed at 10m

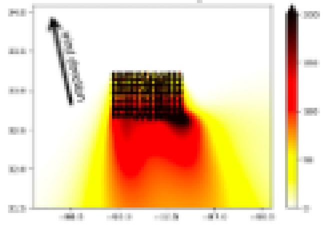


Surface pressure

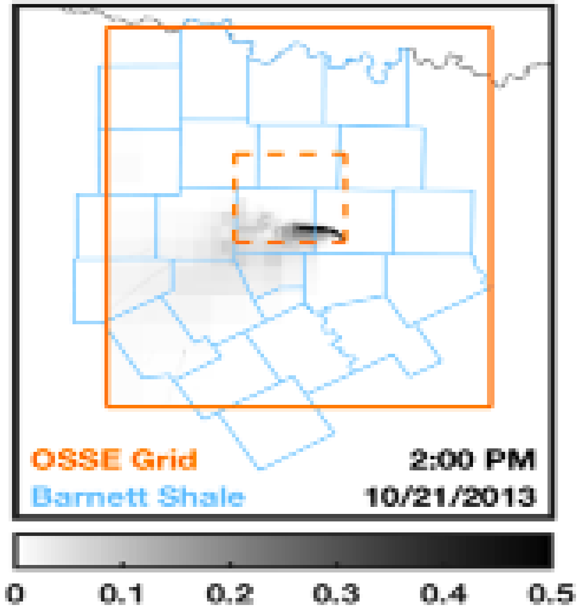


⋮

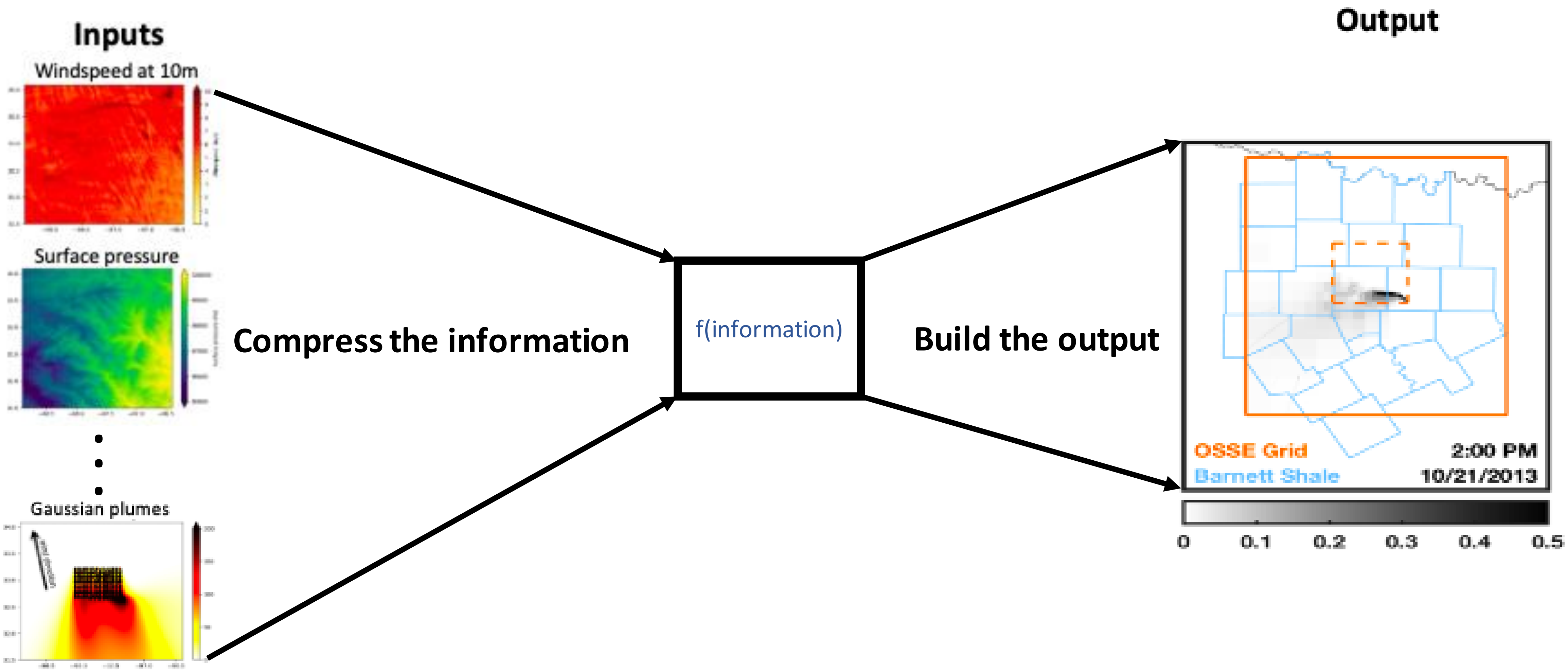
Gaussian plumes



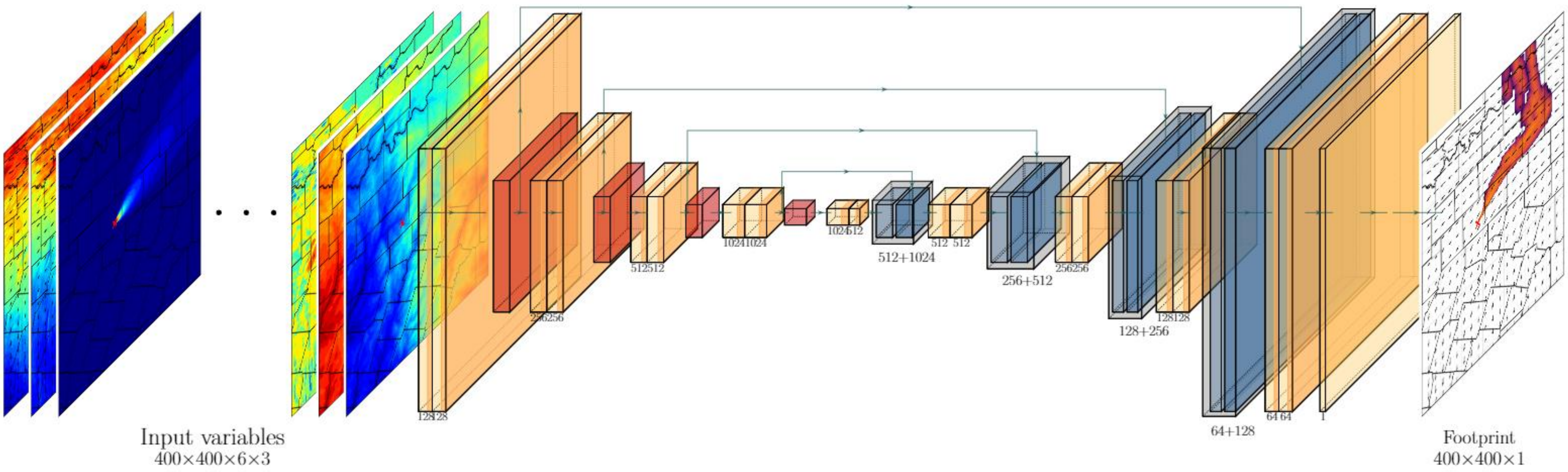
Output



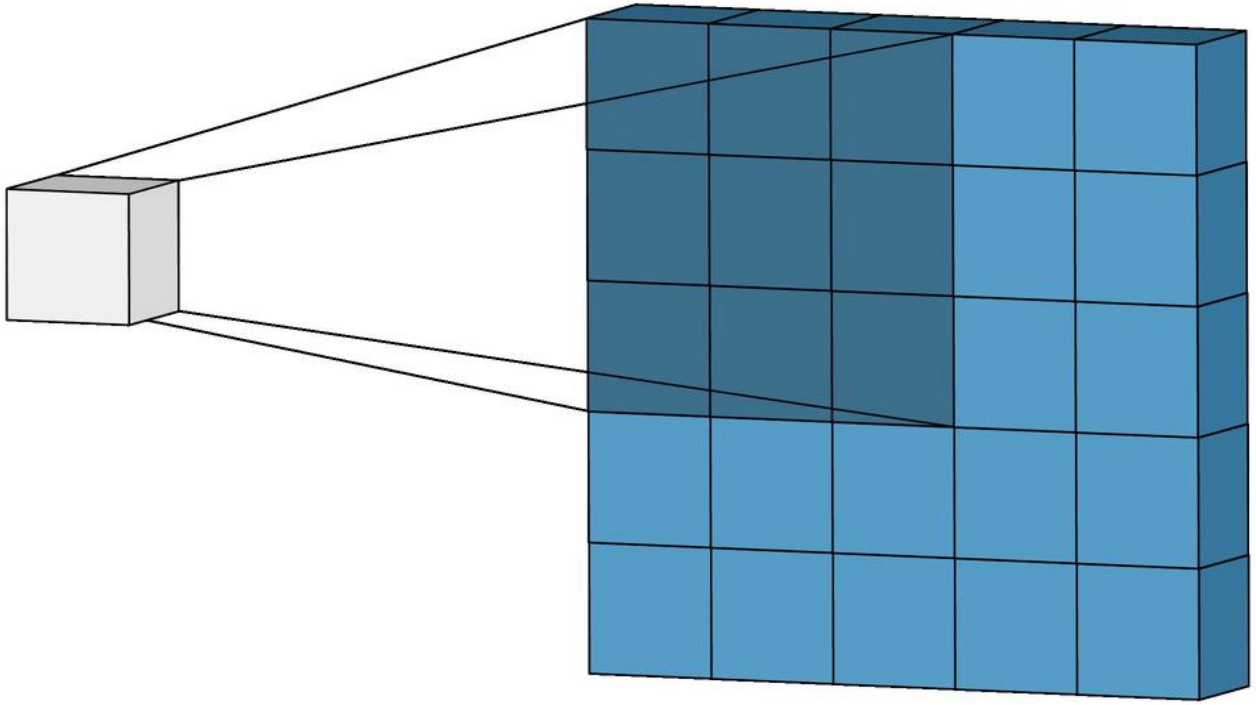
Can machine learning help?

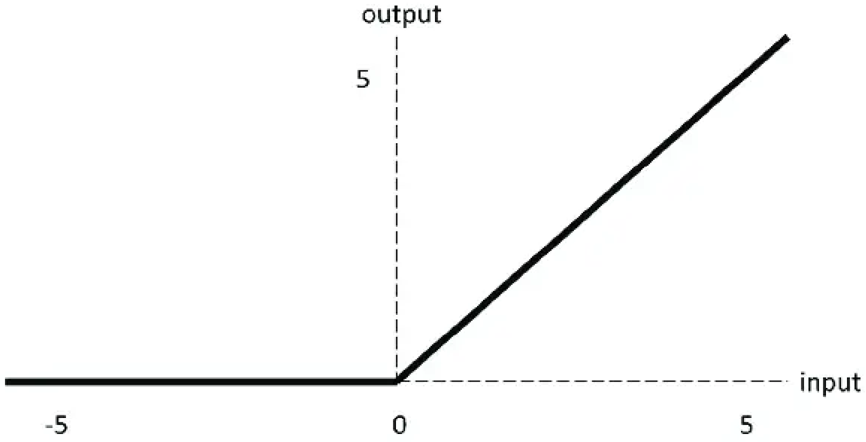


U-Net Model Architecture



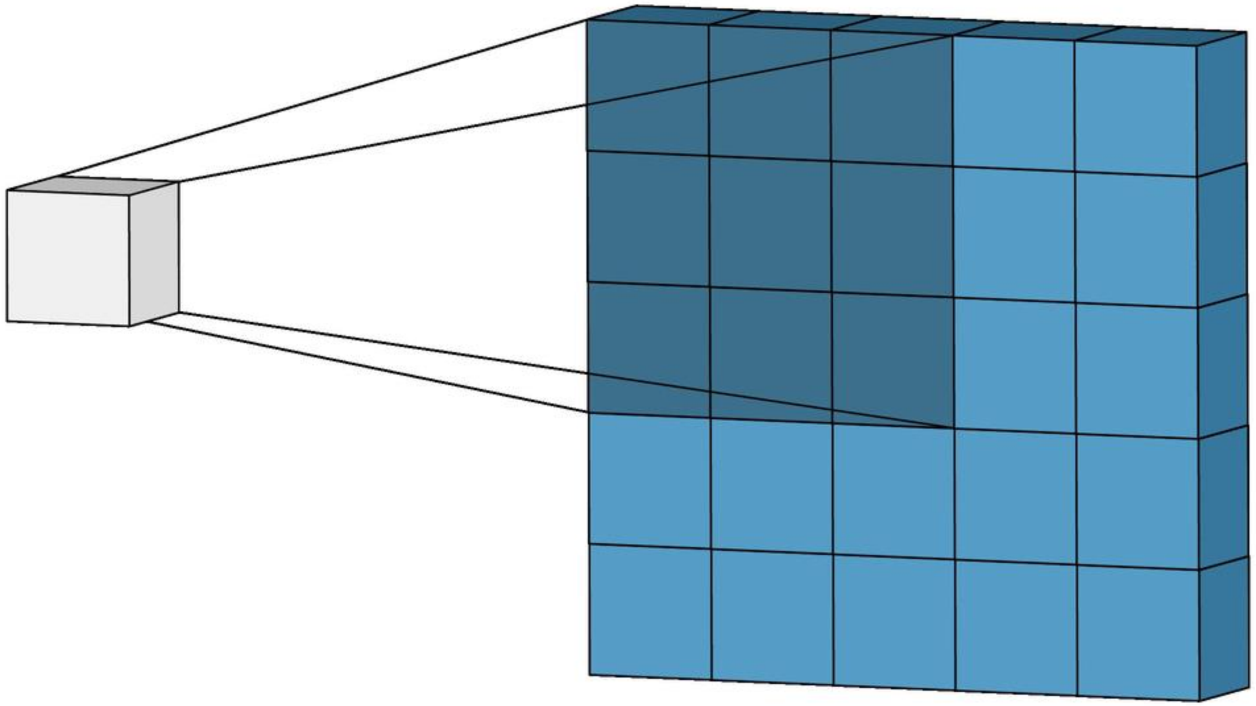
Convolution and Pooling

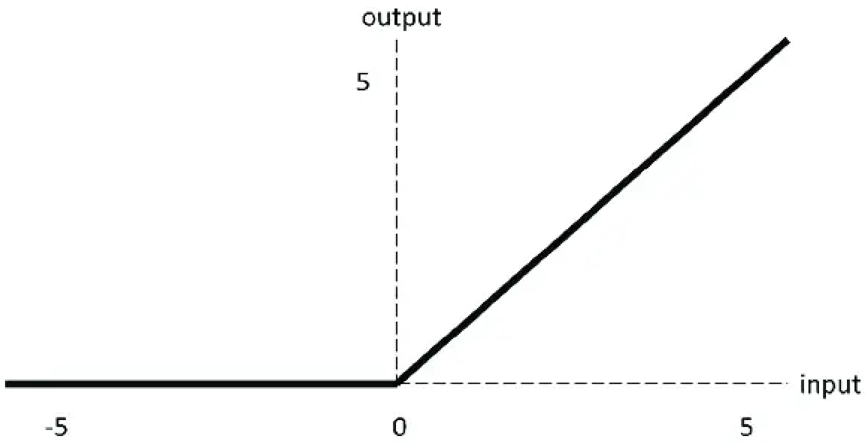




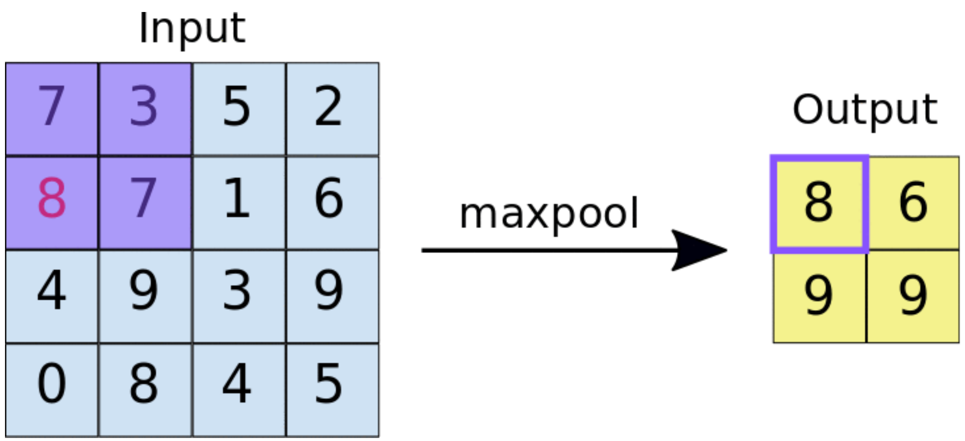
ReLU activation function

Convolution and Pooling

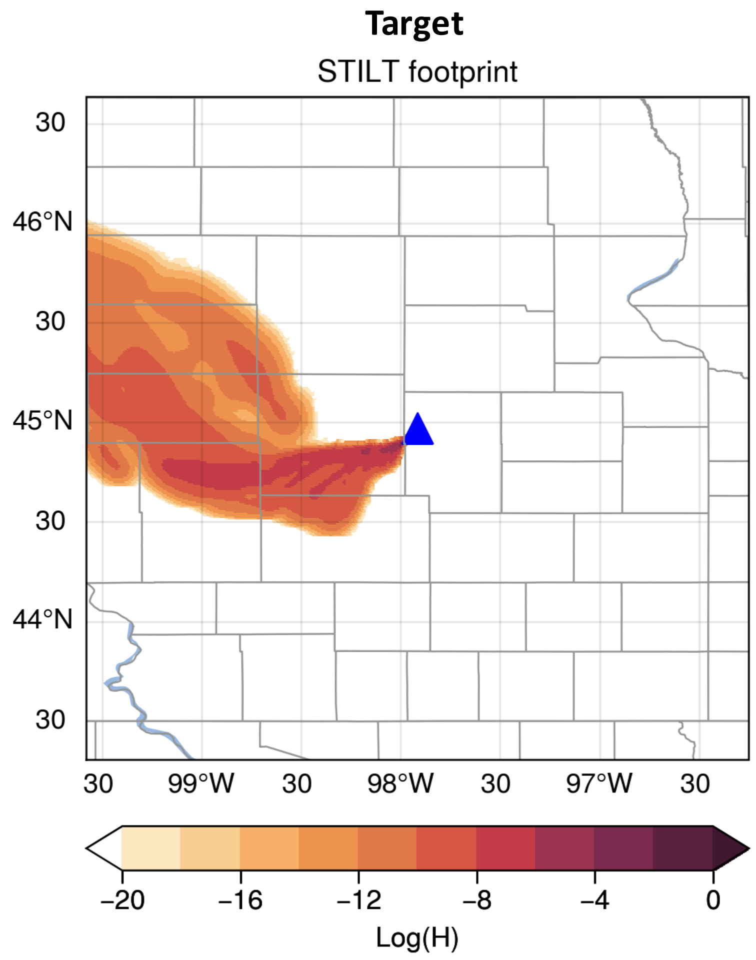




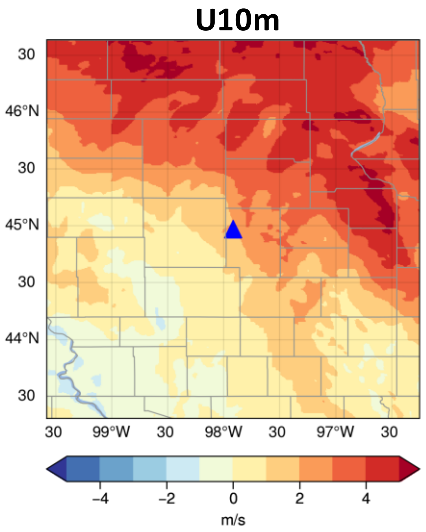
ReLU activation function

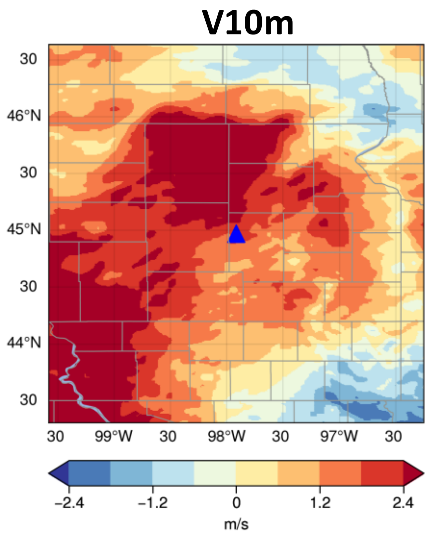


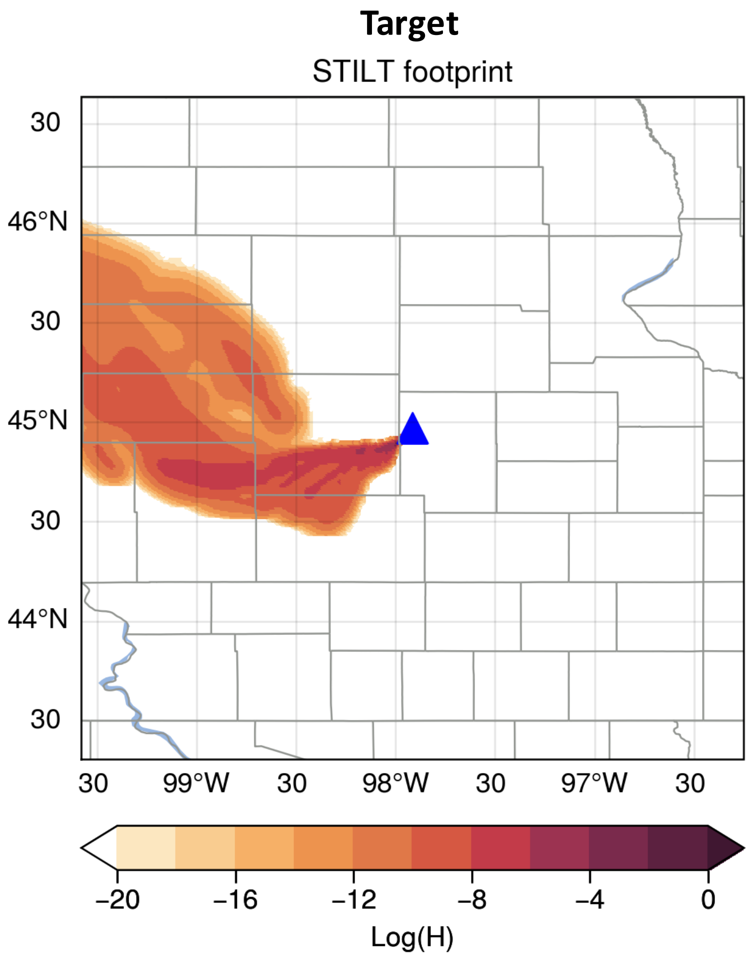
Input variables and output



Input variables and output

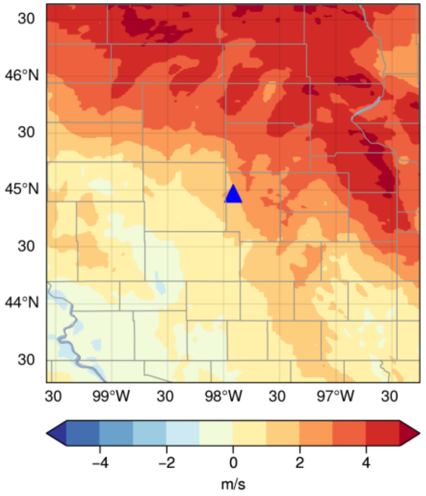




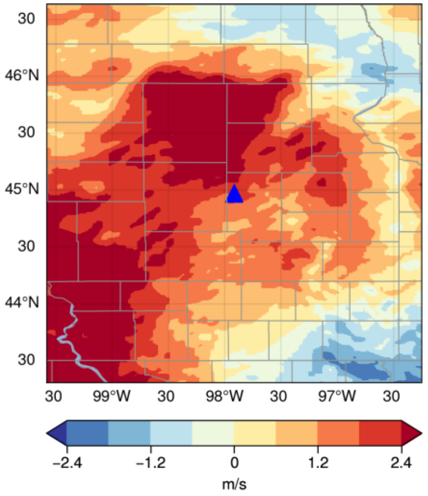


Input variables and output

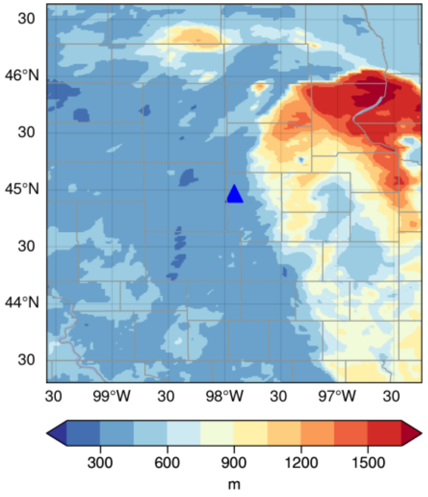
U10m



V10m

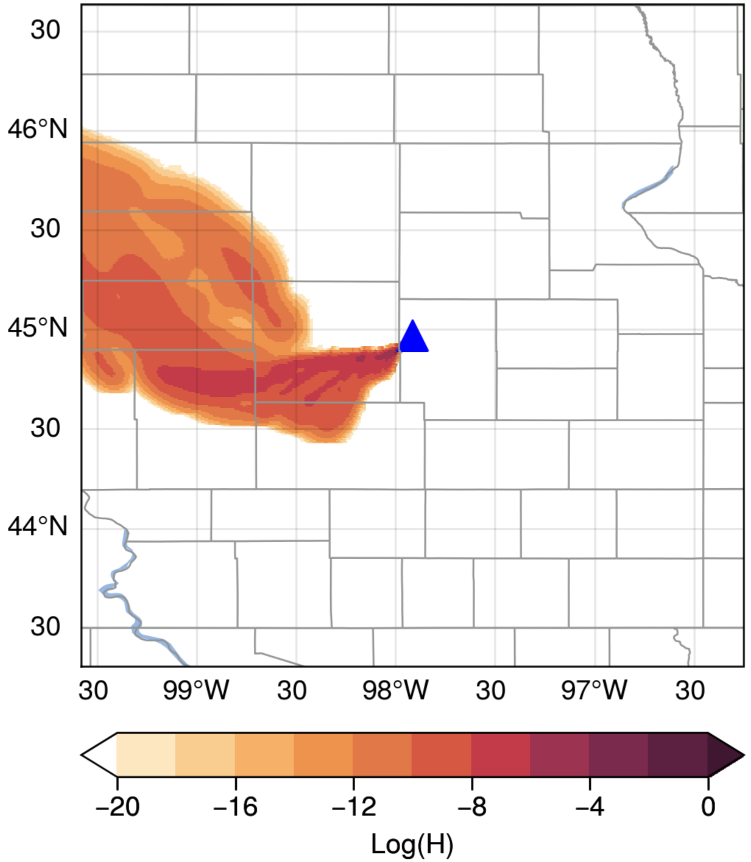


PBLH



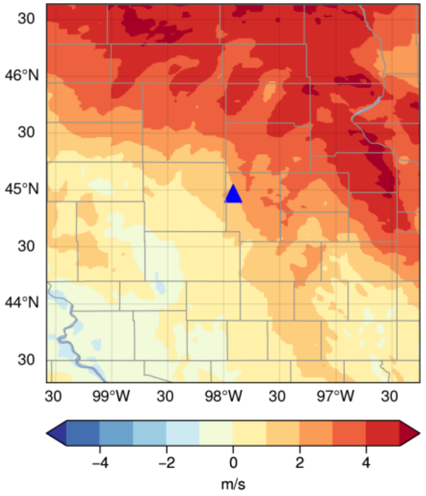
Target

STILT footprint

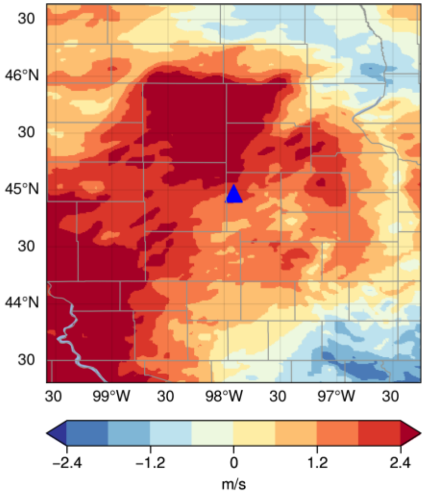


Input variables and output

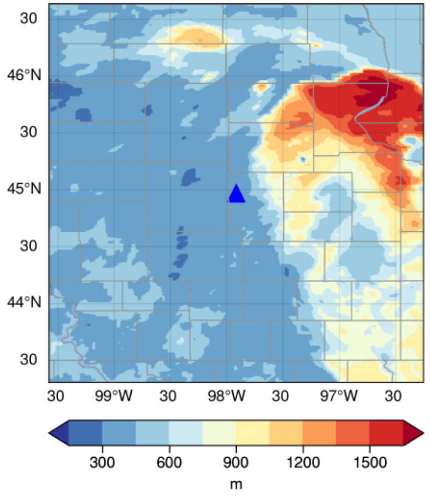
U10m



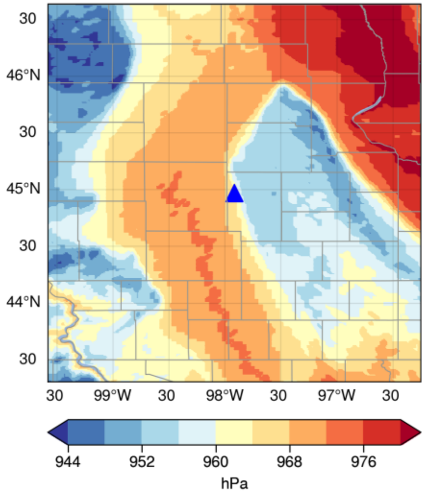
V10m



PBLH

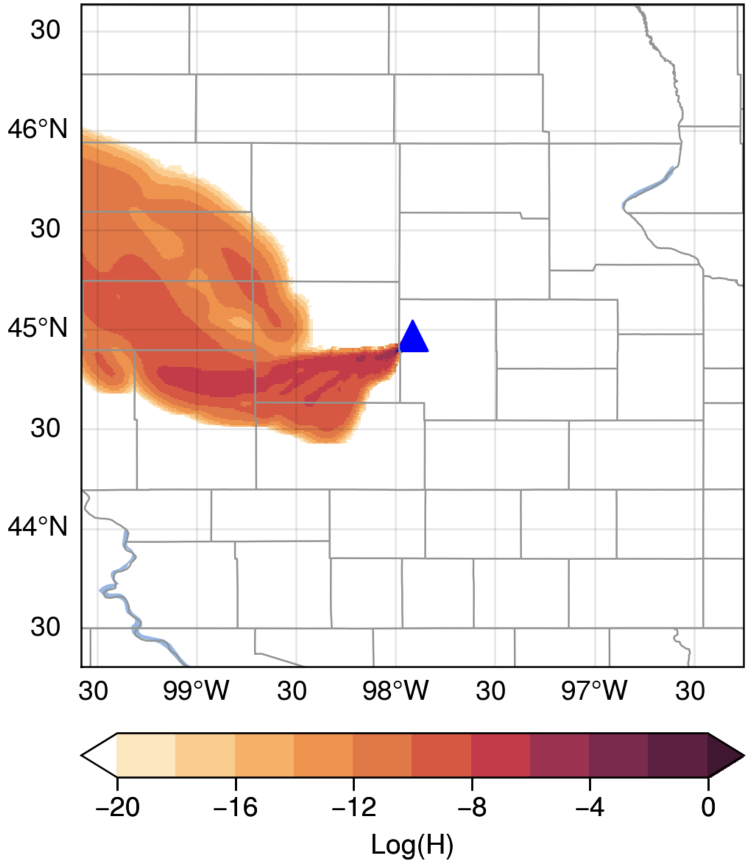


Psurf

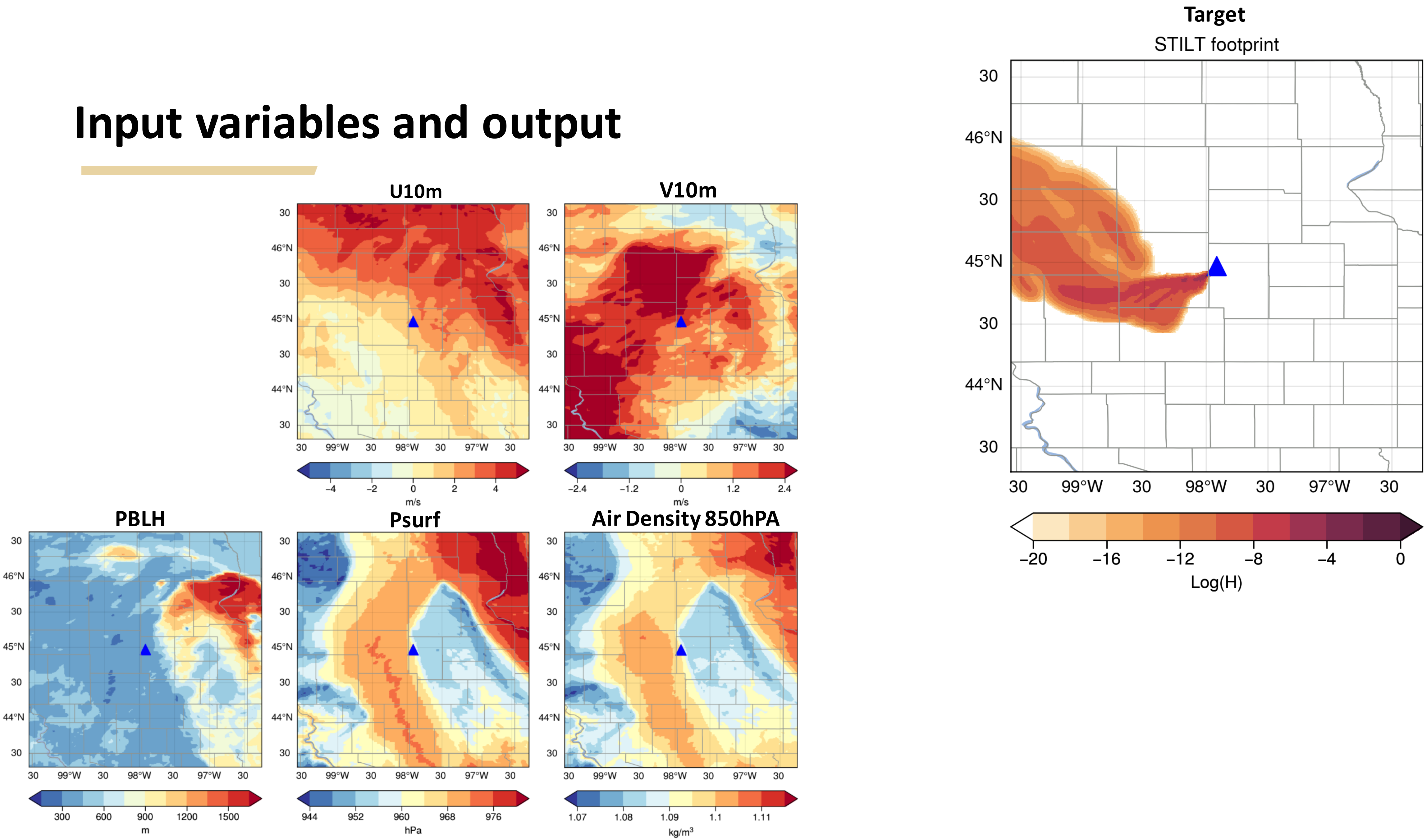


Target

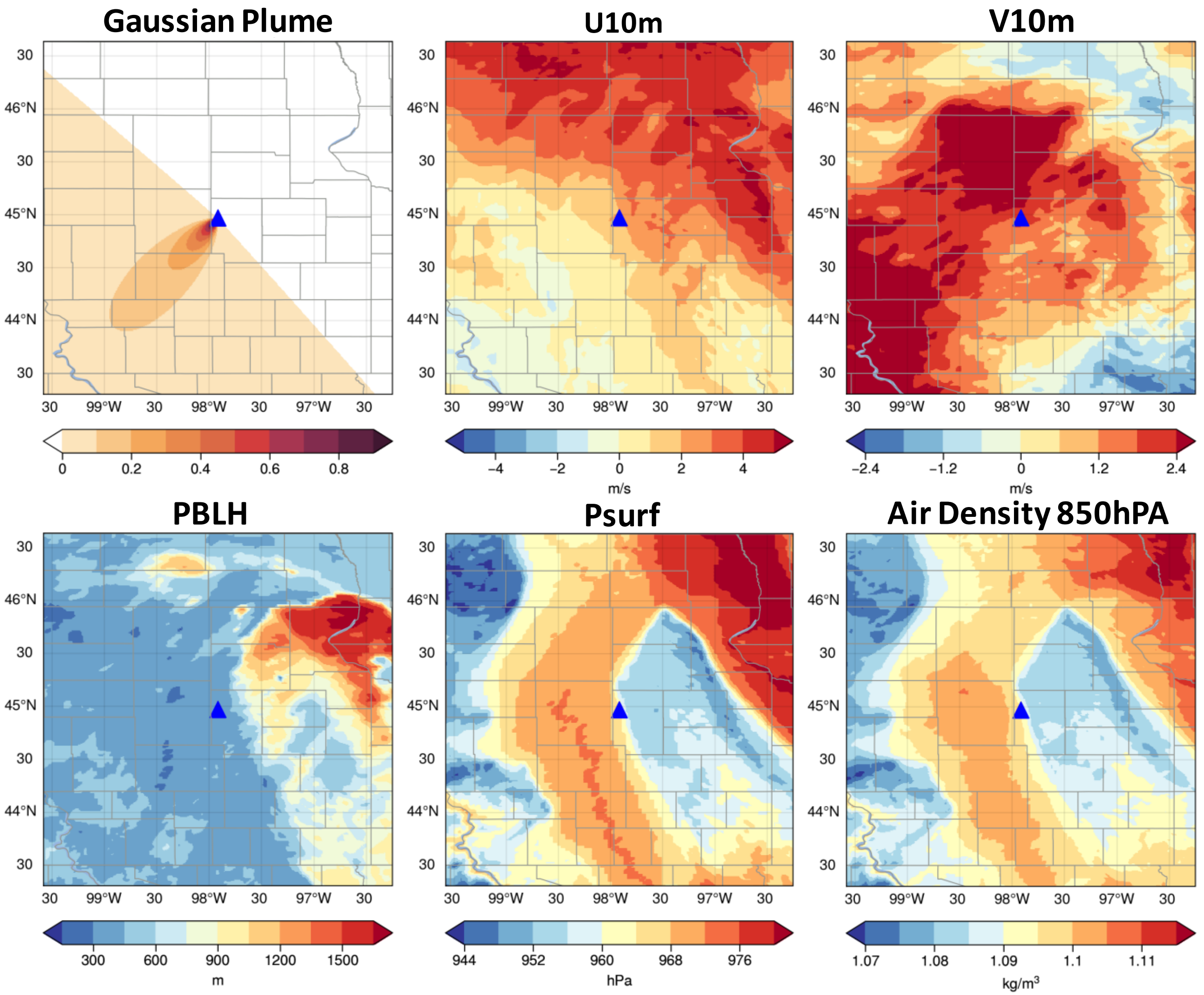
STILT footprint

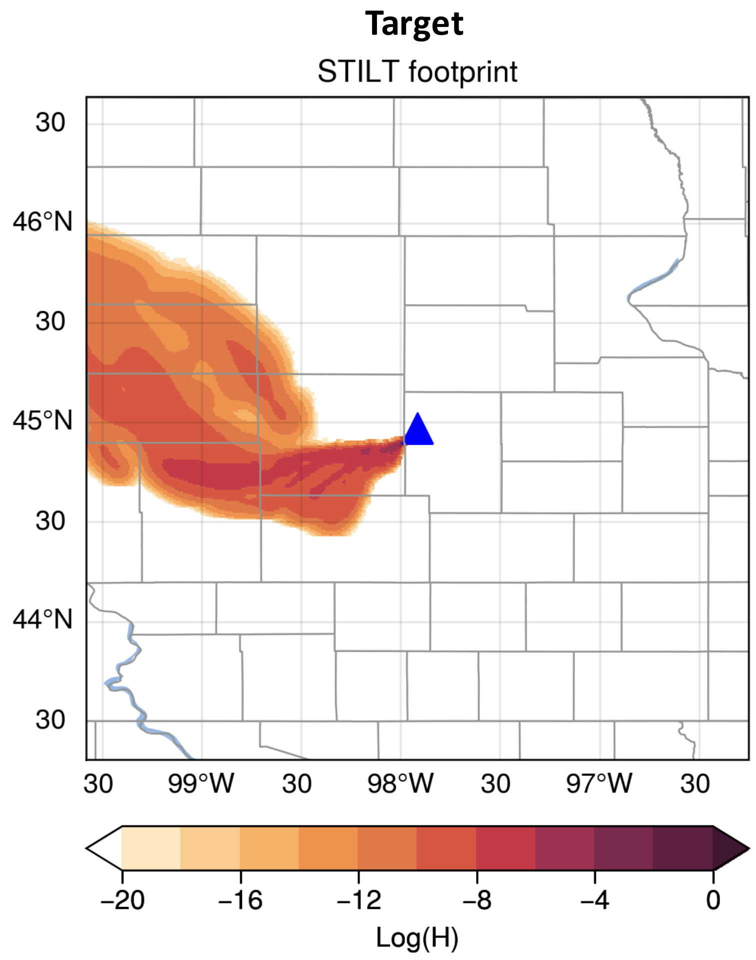


Input variables and output

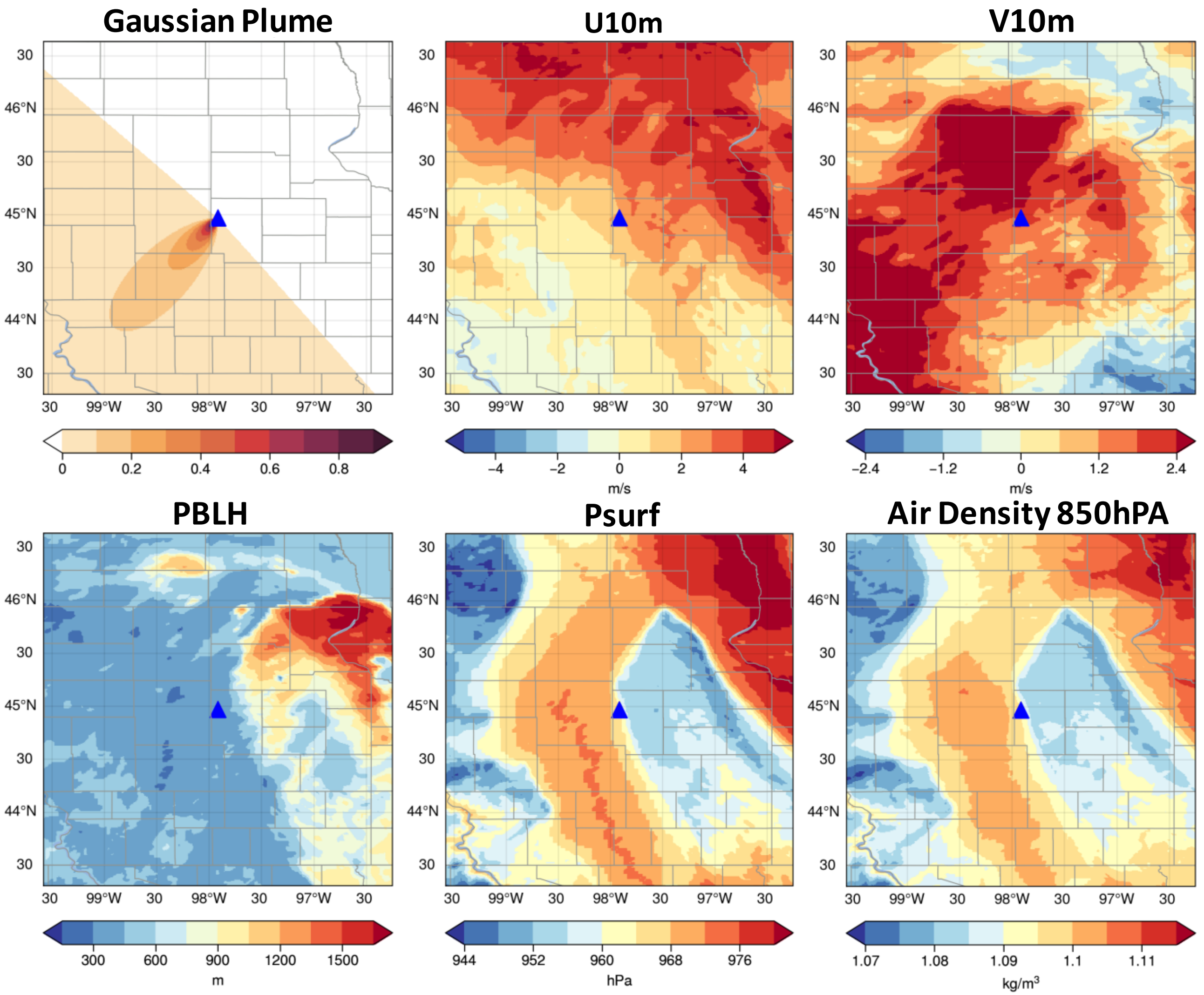


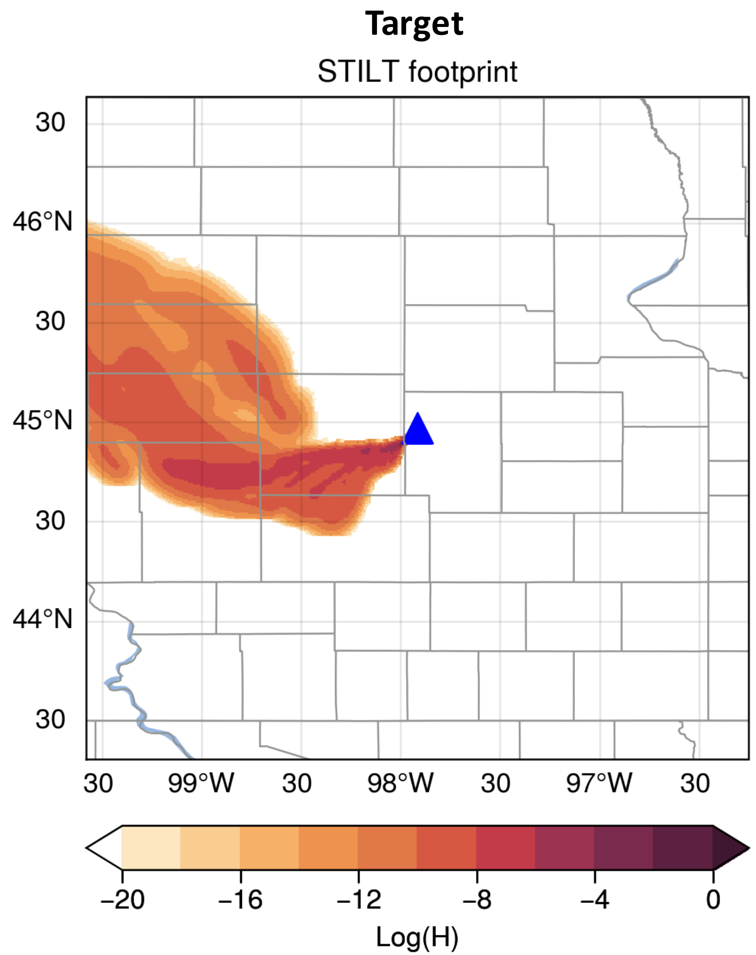
Input variables and output





Input variables and output

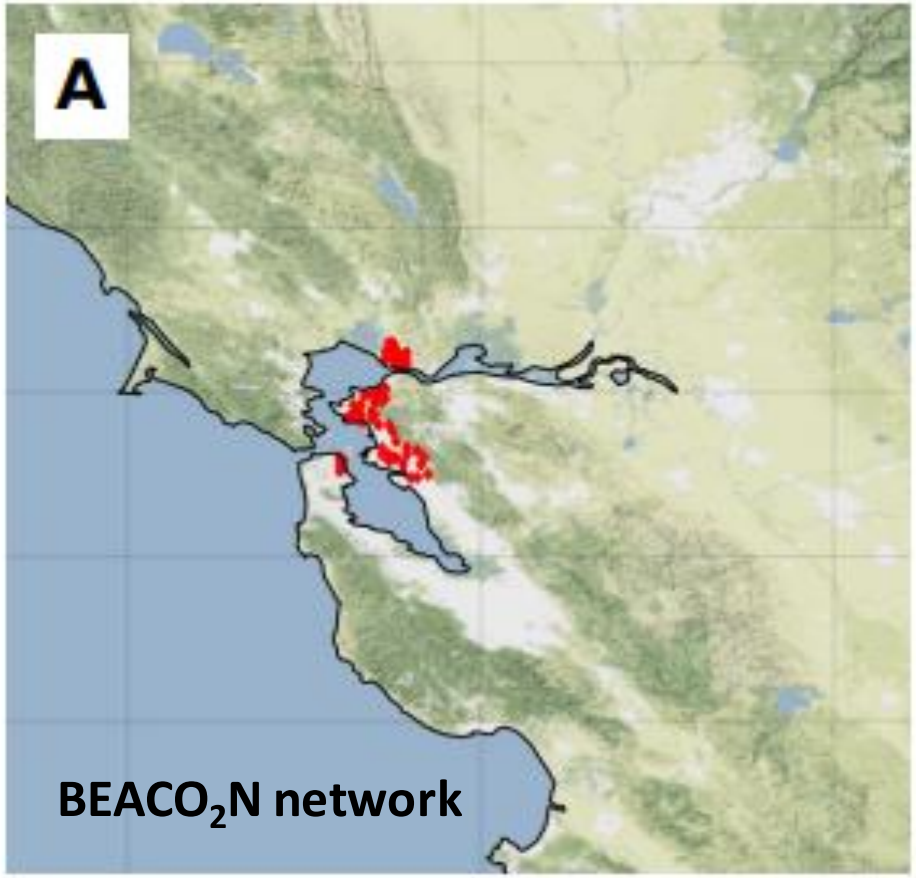


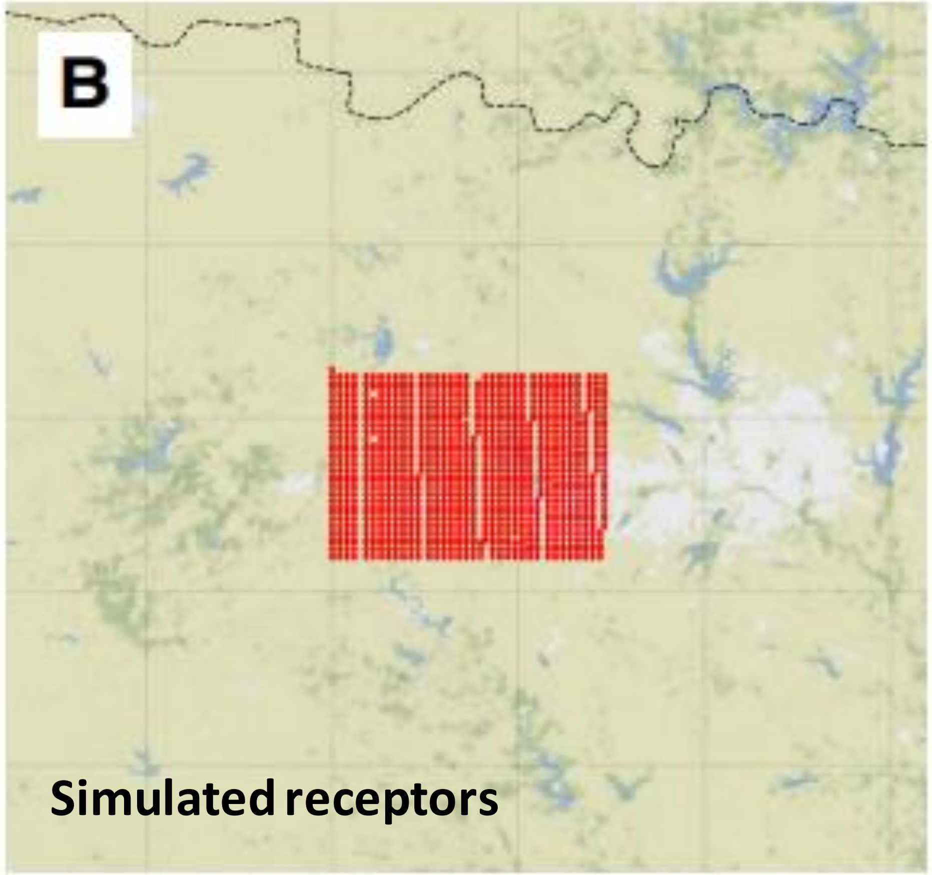


- Meteorological variables are from the NOAA HRRR data product.
- We use footprints simulated by the STILT model as the “truth” to train the U-net.
- All the input and output fields are at 1 km resolution.

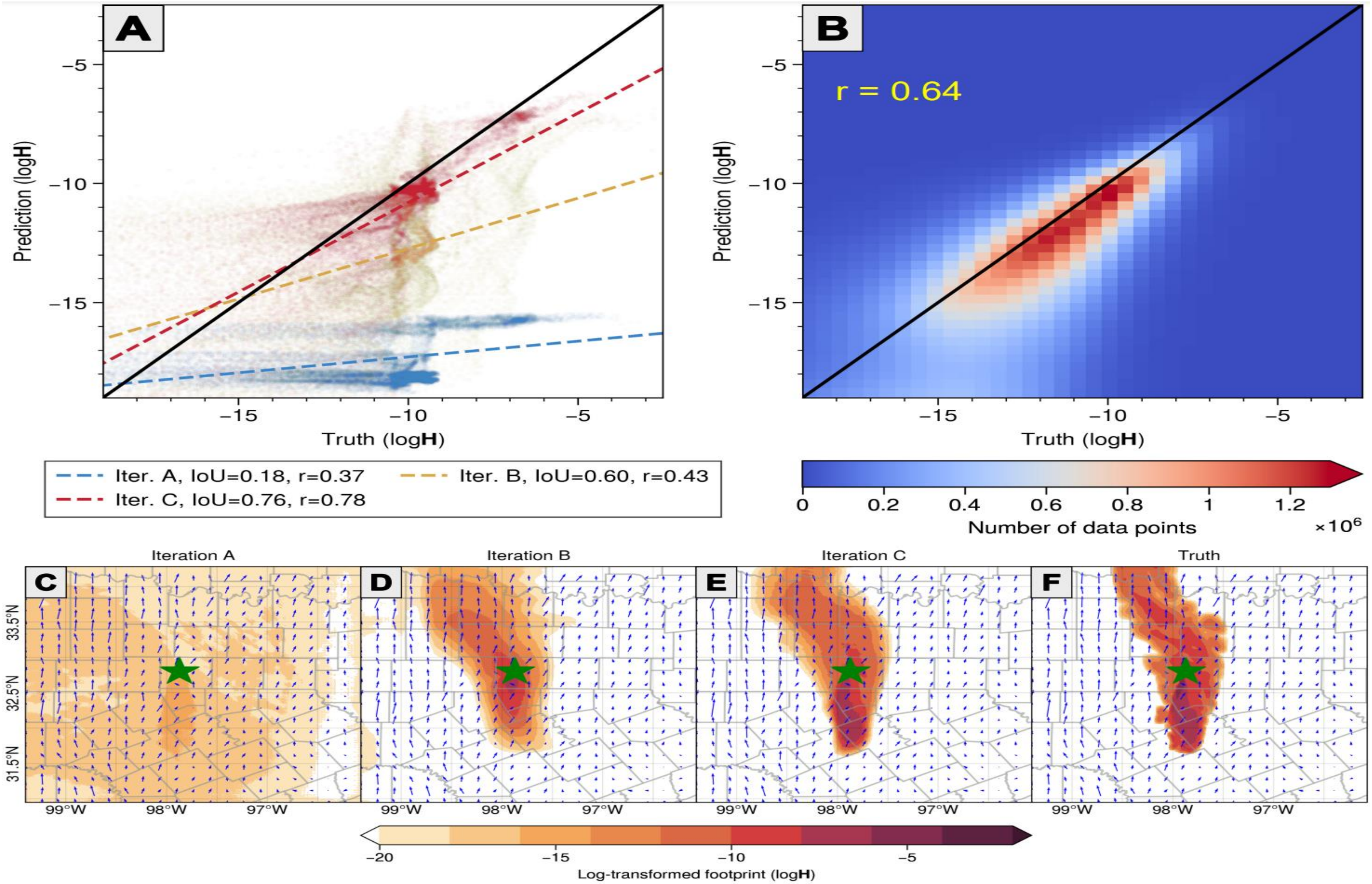
2 Case studies

SF Bay Area	Barnett Shale, TX





Model Training

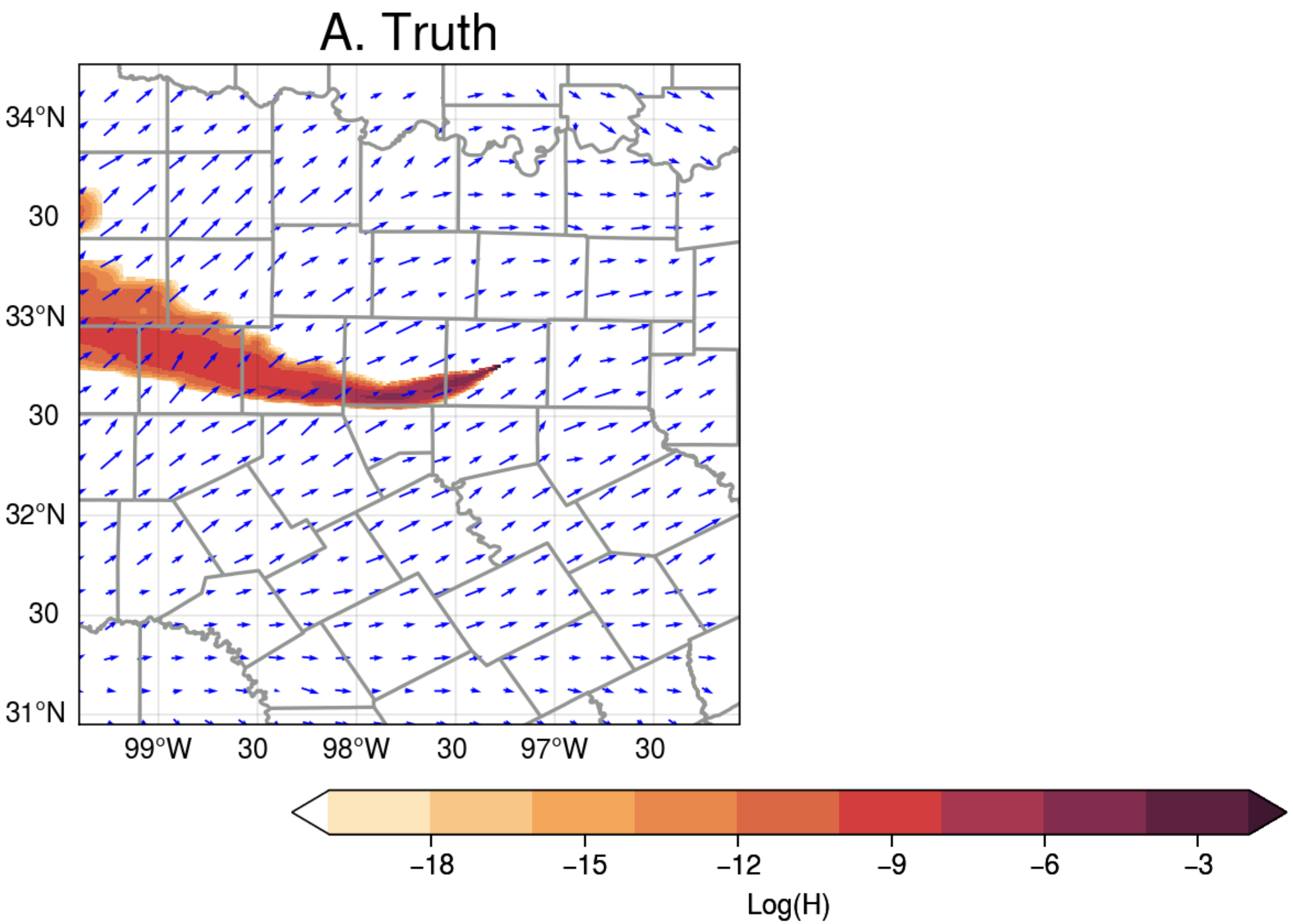


Results

> Let's start with an easy area for model comparison

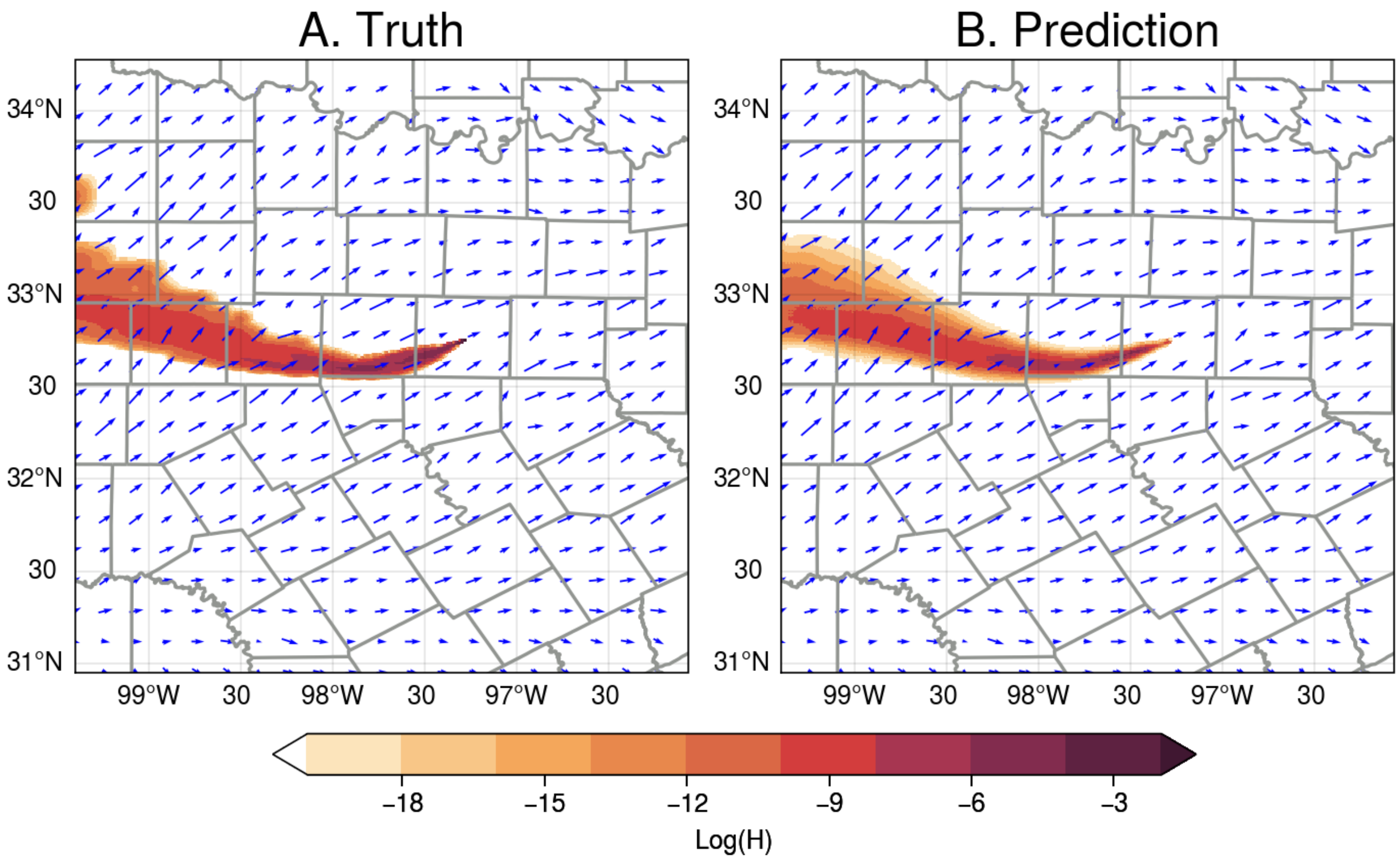
Results


caseB


A. Truth
34°N
30
33°N
30
32°N
30
31°N
99°W
30
98°W
30
97°W
30
–18
–15
–12
-9
-6
-3
Log(H)

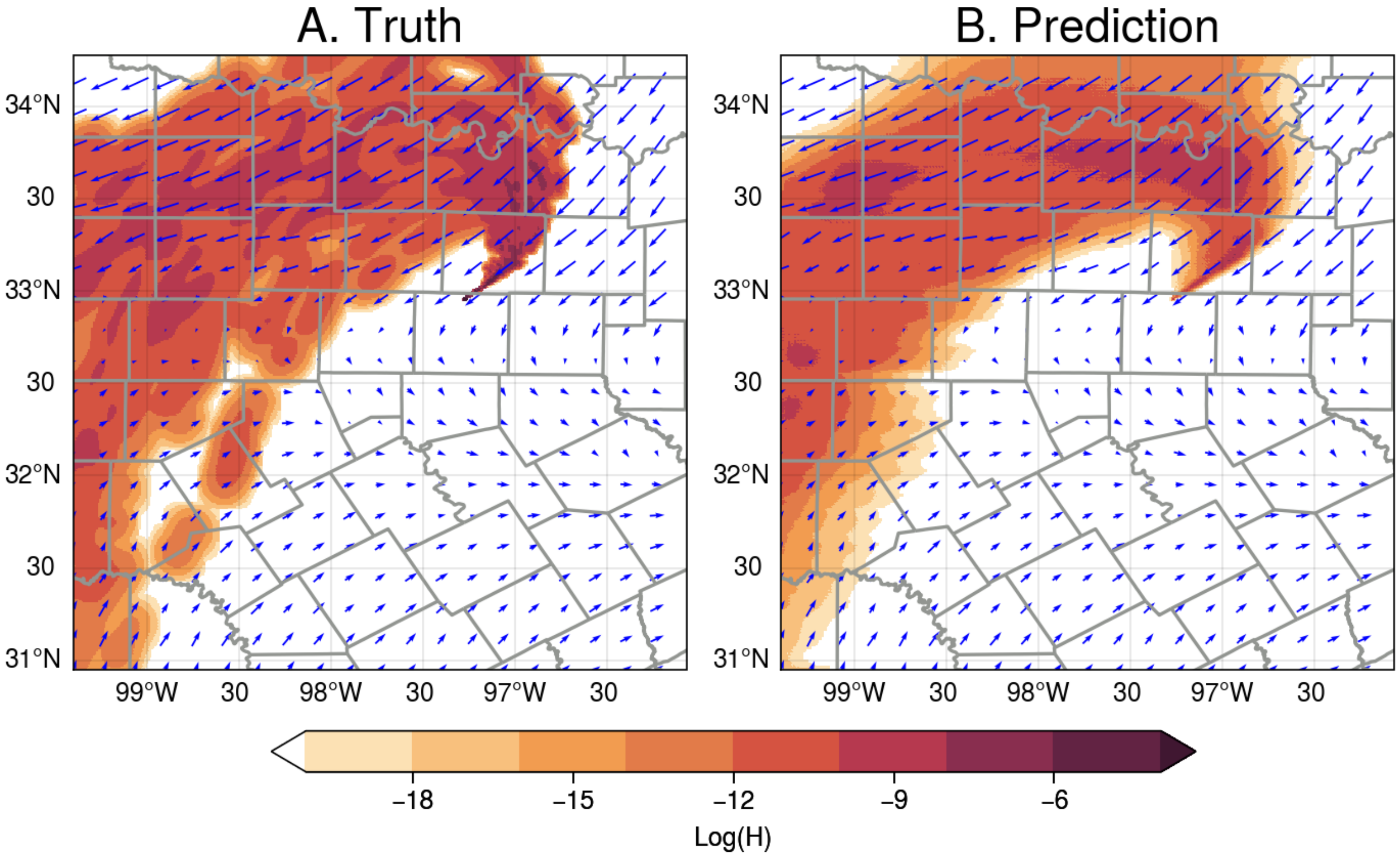
Results

caseB



Results


caseB

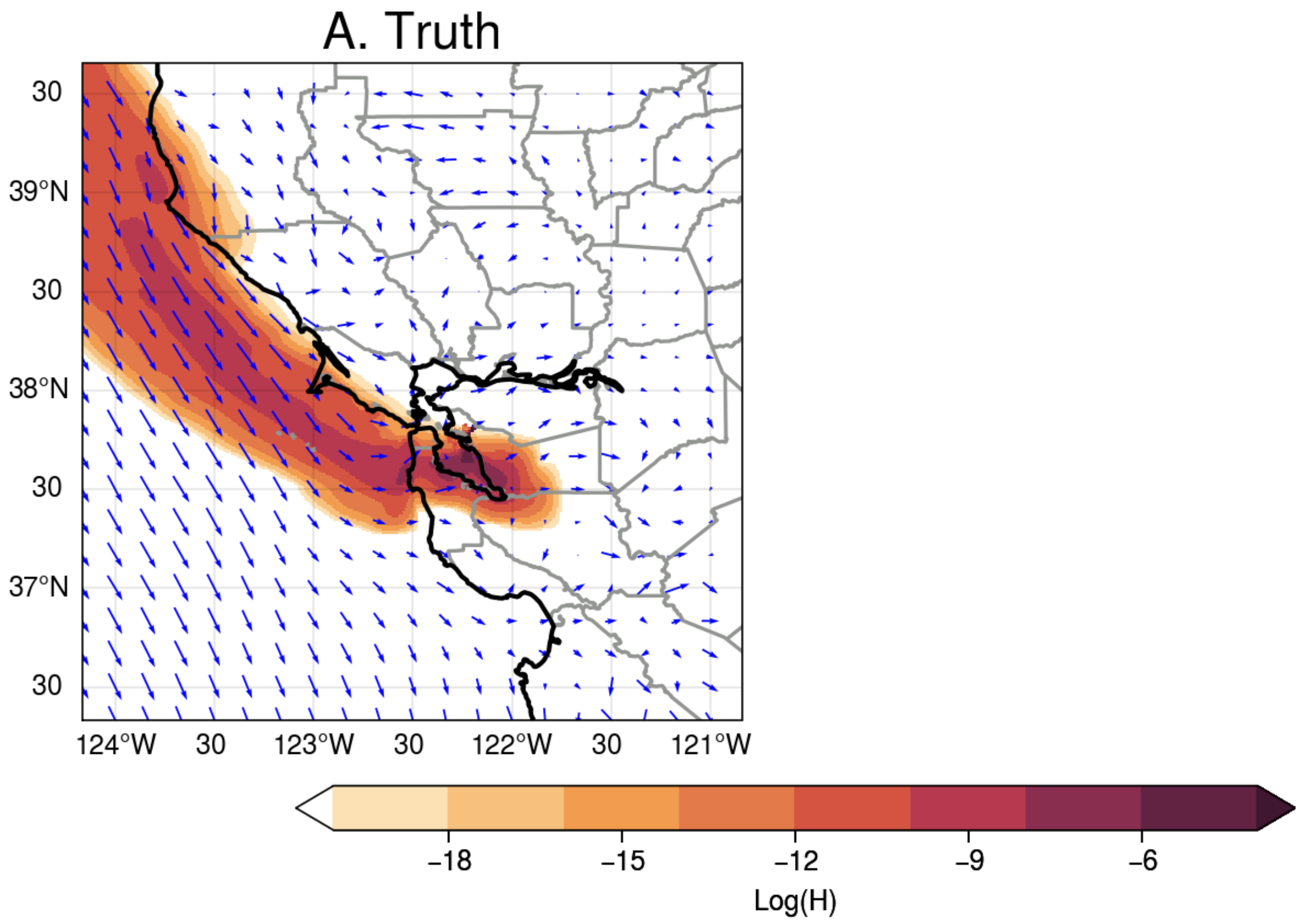

A. Truth
B. Prediction
34°N
30
33°N
30
32°N
30
31°N
99°W
30
98°W
30
97°W
30
−18
−15
−12
−9
−6
Log(H)

Results

> **Now a difficult area for model comparison**

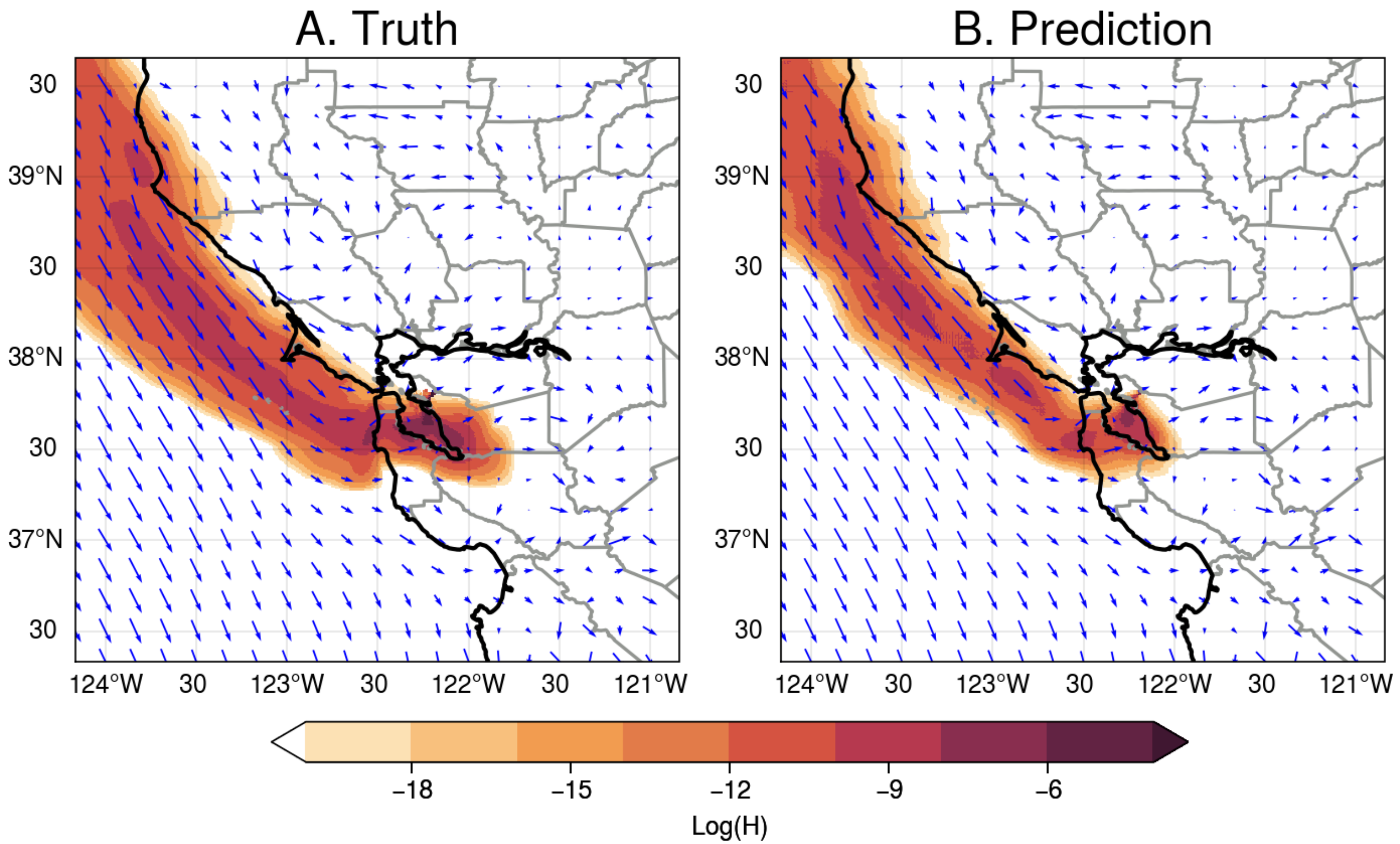
Results


caseA


A. Truth
30
39°N
30
38°N
30
37°N
30
124°W
30
123°W
30
122°W
30
121°W
–18
–15
–12
–9
–6
Log(H)

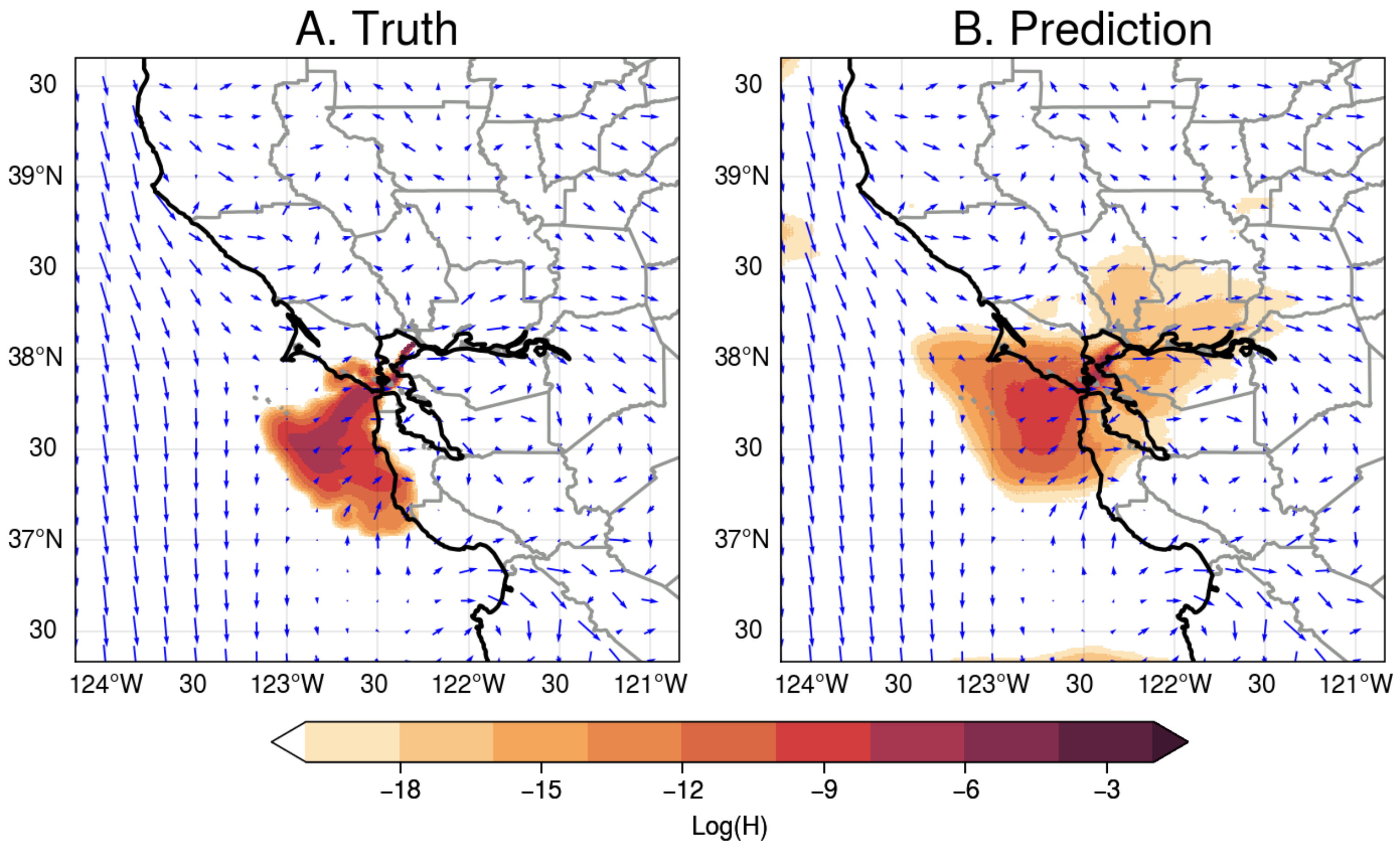
Results


caseA


A. Truth
B. Prediction
30
39°N
30
38°N
30
37°N
30
124°W
30
123°W
30
122°W
30
121°W
–18
–15
–12
–9
–6
Log(H)

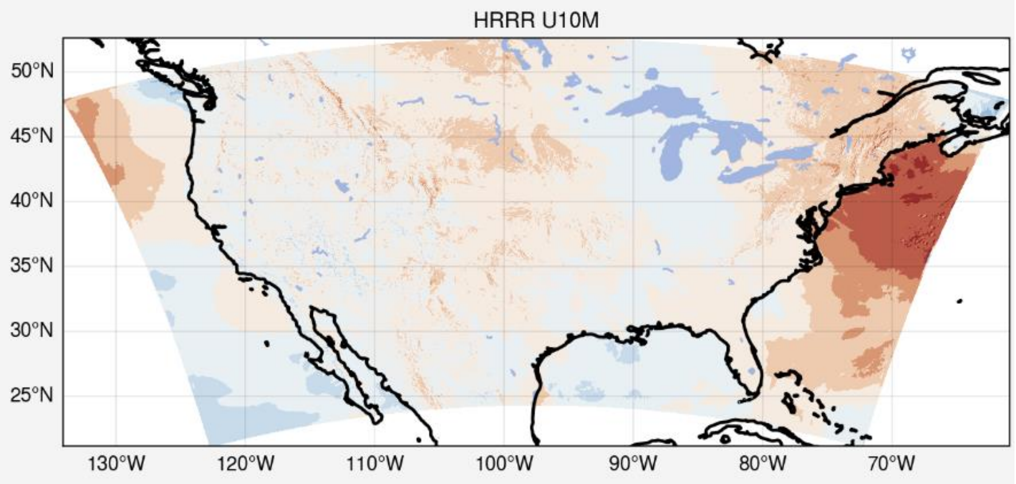
Results


caseA


A. Truth
B. Prediction
30
39°N
30
38°N
30
37°N
30
124°W
30
123°W
30
122°W
30
121°W
–18
–15
–12
–9
–6
–3
Log(H)

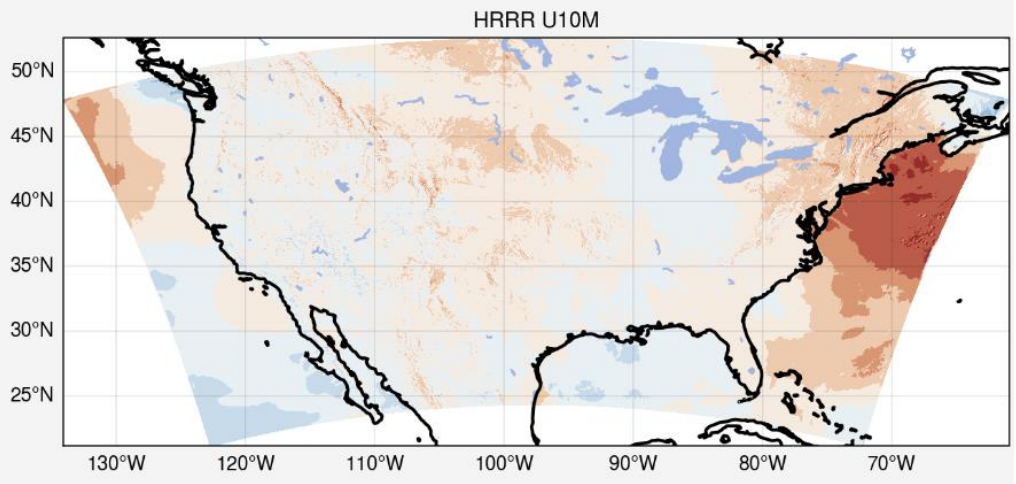
How does the machine learning model solve the computational bottlenecks?

Computational cost



1. HRRR-lite over CONUS

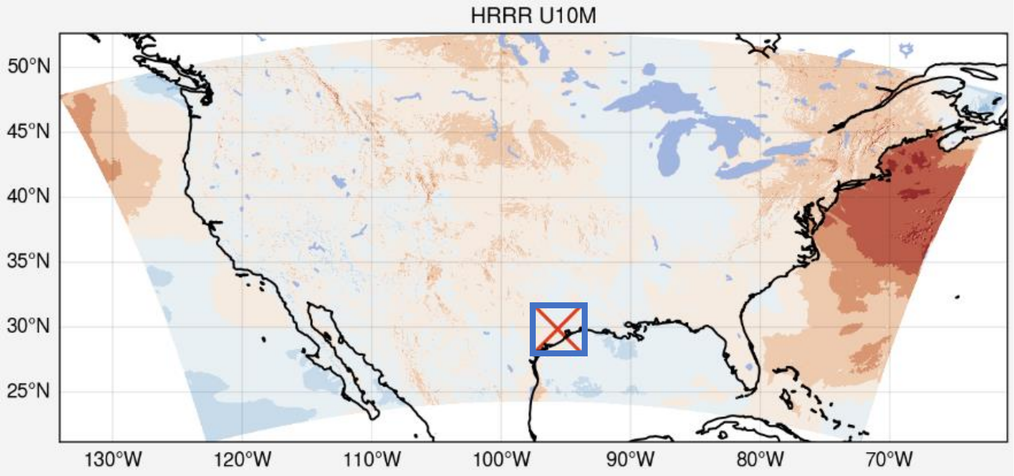
Computational cost



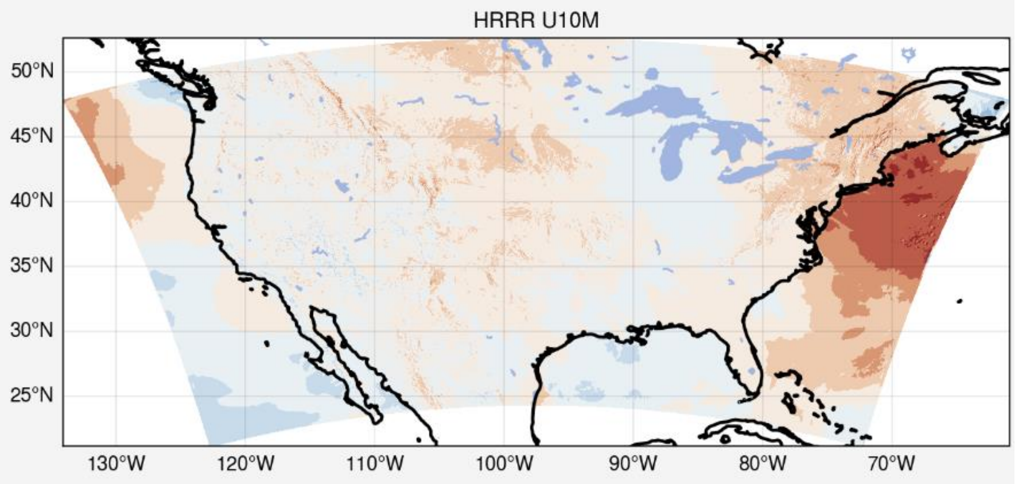
1. HRRR-lite over CONUS

2. I/O and zoom in

0.001 s



Computational cost



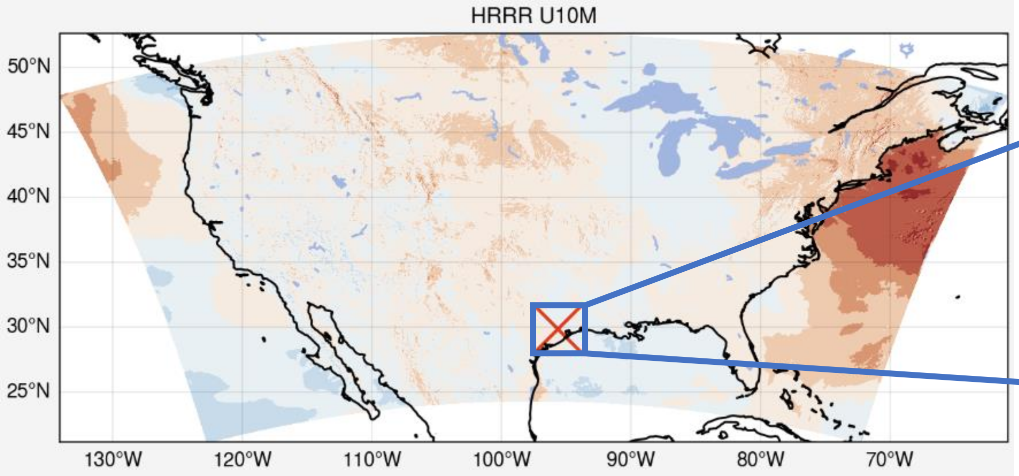
1. HRRR-lite over CONUS

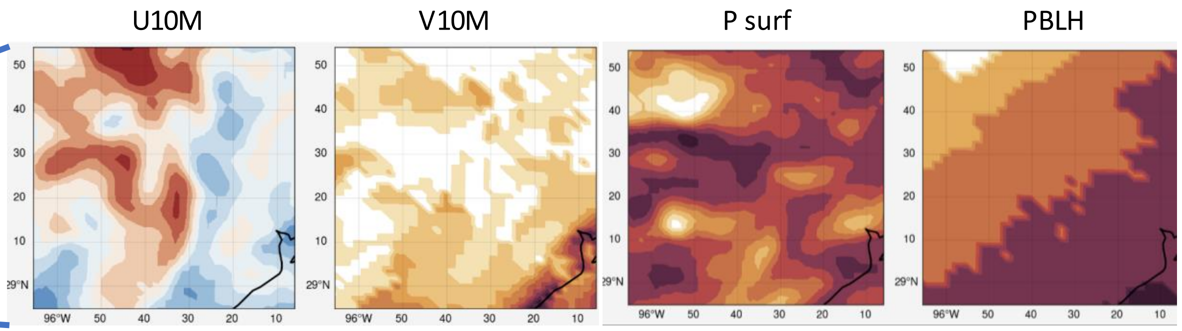
2. I/O and zoom in

0.001 s

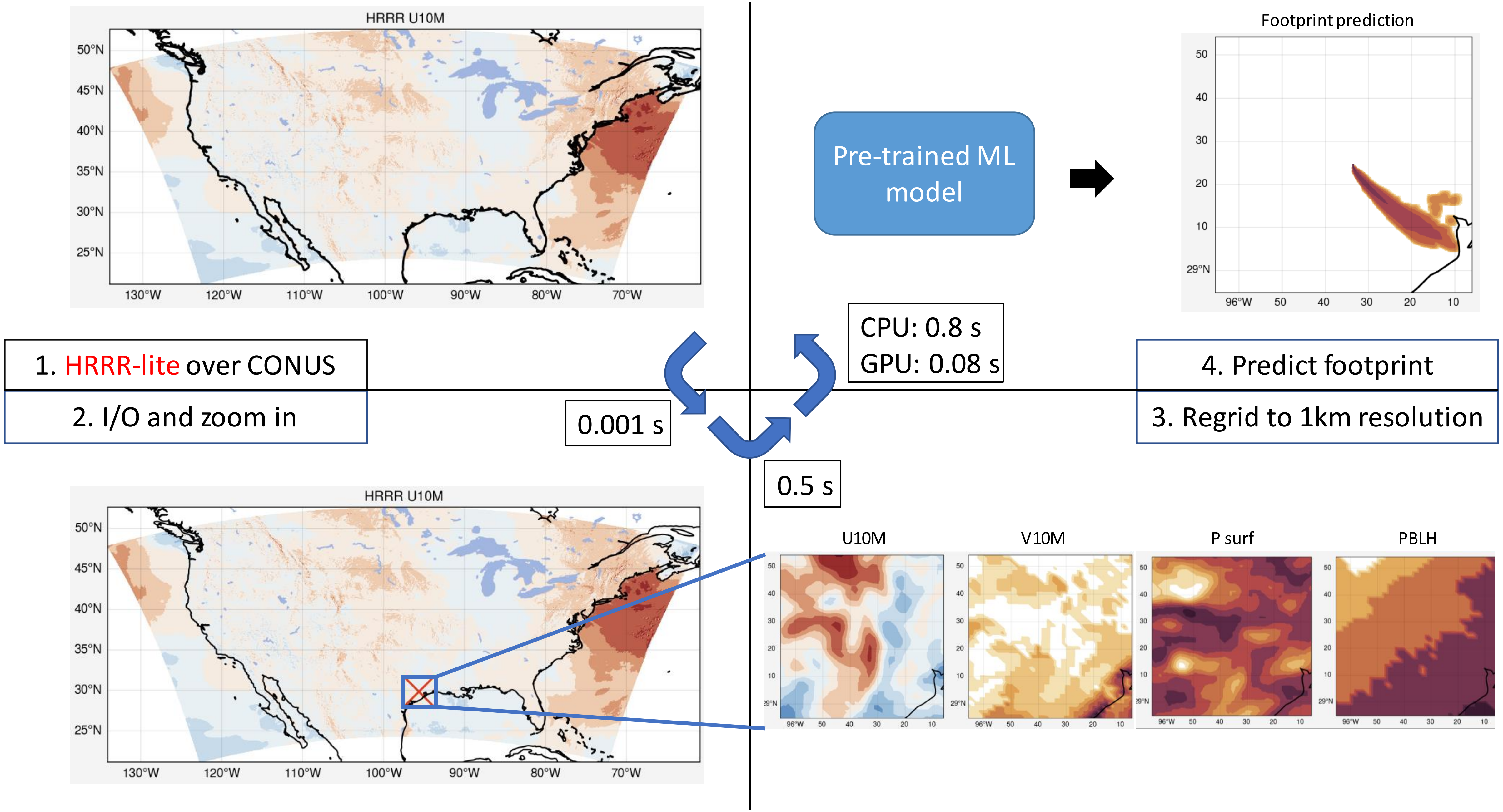
3. Regrid to 1km resolution

0.5 s

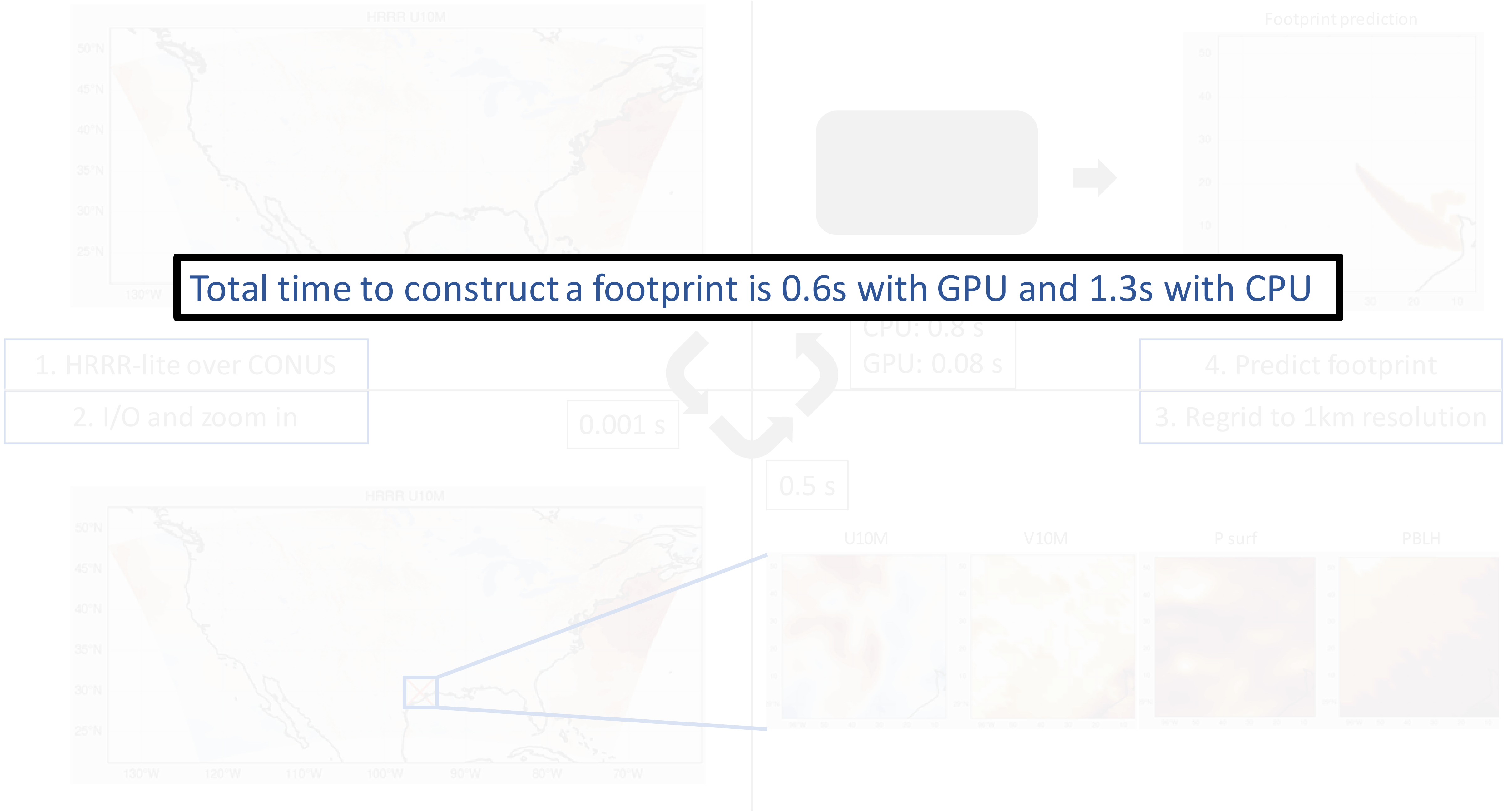




Computational cost

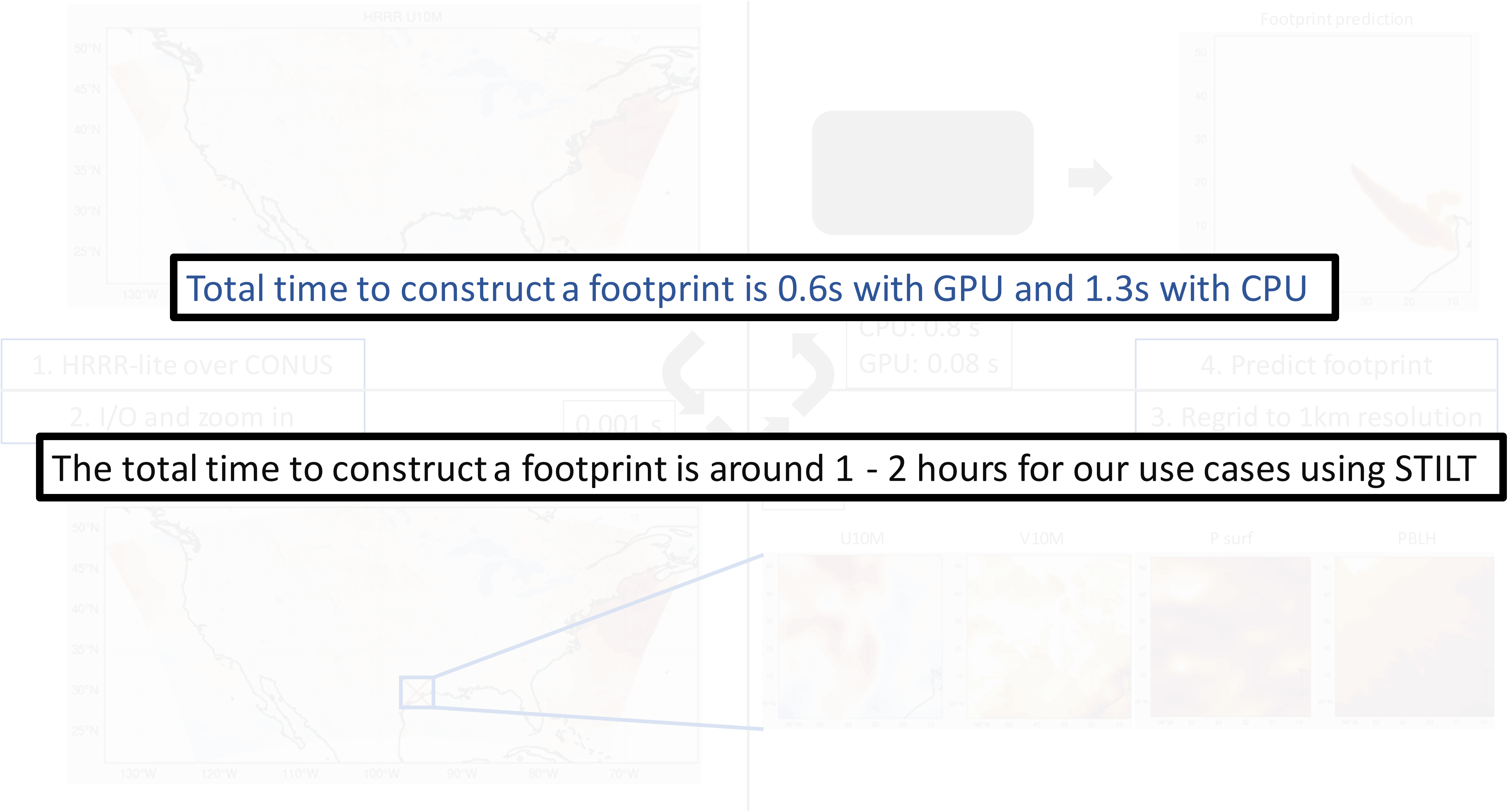


Computational cost



Total time to construct a footprint is 0.6s with GPU and 1.3s with CPU

Computational cost



Total time to construct a footprint is 0.6s with GPU and 1.3s with CPU

The total time to construct a footprint is around 1 - 2 hours for our use cases using STILT

Computational cost analysis

Footprint (STILT)

- **1 footprint simulation ~2 hours**

Footprint (machine learning emulator)

- **1 footprint simulation ~0.6 seconds**

Computational cost analysis

Footprint (STILT)

- 1 footprint simulation ~2 hours
- Footprints for observations of a single day (~700 observations)
  - Sequentially ~ 1400 hours (58 days)
  - Parallel on 32 cores machine ~44 hours (2 days)
  - Parallel on 10 nodes ~4 hours

Footprint (machine learning emulator)

- 1 footprint simulation ~0.6 seconds
- Footprints for observations of a single day (~700 observations)
  - Predictions over GPU ~5 minutes (4.5 mins for loading the data and rest 30 seconds for model predictions)
  - Predictions over CPU ~20 minutes (4.5 mins for loading the data and rest 15.5 minutes for model prediction)

Computational cost analysis

Footprint (STILT)

- 1 footprint simulation ~2 hours
- Footprints for observations of a single day (~700 observations)
  - Sequentially ~ 1400 hours (58 days)
  - Parallel on 32 cores machine ~44 hours (2 days)
  - Parallel on 10 nodes ~4 hours
- Footprints for observations (Feb – April 2020)
  - Parallel on 10 nodes (32 cores machines) ~16 days
  - Storage cost ~400GB

Footprint (machine learning emulator)

- 1 footprint simulation ~0.6 seconds
- Footprints for observations of a single day (~700 observations)
  - Predictions over GPU ~5 minutes (4.5 mins for loading the data and rest 30 seconds for model predictions)
  - Predictions over CPU ~20 minutes (4.5 mins for loading the data and rest 15.5 minutes for model prediction)
- Footprints for observations (Feb – April 2020)
  - Predictions over GPU ~7.5 hours
  - If we optimize the data loading ~15 minutes
  - Can be computed on fly

Computational cost analysis

Footprint (STILT)

- 1 footprint simulation ~2 hours
- Footprints for observations of a single day (~700 observations)
  - Sequentially ~ 1400 hours (58 days)
  - Parallel on 32 cores machine ~44 hours (2 days)
  - Parallel on 10 nodes ~4 hours
- Footprints for observations (Feb – April 2020)
  - Parallel on 10 nodes (32 cores machines) ~16 days
  - Storage cost ~400GB

Requirements

- 10 nodes for simulations
- 32 cores processor on each node

Footprint (machine learning emulator)

- 1 footprint simulation ~0.6 seconds
- Footprints for observations of a single day (~700 observations)
  - Predictions over GPU ~5 minutes (4.5 mins for loading the data and rest 30 seconds for model predictions)
  - Predictions over CPU ~20 minutes (4.5 mins for loading the data and rest 15.5 minutes for model prediction)
- Footprints for observations (Feb – April 2020)
  - Predictions over GPU ~7.5 hours
  - If we optimize the data loading ~15 minutes
  - Can be computed on fly

Requirements

- 1 node for predictions
- 1 GPU card (16GB)

Computational cost analysis

Footprint (STILT)

- 1 footprint simulation ~2 hours
- Footprints for observations of a single day (~700 observations)
  - Sequentially ~ 1400 hours (58 days)
  - Parallel on 32 cores machine ~44 hours (2 days)
  - Parallel on 10 nodes ~4 hours
- Footprints for observations (Feb – April 2020)
  - Parallel on 10 nodes (32 cores machines) ~16 days
  - Storage cost ~400GB

Footprint (machine learning emulator)

- 1 footprint simulation ~0.6 seconds
- Footprints for observations of a single day (~700 observations)
  - Predictions over GPU ~5 minutes (4.5 mins for loading the data and rest 30 seconds for model predictions)
  - Predictions over CPU ~20 minutes (4.5 mins for loading the data and rest 15.5 minutes for model prediction)
- Footprints for observations (Feb – April 2020)
  - Predictions over GPU ~7.5 hours
  - If we optimize the data loading ~15 minutes
  - Can be computed on fly

Construction of footprints is currently 50 times faster with emulator than STILT model.

Goal of this work

> Developing an efficient method to compute source-receptor relationship using machine learning emulator (FootNet)

> **Estimating GHG emission fluxes by emulating atmospheric transport using FootNet**

Case Study: Impacts of COVID-19 on urban CO_2 emissions (Turner et. al., 2020)

Observed Impacts of COVID-19 on Urban CO_2 Emissions

Alexander J. Turner[1,2,3,4], Jinsol Kim[1], Helen Fitzmaurice[1], Catherine Newman[2], Kevin Worthington[2], Katherine Chan[2], Paul J. Wooldridge[2], Philipp Köehler[5], Christian Frankenberg[3,5], and Ronald C. Cohen[1,2]

[1]Department of Earth and Planetary Sciences, University of California, Berkeley, CA, USA, [2]College of Chemistry, University of California, Berkeley, CA, USA, [3]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA, [4]Now at Department of Atmospheric Sciences, University of Washington, Seattle, WA, USA, [5]Division of Geological and Planetary Sciences, California Institute of Technology, Pasadena, CA, USA

Case Study: Impacts of COVID-19 on urban CO_2 emissions (Turner et. al., 2020)

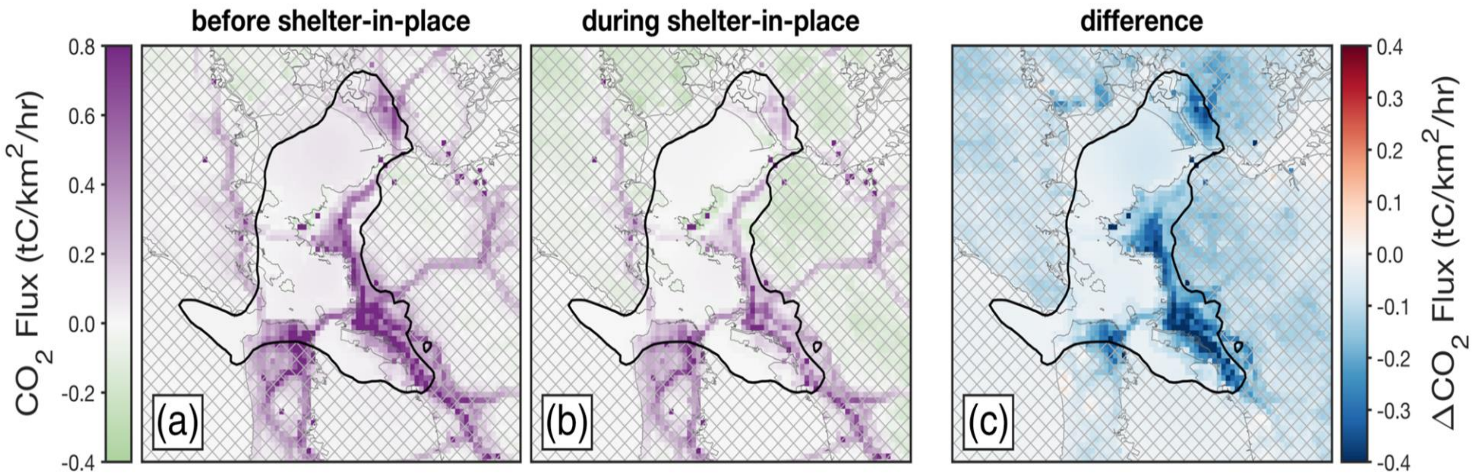


Figure 3. Spatial patterns of CO_2 fluxes in the San Francisco Bay Area. Panel (a) shows the average CO_2 fluxes for 6 weeks before shelter-in-place (2 February 2020 through 14 March 2020). Panel (b) shows the average over 6 weeks during shelter-in-place (22 March 2020 through 2 May 2020). Panel (c) is the difference. Black contour in all panels encompasses the top 40% of the total network influence (BEACO_2N Domain). Cross hatching indicates regions with low sensitivity to the BEACO_2N nodes.

2 Feb – 14 Mar 2020
(Pre covid)

22 Mar – 2 May 2020
(During covid)

Case Study: Impacts of COVID-19 on urban CO_2 emissions (Turner et. al., 2020)

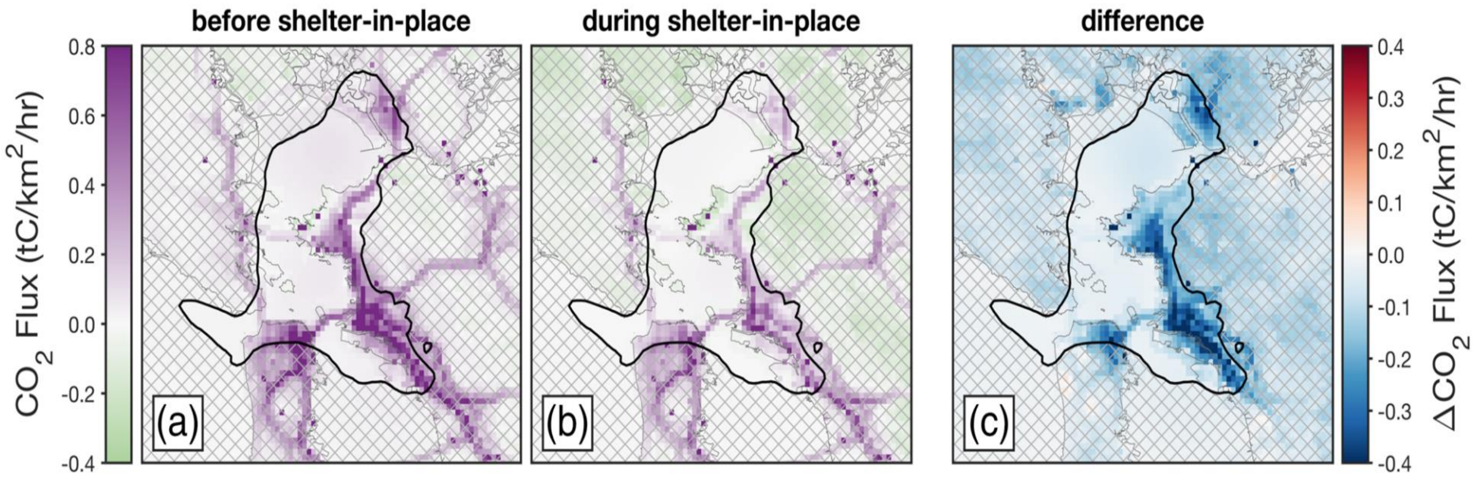
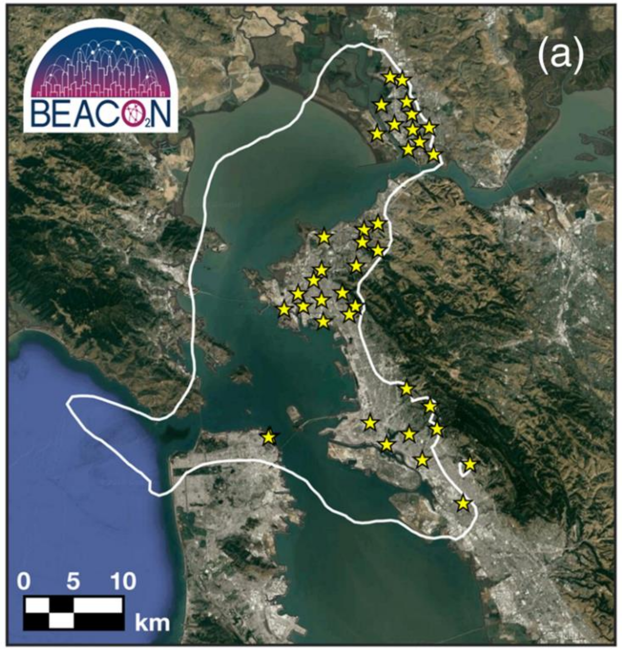


Figure 3. Spatial patterns of CO_2 fluxes in the San Francisco Bay Area. Panel (a) shows the average CO_2 fluxes for 6 weeks before shelter-in-place (2 February 2020 through 14 March 2020). Panel (b) shows the average over 6 weeks during shelter-in-place (22 March 2020 through 2 May 2020). Panel (c) is the difference. Black contour in all panels encompasses the top 40% of the total network influence (BEACO$_2$N Domain). Cross hatching indicates regions with low sensitivity to the BEACO$_2$N nodes.

We are aiming to recompute CO_2 emission fluxes using emulator for both before and during covid periods.

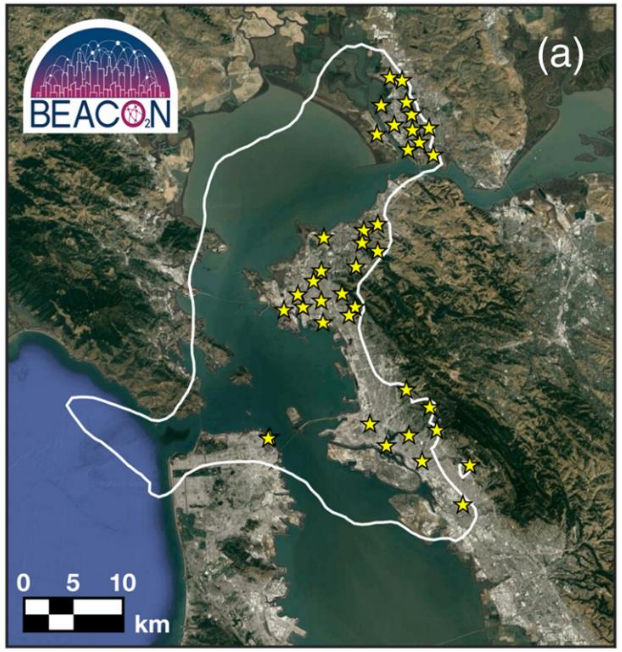
Case Study: Impacts of COVID-19 on urban CO_2 emissions (Turner et. al., 2020)



BEACO$_2$N Surface Network

- Hourly atmospheric CO_2 measurements
- Footprints computed with WRF-STILT
- Prior fluxes are taken from bottom-up inventories (adapted from McDonald et al., 2014; Turner et al., 2016; Turner et al. 2019)

Bayesian Inference (GHG emission estimation)



BEACO$_2$N Surface Network

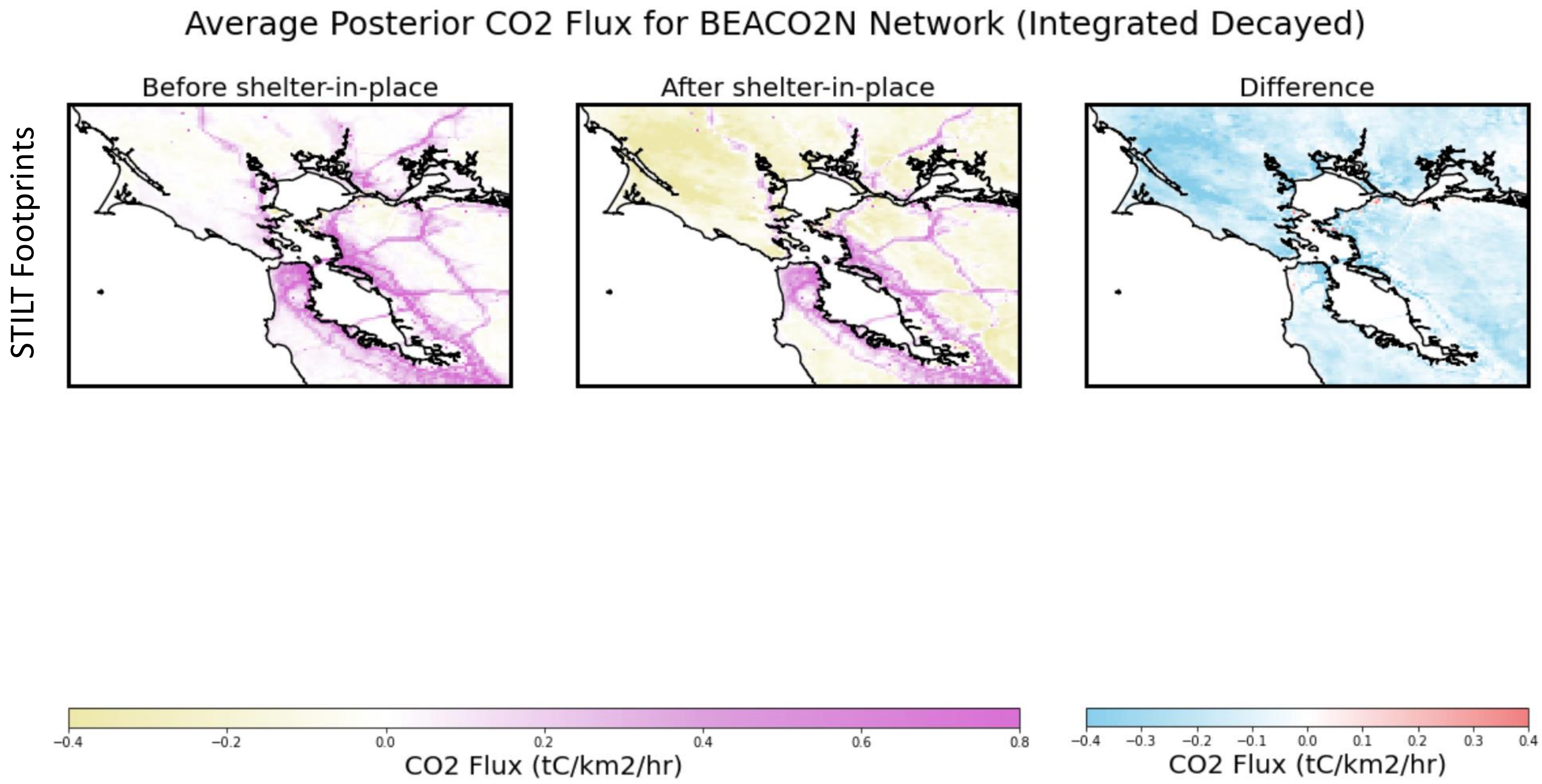
$$\mathbf{y} = \mathbf{Hx} + \in$$

Emissions

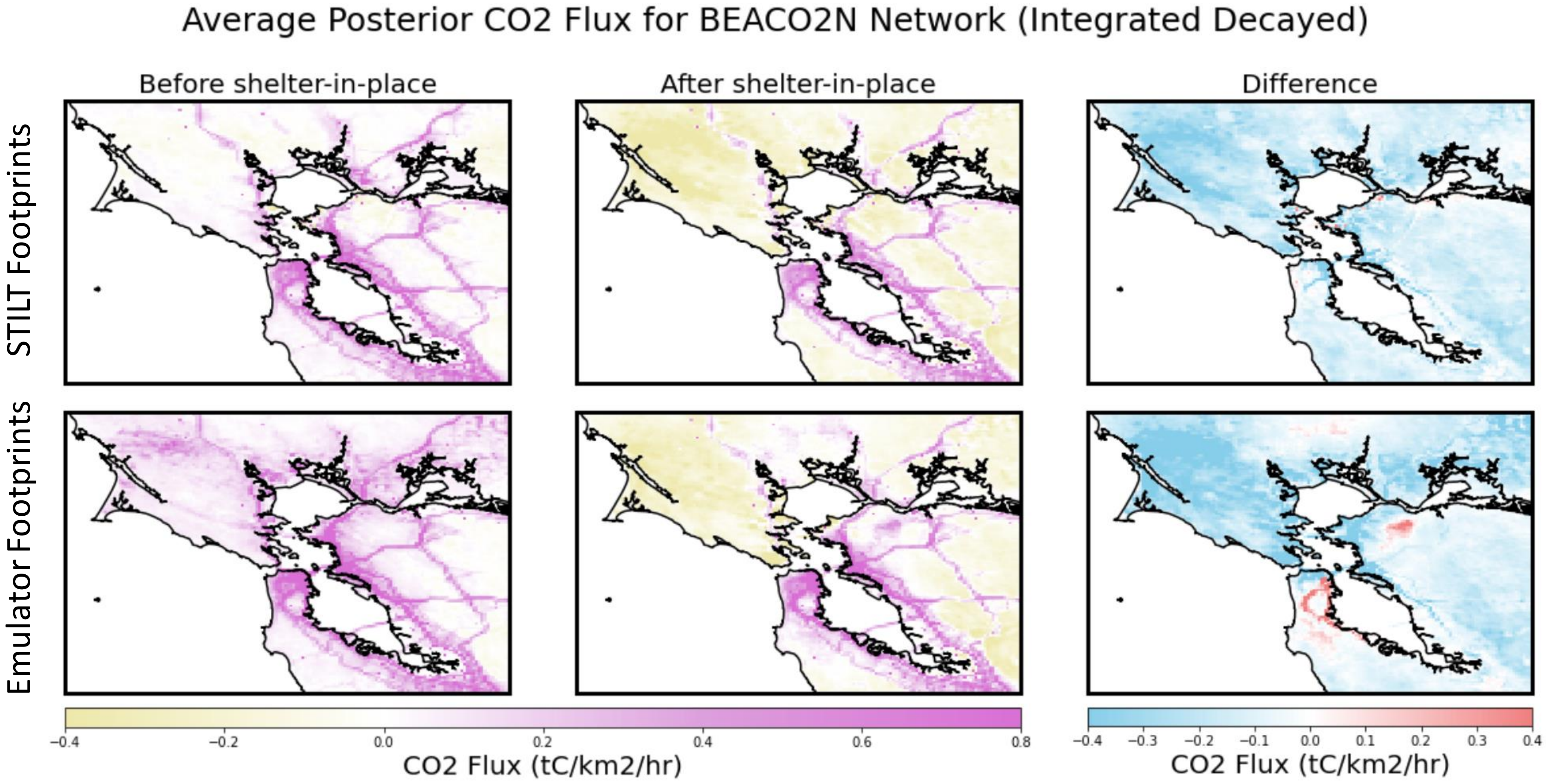
Error

*We need to compute footprints (**H**) to solve for posterior fluxes (**x**)*

Posterior Emission Fluxes (replicating Turner et. al., 2020)



Posterior Emission Fluxes (replicating Turner et. al., 2020)

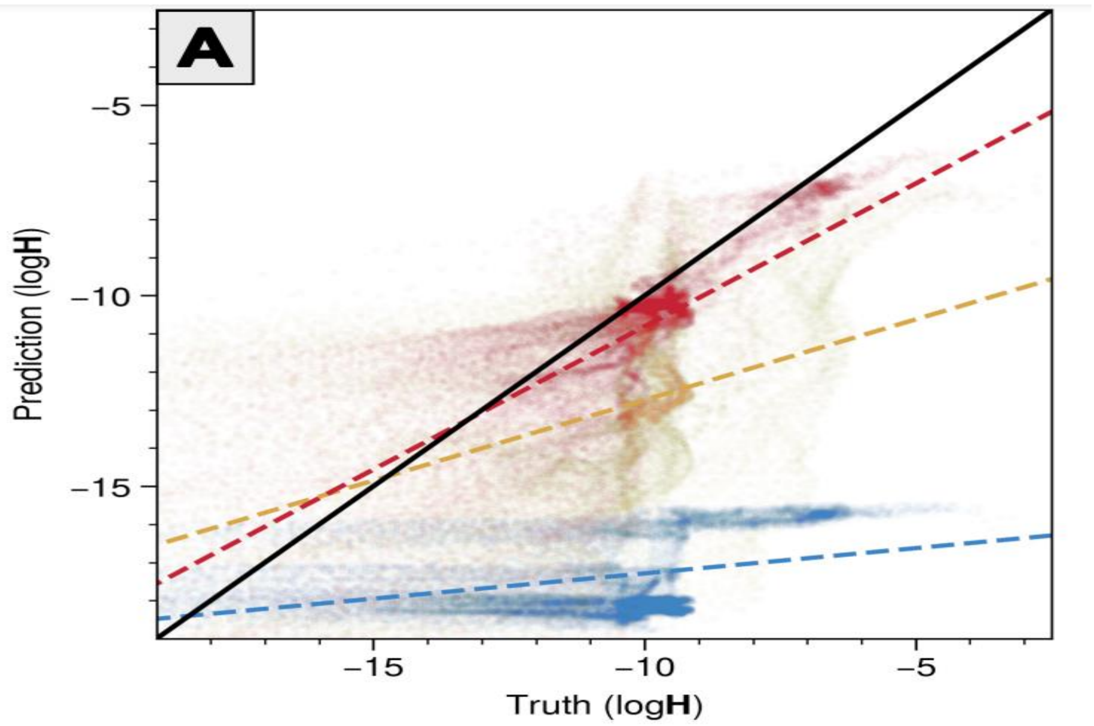


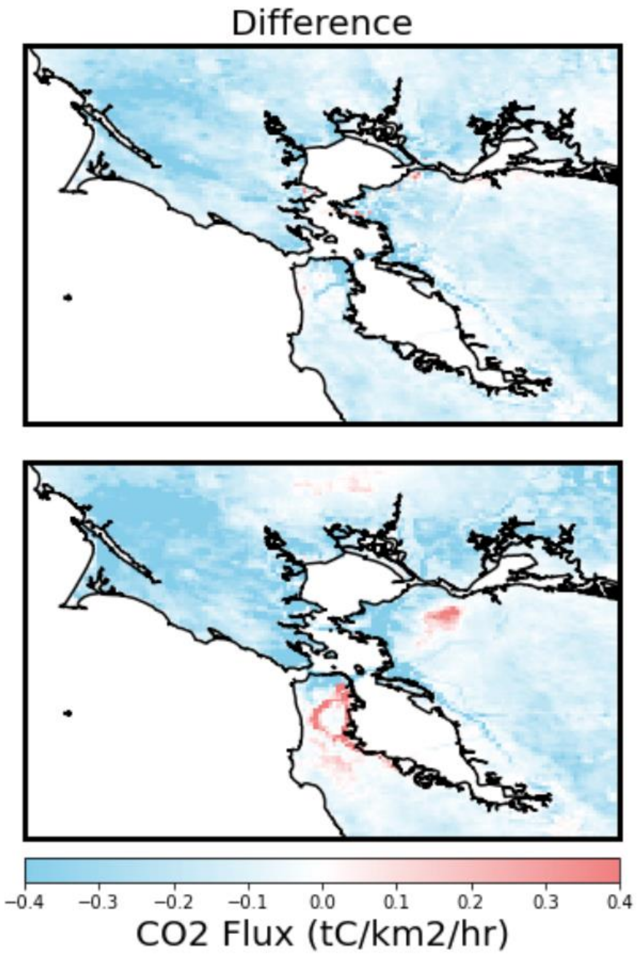
Posterior Emission Fluxes (replicating Turner et. al., 2020)

Average Posterior CO2 Flux for BEACO2N Network (Integrated Decayed)

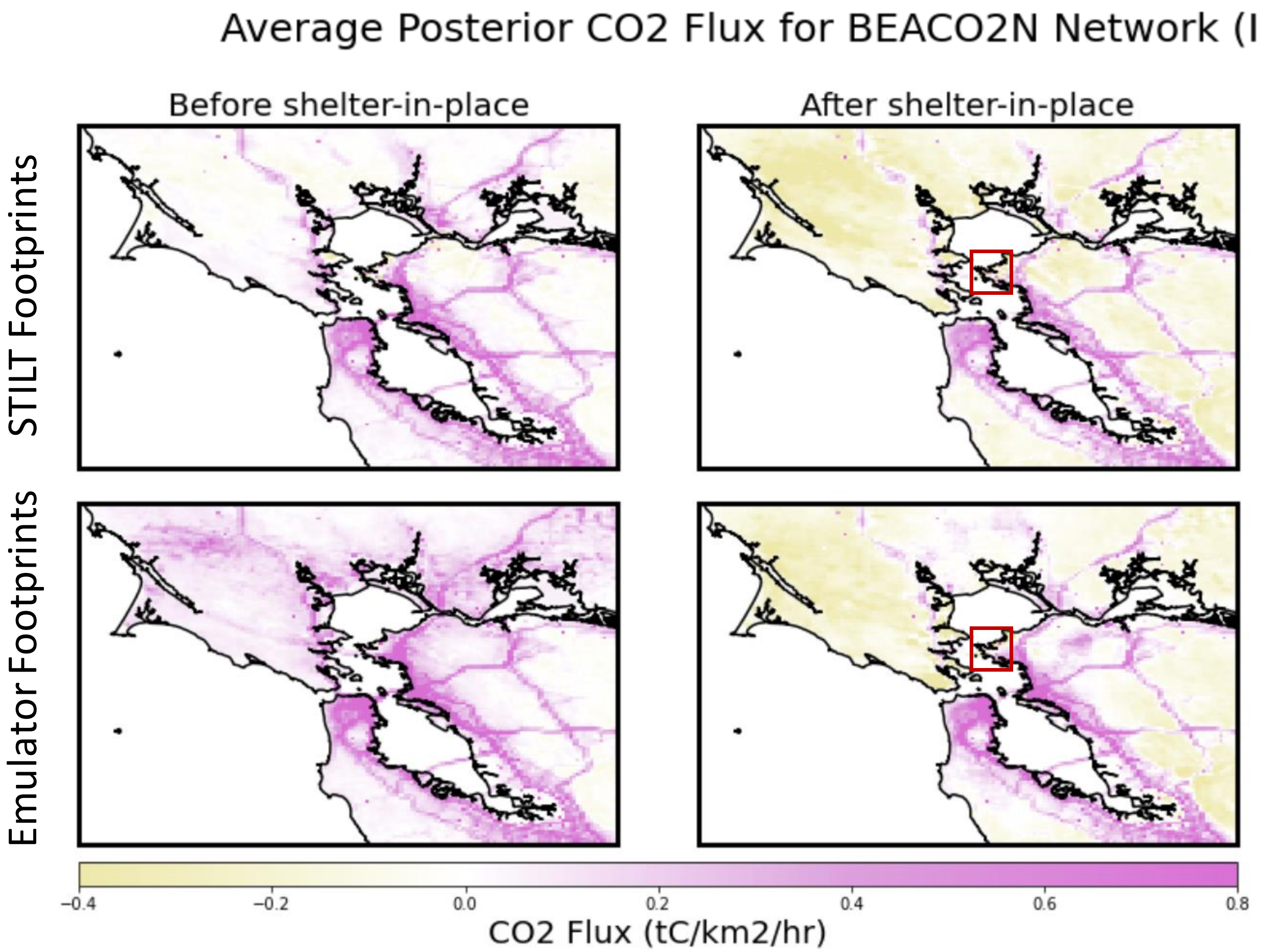
STILT Footprints

Emulator Footprints





Posterior Emission Fluxes (replicating Turner et. al., 2020)



Landsat true color image (March 18, 2019)

Computational cost analysis

Footprint (STILT)

> 1 footprint simulation ~2 hours
> Footprints for observations of a single day (~700 observations)

Bayesian Inference

> Computing emission fluxes for 1 day
- Sequential HB matrix multiplication ~135 minutes
- Parallel HB matrix multiplication ~15 minutes

Total Time

> Computing emission fluxes (Feb –April 2020)
- Time ~16 days 6 hours
- Storage ~403 GB

Footprint (machine learning emulator)

> 1 footprint simulation ~0.6 seconds
> Footprints for observations of a single day (~700 observations)
- Can be computed on fly

Bayesian Inference

> Computing emission fluxes for 1 day
- Sequential HB matrix multiplication ~135 minutes
- Parallel HB matrix multiplication ~15 minutes

Total Time

> Computing emission fluxes (Feb – April 2020)
- Time ~13.5 hours (can be optimized to 6 hours 15 minutes)
- Storage ~3 GB

- Construction of footprints is 50 times faster with emulator than STILT model
- The parallel computation of HB matrix is ~13.5 times faster than sequential approach
- The total time taken for the end-to-end simulation is currently ~37 times faster with emulator

Computational cost analysis

Footprint (STILT)

- 1 footprint simulation ~2 hours
- Footprints for observations of a single day (~700 observations)

Bayesian Inference

- Computing emission fluxes for 1 day
  - Sequential HB matrix multiplication ~135 minutes
- Computing emission fluxes (Feb –April 2020)
  - Time ~16 days 6 hours
  - Storage ~403 GB

Footprint (machine learning emulator)

- 1 footprint simulation ~0.6 seconds
- Footprints for observations of a single day (~700 observations)
  - Can be computed on fly

Bayesian Inference

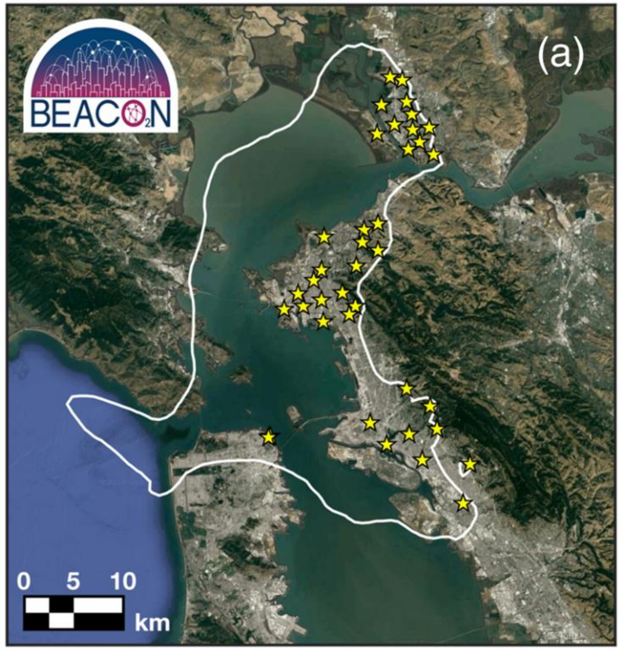
- Computing emission fluxes for 1 day
  - Sequential HB matrix multiplication ~135 minutes
- Computing emission fluxes (Feb – April 2020)
  - Time ~13.5 hours (can be optimized to 6 hours 15 minutes)
  - Storage ~3 GB

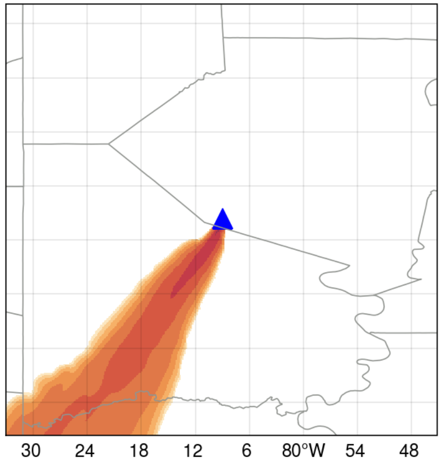
- Construction of footprints is 50 times faster with emulator than STILT model
- The parallel computation of HB matrix is ~13.5 times faster than sequential approach
- The total time taken for the end-to-end simulation is currently ~37 times faster with emulator

Computing emission fluxes using STILT footprints is 134 times storage expensive

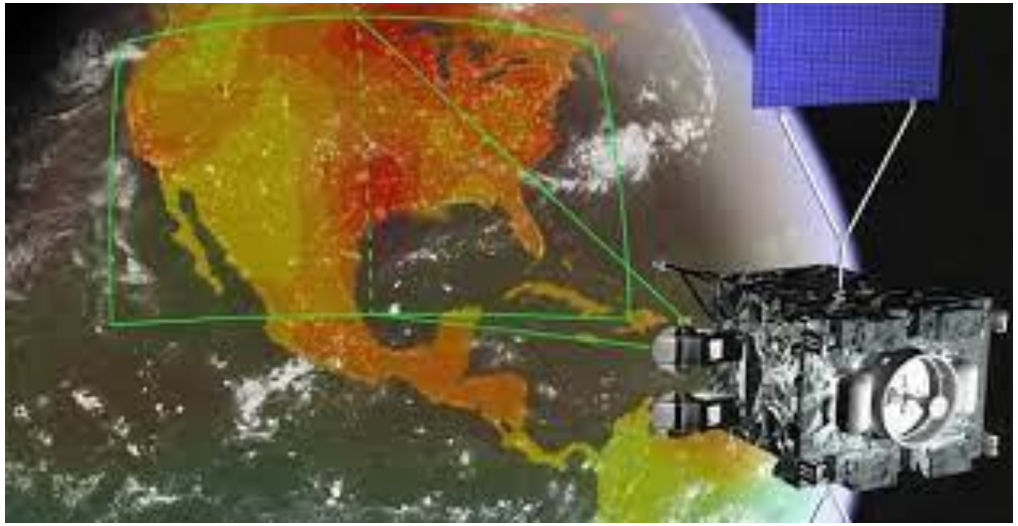
Current and Future Work

> **Generalized training for emulator**





Footprints at 200m resolution
(Funded by EDF)

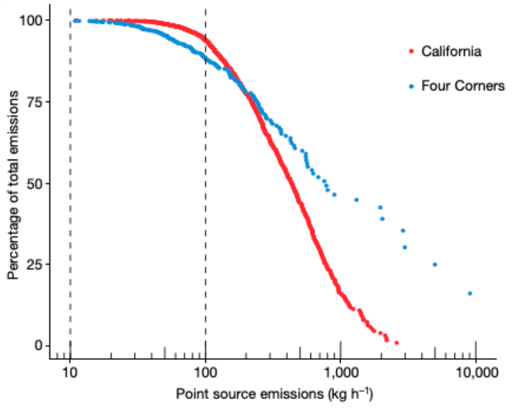


TROPOMI GHGSAT MethaneSAT™

Extend the machine learning model to footprints for satellites observations

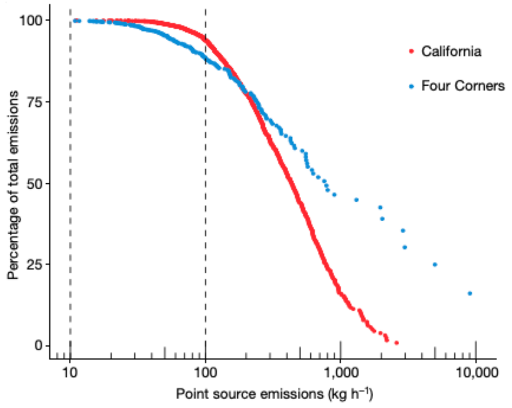
Summary

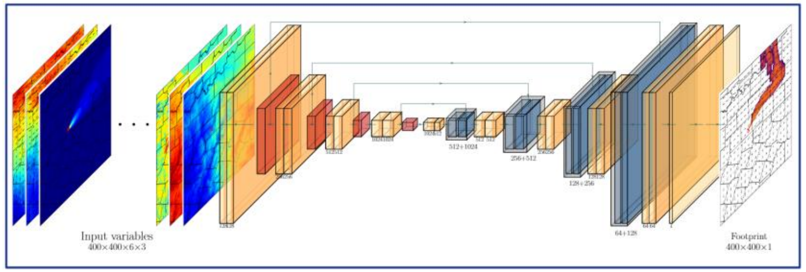
> Emissions from point sources dominate the total emission budget
> Next-generation observing systems provide dense coverage of the GHGs



Summary

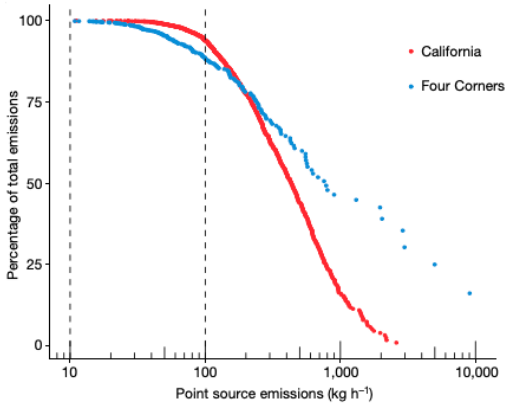
- Emissions from point sources dominate the total emission budget
- Next-generation observing systems provide dense coverage of the GHGs
- Computational bottlenecks limit our understanding of point sources
- We propose a deep learning-based model which can efficiently construct footprints in near-real-time

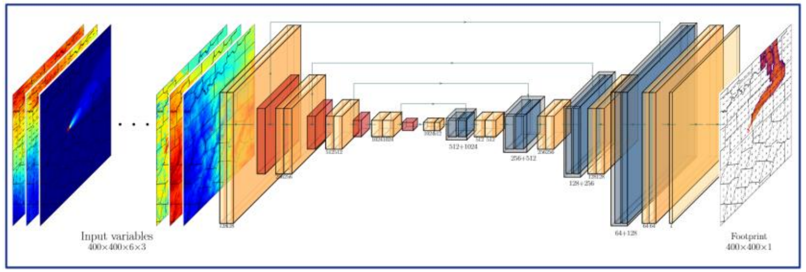


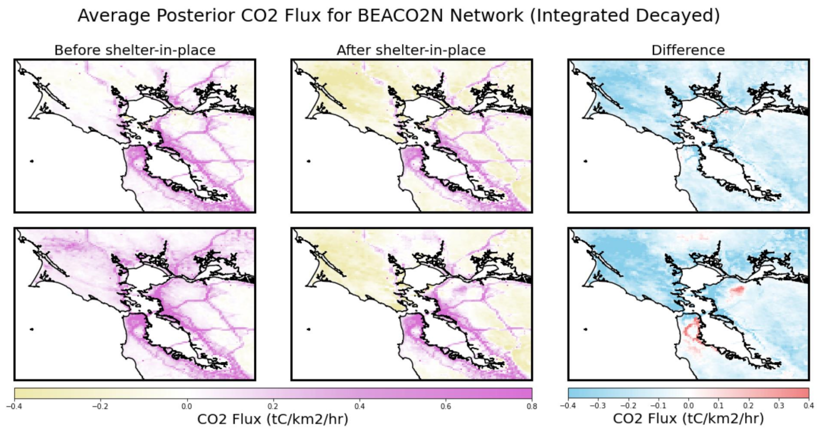


Summary

- Emissions from point sources dominate the total emission budget
- Next-generation observing systems provide dense coverage of the GHGs
- Computational bottlenecks limit our understanding of point sources
- We propose a deep learning-based model which can efficiently construct footprints in near-real-time
- Footprints from emulator can be used to estimate GHG emissions







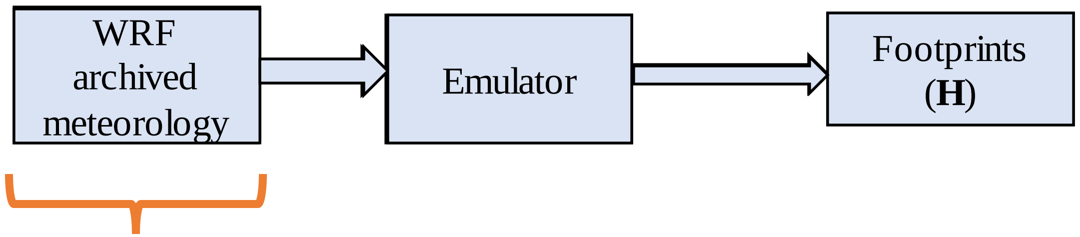
Supplementary Slides

How will this be useful?

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \in$$

- Computing **H** through U-Net model will be near-real-time
- We do not need to store footprints anymore
- We can quantify latent biases in the meteorological data

We will be computing **H** through a deep learning-based model



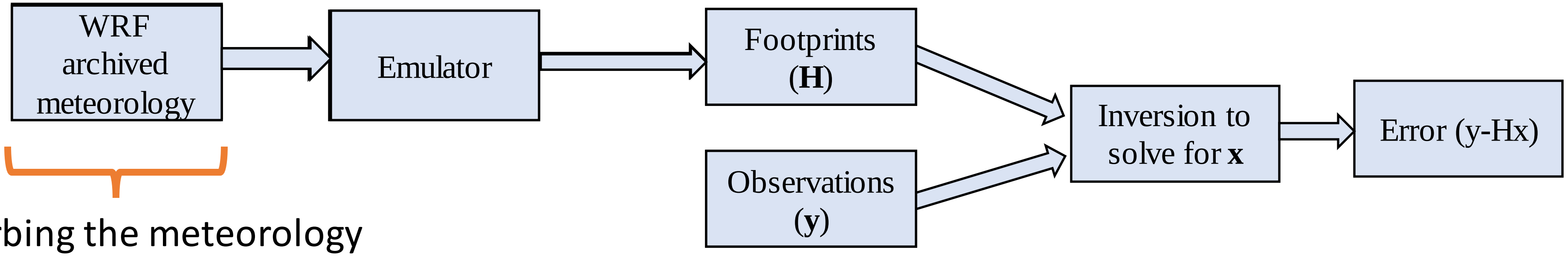
Perturbing the meteorology
(for e.g. PBL height)

How will this be useful?

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \in$$

We will be computing **H** through a deep learning-based model

- Computing **H** through U-Net model will be near-real-time
- We do not need to store footprints anymore
- We can quantify latent biases in the meteorological data



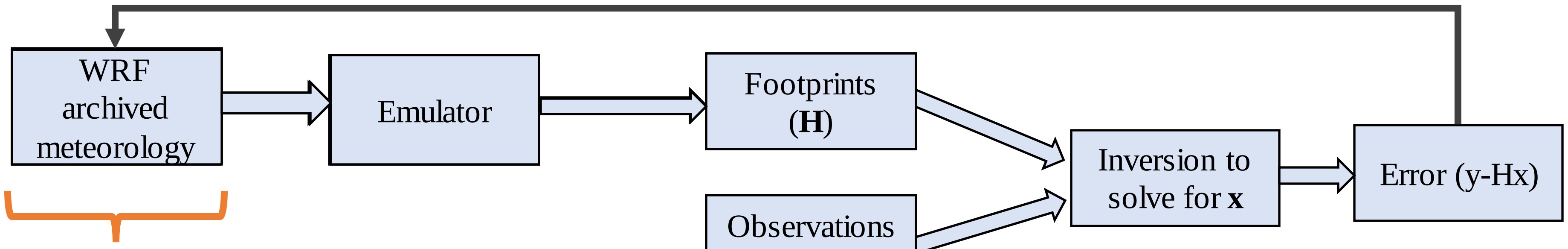
Perturbing the meteorology
(for e.g. PBL height)

How will this be useful?

$$\mathbf{y} = \mathbf{Hx} + \in$$

- Computing **H** through U-Net model will be near-real-time
- We do not need to store footprints anymore
- We can quantify latent biases in the meteorological data

We will be computing **H** through a deep learning-based model



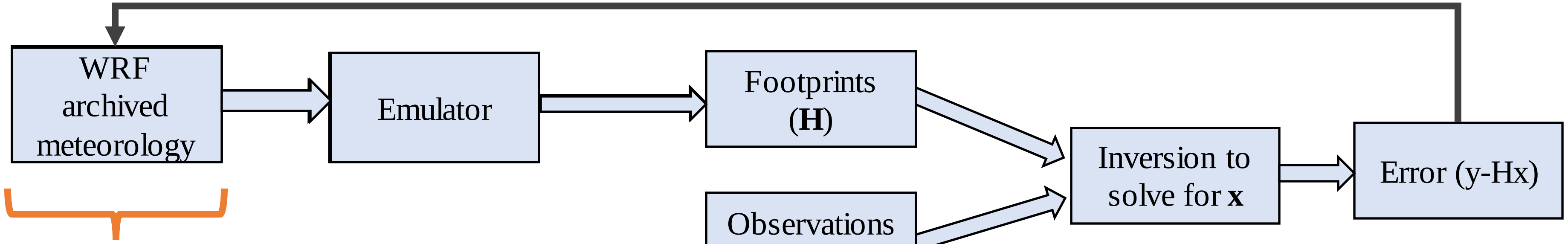
Perturbing the meteorology
(for e.g. PBL height)

How will this be useful?

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \in$$

We will be computing **H** through a deep learning-based model

- Computing **H** through U-Net model will be near-real-time
- We do not need to store footprints anymore
- We can quantify latent biases in the meteorological data
- Inverting for emissions can become more accurate (iterative methods such as MCMC methods can be used)



Perturbing the meteorology (for e.g. PBL height)

Non-Gaussian errors

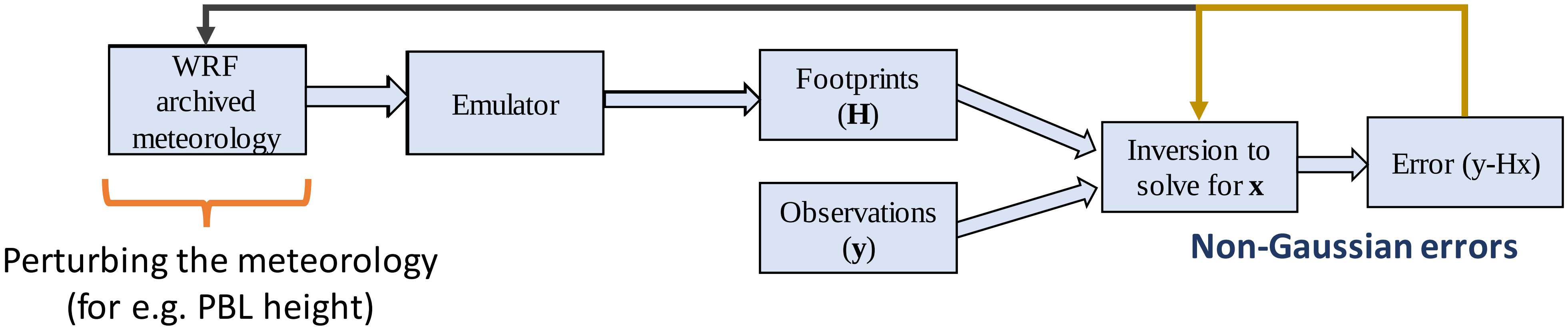
How will this be useful?

- Computing **H** through U-Net model will be near-real-time
- We do not need to store footprints anymore
- We can quantify latent biases in the meteorological data
- Inverting for emissions can become more accurate (iterative methods such as MCMC methods can be used)

$$\mathbf{y} = \mathbf{Hx} + \in$$

We will be computing **H** through a deep learning-based model

Iterative methods for inversion



$$\hat{x} = x_a + (HB)^T(HBH^T + R)^{-1}(y - Hx_a)$$

Computational cost analysis

Footprint (STILT)

- 1 footprint simulation ~2 hours
- Footprints for observations of a single day (~700 observations)
  - Sequentially ~ 1400 hours (58 days)
  - Parallel on 32 cores machine ~44 hours (2 days)
  - Parallel on 10 nodes ~4 hours
- Footprints for observations (Feb – April 2020)
  - Parallel on 10 nodes (32 cores machines) ~16 days
  - Storage cost ~400GB

Bayesian Inference

- Computing emission fluxes for 1 day
  - Sequential HB matrix multiplication ~135 minutes

Footprint (machine learning emulator)

- 1 footprint simulation ~0.6 seconds
- Footprints for observations of a single day (~700 observations)
  - Predictions over GPU ~5 minutes (4.5 mins for loading the data and rest 30 seconds for model predictions)
  - Predictions over CPU ~20 minutes (4.5 mins for loading the data and rest 15.5 minutes for model prediction)
- Footprints for observations (Feb – April 2020)
  - Predictions over GPU ~7.5 hours
  - If we optimize the data loading ~15 minutes
  - Can be computed on fly

Bayesian Inference

- Computing emission fluxes for 1 day
  - Sequential HB matrix multiplication ~135 minutes

$$\hat{x} = x_a + (HB)^T(HBH^T + R)^{-1}(y - Hx_a)$$

Computational cost analysis

Footprint (STILT)

- 1 footprint simulation ~2 hours
- Footprints for observations of a single day (~700 observations)
  - Sequentially ~ 1400 hours (58 days)
  - Parallel on 32 cores machine ~44 hours (2 days)
  - Parallel on 10 nodes ~4 hours
- Footprints for observations (Feb – April 2020)
  - Parallel on 10 nodes (32 cores machines) ~16 days
  - Storage cost ~400GB

Bayesian Inference

- Computing emission fluxes for 1 day
  - Sequential HB matrix multiplication ~135 minutes
  - Parallel HB matrix multiplication ~10 minutes

Footprint (machine learning emulator)

- 1 footprint simulation ~0.6 seconds
- Footprints for observations of a single day (~700 observations)
  - Predictions over GPU ~5 minutes (4.5 mins for loading the data and rest 30 seconds for model predictions)
  - Predictions over CPU ~20 minutes (4.5 mins for loading the data and rest 15.5 minutes for model prediction)
- Footprints for observations (Feb – April 2020)
  - Predictions over GPU ~7.5 hours
  - If we optimize the data loading ~15 minutes
  - Can be computed on fly

Bayesian Inference

- Computing emission fluxes for 1 day
  - Sequential HB matrix multiplication ~135 minutes
  - Parallel HB matrix multiplication ~10 minutes

$$\hat{x} = x_a + (HB)^T(HBH^T + R)^{-1}(y - Hx_a)$$

Computational cost analysis

Footprint (STILT)

- 1 footprint simulation ~2 hours
- Footprints for observations of a single day (~700 observations)
  - Sequentially ~ 1400 hours (58 days)
  - Parallel on 32 cores machine ~44 hours (2 days)
  - Parallel on 10 nodes ~4 hours
- Footprints for observations (Feb – April 2020)
  - Parallel on 10 nodes (32 cores machines) ~16 days
  - Storage cost ~400GB

Bayesian Inference

- Computing emission fluxes for 1 day
  - Sequential HB matrix multiplication ~135 minutes
  - Parallel HB matrix multiplication ~10 minutes

Footprint (machine learning emulator)

- 1 footprint simulation ~0.6 seconds
- Footprints for observations of a single day (~700 observations)
  - Predictions over GPU ~5 minutes (4.5 mins for loading the data and rest 30 seconds for model predictions)
  - Predictions over CPU ~20 minutes (4.5 mins for loading the data and rest 15.5 minutes for model prediction)
- Footprints for observations (Feb – April 2020)
  - Predictions over GPU ~7.5 hours
  - If we optimize the data loading ~15 minutes
  - Can be computed on fly

Bayesian Inference

- Computing emission fluxes for 1 day
  - Sequential HB matrix multiplication ~135 minutes
  - Parallel HB matrix multiplication ~10 minutes

The parallel HB matrix multiplication is 13.5 times faster than sequential HB matrix multiplication

Computational cost analysis

Footprint (STILT)

> 1 footprint simulation ~2 hours

> Footprints for observations of a single day (~700 observations)
- Sequentially ~ 1400 hours (58 days)
- Parallel on 32 cores machine ~44 hours (2 days)
- Parallel on 10 nodes ~4 hours

> Footprints for observations (Feb – April 2020)
- Parallel on 10 nodes (32 cores machines) ~16 days
- Storage cost ~400GB

Bayesian Inference

> Computing emission fluxes for 1 day
- Sequential HB matrix multiplication ~135 minutes
- Parallel HB matrix multiplication ~10 minutes

Total Time

> Computing emission fluxes (Feb –April 2020)
- Time ~16 days 3 hours
- Storage ~403 GB

Footprint (machine learning emulator)

> 1 footprint simulation ~0.6 seconds

> Footprints for observations of a single day (~700 observations)
- Predictions over GPU ~5 minutes (4.5 mins for loading the data and rest 30 seconds for model predictions)
- Predictions over CPU ~20 minutes (4.5 mins for loading the data and rest 15.5 minutes for model prediction)

> Footprints for observations (Feb – April 2020)
- Predictions over GPU ~7.5 hours
- If we optimize the data loading ~15 minutes
- Can be computed on fly

Bayesian Inference

> Computing emission fluxes for 1 day
- Sequential HB matrix multiplication ~135 minutes
- Parallel HB matrix multiplication ~10 minutes

Total Time

> Computing emission fluxes (Feb – April 2020)
- Time ~10.5 hours (can be optimized to 3 hours 15 minutes)
- Storage ~3 GB (storage for inversion results)