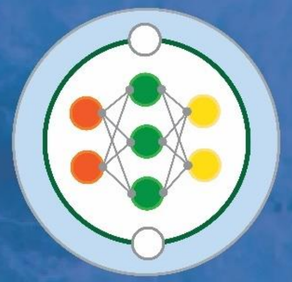


m2lines.github.io

Building Ocean Climate Emulators

Adam Subel and Laure Zanna

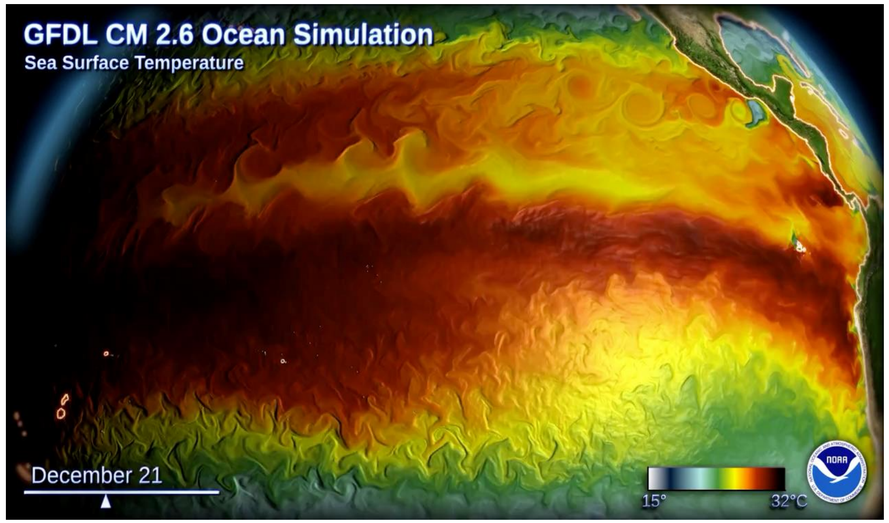
Courant Institute of Mathematical Sciences, New York University

Paper

https://arxiv.org/abs/2402.04342

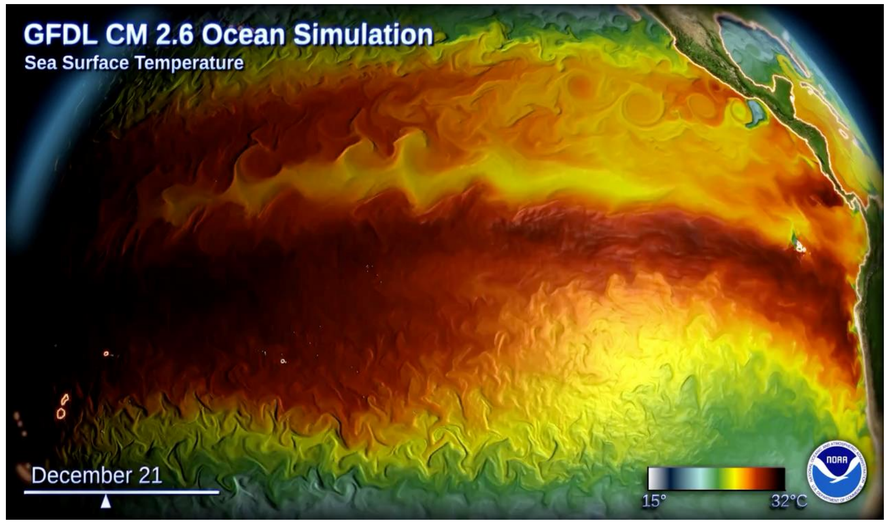
Why Emulation?

Numerical Models



Why Emulation?

Numerical Models



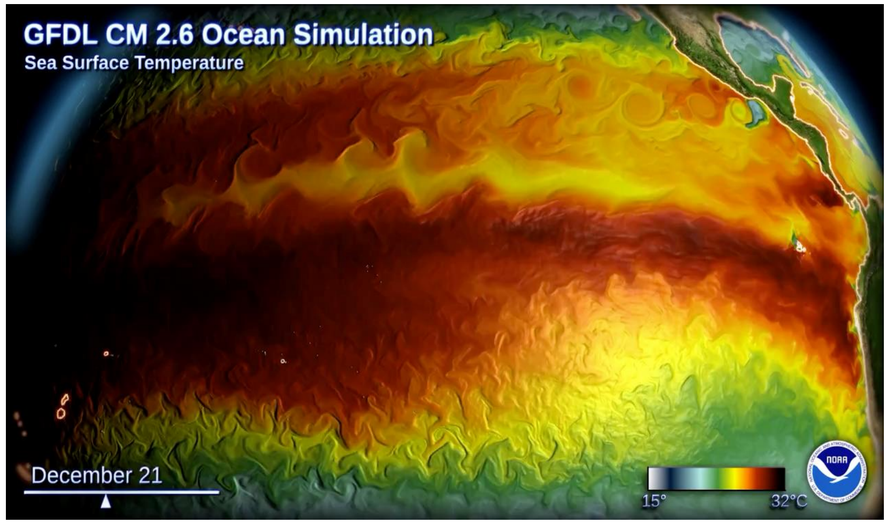
Trustworthy Data

\+ Full range of physics
Response to external forcings

— Expensive!!

Why Emulation?

Numerical Models

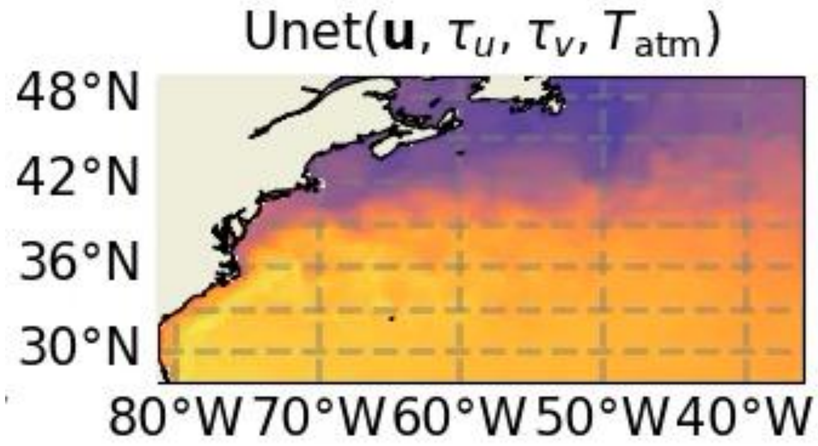


Trustworthy Data

\+ Full range of physics
Response to external forcings

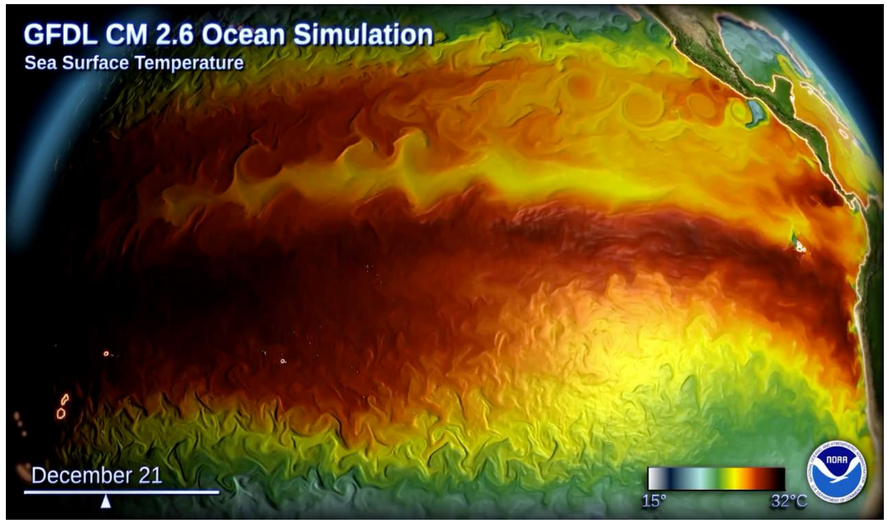
\- Expensive!!

ML Emulation

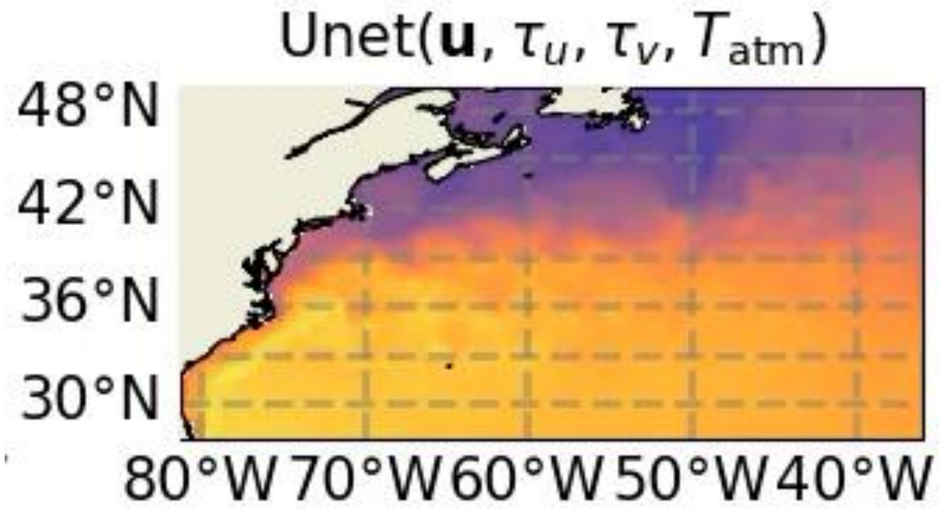


Why Emulation?

Numerical Models



ML Emulation

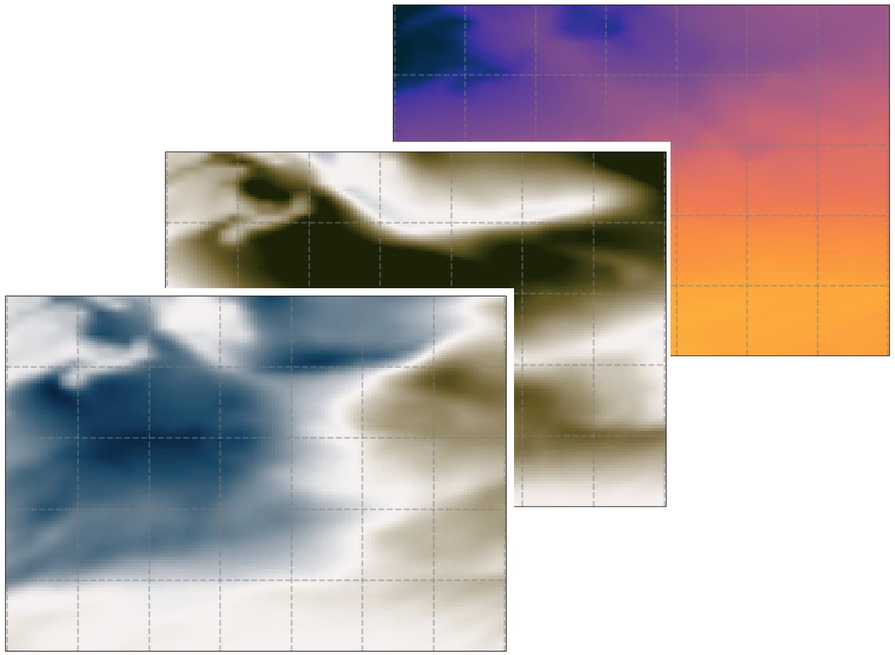


~ 200 x Faster

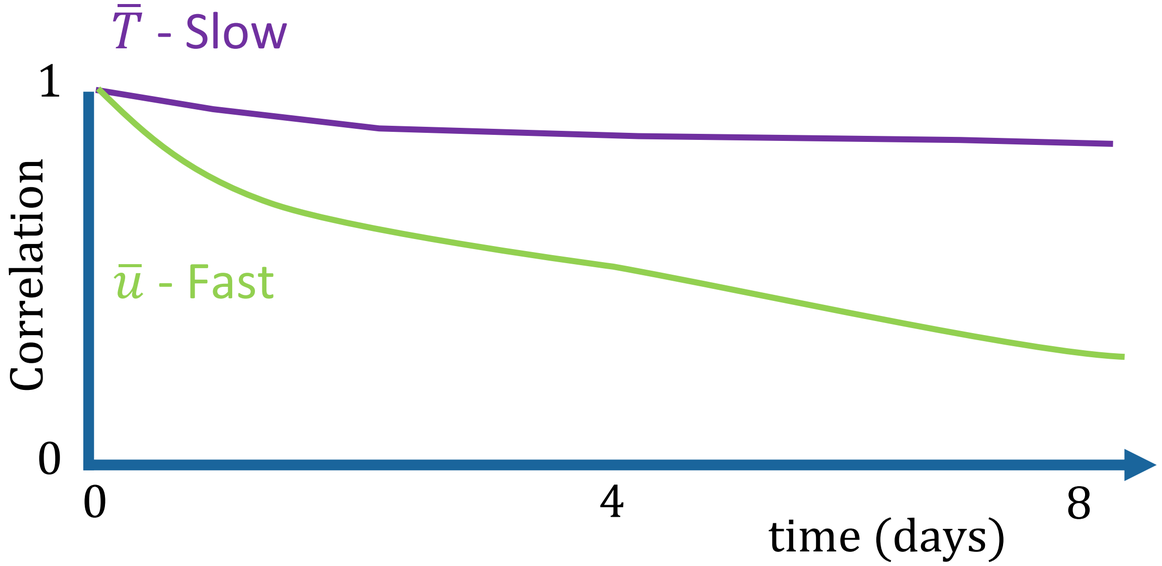
Runs on 1 GPU

How to Build an Ocean Emulator?

What boundary information should you use?



What time-step, or training horizon does the model need?



Formulating the Problem

Input:

full fields $\mathbf{\Phi} = (u, v, T)$

boundary halo

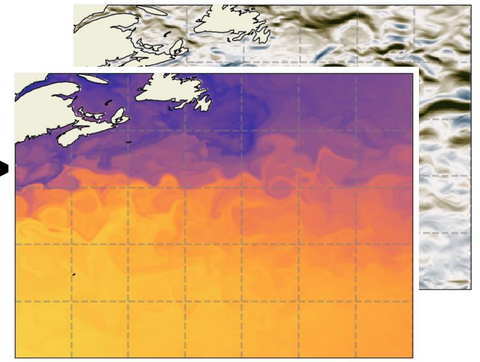
Forcing

$\boldsymbol{\tau} = (\tau_u, \tau_v, \bar{T}_{\text{atm}})$

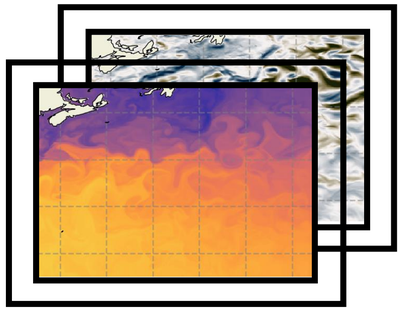
Emulator

Output:

full fields $\mathbf{\Phi} = (u, v, T)$



$\mathbf{\Phi} = (u, v, T)$



Pass back to the network

+

Replace the boundary of the output with the boundary halo

Mixed Loss

MSE + Kinetic energy

N-recurrent passes during training

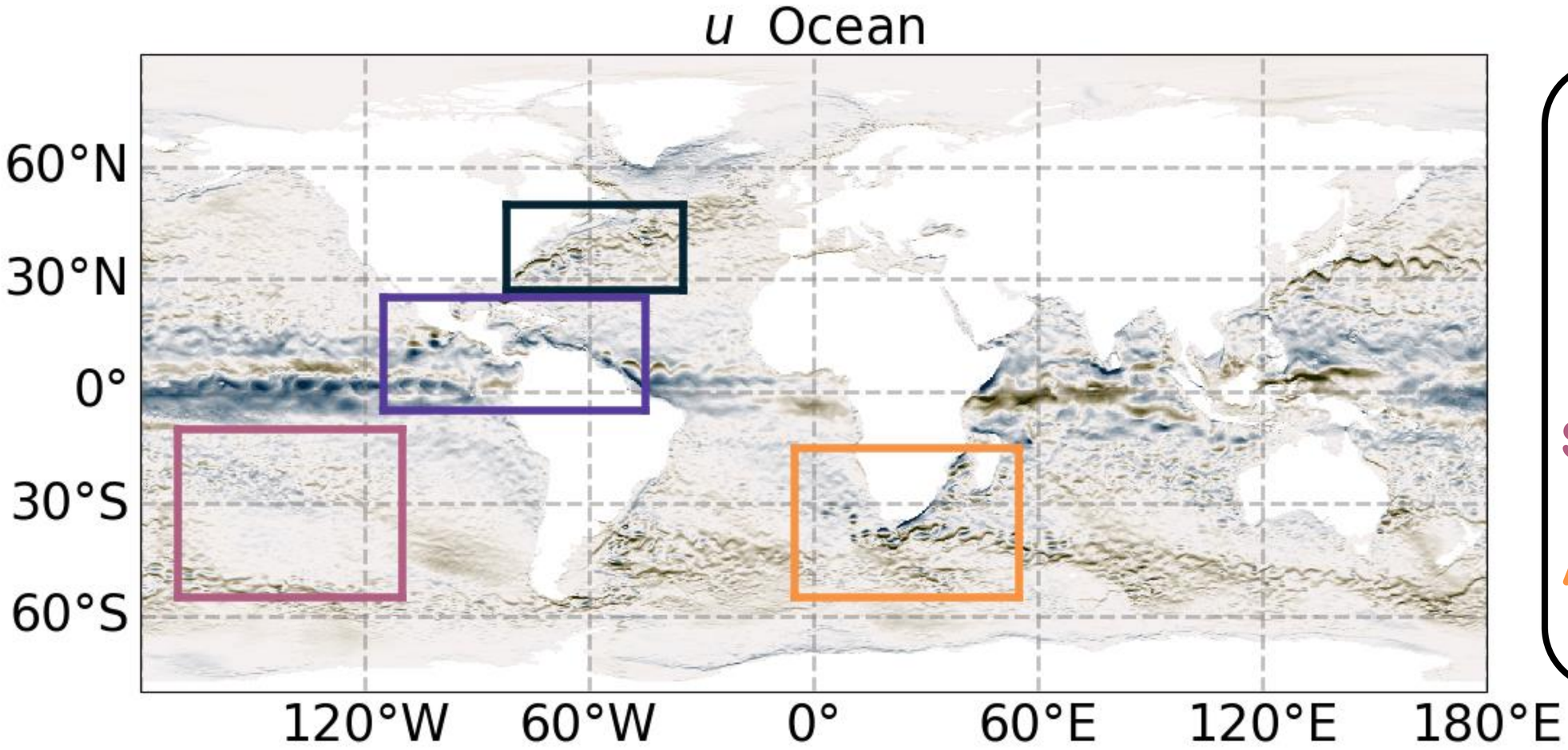
$$\mathcal{L}^{(N)} = \sum_{n=1}^{N} \lambda_n \left((1-\alpha)\, \mathcal{L}_{\text{mse}}^{(n)} + \alpha \mathcal{L}_{\text{KE}}^{(n)} \right)$$

$$\tilde{\mathbf{\Phi}}_{t+n\Delta t} = F_\theta^{(n)}(\mathbf{\Phi}_t, \boldsymbol{\tau}_t),$$

$$\mathcal{L}_{\text{mse}} = \left\| \mathbf{\Phi}_{t+n\Delta t} - F_\theta^{(n)}(\mathbf{\Phi}_t, \boldsymbol{\tau}_t) \right\|_2^2$$

$$\mathcal{L}_{\text{KE}} = \left\| (u_{t+n\Delta t}^2 + v_{t+n\Delta t}^2) - (\tilde{u}_{t+n\Delta t}^2 + \tilde{v}_{t+n\Delta t}^2) \right\|_2^2$$

Regions Explored



Gulf Stream

Tropics

South Pacific

African Cape

Incorporating Boundary Information

Input:

full fields

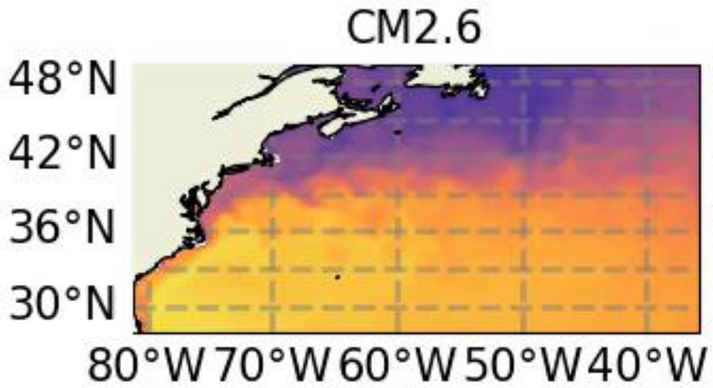
boundary halo

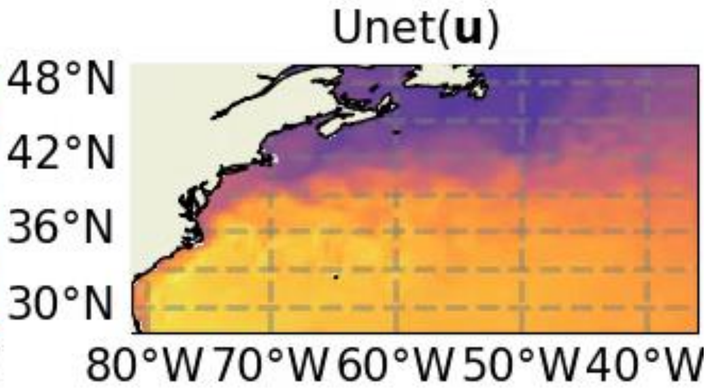
$\mathbf{\Phi} = (u, v, t)$

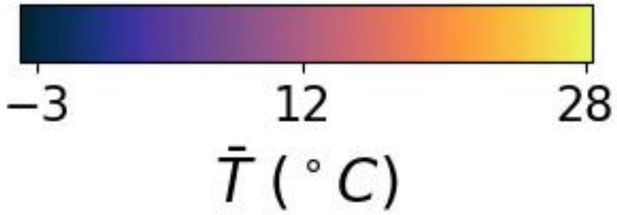
Emulator

Forcing

$\boldsymbol{\tau} = (\tau_u, \tau_v, T_{atm})$







Incorporating Boundary Information

Input:

full fields

boundary halo

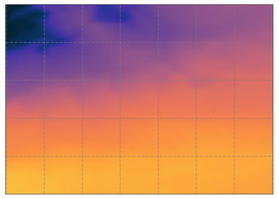
$\mathbf{\Phi} = (u, v, t)$

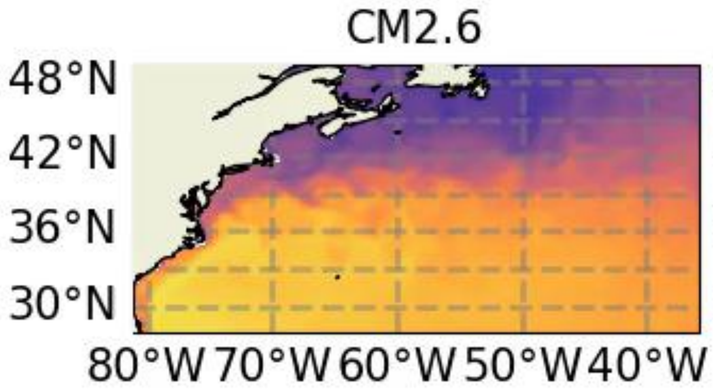
Emulator

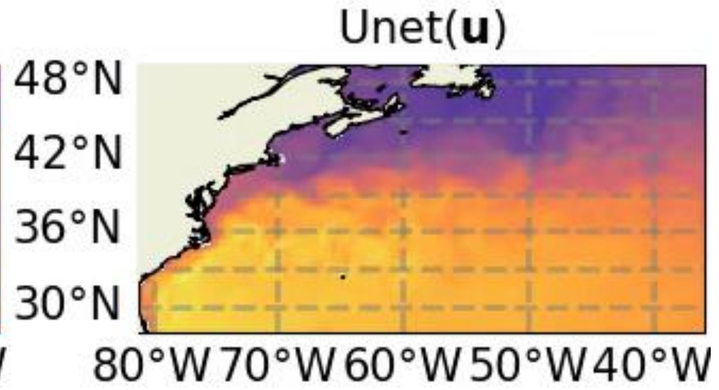
Forcing

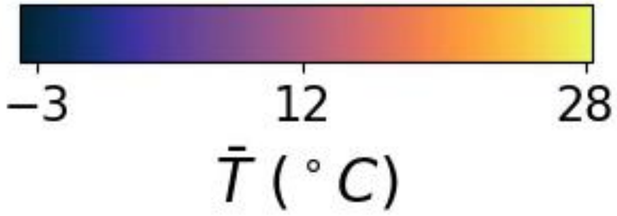
$\boldsymbol{\tau} = (\tau_u, \tau_v, T_{atm})$

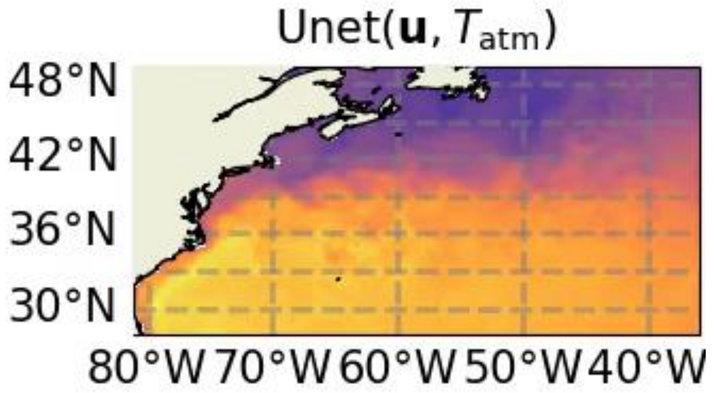
+











Incorporating Boundary Information

Input:

full fields

boundary halo

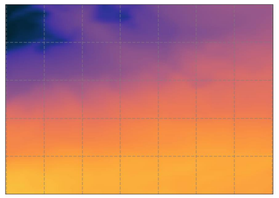
$\mathbf{\Phi} = (u, v, t)$

Emulator

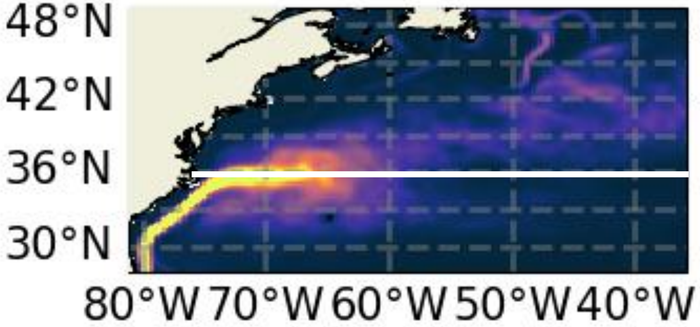
Forcing

$\boldsymbol{\tau} = (\tau_u, \tau_v, T_{atm})$

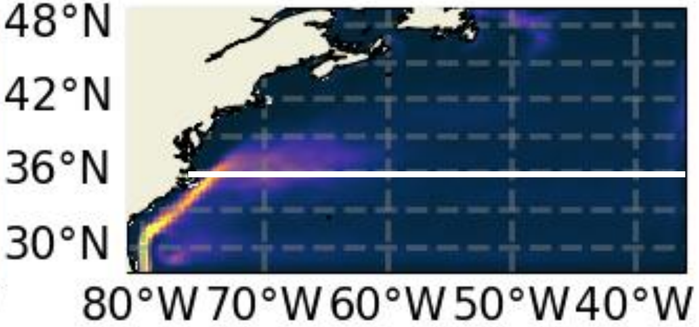
+

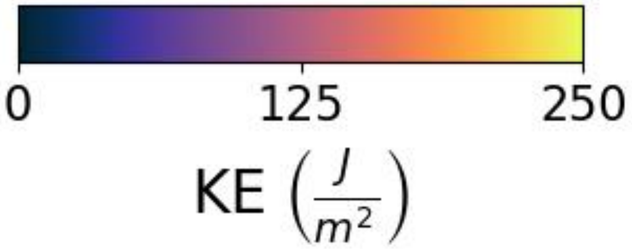


CM2.6

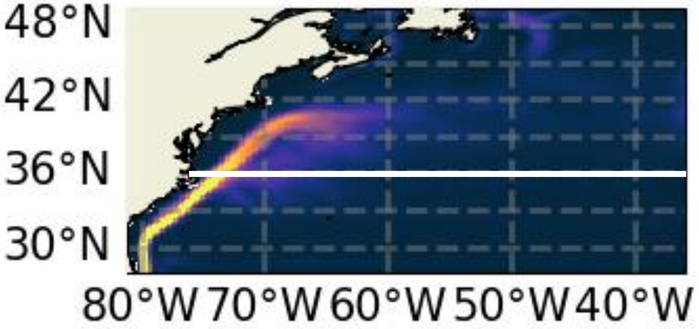


Unet(**u**)





Unet($\mathbf{u}, T_{atm}$)



Incorporating Boundary Information

Input:

full fields

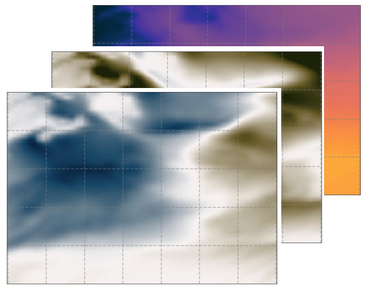
boundary halo

$\mathbf{\Phi} = (u, v, t)$

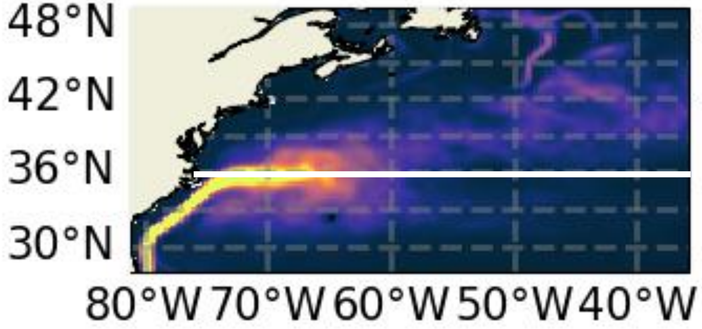
Emulator

Forcing

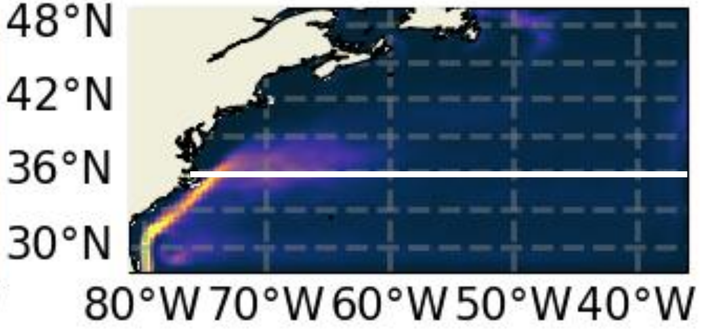
$\boldsymbol{\tau} = (\tau_u, \tau_v, T_{atm})$



CM2.6

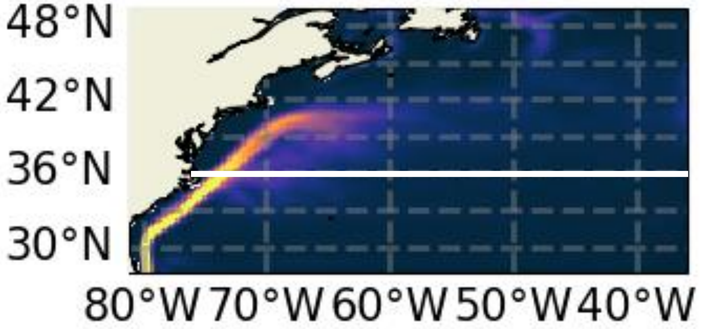


Unet(**u**)

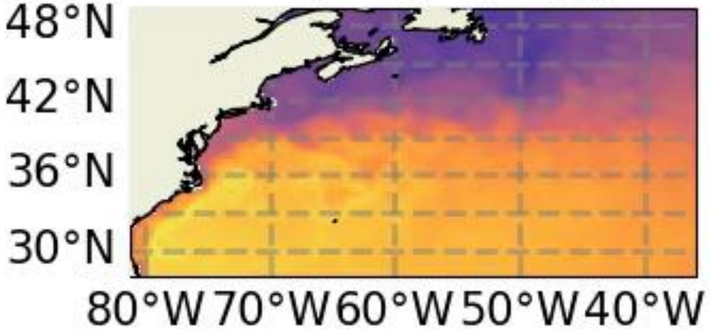


KE $\left(\frac{J}{m^2}\right)$

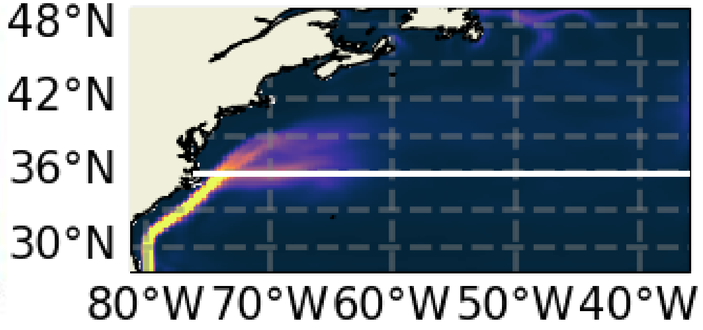
Unet(**u**, T_{atm})



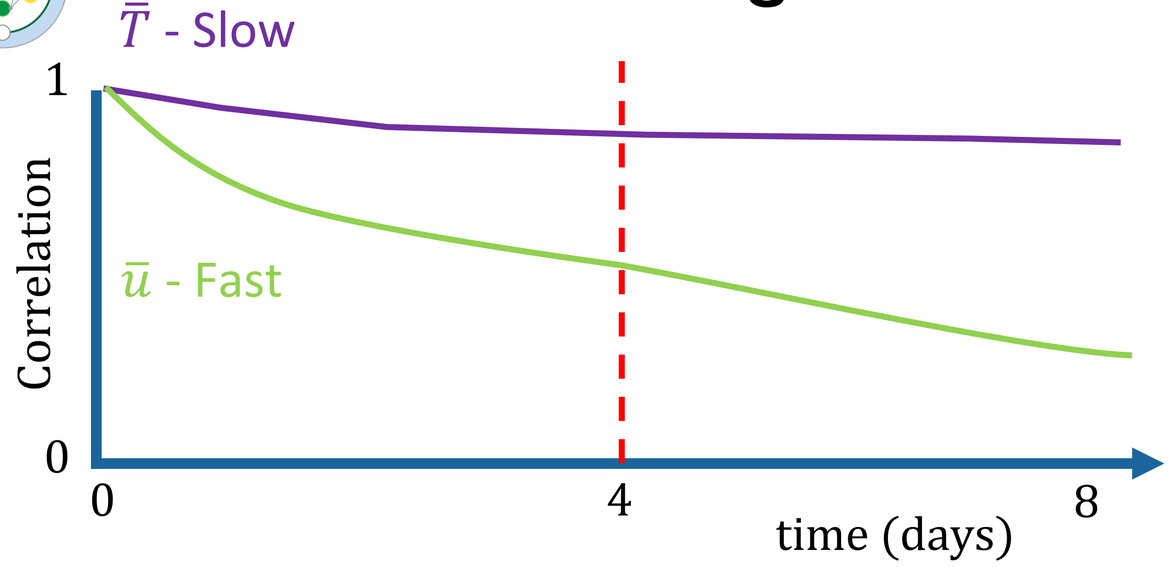
Unet(**u**, τ_u, τ_v, T_{atm})

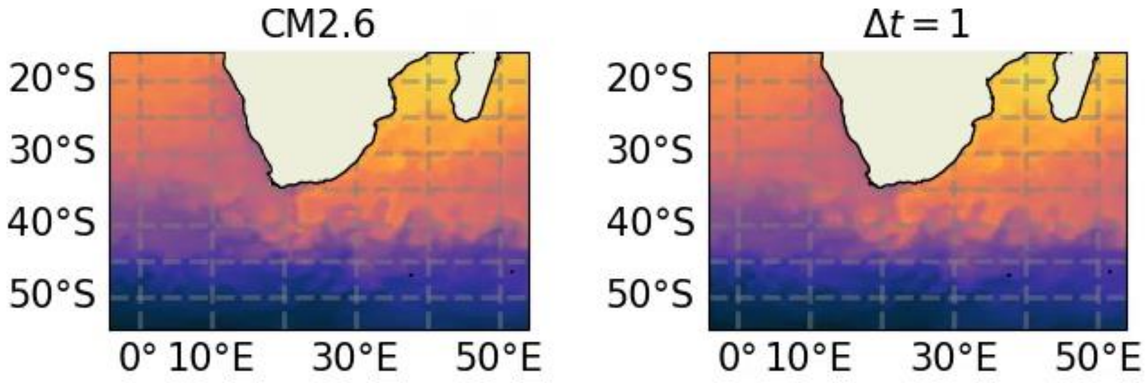


Unet(**u**, τ_u, τ_v, T_{atm})



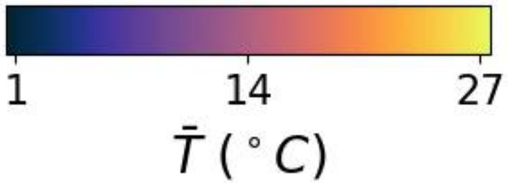
Extending the Training Window



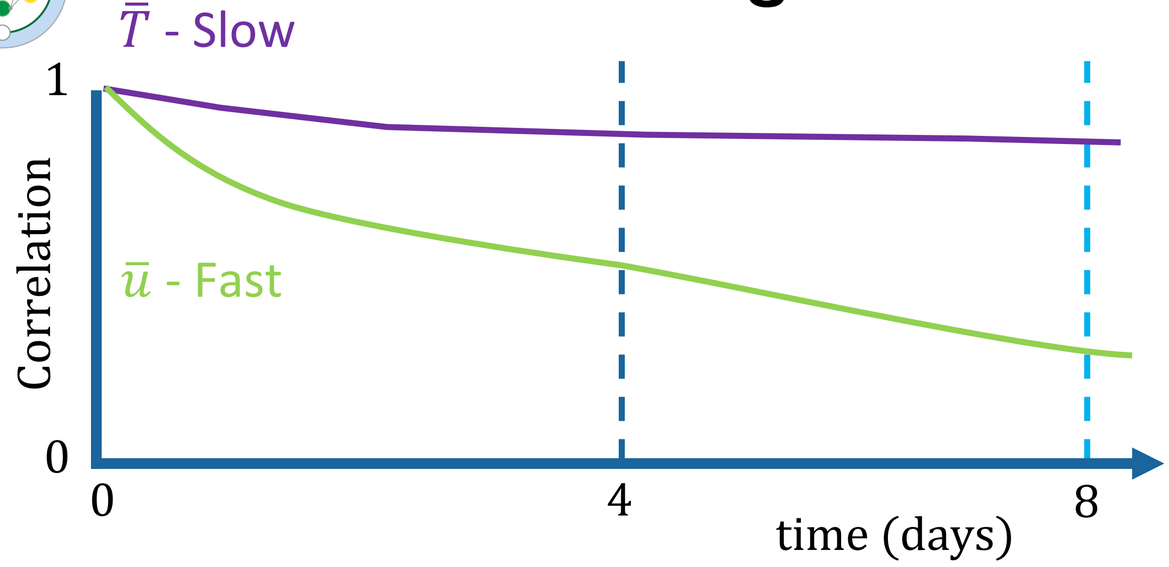


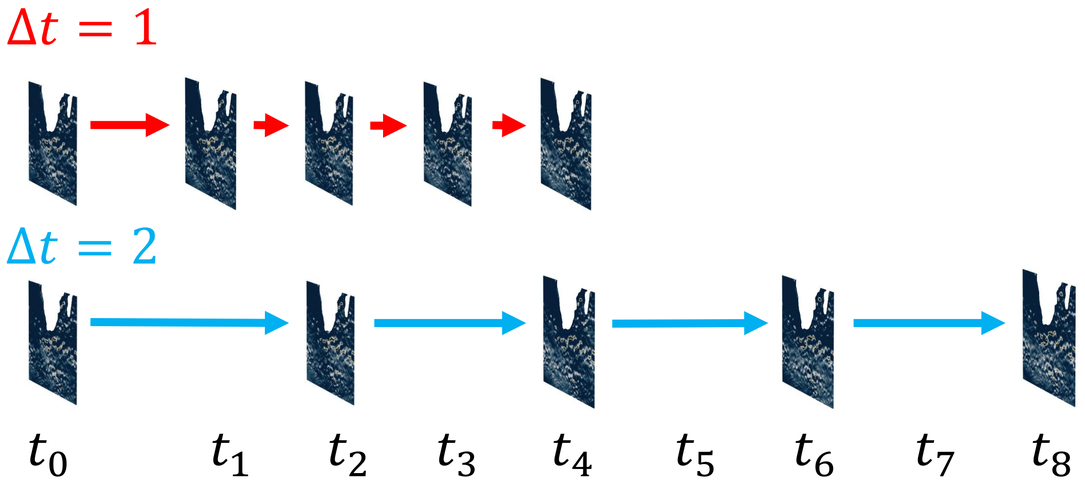
$\Delta t = 1$

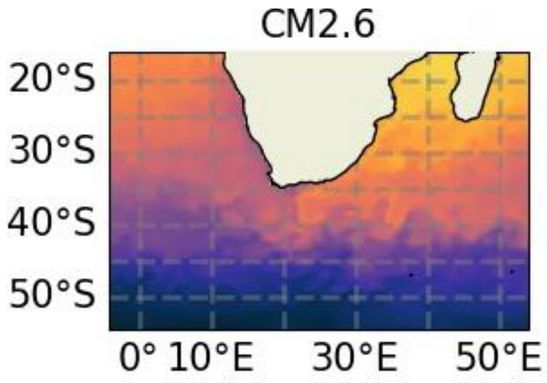
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8

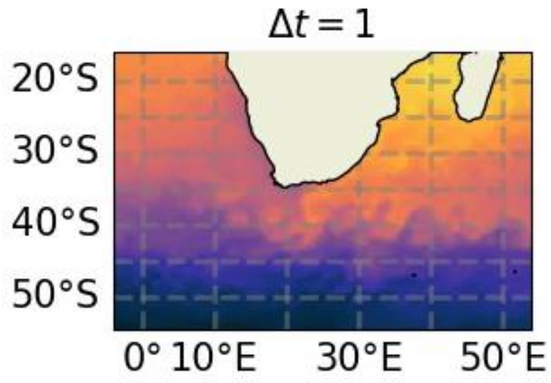


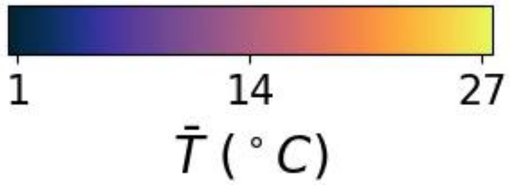
Extending the Training Window

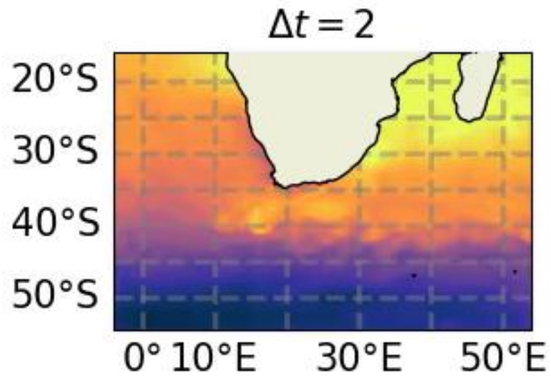




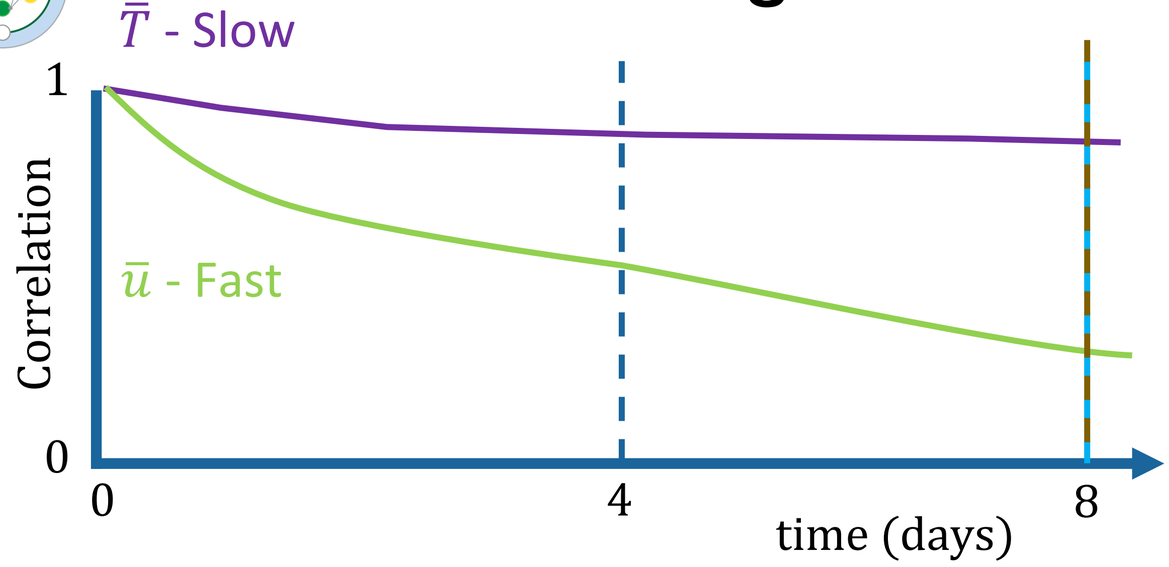






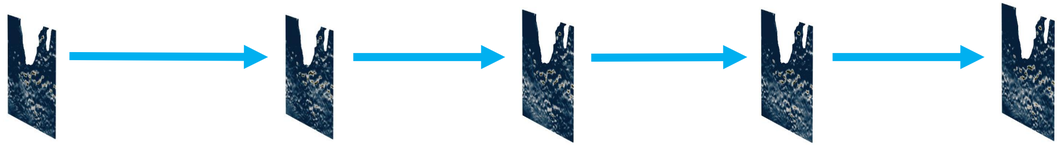


Extending the Training Window

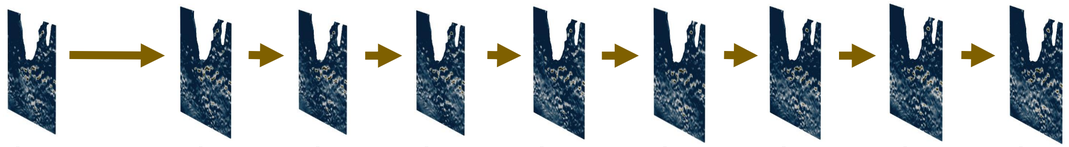


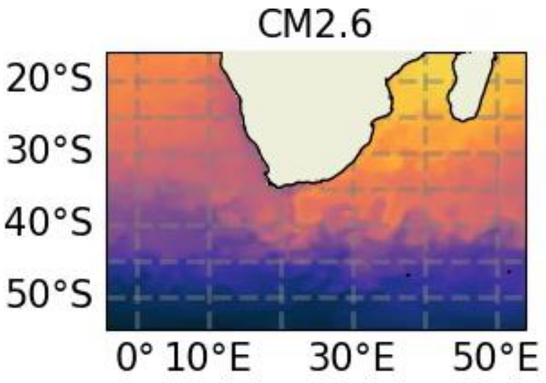
$\Delta t = 1$

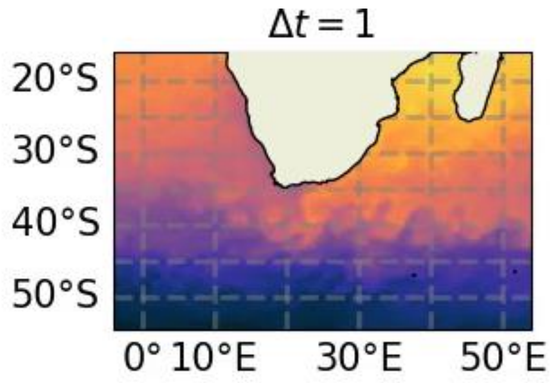
$\Delta t = 2$

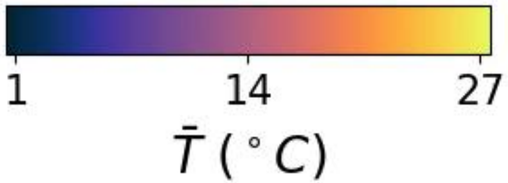


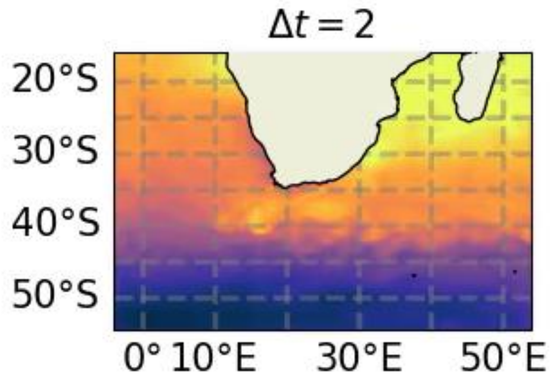
$\Delta t = 1, N = 8$

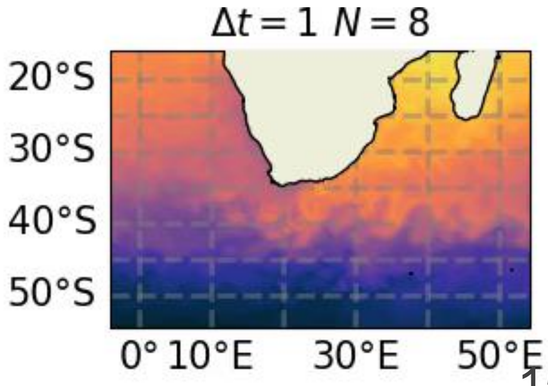






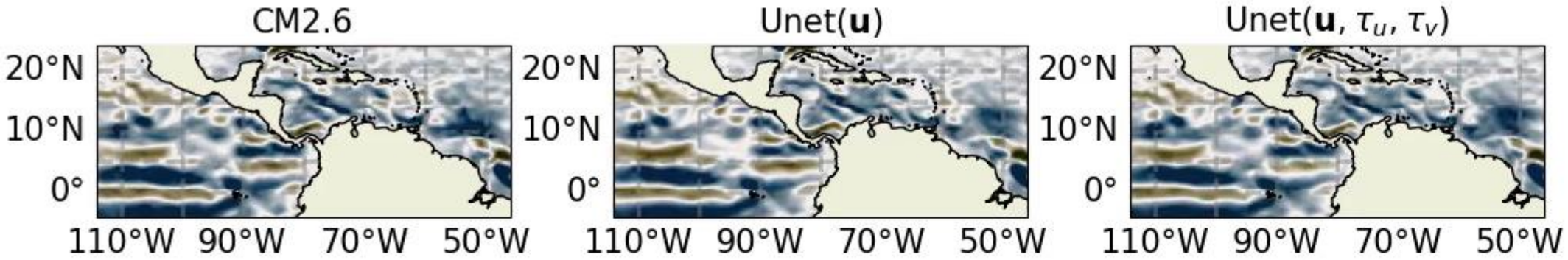






Conclusions

- Successfully produce emulators stable for 8 years
- Need both fast and slow boundary conditions, τ_u, τ_v and T_{atm}
- Use longer windows to capture dynamics of slow variables



https://arxiv.org/abs/2402.04342