

Lightweight Fine-tuning of An Earth System Foundation Model for Sea Surface Temperature Forecasting

Rakib Hossain

December 8, 2025

Why AI Foundation Models for Weather?

Operational Numerical Weather Prediction (NWP) Systems:

- Complex physics-based models (e.g., IFS, GFS)
- Require extensive computational infrastructure
- High operational cost: millions per year for weather agencies
- Training and maintenance: domain expertise-heavy

AI Foundation Models (e.g., Aurora):

- **Order of magnitude cheaper** to train and run
- Inference: seconds on modern GPUs vs. hours on supercomputers
- Competitive forecast skill on standard variables
- Democratizes weather AI to smaller institutions

Trade-off: Aurora's 1.3B parameters are still substantial—full retraining prohibitively expensive

Challenge: Adapting to Unseen Variables

The Problem:

- Aurora's original decoder outputs only standard variables (2m temp, wind, etc.)
- New physical processes (e.g., SST) require adaptation
- Full model retraining: ~80 GB GPU memory, weeks of compute

Our Approach (inspired by Lehmann et al.):

- Freeze Aurora's encoder & backbone (1.3B parameters)
- Train only lightweight MLP decoder (~2M parameters)
- Cost reduction: **orders of magnitude**
- Proof of concept: can Aurora's frozen representations generalize to unseen processes?

reference: Lehmann, Ozdemir, et al., "Finetuning a Weather Foundation Model with Lightweight Decoders for Unseen Physical Processes," arXiv:2506.19088, 2025. doi: 10.48550/arXiv.2506.19088.

Overview

Goal: Predict sea surface temperature (SST) without retraining Aurora's 1.3B parameters

Approach:

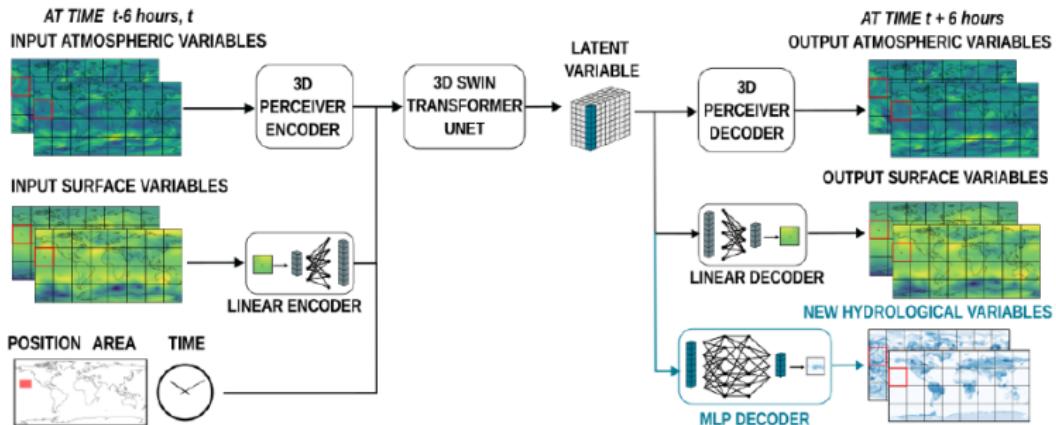
- Freeze AuroraLite encoder and backbone
- Train lightweight MLP decoder on latent representations
- Data- and compute-efficient adaptation

Input: ERA5 fields (surface, static, atmospheric variables) over global grid

Target: SST at specific times and coordinates

Setting: Supervised learning with MAE loss

AuroraLite Architecture



Pipeline: Perceiver3D Encoder → Swin3D Transformer Backbone (frozen) → MLP Decoder. This was used by Lehmann et al.

reference: Lehmann, Ozdemir, et al., "Finetuning a Weather Foundation Model with Lightweight Decoders for Unseen Physical Processes," arXiv:2506.19088, 2025. doi: 10.48550/arXiv.2506.19088.

Perceiver3D Encoder

Input Types:

- ① **Surface variables:** 2m temperature, 10m winds, mean sea level pressure
- ② **Static variables:** land-sea mask, orography, soil type
- ③ **Atmospheric variables:** temperature, winds, humidity, geopotential (13 pressure levels)

Processing:

- ① Normalization with precomputed statistics
- ② Spatial cropping to align with patch size
- ③ Broadcasting static fields across batch and time

Output: Sequence of latent tokens representing 3D atmospheric state and history

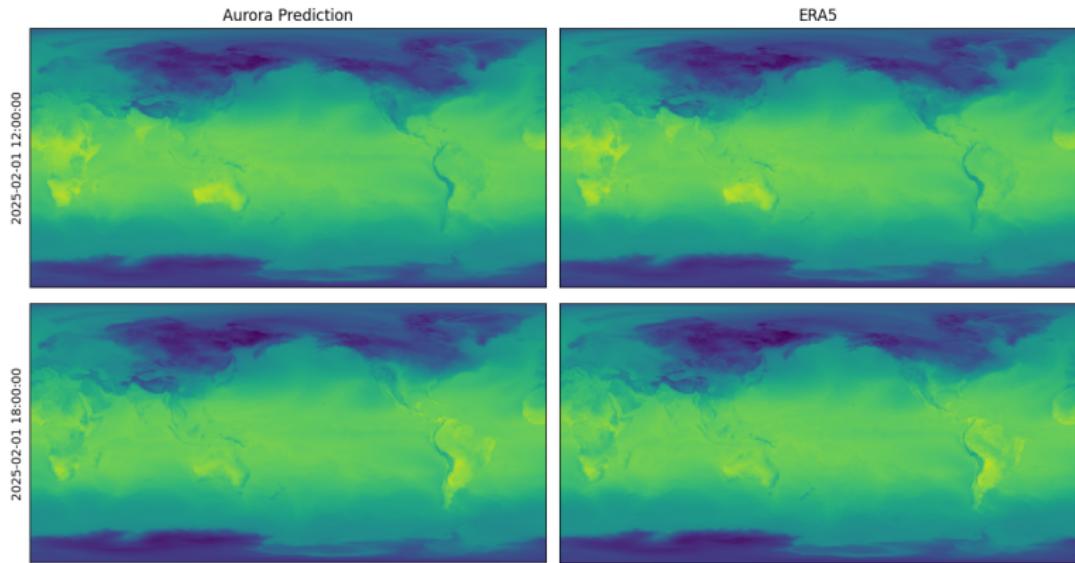
MLPDecoderLite

Architecture:

- ① Input: latent_decoder + lat/ion metadata
- ② Per-variable MLP head:
 - ① Input dimension: D_{embed} (encoder embedding $\times 2$)
 - ② Hidden layers: $512 \rightarrow 512 \rightarrow 256$
 - ③ Output dimension: P^2 (patch-wise predictions)
- ③ Unpatchify to full resolution
- ④ Denormalize to physical units

Scale: $\approx 300,000$ parameters for a decoder

Baseline - 2m Temperature (Aurora Native)



Lead Time	RMSE (K)	Correlation
Step 1 (+6h)	0.6450	0.9995
Step 2 (+12h)	0.8337	0.9992
Average	0.7393	0.9994

Methodology - Inference

SST Inference Workflow

Step 1: Fill missing SST values using spatial median

Preprocess ERA5 SST data to handle missing/invalid values.

Step 2: Run Aurora inference using SST as 2m temperature

Feed the preprocessed SST into the frozen AuroraLite model as if it were 2m temperature input.

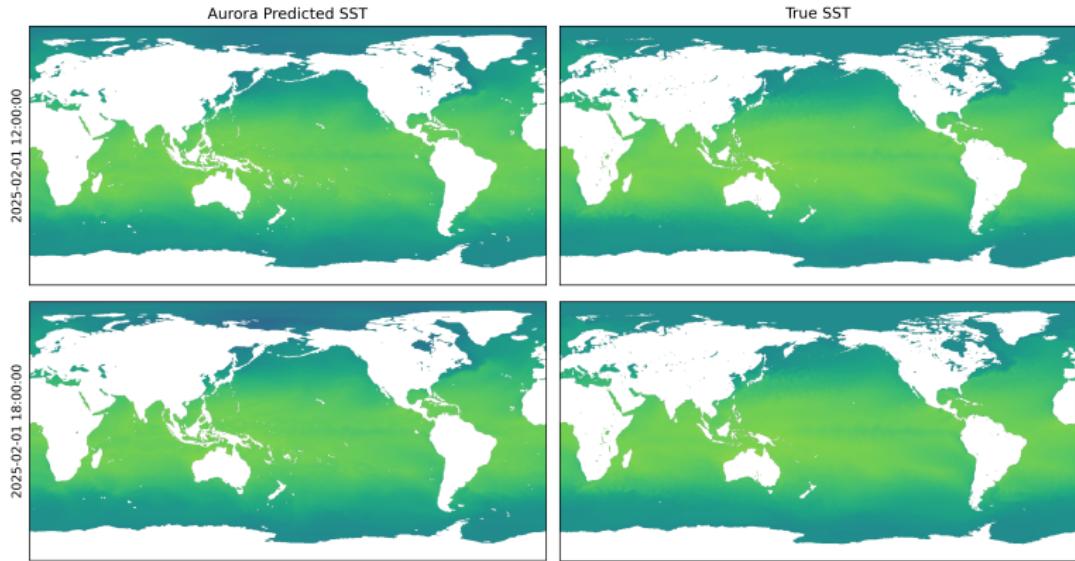
Step 3: Mask land grid points in model prediction

Apply land-sea mask to remove predictions over land regions (keep only ocean predictions).

Step 4: Compare masked predictions with original SST

Evaluate model output against ground truth ERA5 SST over valid ocean pixels.

Sea Surface Temperature - Inference



Lead Time	RMSE (K)	Correlation
Step 1 (+6h)	1.4204	0.9952
Step 2 (+12h)	2.3195	0.9880
Average (ocean)	1.8699	0.9916

Data

Training: ERA5 Reanalysis 2020–2021, 6-hour intervals

Validation: Jan–Aug 2022

Total Dataset Size: 445 GB

Data Configuration:

- ① Grid: 720 lat × 1440 lon
- ② History: 2 time steps (current timestep and immediately previous timestep)
- ③ Pressure levels: [50, 100, 150, ..., 1000] hPa (13 levels)
- ④ Sequential batching: 22 timesteps (\approx 5.5 days)

Target: Single 2D SST field per time step

Methodology - Training

Loss: Mean Absolute Error (MAE)

Optimizer: Adam with learning rate $\eta = 3 \times 10^{-4}$

Validation: Jan–Aug 2022, evaluated every 22 timesteps

Sequential Pass: Gradients accumulated over sequential data to capture seasonal patterns

Hyperparameters: All fixed; validation used only to monitor convergence

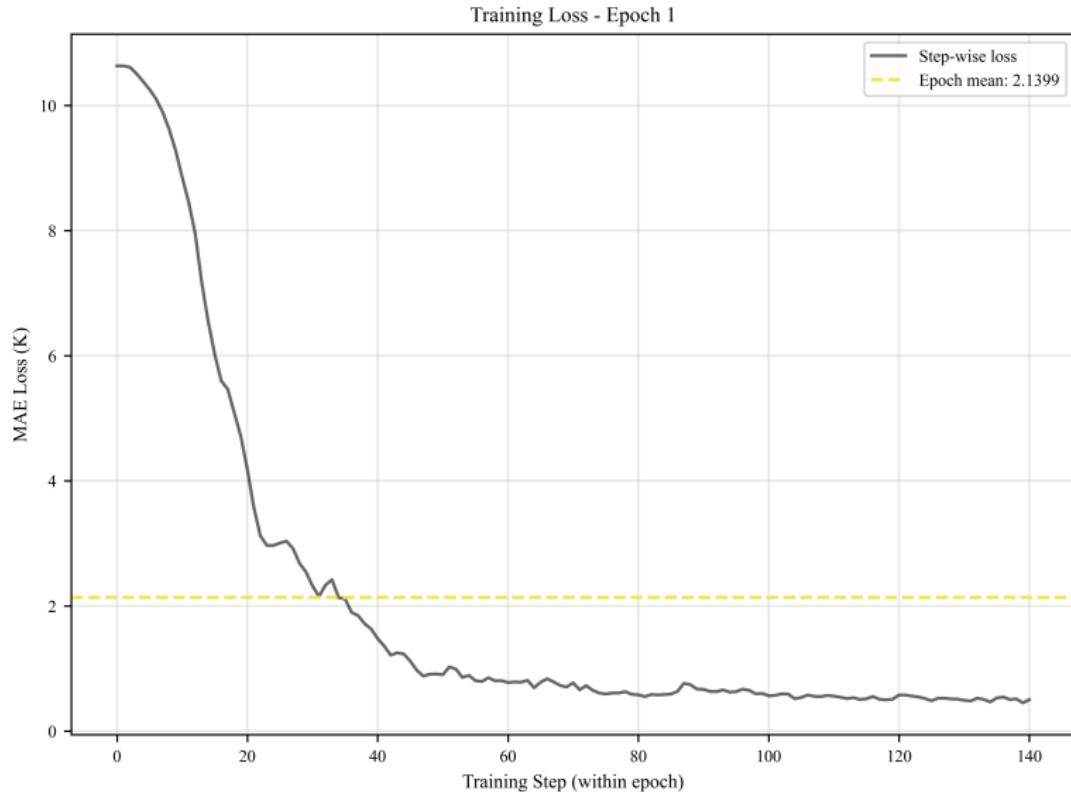
Computational Resources

Hardware: Requested on Fir Cluster of Research Alliance Canada

- ① GPU: 1× NVIDIA H100
- ② CPU: 12 cores
- ③ RAM: 288 GB system memory

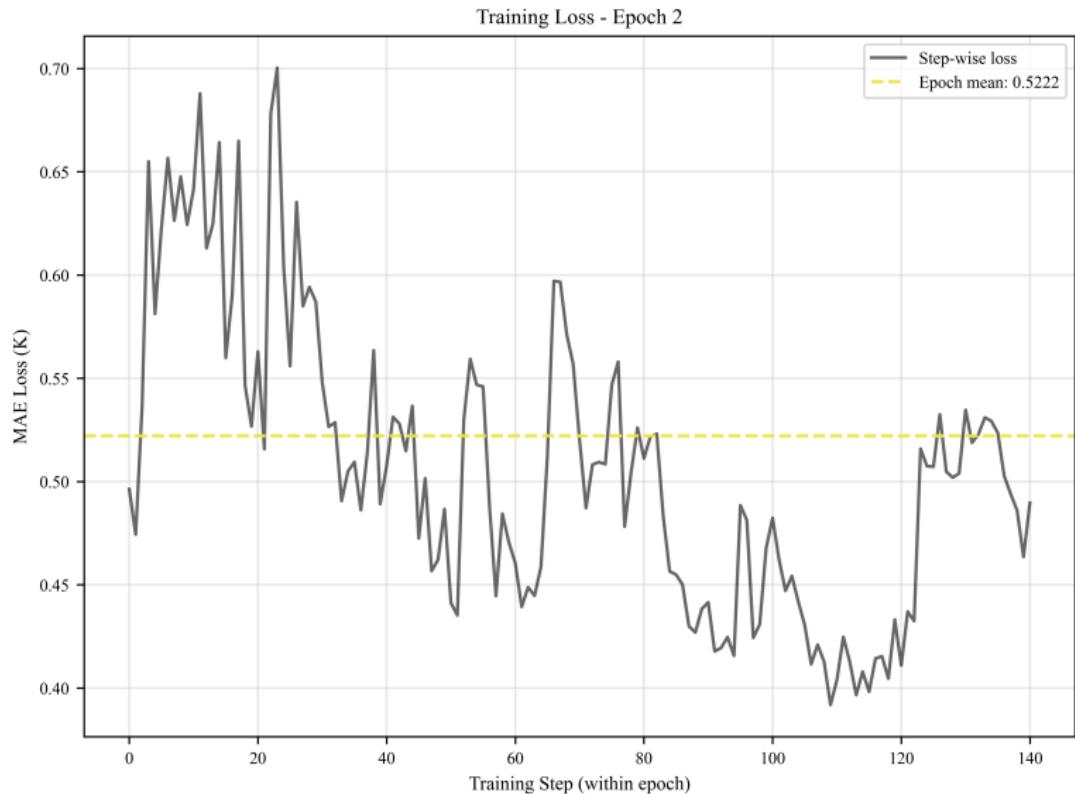
Runtime: ~3 days for 2 epochs

Training Loss - Epoch 1



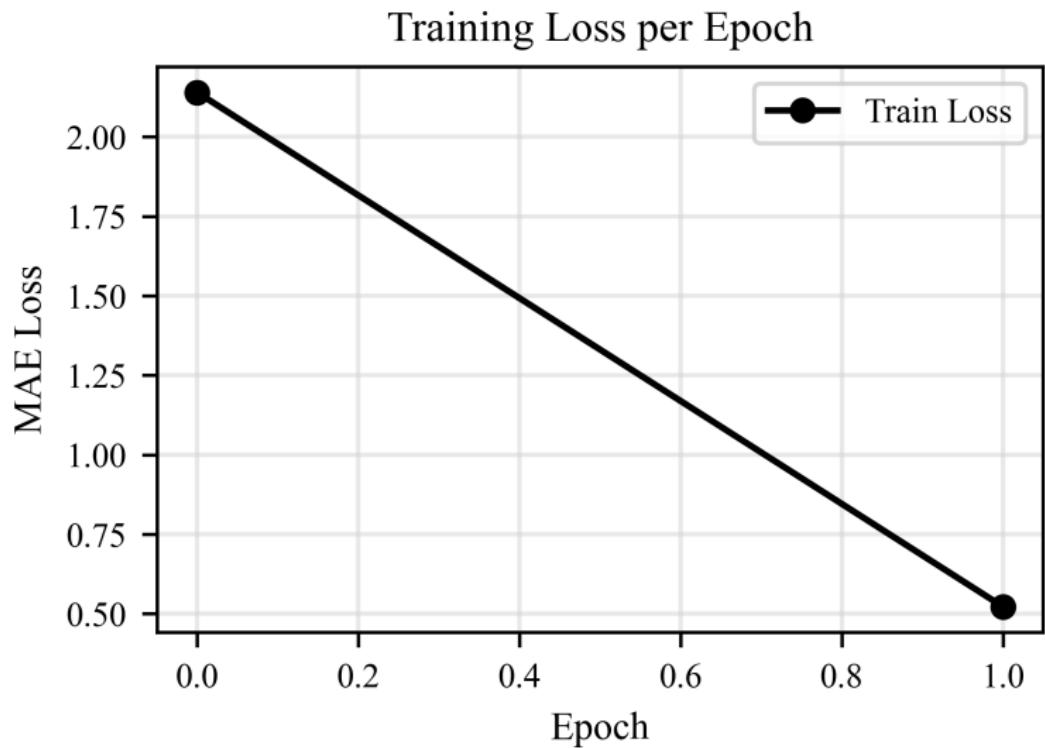
Epoch 1: Train Loss 2.1399 → Val Loss 0.7272

Training Loss - Epoch 2



Epoch 2: Train Loss 0.5222 → Val Loss 0.6218

Training Loss Over Epochs

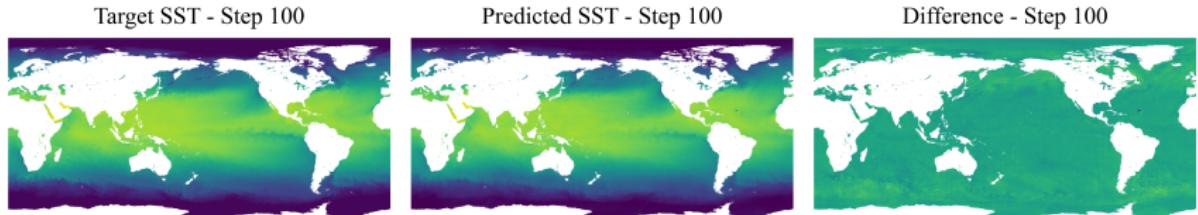


Training Progression - Step 0



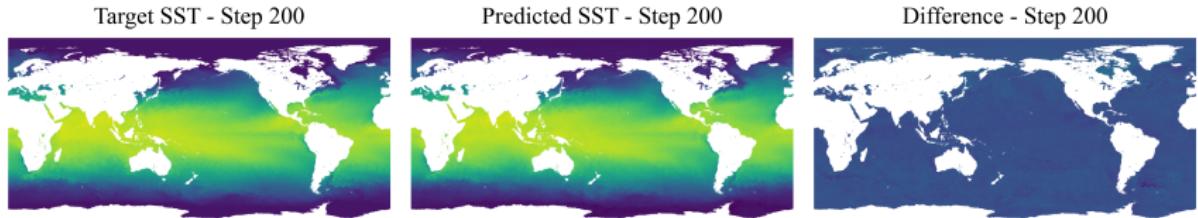
Initial training state: raw predictions before convergence

Training Progression - Step 100



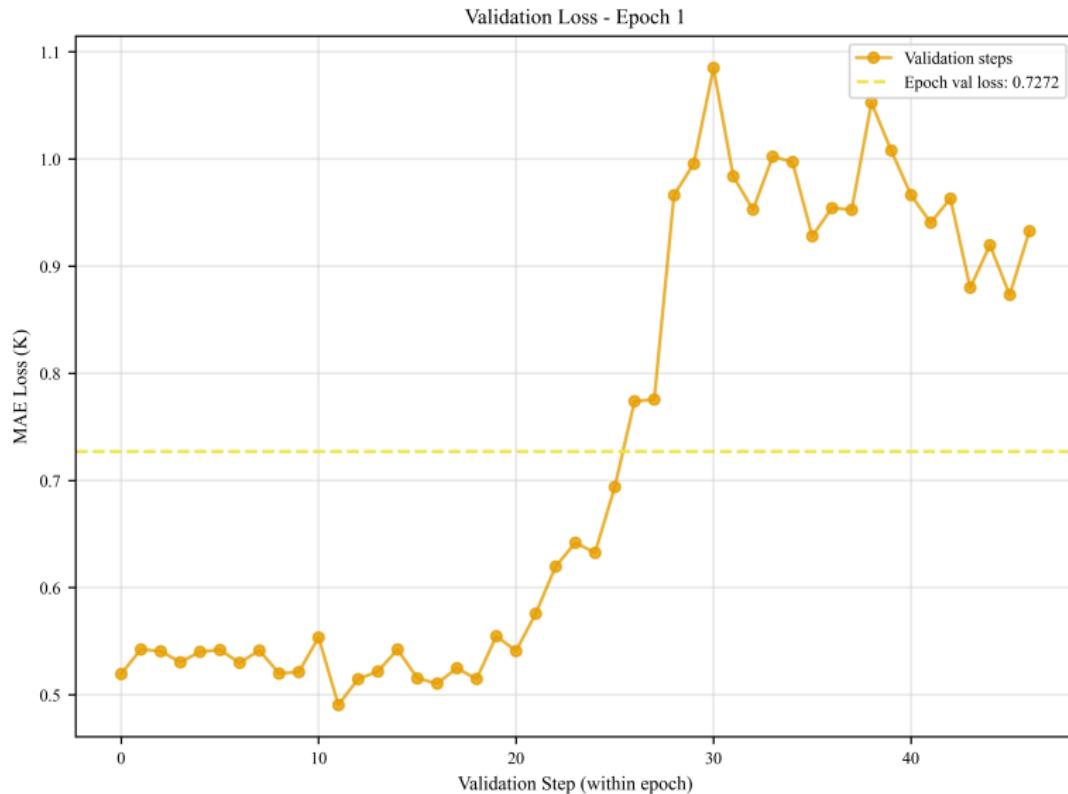
Early training: spatial structure begins to emerge

Training Progression - Step 200



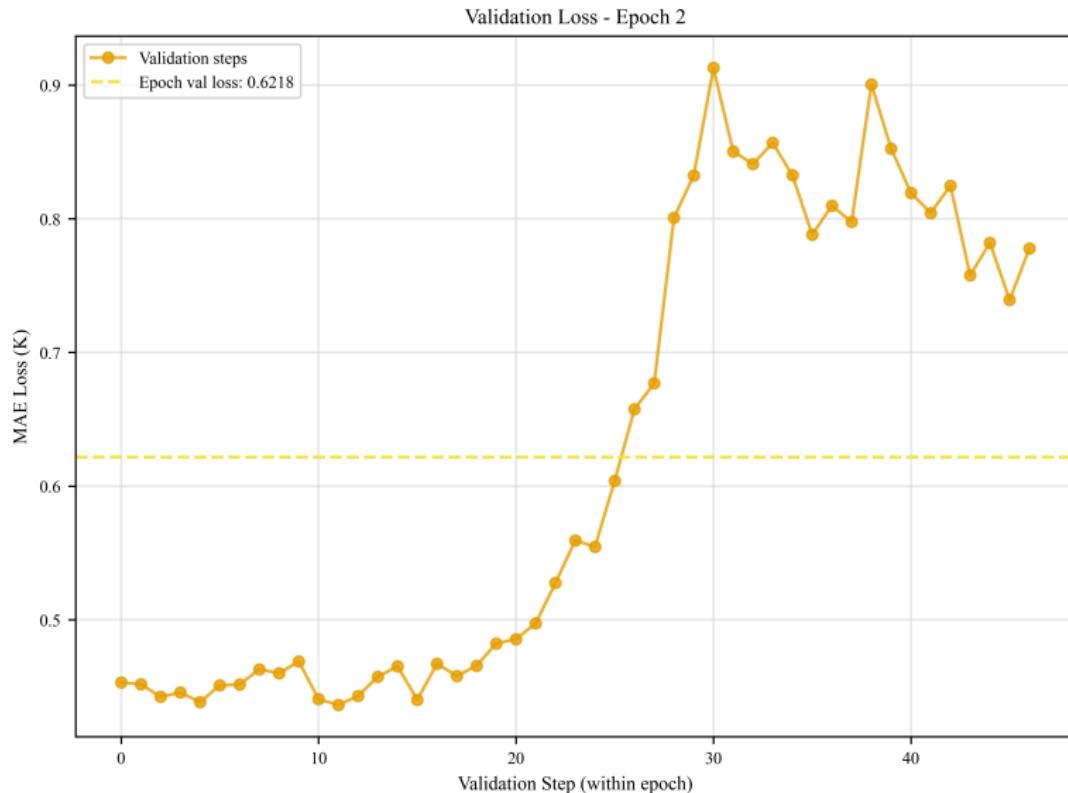
Convergence: model captures realistic ocean temperature patterns

Validation Loss - Epoch 1



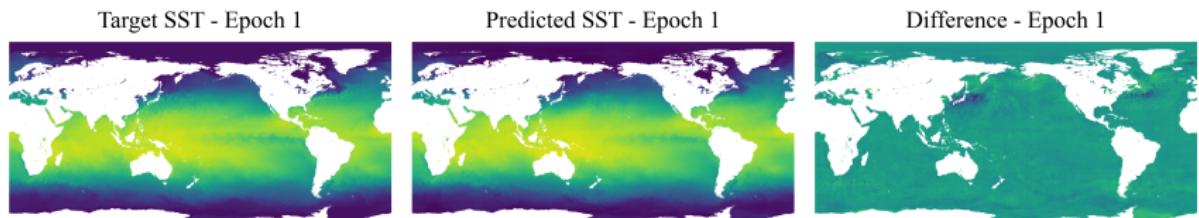
Performance better on data from Jan - Apr

Validation Loss - Epoch 2



Validation loss decreases at each point, yet same pattern

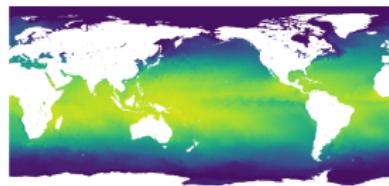
Validation Predictions - Epoch 1



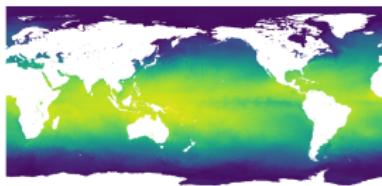
Early epoch: model already captures significant SST structure

Validation Predictions - Epoch 2

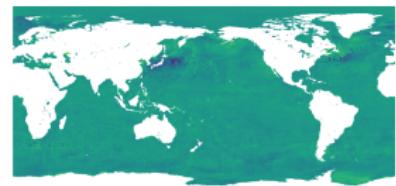
Target SST - Epoch 2



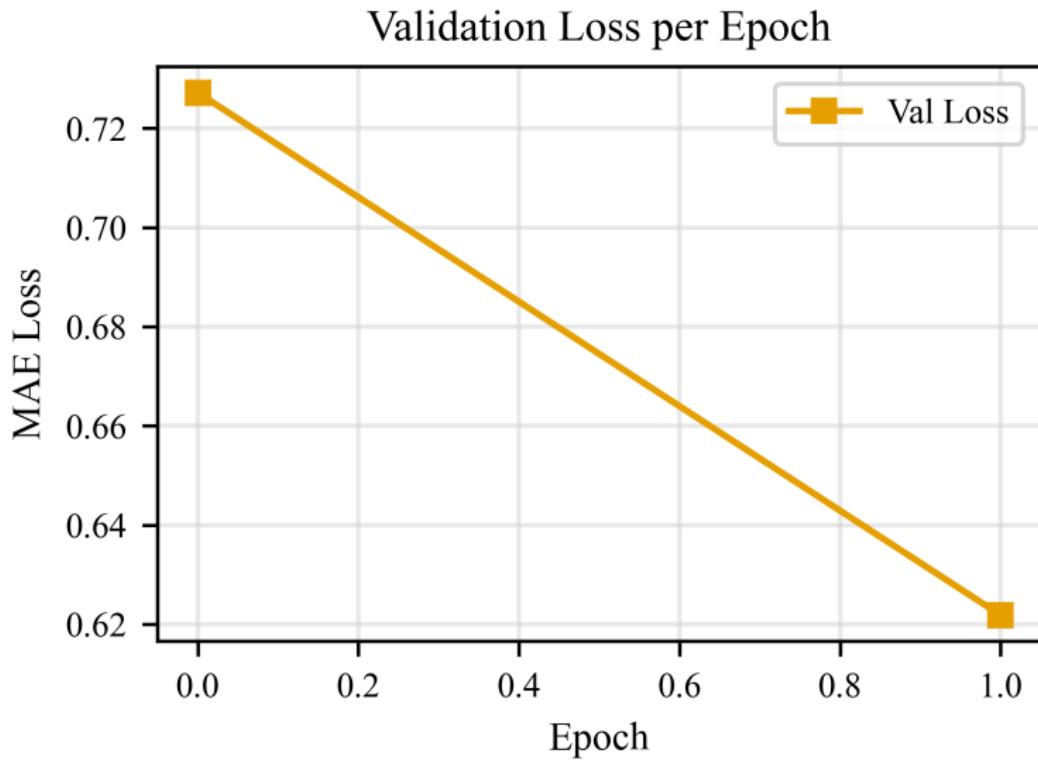
Predicted SST - Epoch 2



Difference - Epoch 2



Validation Loss Over Epochs



Validation loss trajectory showing tendency of model convergence

Spatial Consistency Regularization

Gradient Penalty Loss:

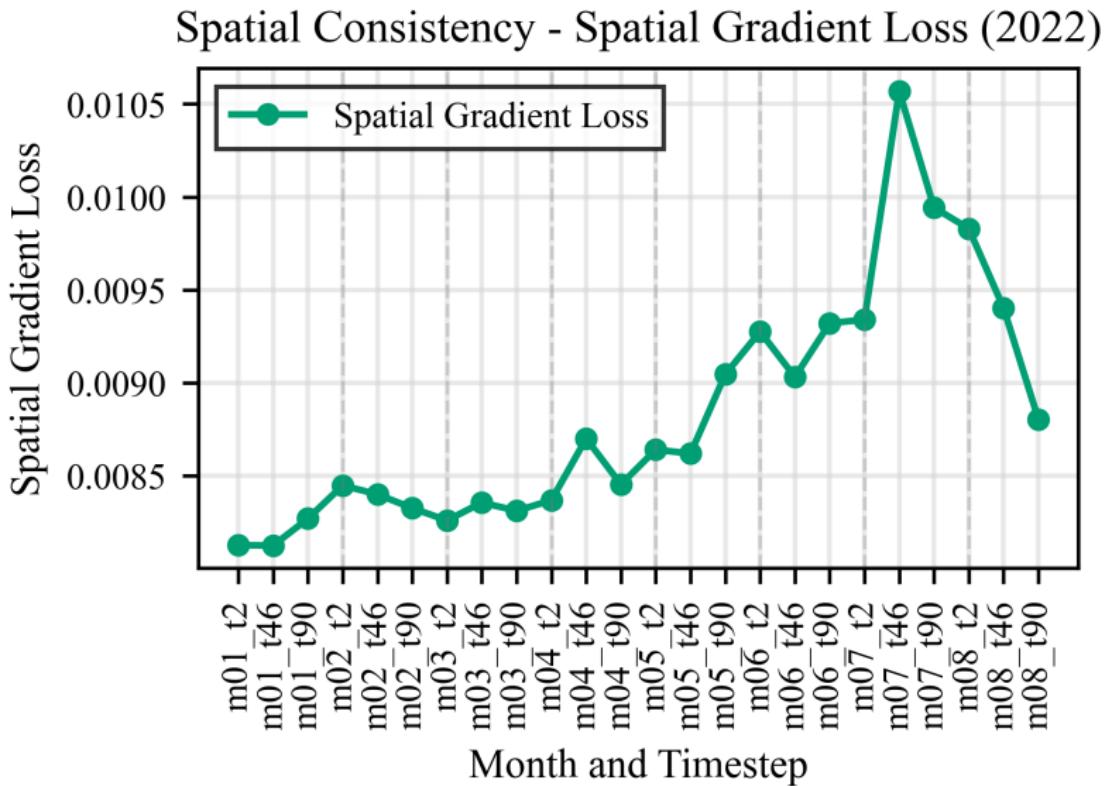
$$\Delta_{\text{lat}} = \frac{1}{N} \sum_{i,j} |\text{SST}_{i+1,j} - \text{SST}_{i,j}|^2 \quad (1)$$

$$\Delta_{\text{lon}} = \frac{1}{N} \sum_{i,j} |\text{SST}_{i,j+1} - \text{SST}_{i,j}|^2 \quad (2)$$

$$L_{\text{gradient}} = \lambda (\Delta_{\text{lat}} + \Delta_{\text{lon}}), \quad \lambda = 0.05 \quad (3)$$

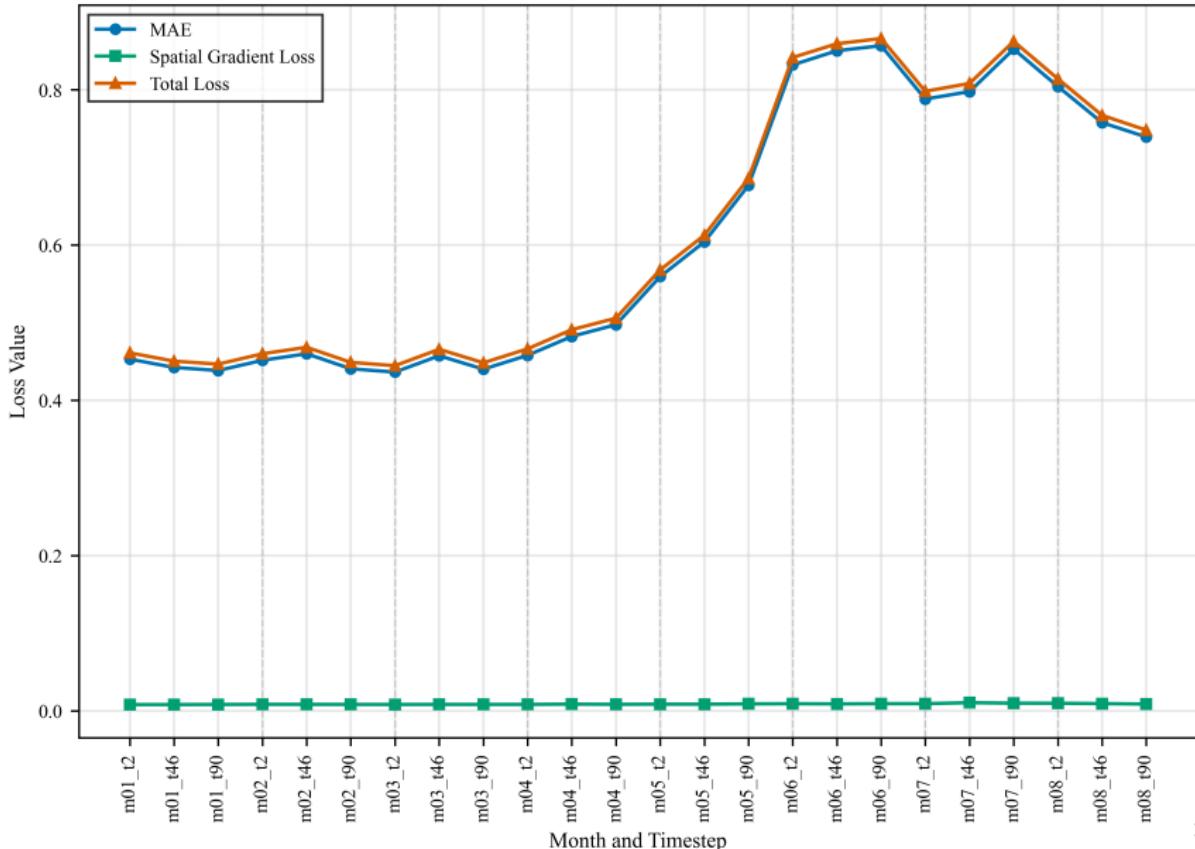
These gradients enforce physical smoothness: SST should vary smoothly across space, preventing unrealistic sharp discontinuities in temperature fields. It was only used for testing.

Spatial Consistency Check



Spatial Consistency-based Validation

SST Forecasting Losses Across 2022 Test Period



Key Results

Forecasting Performance:

- ① We can conclude that without explicitly seeing SST data during training, the Aurora Model can predict SST.
- ② However the model has generalized better on SST data from January to April compared to May to August.
- ③ We can see the model already maintains spatial consistency (SST does not rise or fall from one coordinate to another rapidly). Yet in this case we see better performance in first 4 months.

Scopes of improvement:

- ① Finetune on at least 8-10 years of data.
- ② Validate on more years. Both of the tasks will require much more resources and time.

Takeaway: Foundation models + lightweight decoders enable practical adaptation to new variables with minimal compute

Acknowledgements

- ① This slide demonstrates my endeavours to explore Aurora model.
<https://github.com/mridul0837/aurora-sst-forecasting>
- ② This work is a part of course project done for COMP6936: Advanced Machine Learning Course of MUN instructed by Dr. Jingrong Wang. Link to the full course project by all groupmates:
<https://github.com/Aiden0609/MLProject-G1-F2025>
- ③ Special thanks to the Research Alliance Canada for all the resources.

Thank You!

For any questions, please feel free to contact:

Email: mdrh@mun.ca

LinkedIn: md-rakib-hossain-mridul

GitHub: github.com/mridul0837