CSE 151B Project Final Report

GitHub Repository

Adrian Apsay - adapsay@ucsd.edu
Ethan Flores - etflores@ucsd.edu
Ryan Izadshenas - rizadshenas@ucsd.edu
Andrew Yin - anyin@ucsd.edu

Abstract

Predicting future climate variables efficiently and accurately is a critical challenge in both climate science and machine learning. Traditional climate emulation models, while powerful, are computationally expensive and difficult to scale for rapid scenario evaluation. In this work, we propose a lightweight, deep learning-based surrogate model for climate emulation, using a U-Net convolutional neural network with CoordConv inputs and dual decoder heads to predict surface air temperature (tas) and precipitation (pr) across spatial grids. Our design addresses common challenges in multitask geospatial prediction, such as spatial bias and gradient interference between output targets. We conduct extensive experiments comparing architectural variants, including ConvLSTM modules and separate decoder architectures, and explore multiple loss functions tailored for skewed data distributions. Our final model achieves a private test RMSE of 0.9800 on the CSE 151B Kaggle competition and ranks 33th out of 83 teams, outperforming baseline U-Net configurations. These results highlight the importance of task-specific architectural separation and spatial-awareness techniques in improving the stability and generalization of climate emulators. Our findings support the use of lightweight, spatially structured models as alternatives to large-scale simulators for climate projection tasks.

1 Introduction

Climate simulation tools like those in CMIP6 offer powerful projections but are computationally expensive, limiting their scalability under new emissions scenarios. Deep learning provides a promising alternative: fast, learned emulators can approximate these simulations and enable real-time scenario testing.

In this project, we develop neural network models to emulate monthly climate variables—surface air temperature (*tas*) and precipitation rate (*pr*)—based on variables such as greenhouse gas concentrations and solar radiation. Accurate emulation supports faster policy decisions, richer ensemble forecasting, and broader scientific access in climate-sensitive domains.

The starter code provides a shallow CNN with residual blocks but lacks spatial skip connections, temporal modeling, and separates outputs using a shared decoder head, which we found to hinder convergence.

Our contributions include:

- **Dual decoder heads** to mitigate task interference between *tas* and *pr*.
- **CoordConv input augmentation** to encode spatial position via coordinate channels.

Preprint. Under review.

- **Deeper and temporal architectures** (e.g., U-Net and ConvLSTM) to model spatial and sequential patterns.

These improvements led to a final private leaderboard RMSE of 0.9800, reflecting strong generalization beyond the baseline.

2 Related Work

Recent works have explored deep learning as a cost-effective alternative to full-scale climate simulations. Our approach draws architectural inspiration from prior models designed for spatiotemporal and geospatial data.

Yu et al. [5] introduced ST-UNet for graph-structured time series, demonstrating the U-Net's ability to capture local and global structure. We adopted this backbone for climate variable prediction.

Wang et al. [4] proposed DeepSTCL, using ConvLSTMs to model temporal dependencies. We experimented with similar architectures, finding that while they captured emission trends (e.g., in SSP370), they overfit on smoother scenarios like SSP245.

Ozbulak et al. [3] proposed USE-Net, which enhances U-Net with Squeeze-and-Excitation (SE) blocks to recalibrate channel-wise features for better segmentation of MRI scans. While SE blocks improved generalization in medical imaging, our experiments in climate emulation found minimal gains—likely due to the structured nature of climate grids and limited benefit from per-channel recalibration.

CoordConv, introduced by Liu et al. [2], improves positional awareness by appending spatial coordinates to the input. This proved effective in our domain and improved tas accuracy.

Lastly, Lin and Michailidis [1] demonstrated that task-specific decoders reduce interference in multi-output settings. Motivated by this, we implemented dual decoder heads for tas and pr, improving stability and pr RMSE.

These prior methods informed key components of our model, which combines spatial reasoning, task separation, and empirical validation into a stable climate emulation pipeline. We've acknowledged these methods going forward throughout the course of the project.

3 Problem Statement

The task is to build a climate model that emulates the output of a computationally expensive climate simulation. Specifically, we aim to predict future climate variables, surface air temperature (*tas*) and precipitation rate (*pr*), under a previously unseen emissions scenario (SSP245), given a set of input climate forcings such as greenhouse gas concentrations and solar radiation.

This problem is critical because climate models are slow and resource-intensive to run. A reliable machine learning climate emulator can significantly accelerate climate projections, enabling faster scenario analysis for those interested in climate change.

Mathematically, we are given training data in the form of input-output pairs:

$$S = \{(x_i, y_i)\}_{i=1}^{N}$$

where $x_i \in \mathbb{R}^{C \times H \times W}$ is an input tensor containing $C = 5$ variables (e.g., CO_2, CH_4, SO_2, BC, rsdt) over a 48 (H) $\times$ 72 (W) spatial grid at time i, and $y_i \in \mathbb{R}^{H \times W \times 2}$ is the corresponding output tensor of predicted *tas* and *pr* values.

The model $f_\theta(x)$ is parameterized by weights θ, and aims to predict $\hat{y}_i = f_\theta(x_i)$. The learning objective is to minimize a loss function $\mathcal{L}$ that combines several terms reflecting spatial and temporal accuracy. Mathematically, the objective can be expressed as:

$$\min_\theta \sum_{i=1}^{N} \mathcal{L}(y_i, f_\theta(x_i))$$

Where $\mathcal{L}$ could include:

- Monthly Area-Weighted RMSE to capture pointwise monthly accuracy
- Decadal Mean Area-Weighted RMSE to capture long-term climate means
- Decadal Std Dev Area-Weighted MAE to capture temporal variability at each location

Data Splits and Preprocessing. We used monthly climate data from three SSP scenarios for training: SSP126 (low emissions), SSP370 (medium-high emissions), and SSP585 (high emissions). Specifically, we trained on:

- All months from SSP126 and SSP585
- The first 901 months of SSP370

The validation set consisted of the final 120 months of SSP370. The test set consisted of the full 360 months of SSP245, a scenario completely held out during training.

Each input tensor x_i contains the five forcing variables `[CO2, CH4, SO2, BC, rsdt]`, either as scalars broadcasted over the spatial grid or with inherent spatial variation. The model predicts *tas* and *pr* on a global 48×72 lat-lon grid.

Normalization. All input variables were standardized using z-score normalization based on statistics from the training set. Both output variables—*tas* and *pr*—were also z-score normalized. Although we experimented with log-transforming *pr* to address its skewness, this hurt performance, so we retained the raw (non-log) values and normalized them using z-scores.

Key Deviations from Starter Code. The starter code implements a shallow residual CNN with a shared decoder head. We experimented with several important architectural and preprocessing methods:

- Replaced the shallow residual CNN with a deeper U-Net-style encoder–decoder with skip connections.
- Introduced CoordConv layers by appending normalized latitude and longitude channels to the inputs, enhancing spatial awareness.
- Split the output decoder into two separate heads for *tas* and *pr*, reducing task interference between the two output channels.
- Experimented with ConvLSTM-based models to improve temporal pattern recognition across multi-month windows, represented as T.

These changes significantly improved generalization to unseen SSP scenarios and reduced both short-term and long-term prediction errors for the target variables.

4 Methods

We formulate climate emulation as a supervised learning problem over geospatial climate variables. Given a sequence of monthly climate forcings, the goal is to predict surface air temperature (*tas*) and precipitation rate (*pr*) on a global 48×72 grid.

4.1 Problem Formulation

Each input $x_i \in \mathbb{R}^{C \times H \times W}$ represents $C = 5$ climate variables—CO_2, CH_4, SO_2, BC, and rsdt—broadcast or spatially distributed over latitude and longitude. The target $y_i \in \mathbb{R}^{2 \times H \times W}$ corresponds to monthly *tas* and *pr* at the same spatial resolution.

We seek to learn a model $f_\theta : \mathbb{R}^{C \times H \times W} \to \mathbb{R}^{2 \times H \times W}$ that minimizes a loss $\mathcal{L}$ across all training samples:

$$\min_\theta \sum_{i=1}^{N} \mathcal{L}\left(f_\theta(x_i), y_i\right)$$

The exact form of $\mathcal{L}$—a multi-target error function—will be discussed in Section 5.3.

4.2 Model Overview

Our final model is a modified U-Net CNN with two key enhancements: coordinate-aware inputs and task-specific decoder heads. This architecture was selected after iterative testing for its superior generalization and stability across emission scenarios.

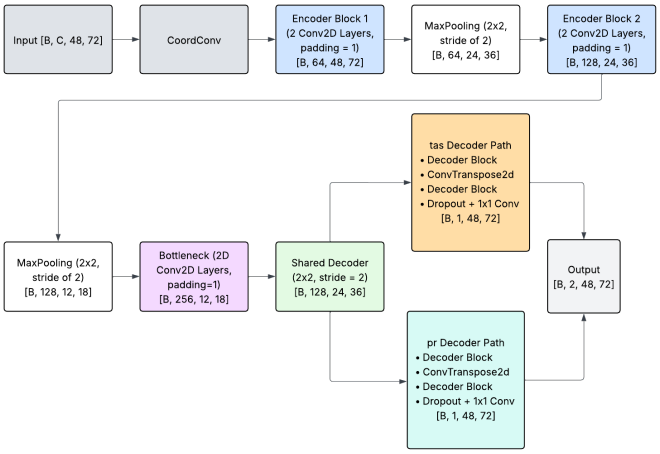

Figure 1: Overview of our final U-Net-based architecture. The model takes a 5-channel input grid (with appended CoordConv channels), passes it through two encoder blocks and a bottleneck, and splits into dual decoder heads—one for *tas* and one for *pr*. Each decoder contains upsampling layers, skip connections, and a final 1×1 convolution. The outputs are concatenated into a $[2, 48, 72]$ tensor representing climate variable predictions.

CoordConv Input Channels. To incorporate spatial position, we append two additional coordinate channels (normalized latitude and longitude grids ranging from -1 to 1) to each input tensor. This modification, inspired by CoordConv [2], provides the model with spatial context and improves localization of features across global regions.

Encoder–Decoder Backbone. The model follows a 2-level U-Net structure:

- **Encoder:** Two convolutional blocks with increasing depth (64, 128), each followed by BatchNorm, ReLU, and max-pooling.
- **Bottleneck:** A pair of convolutional layers with 256 channels, capturing mid-level spatial abstractions.

Dual Decoder Heads. After the bottleneck, the network branches into two parallel decoders—one each for *tas* and *pr*. Each branch contains upsampling layers with skip connections and ends in a 1×1 convolutional layer that restores the original 48×72 spatial resolution. The outputs are concatenated into a $[2, 48, 72]$ tensor representing the final predictions.

This separation was motivated by empirical observations of gradient interference in shared-head models, especially given the distinct statistical behaviors of temperature and precipitation.

4.3 Alternative Architectures Considered

We explored several architectural variants during model development:

- **Shared Decoder U-Net:** A single decoder predicting both outputs jointly.
- **ConvLSTM-UNet Hybrid:** A temporal variant processing $T = 12$ months of climate forcings via ConvLSTM layers before decoding.
- **UNetCNN + SE:** Used Squeeze-and-Excitation blocks to enhance channel-wise attention.

- **UNetCNN + Attention Gates:** Added attention mechanisms on skip connections for dynamic feature gating.

Design tradeoffs and performance differences between these variants are analyzed in Section 5.

5 Experiments

In this section, we describe our empirical exploration of climate emulation models, detailing the baselines tested, evaluation metrics used, training methodology, and key results from our experiments. We conclude with an ablation study highlighting the contribution of each architectural component.

5.1 Baselines

We compare our proposed model against two baselines of increasing architectural complexity:

- **SimpleMLP:** A non-spatial feedforward MLP that flattens the $[C, 48, 72]$ input into a 1D vector and processes it through two dense layers. It discards all spatial structure, making it severely limited in learning geospatial patterns. This model serves as a minimal baseline to test whether spatial awareness is essential for climate emulation.
- **SimpleCNN (Starter Code):** A shallow convolutional network with residual blocks and a shared output head for `tas` and `pr`. It captures local spatial patterns better than MLP but lacks the depth and flexibility for multiscale learning. It often overfits quickly and underperforms on pr due to limited capacity.
- **Shared Decoder U-Net:** A standard U-Net with shared decoder for `tas` and `pr`. It introduces encoder-decoder symmetry, skip connections, and multiscale context aggregation, but the shared output path led to task interference and gradient conflicts. This baseline was essential for evaluating the impact of separate output heads.

5.2 Evaluation

We use three official Kaggle metrics to evaluate model performance:

1. **Monthly RMSE:** Area-weighted root mean squared error over all grid points and time steps.
2. **Time-Mean RMSE:** Area-weighted RMSE on decadal mean predictions at each location.
3. **Stddev MAE:** Area-weighted mean absolute error on decadal standard deviation, capturing temporal variability.

These metrics are computed separately for surface air temperature (tas) and precipitation rate (pr), and the final leaderboard score is a weighted sum.

5.3 Implementation Details

All models were implemented in PyTorch Lightning and trained on UC San Diego's DataHub GPU infrastructure. We used a consistent training pipeline with robust logging and early stopping to ensure reproducibility and metric stability.

Training Configuration. We trained for 50–200 epochs depending on convergence, with early stopping (patience = 20) based on validation RMSE for tas. The batch size was fixed at 64 to balance GPU memory usage and training stability. We used the Adam optimizer with the `CosineAnnealingLR` scheduler, which gradually reduced the learning rate for smoother convergence in later epochs. This was also utilized with Early Stoppage.

Loss Functions and Transformations. We explored various loss functions to improve precipitation (pr) predictions, including SmoothL1, Huber, and Tweedie losses, as well as log-transformation of pr due to its skewness. These experiments were enabled by our dual-head architecture, which allowed per-output customization. However, none of these alternatives consistently outperformed mean squared error (MSE), which we ultimately used across both tas and pr targets with z-score normalization. Weighted loss tuning (e.g., 80/20 in favor of pr) was also tested but degraded tas performance without improving pr, so we retained equal loss weighting.

5.4 Results

Table 1: Validation performance of models across *tas* and *pr* using Kaggle evaluation metrics.

Model	tas RMSE	Time-Mean RMSE	Stddev MAE	pr RMSE	Time-Mean RMSE	Stddev MAE
UNetCNN (Dual + Coord)	1.3437	0.5271	0.2522	1.9586	0.2952	0.7627
UNetCNN + SE	1.3401	0.5173	0.2623	1.9450	0.2642	0.7500
ImprovedUNet	1.3438	0.5300	0.2551	2.2561	1.0635	1.0770
ConvLSTMNet	1.7863	1.0922	0.4423	1.9683	0.3332	0.8340
SimpleMLP (Baseline)	2.7759	1.6414	0.7919	2.1343	0.5392	1.0876

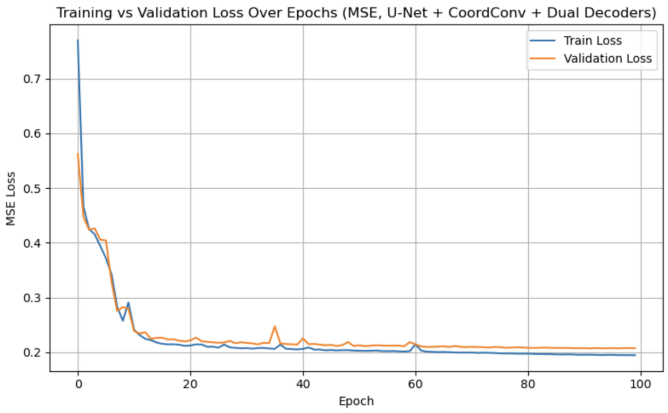

Figure 2: Training and validation MSE loss curves for the U-Net + CoordConv + Dual Decoder architecture. The model converges smoothly with minimal overfitting, indicating strong training stability and generalization.

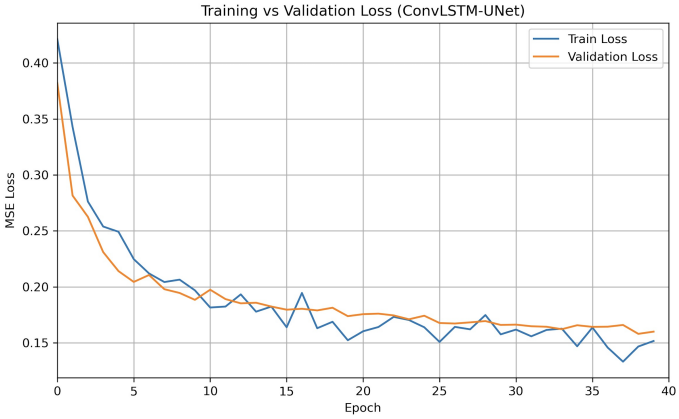

Figure 3: Loss curves for ConvLSTM-UNet. While initial training is rapid, validation loss diverges due to overfitting, especially on the smoother SSP245 scenario.

As shown in Table 1, convolutional models vastly outperformed the MLP baseline, especially for surface air temperature (*tas*). Our U-Net model with dual decoder heads and CoordConv inputs achieved the best overall balance between performance and stability, as further supported by its smooth convergence in Figure 2. Adding SE blocks (UNetCNN + SE) yielded marginal improvements in *pr* but did not significantly outperform the base U-Net. The ImprovedUNet—which combined SE blocks with Attention Gates—introduced training instability and worsened generalization, particularly for *pr*. ConvLSTMNet initially performed well and converged with lower epochs during training, but as seen in Figure 3, it overfit and failed to generalize to the unseen SSP245 scenario. Overall, our U-Net variant with dual-head decoders and CoordConv delivered the most consistent results across metrics and scenarios, earning a private leaderboard RMSE of 0.9800 and placing 33rd out of 83 teams.

5.5 Ablation Studies

To evaluate the contribution of individual architectural and training decisions, we conducted targeted ablation studies. Below, we summarize the key findings and lessons learned.

Shared vs. Dual Decoder Heads. Using a shared decoder led to unstable training due to conflicting gradient signals from the two targets—tas and pr. Introducing dual decoder heads enabled task-specific optimization and reduced interference, yielding greater performance in pr RMSE. This was one of the most impactful changes in our architecture.

CoordConv Inputs. Appending normalized (x, y) coordinate channels to the input improved tas accuracy and enhanced spatial generalization across all SSP scenarios. Removing CoordConv consistently increased tas RMSE by 0.1, confirming its value in embedding geospatial context early.

SE Blocks. Squeeze-and-Excitation (SE) blocks provided minor early-epoch improvements and slightly enhanced pr metrics by reweighting noisy features. However, they increased computational overhead and did not improve final performance. We dropped them in favor of a leaner architecture.

Attention Modules. Attention gates on skip connections were explored for selective feature passing. Despite theoretical appeal, they introduced training instability and frequently caused gradient divergence—especially when predicting sparse pr. These modules were ultimately abandoned due to low reliability.

Loss Function and Weighting. Standard MSE loss consistently outperformed alternatives such as SmoothL1, Huber, and Tweedie. Robust losses did not improve generalization and sometimes degraded tas predictions. Additionally, reweighting the loss to favor pr (e.g., 70/30) did not reduce pr RMSE and came at the cost of worse tas performance. Equal weighting remained the most balanced choice.

Deeper U-Net Variants. We experimented with a deeper U-Net (3-level encoder-decoder and wider bottleneck) to increase capacity. While it achieved slight validation gains, it also introduced longer training times and higher overfitting risk. Given the marginal improvement, we retained the simpler 2-level configuration.

Log-Transformation of pr. Applying a log-transform to precipitation introduced inverse-complexity at inference and led to training instability. RMSE scores worsened despite theoretical benefits for handling skew. We reverted to using raw pr values with MSE loss.

Quantitative Highlights.

- Dual decoder heads reduced pr RMSE by 0.2 and stabilized training.
- CoordConv improved tas RMSE by 0.1 and enhanced spatial pattern recognition.
- MSE loss was consistently more stable and accurate than robust loss alternatives.

Qualitative Insights. Visual inspection of predicted maps showed strong alignment with tas spatial trends and general precipitation regions. However, the model struggled to localize high-intensity pr spikes, reflecting the inherent sparsity and noise of precipitation data.

Negative Results. Several interventions yielded no benefits or degraded performance:

- Log-transform of pr increased training complexity and reduced accuracy.
- Attention modules caused gradient instability and poor reproducibility.
- SE blocks slowed training with minimal improvement in final metrics.

6 Discussion

What Was Learned

We found that simple, well-aligned architectures outperformed deeper or more complex ones when tailored to the task. Our dual-head decoder design significantly improved training stability by reducing gradient interference between *tas* and *pr*. Adding CoordConv inputs improved spatial awareness early in the network, helping the model generalize across SSP scenarios.

While we explored temporal architectures like ConvLSTM, we found them prone to overfitting to SSP-specific trends, making spatial-only models more robust under distribution shift. Similarly, techniques like log-transforms, robust loss functions, and attention gates did not improve generalization and often added instability or unnecessary complexity. Our findings reinforce that empirical rigor and reproducibility matter more than theoretical appeal—especially when working with noisy scientific data.

Strengths and Limitations

Our model achieved strong validation and leaderboard performance (RMSE = 0.9800, rank 33/83) by emphasizing task separation, early spatial context, and stable training. CosineAnnealingLR with early stopping enabled reproducible convergence and prevented overfitting. However, we did not systematically tune architectural hyperparameters (e.g., depth, width), and our model operates on single-month inputs, limiting temporal understanding. It also lacks uncertainty quantification, which is important for noisy targets like precipitation.

Innovation Summary

We extended the baseline by introducing a dual-head U-Net tailored to the statistical differences of *tas* and *pr*, and appended CoordConv channels to improve spatial grounding. Empirical evaluation led us to reject additions like robust losses, log-transforms, and attention due to either instability or negligible benefit. These design decisions prioritized validation robustness, interpretability, and generalization across SSPs.

Future Work

Future directions include exploring temporal modeling via ConvLSTMs or Transformer-based encoders to capture long-range dependencies across months. Integrating uncertainty-aware loss functions (e.g., quantile loss) may improve confidence estimation for pr. Multi-step forecasting, either through sequence-to-sequence models or autoregressive decoding, could enable richer temporal forecasting. Finally, data augmentation or ensemble learning may further improve model robustness to unseen spatial or scenario shifts.

7 Contributions

Adrian: Major contributor to model architecture building and hyperparameter tuning. Contributed heavily to the final presentation and write-up. Project lead. Oversaw all team members throughout the entirety of the project.

Ethan: Main contributor to model architecture design and implementation. Built and tested all core models, maintained the training pipeline, and assisted with the final presentation and report.

Ryan: Contributed to model design choices along with creation of loss curves. Contributed to milestone, presentation, and final report write-ups.

Andrew: Conducted hyperparameter tuning, experimented with temporal features and larger models, and contributed to the final presentation and write-up.

References

[1] Jianhao Lin and George Michailidis. A multi-task encoder–dual-decoder framework for mixed frequency data prediction. *International Journal of Forecasting*, 40(3):942–957, 2024. doi:10.1016/j.ijforecast.2023.08.003.

[2] Rosanne Liu, Joel Lehman, Piero Molino, Felix Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/60106888f8977b71e1f15db7bc9a88d1-Paper.pdf.

[3] Utku Ozbulak, Burak Akin, Bülent Sankur, and Hazim Kemal Ekenel. Use-net: Incorporating squeeze-and-excitation blocks into u-net for prostate zonal segmentation of multi-institutional mri datasets. *arXiv preprint arXiv:1904.08254*, 2019.

[4] Dongsheng Wang, Yanyan Yang, and Shengchao Ning. Deepstcl: A deep spatio-temporal convlstm for travel demand prediction. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018. doi:10.1109/IJCNN.2018.8489530.

[5] Bing Yu, Haoteng Yin, and Zhanxing Zhu. St-unet: A spatio-temporal u-network for graph-structured time series modeling. 2021. URL https://arxiv.org/abs/1903.05631.