CSE 151B Project Final Report

Github Link

Colin Commans [redacted]

Victor Limouzi [redacted]

Jeffrey Meredith [redacted]

Anthony Pertsel [redacted]

Abstract

Deep learning models are now the state of the art for many challenging climate and weather prediction tasks. One of the most important issues in this domain is climate prediction in the context of rising levels of various gases such as CO2, CH4, and more. We propose several deep learning model architectures that we believe are best suited to this task and our motivations for experimenting with these architectures. In this paper we discuss several model types that we experimented with and tested on the climate dataset provided in the Kaggle competition: (1) a convolutional baseline with ResNet-style blocks; (2) a CNN-Transformer with positional and temporal encodings; and (3) a Fourier Neural Operator (FNO) with SpectralConv2d blocks. Our best model, which used the FNO architecture, got a private Kaggle score of 1.0262, achieving 40th on the leaderboard. The results we obtained give key insight as to what model architectures are best suited to climate prediction tasks, showcasing the importance of exploiting data locality and temporality in obtaining quality prediction results.

1 Introduction

One of the key issues in climate science right now is predicting the effects of climate change, in the form of increasing levels of greenhouse gases in the atmosphere due to human activity, will have on our planet. The task we tackled was given 5 inputs (CO2, SO2, CH4, Black Carbon, and Solar Radiation) on a latitude-longitude grid, predict surface air temperature and precipitation for the points on the grid. Solving this task will greatly increase our ability to understand how our climate will change, which is crucial to forming plans to mitigate climate change. If we are able to accurately predict surface temperature and precipitation given gas levels in the atmosphere, we can determine what levels of these gases can be considered safe and create clear goals to ensure our planet remains safe.

The provided starter code preprocesses data, normalizes inputs, splits data into training and validation sets, and implements a simple CNN with residual layers. However, this starter approach had several limitations:

- Combined Predictions: Initially, the model simultaneously predicted temperature and precipitation, limiting its performance. We found training separate models for each improved accuracy significantly.
- Irrelevant Inputs: The baseline included Black Carbon and SO2 readings, mostly zeros, which provided minimal predictive power. Our analysis indicated improved results when omitting these inputs.
- Spatial and Temporal Limitations: The baseline CNN model neglected temporal sequences and essential spatial correlations inherent in global climate data. Specifically, it failed to recognize relationships between grid points wrapping around the globe and across consecutive time steps.

To address these issues, our key contributions are:

- Data Processing Improvements: Using separate models for temperature and precipitation predictions and omitting low-value inputs like Black Carbon and SO2, enhancing model performance.
- CNN-Transformer Architecture (ViT): Exploiting spatial and temporal relationships via attention mechanisms across spatial patches and incorporating temporal embeddings.
- Fourier Neural Operator (FNO) Architecture: Leveraging frequency-space transformations to effectively model global spatial relationships.

2 Problem Statement

We can think of our inputs in two ways: (1) ignoring all spatial and temporal and situational information, each input is simply a vector in $\mathbb{R}^5$ representing each forcing; or (2) the more-complete picture is that each of these data points is at a particular latitude and longitude on earth, which is at a particular month since January 2015, which is a prediction from one of three physics-based simulations / ensembles, which is assuming one of three levels of emissions scenarios. Hence we can think of our entire input as a vector in $\mathbb{R}^5 \times \mathbb{Z}^2 \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$.[1]

In the same way, our output is either simply a vector in $\mathbb{R}^2$ representing temperature and precipitation, or it is a complete prediction of these variables over (lat, long) pairs on earth at each month (this is exactly what the Kaggle competition is expecting as input when calculating scores), which would be a vector in $\mathbb{R}^2 \times \mathbb{Z}^2 \times \mathbb{Z}$.

To split the train/validation data, we just used the starter code split method, with the same split of 0.1 for validation. We didn't deviate from the starter code for normalizing inputs/outputs either. Our only deviation was regarding splitting the models for some experiment, so the output was only either precipitation or temperature for the model. In addition, we also dropped the BC / SO2 columns for some experiments, as these values were mostly zeros.

3 Methods

3.1 Task and Objective

The task of our deep learning models is to accurately predict climate data over a ten-year span. Concretely, let each example consist of an input tensor $X \in \mathbb{R}^{C_{\text{in}} \times H \times W}$ representing climate forcings over a spatial grid at a given timestep, and a target tensor $Y \in \mathbb{R}^{C_{\text{out}} \times H \times W}$ representing the temperature and precipitation at that timestep. Our goal is to learn a function $f : \mathbb{R}^{C_{\text{in}} \times H \times W} \to \mathbb{R}^{C_{\text{out}} \times H \times W}$, parameterized by weights θ, such that $\widehat{Y} := f(X;\theta)$ approximates Y.

We approach this task as a supervised regression problem and hence train by minimizing over all training examples. Thus our optimization problem looks like:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L_{MSE}\big(f(X^{(i)};\theta), Y^{(i)}\big)$$

where L_{MSE} is the mean squared error loss function:

$$L_{MSE}(\widehat{Y}, Y) = \frac{1}{C_{out}HW} \big\|\widehat{Y} - Y\big\|_2^2,$$

3.2 Fourier Neural Operator

3.2.1 Background and Motivation

We began working with this choice of architecture with the Spherical Fourier Neural Operator (SFNO) introduced in Bonev et al. [1]. The main contribution of the paper was its general framework to

[1]Here we represent the latitude and longitude position as a pair in $\mathbb{Z}^2$ just to emphasize the discreteness of this feature; in reality this is a pair of two real numbers.

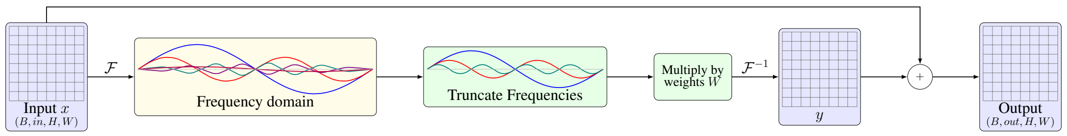


Figure 1: SpectralConv2d process: the spatial-domain input grid is transformed via 2D Discrete Fourier transform $\mathcal{F}$ into frequency-domain coefficients, which after truncation are multiplied by learnable weights W, then inverse-transformed via $\mathcal{F}^{-1}$ back to the spatial domain, producing y. A residual connection adds the original input x to yield the final output.

predict (1) spatio-temporal data (2) over long time periods and (3) on spherical geometry, which the authors applied to climate data. All three of these factors are relevant for this project's task, thus we proceeded to implement it. However, we were unable to effectively use the `torch-harmonics` library from [1] in our models nor able to implement the SFNO from scratch. However, we decided to simplify our approach and just look at a Fourier Neural Operator (FNO) introduced in Li et al. [2]. This still has benefits (1) and (2) of the SFNO, but not explicitly working now in a spherical domain.

The FNO, as the name implies, leverages a Fourier transform to send the data to frequency space. This allows any consequent convolutions to become "global" and theoretically capture long-term patterns. In particular, this contrasts with a standard CNN which operates (locally) only in the signal space and does not incorporate anything temporal. As a side note, the FNO in [2] was developed to numerically solve PDEs in a branch of applied mathematics called spectral methods, which is relevant to our climate prediction task as we treat the climate as a smooth function to approximate with known dynamics and spatial and temporal correlations. Therefore, we can adopt the FNO (similarly to [1]) to use these spectral properties on our coarse latitude-longitude grid to learn the global patterns more effectively than the purely local convolutions in our baseline.

3.2.2 Architecture

The key to our FNO is the SpectralConv2d block. Briefly, given input x of shape (B, C, H, W), we compute its 2D discrete Fourier transform and take the lowest n frequencies in the x dimension and m frequencies in the y dimension. This gives us $n \times m$ Fourier coefficients that we can sum and multiply by learnable weights to create an output y after taking an inverse Fourier transform. We also add a residual connection (i.e. add original input x to y at the end) to help gradient flow. Figure 1 visualizes this process of spectral convolution. Some more implementation details include:

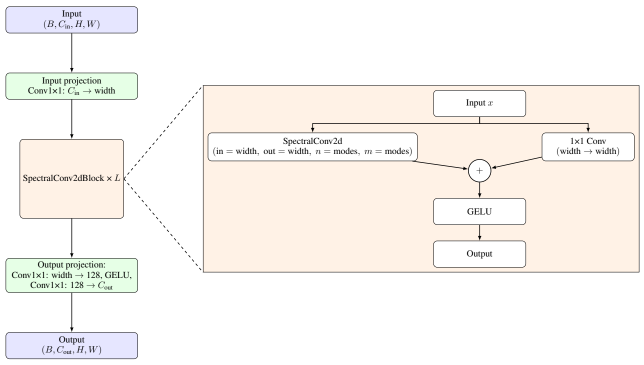
- Our Fourier coefficients are complex numbers, so we are in fact doing complex multiplication with our learnable weights rather than straight matrix multiplication and each weight matrix W has real $\Re(W)$ and imaginary $\Im(W)$ counterparts.
- The numbers n and m are called the modes, and we typically have $n = m$.
- We normalize our Fourier transform and inverse transform by the "orthonormal" mode, as the usual transform does not normalize in the forward direction and instead recovers it in the inverse. However, since we are using our weights after the transform, we first normalize.

Now building the entire FNO model, see Figure 2 for the architectural diagram.

3.3 Vision Transformer

The key elements in this model architecture consist of (1) a dual position encoding system to properly capture geographic position information in climate data, and (2) a time embedding, a learnable representation for each month to capture seasonal patterns. Along with these embeddings, we used a CNN-Transformer Hybrid, with both CNNs and an encoder-decoder transformer.

The motivation behind our model design was to find a way to capture the temporal seasonality from the climate, which is why we included the time embedding. In addition, we used attention and an encoder-decoder transformer model to capture the spatial relationships in the data. The transformer is better at capturing global connections than a CNN alone, and can relate image patches to each other across the whole image. The CNN component of the model, built to extract features from the grid, has 3 pathways to capture different local patterns at different resolutions. The goal was to be



Input
(B, C_{in}, H, W)
Input projection
Conv1×1: $C_{\text{in}} \to$ width
SpectralConv2dBlock × L
Output projection:
Conv1×1: width $\to$ 128, GELU,
Conv1×1: 128 $\to C_{\text{out}}$
Output
$(B, C_{\text{out}}, H, W)$
Input x
SpectralConv2d
(in = width, out = width, n = modes, m = modes)
1×1 Conv
(width $\to$ width)
+
GELU
Output

Figure 2: FNO2D architecture: Each SpectralConv2dBlock splits its input into a spectral branch and a 1×1 convolution branch, sums them, then applies GELU. Note that "width" and "modes" are hyperparameters controlling intermediate channels and frequency truncation respectively.

able to extract patterns at various scales to get local and global feature information. See Appendix A (Figure 8) for a diagram of the architecture.

4 Experiments

4.1 Baselines

- **Starter Code ResNet-CNN** - We first tested the baseline provided in the starter code. This model is composed of an initial convolution layer followed by 4 residual blocks. Each residual block contains 2 convolution layers with batch normalization after each layer and a residual connection between the beginning and end of the block. This model achieved a public Kaggle score of **3.33** when trained for 10 epochs and **2.02** on 40 epochs.

4.2 Evaluation

Our primary measure of how the model performs is its score on the Kaggle leaderboard. This score is a combination of metrics for each output variable calculated in the starter code: (1) Monthly Area-Weighted RMSE, (2) Decadal Mean Area-Weighted RMSE, and (3) Decadal Standard Deviation Area-Weighted MAE. In particular, we have the formula:

$$\text{Kaggle Score} = \frac{1}{2}\left[\frac{1}{10}\text{RMSE}_{tas} + \text{Time-Mean}_{tas} + \text{Time-Std}_{tas}\right] + \frac{1}{2}\left[\frac{1}{10}\text{RMSE}_{pr} + \text{Time-Mean}_{pr} + \frac{3}{4}\text{Time-Std}_{pr}\right]$$

Hence any progress (and all other things being equal) in one of these metrics indicates a better-performing model to us.

4.3 Implementation details

4.3.1 Computation Details

We trained our models in different environments:

- **Starter Code**: This model was trained on the UCSD Datahub GPUs, taking about 25 seconds per epoch.
- **FNO**: This model was trained on the UCSD Datahub GPUs, and it was very fast taking only about 7 seconds per epoch.
- **ViT**: This model was trained on a T4 GPU with the free tier of Google Colab. It took around 1 minute per epoch to train this model.

4.3.2 Hyperparameters

We summarize in Table 1 the final hyperparameters of the models we discuss in this report.

Some more model-specific hyperparameters include:

Table 1: Hyperparameters of models

Model	Epochs	LR	Batch Size	Optimizer	Dropout	Depth
	Hyperparameter					
Starter	40	1e-3	64	Adam	0.1	4
FNO	50	1e-3	32	Adam	0.0	4
ViT	40	1e-3	64	Adam	0.1	6

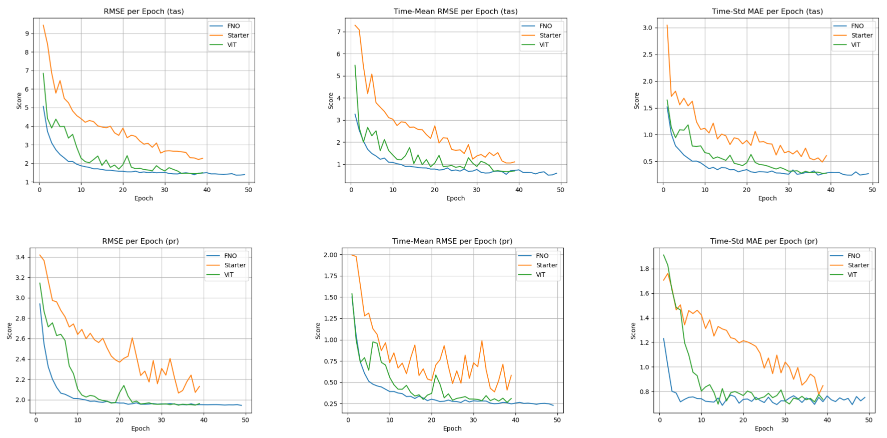


Figure 3: Validation metrics of FNO (blue), ViT (Green), and Starter Code (Orange). Columns are RMSE, Time-Mean RMSE, and Time-Std MAE respectively and rows are temperature and precipitation respectively.

- **FNO**: width = 64, modes = 12
- **ViT**: hidden dimension = 256, attention heads = 8, patch size = 4×4

To make these choices, we just began with some initial hyperparameters, with usually smaller depths, and just tuned based on trends we saw with training / validation data. For the ViT, we played with doubling the depth and number of attention heads, but found no improvement.

4.4 Results

Result 1: Kaggle Score
The results on the Kaggle leaderboard (both public and private) are summarized in Table 2. The FNO performed best, giving our team a rank of 40 in the competition.

Table 2: Kaggle Scores

Model	**Private Score**	**Public Score**
Starter	2.1516	2.0233
FNO	1.0262	0.9528
ViT	1.1002	0.9669

Result 2: Validation Metrics
We also show in Figure 3 the validation metrics over time of our models. Table 3 shows the values of these metrics after training for each model.

Result 3: Prediction Visualizations
Finally, we visualize the difference between the validation and the ground truth Time-Mean RMSE in Figure 4. We do the same for Time-Std MAE in Figure 5.

Table 3: Validation metrics at end of training in the form tas / pr

Model	RMSE	Time-Mean RMSE	Time-Std MAE
Starter	2.2632 / 2.1301	1.1000 / 0.5814	0.6067 / 0.8473
FNO	1.3880 / 1.9437	0.5867 / 0.2286	0.2688 / 0.7507
ViT	1.4849 / 1.9611	0.6892 / 0.3095	0.2796 / 0.7340

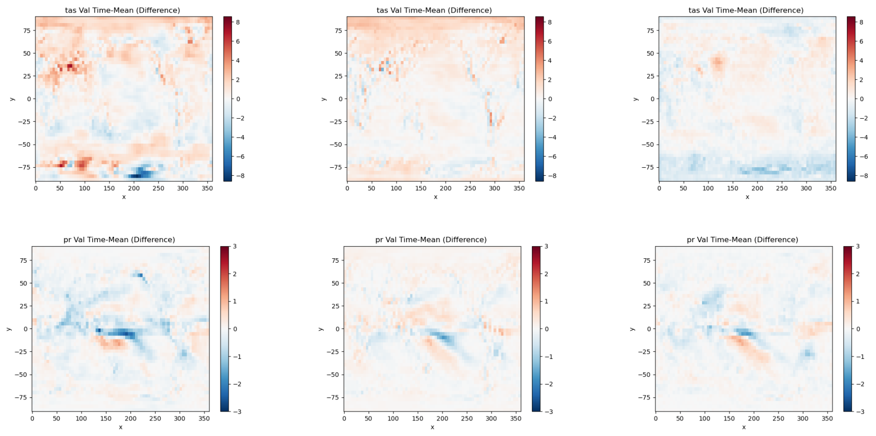

tas Val Time-Mean (Difference)
tas Val Time-Mean (Difference)
tas Val Time-Mean (Difference)
pr Val Time-Mean (Difference)
pr Val Time-Mean (Difference)
pr Val Time-Mean (Difference)
x
y

Figure 4: Difference maps on the lat-long grid of the Time-Mean RMSE. The top row is temperature and the bottom row is precipitation, and the same scale is used per row for easier comparisons. The columns are the starter code, FNO, and ViT respectively

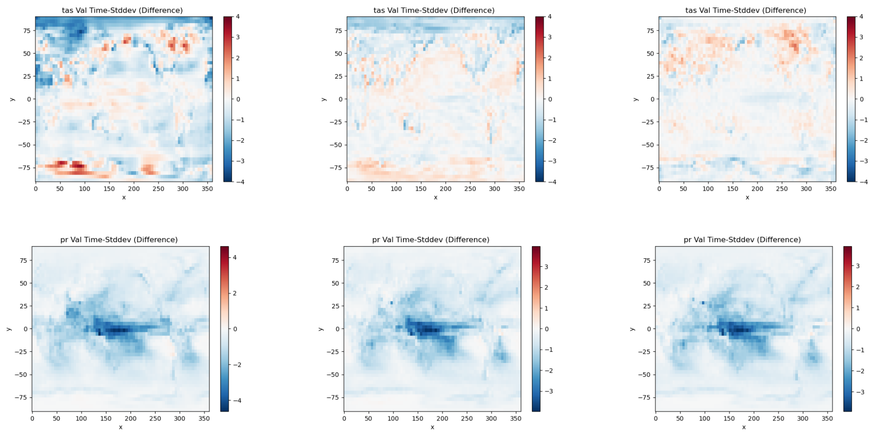

tas Val Time-Stddev (Difference)
tas Val Time-Stddev (Difference)
tas Val Time-Stddev (Difference)
pr Val Time-Stddev (Difference)
pr Val Time-Stddev (Difference)
pr Val Time-Stddev (Difference)
x
y

Figure 5: Difference maps on the lat-long grid of the Time-Std MAE. The top row is temperature and the bottom row is precipitation, and the same scale is used per row for easier comparisons. The columns are the starter code, FNO, and ViT respectively

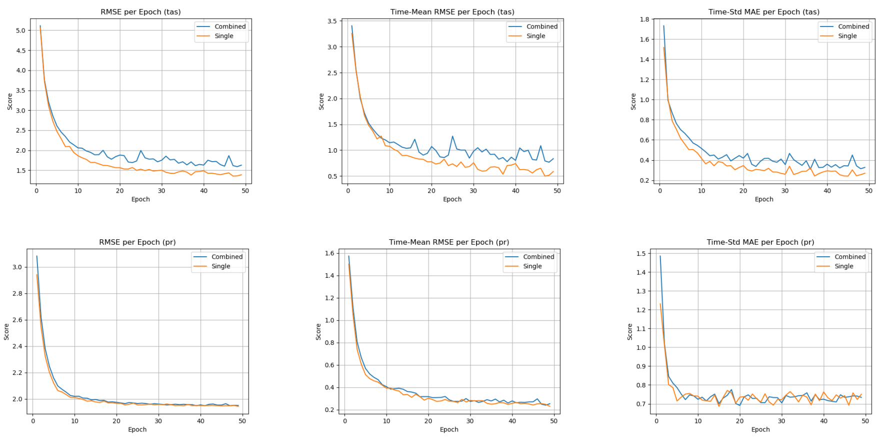


Figure 6: Validation metrics of a 2-output FNO (blue) and of two 1-output FNOs (orange). Columns are RMSE, Time-Mean RMSE, and Time-Std MAE respectively and rows are temperature and precipitation respectively.

4.5 Ablations

4.5.1 Temporal VS No Temporal Vision Transformer

A small ablation experiment was performed to determine whether the Vision Transformer would benefit from attention over multiple timesteps. Using a window size of $w = 12$, the model performed attention over 12 timesteps $\times$ 216 patches $=$ 2592 tokens per example. With temporal attention, the ViT got **1.0629** on Kaggle's public test, which was no improvement over the standard ViT's score of **0.967**.

4.5.2 FNO VS LSTM-FNO Models

Another ablation experiment was evaluating the addition of LSTM to the FNO model to introduce the temporal dimension. This model also used a window size of $w = 12$. The FNO layers were the exact same as the FNO model described above. 4 LSTM layers were used with a hidden dimension of 128. The LSTM-FNO model got a public Kaggle score of **1.152**, which was no improvement over our standard FNO model.

4.5.3 Combined VS Split Model

Here we justify the design choices of the FNO. First, we compare the results of a FNO that outputs both tas and pr with a FNO trained just to predict tas and another to just predict pr. Notably we keep all other hyperparameters the same. Figure 6 shows the validation metrics of these two approaches, and in all cases the split model performs better over epochs and crucially at the final epoch.

In terms of validation Kaggle scores, the combined model scored a **1.324** on tas and **0.998** on pr. In contrast, the tas-only model scored a **0.994** and the pr-only model scored a **0.986**. Hence the split models unequivocally perform better than the combined model.

4.5.4 Include BC & SO2 VS Exclude

We also made the decision to not include black carbon and SO2 as inputs to the FNO due to them being nearly all zeros. To justify this choice, we trained identical models that only differed by including or excluding BC and SO2. Figure 7 summarizes our results, where we show the validation metrics of these two models over time. Notably, the 3-input model always began with worse metrics but is able to catch up and beat the 5-input model by the end of 50 epochs (excluding tas RMSE, where the 5-input model scores better by 0.037).

In terms of validation Kaggle scores, the 5-input model scored a **1.089** on tas and **1.114** on pr; the 3-input model scored a **0.994** on tas and **0.986** on pr. Thus this design choice is justified.

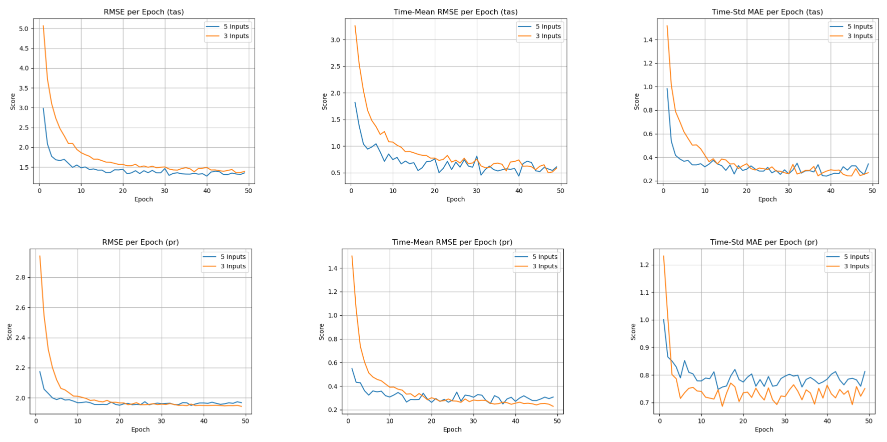


Figure 7: Validation metrics of a 5-input FNO (blue) and a 3-input FNO (orange), excluding BC and SO2. Columns are RMSE, Time-Mean RMSE, and Time-Std MAE respectively and rows are temperature and precipitation respectively.

5 Discussion

Our results and experiments show that the FNO consistently outperforms both the baseline and ViT on the prediction task. The ViT yields competitive results, but slightly lags behind FNO, especially on temperature metrics at the poles (see Figure 4). Examining the validation metric curves in Figure 3 also demonstrates the FNO trains much smoother than the other models. A recurring challenge across all models, however, is the precipitation prediction, especially around the equator (see Figure 5). This became the main bottleneck for improving the overall Kaggle score. Ablation studies reinforce that splitting models for separate temperature and precipitation predictions and excluding near-zero inputs (BC, SO2) yield clear benefits.

Limitations. Several factors likely contribute to the persistent challenges observed:

- **Precipitation variability and skewed distribution.** Precipitation often exhibits heavy-tailed or highly skewed behavior, especially in equatorial regions with convective patterns. Using MSE as the sole objective does not take into account these patterns.
- **Static, per-timestep prediction without explicit temporal dynamics.** Although we applied temporal embeddings in ViT and experimented with LSTM-FNO, our main FNO and baseline treat each timestep largely independently and the model cannot fully leverage temporal correlations, seasonal cycles, or interannual dependencies.
- **Approximating spherical geometry on a planar grid.** Implementing a full SFNO was not feasible, but operating on a regular lat-long grid introduces distortions near poles.
- **Computational and data limitations.** Training was constrained by available GPU resources and time, limiting extensive hyperparameter searches (e.g., deeper FNO widths or more modes). The ten-year span and ensemble scenarios provide a finite data regime; data-hungry architectures (like large Transformers) may underfit or overfit without more data or regularization.

Future work. Possible directions to address the above limitations include:

- **Custom loss functions.** Construct a loss function that is exactly the Kaggle score formula and directly optimize that when training.
- **Spherical and multi-scale architectures.** Implement a spherical FNO or use spherical harmonics to better respect Earth geometry.
- **Stronger temporal modeling.** Extend FNO with temporal convolution or use spatio-temporal attention mechanisms over sliding windows beyond simple embeddings.

6 Contributions

All of us met once a week to try experiments, discuss results, and discuss next steps. We also all split up the milestone report / final report. In terms of who worked on which experiments:

Anthony worked on toroidal convolution experiments, experiments to get SFNO working with torch-harmonics, experiments with sequential ViT models, and experiments with spatio-temporal ViT. Victor worked with the spatio-temporal ViT, working mainly on the architecture of this model, including the temporal + positional embeddings, CNN architecture, and hyper-parameter tuning. Jeffrey worked on various experiments including LSTM + CNN, LSTM + FNO, ViT with causal attention-masking, and the FNO architecture and hyperparameter tuning. Colin worked on the Conv3D model, torch-harmonics SFNO experiments, experimented with splitting models for ViT, and worked on the FNO architecture.

References

[1] Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical fourier neural operators: Learning stable dynamics on the sphere, 2023. URL `https://arxiv.org/abs/2306.03838`.

[2] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *CoRR*, abs/2010.08895, 2020. URL `https://arxiv.org/abs/2010.08895`.

Appendix

A Vision Transformer Architecture

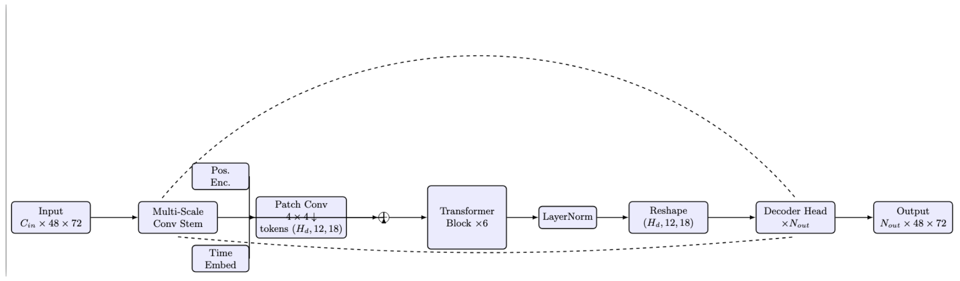


Figure 8: Spatiotemporal Vision Transformer