Missouri University of Science and Technology

Paleoclimate Reconstruction and Forecasting

Viktor Butkovich

Computer Engineering 5310 – Computational Intelligence

Dr. Steven Corns

15 May 2025

# Summary

This project seeks to apply computational intelligence methods such as genetic algorithms and neural networks to the task of predicting temperatures and glacial cycles (ice ages) over the next several hundred-thousand years. Given my skillset and strong computer science background, I selected a challenging project topic and placed a particular emphasis on data acquisition, exploration, and warehousing, in contrast with the common tendency of ML projects and papers to maximize performance through advanced techniques on pre-compiled benchmark datasets without a domain interest or understanding.

The machine learning task at hand here is to, using some regression or forecasting method, predict future climate patterns based on the last million years of climate data and simulated co-variates (such as future orbital parameters). As we will see, this prediction ends up omitting post-industrial measurements from training/validation data due to the anomalous behavior of the modern time period – while modern data is much more readily available and accurate, long-standing relationships such as the magnitude of carbon dioxide’s effect on the climate have greatly changed (carbon dioxide concentration is actually less strongly correlated with temperature than before industrialization).

While of interest, predicting climate patterns that would’ve occurred without the impact of human industry do not immediately seem applicable to our current global crisis – humans have significantly impacted the Earth’s climate, so any attempt to predict the future climate without accounting for recent human activity is very unlikely to be truly accurate. However, the process of making accurate forecasts based on pre-industrial conditions requires gaining a thorough understanding of past events that have shaped the Earth-climate system. Based on our geological record, in the last million years, the Earth has undergone at least 7 major glacial cycles, with quick heating or cooling of as much as 5 °C, sometimes occurring within several thousand years – this is some of the highest-quality data we will ever have on the Earth-climate system’s behavior under wildly varying conditions. While this project doesn’t quite answer the questions of how the Earth-climate system will respond to the modern age’s (relatively) lightning-fast inundation with carbon dioxide, or whether the next ice ages will even happen anymore, it hopes to provide a better understanding of the Earth-climate system as a whole.

# Context

There have been many examples of Earth’s geological record being used to attempt to understand the workings of the Earth-climate system under varying conditions. The presence of boulders and strange geological features across much of northern Europe, Asia, and North America have long signaled that much of the world used to be covered in ice. Scientists gradually gained an understanding, through dating samples with respect to climate proxies such as fossil pollen and ice cores, that the Earth has experienced wild temperature fluctuations in the last million years. This primarily consist of approximately 100k-year intervals of temperatures 5-8 °C colder than usual, followed by shorter 10k-year interglacial periods with warm temperatures.

In the 1920s, Serbian scientist Milutin Milankovitch [1] proposed a model for these fluctuations – he found that these temperature changes largely coincided with fluctuations in Earth’s orbit, such as its tilt (obliquity) or how circular it is (eccentricity). Milankovitch found that these slight changes affected the insolation received by the Earth from the sun – a slight decrease in sunlight could cause less ice to melt each summer, causing more landscape to be covered in reflective ice that further decreases the sunlight absorbed, resulting in a cascading feedback loop dropping temperatures by several degrees every thousand years, which is near-instant for geological time. These same effects apply in the reverse, with temperature increases resulting in ice melting and more sunlight absorbed. Notably, these orbital fluctuations seem to affect both temperature and CO2 levels, with CO2 fluctuations sometimes significantly lagging those of temperature, so CO2 is not singlehandedly causing the temperature changes.

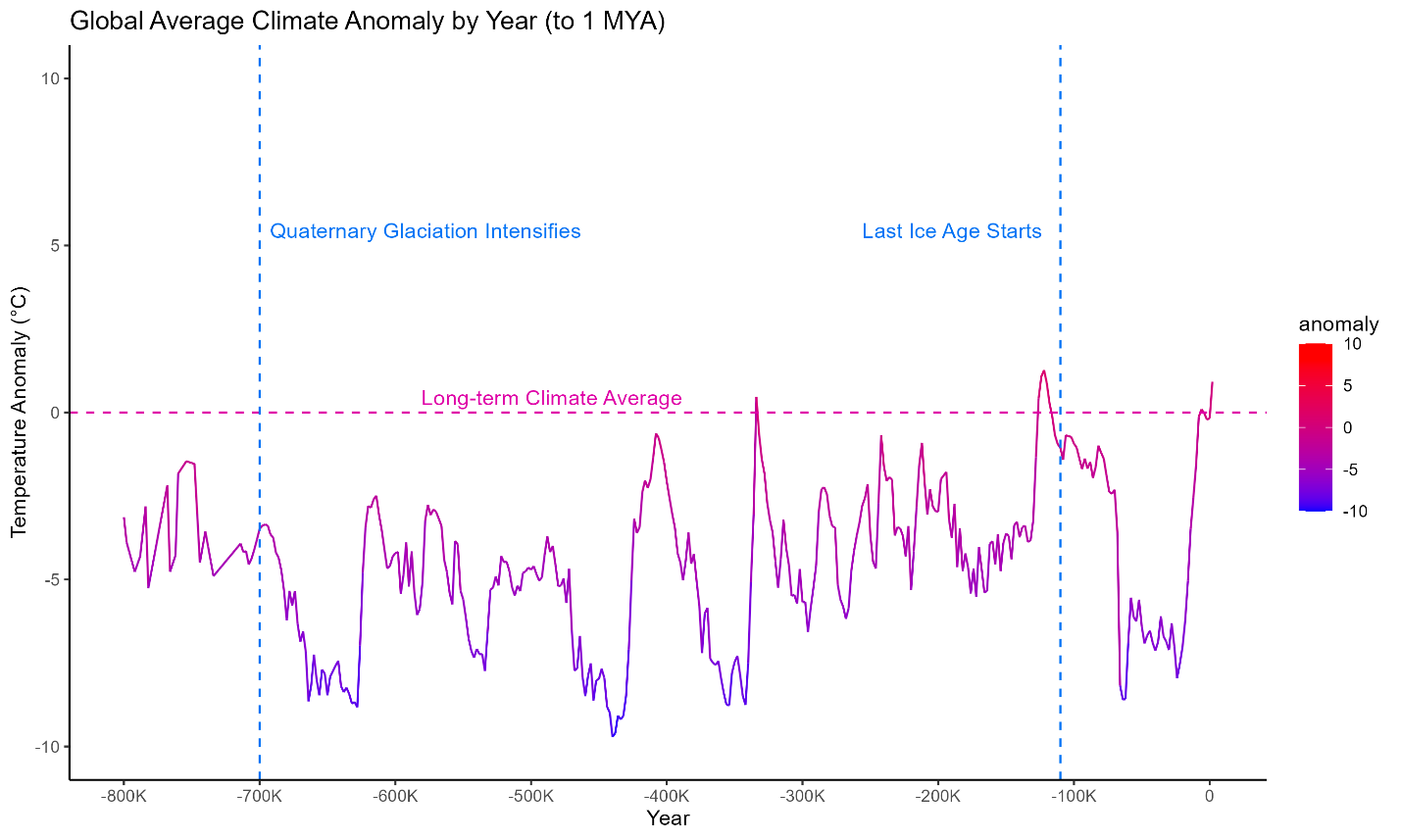


Figure 1: Plot of Earth’s anomaly (°C deviation from long-term climate average) since 800k B.C.

While Milankovitch’s theory is still thought to be largely correct, more recent theories have found more immediate causes for the initiation and termination of glacial cycles. Gerald Marsh [2] theorizes that cosmic ray flux, the extent to which cosmic radiation reaches the Earth, may be the immediate cause of glacial cycle initiation/termination, since cosmic rays result in more low-cloud cover, decreasing sunlight reaching the Earth’s surface, initiating a cascade set up by the configuration of Earth’s orbit. This explains observations that glacial cycles sometimes started or ended some time before the corresponding orbital fluctuations occurred.

This background research aided in informed selection of input features for the climate forecasting, with features such as eccentricity, obliquity, and solar modulation (cosmic flux proxy) being the main drivers of long-term climate variation.

# Code

This project contains a variety of code module. These are designed to work as independently as possible – cloning the repository includes pre-computed .csv files resulting from the ETL and preprocessing steps, meaning predictions can be run with minimal setup, followed by visualizing the new predictions.

## ETL

The Python ETL (Extract, Transform, Load) module performs a variety of initial pre-processing steps required in compiling the data from its respective sources.

* **Extract**: Load data from Temp12k dataset, Berkeley Earth, Sint2000, simulated Milankovitch orbital parameters, and various CO2 and Beryllium-10 datasets (all pulled automatically or included in repository)
* **Transform**: Elaborate data cleaning needs to occur due to the source data’s occasionally sparse resolutions and low accuracy, in addition to the wide variety of structured and semi-structured source data formats. This includes date calculations, extrapolating missing values, detecting/removing samples flagged by researchers as unreliable, calculating the climate anomaly for temperature estimates in each location, and joining/splitting the resulting tables into the desired data warehouse format.
* **Load**: Create the data warehouse and populate with the extracted/transformed data. This interacts with a SQLite database sub-module which handles database connection and any required insertions/updates/deletions.

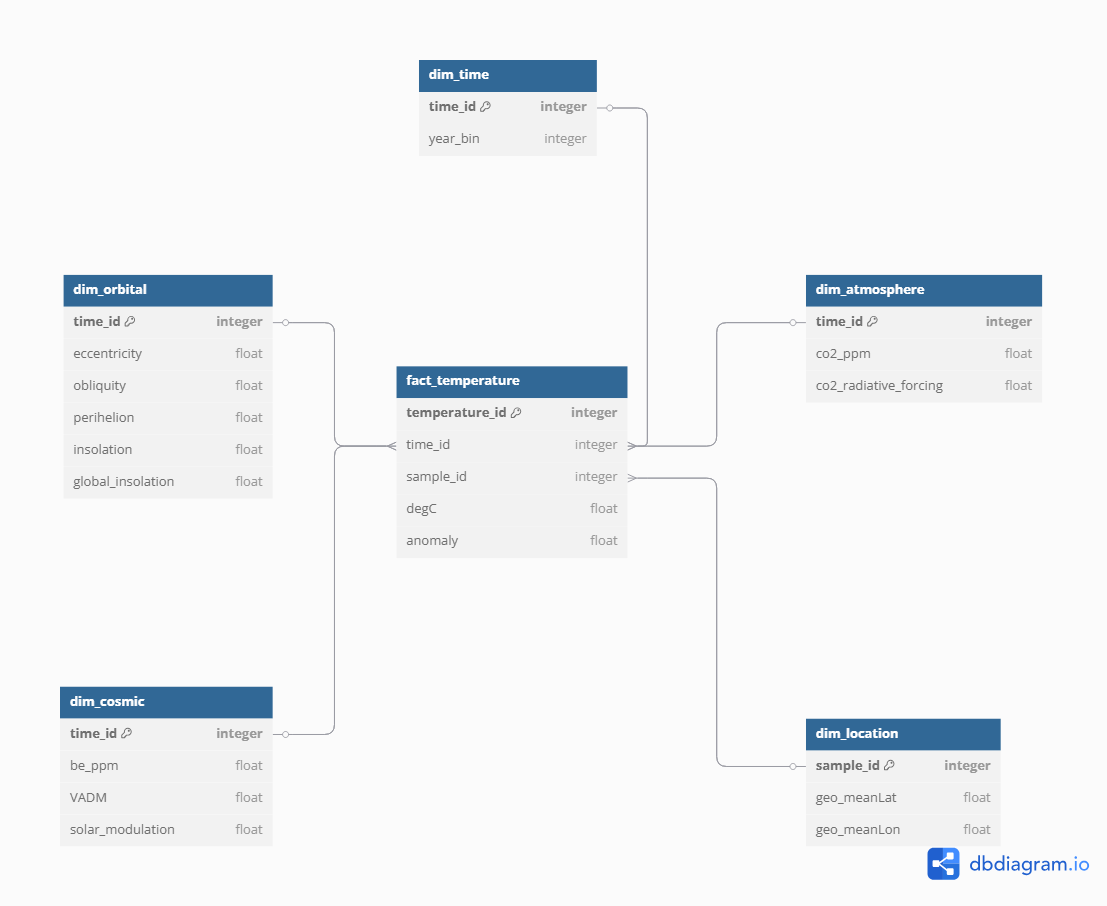


Figure 2: Data warehouse schema providing a central repository to flexibly analyze pre-processed, integrated data

## View Creation

The Python view creation module interacts with the data warehouse with queries to compile desired datasets. In addition, this performs some final pre-processing steps that fundamentally change the source data (while the data warehouse is to be a reflection of the source data), such as standardizing values (mean 0, standard deviation 1), global aggregation, and rounding.

This creates several datasets to be used in visualization and ML tasks:

* **raw\_global\_anomaly\_view.parquet**: Raw data warehouse data with anomaly values aggregated globally (average all samples taken at a particular time), for visualization tasks where original units are desired
* **long\_term\_global\_anomaly\_view.parquet**: Extensively preprocessed data for visualization tasks where simulated future values (after 2025) are desired
* **long\_term\_global\_anomaly\_view\_enriched\_training.parquet**: Extensively preprocessed data with added engineered features (deltas, lags) for ML tasks

The following prediction methods create .csv prediction files where predicted anomaly values are displayed alongside actual ones (where known), which are used as inputs for the visualization module. The MSE is recorded for each non-ARIMA model and used as the primary metric for model comparison.

## Classical Regression

Simple R module performing k-fold linear regression (k=5) to predict anomaly from stepwise-selected training features. This contains an experiment, with a lagged linear model and a normal linear model, with the lagged model including lagged values of the target variable, anomaly, and the normal one excluding these, to evaluate the feature’s effects (all other models used lagged values as well). Past experiments tried polynomial regression, but these exhibited excessive overfitting, and development was focused on computational intelligence techniques.

Resulted in a validation MSE of 3.56 (lagged) and 3.62 (not lagged – including lagged target variable values helped out-of-sample performance)

## Time Series Forecasting

Simple R module performing ARIMA and ARIMAX forecasting to predict future values of anomaly. Pure forecasting methods were found to present many challenges with respect to validation and calculating predictive performance on past data, which were beyond the scope of this project, so this module was not developed beyond basic prediction.

## Neural Regression (Torch forecasting)

Python module performing k-fold regression (k=10) with neural networks to predict anomaly from training features. Each of the 10 folds trains with a partition of 90% of the training data, using the other 10% as validation. MSE is calculated with the average MSE of each fold on that fold’s validation set. The training uses these hyper-parameters, which were chosen by trial and error:

* **Layers**: 18 (input layer) -> 64 -> 32 -> 1 (output layer)
  + The output signal of the output layer’s single neuron is the prediction – model then adjusts parameters with back-propagation based on the errors in the batch
    - A very large batch size of 512 means that most of the training set is involved in each forward/backward pass, which results in faster training
* **Epochs**: 2000
  + Number of iterations of training/adjusting parameters before final prediction
* **Patience**: 10
  + Training is terminated early if at least 10 epochs in a row fail to improve the validation MSE, discouraging overfitting
* **Adam Learning Rate**: 0.001
  + Learning rate of Adam optimizer, which optimizes training by adjusting how quickly model parameters change – helps train faster initially and train more precisely near convergence
* **Regularization Lambda**: 0.05
  + Penalty applied to training loss proportional to the sum of the squared model parameters
    - Discourages large parameter sizes, resulting in strongly fitting only when this is sufficiently rewarded by performance gain, rather than overfitting to weak patterns
    - Training loss = MSE on training set + Σ(parameters2)
    - Penalty not applied to validation MSE, so it is not used in final comparison

Resulted in a validation MSE of 1.98

## Genetically Optimized Neural Regression (Genetic Torch forecasting)

Python module performing k-fold regression (k=10) with neural networks to predict anomaly from training features, with the neural network hyperparameters being optimized through a genetic algorithm (GA). The GA used these hyperparameters, which were (ironically) chosen by trial and error (more advanced methods can actually optimize the GA hyperparameters themselves throughout the evolution).

* **Population size**: 20
  + 20 neural network configurations were evaluated each generation
  + Each generation, 20 new individuals are created through mutation/recombination of existing individuals, and these new individuals are evaluated. Then, keep the 20 best individuals of the 40 present
  + Utilizes full elitism: Individuals from past generations can continue to survive as long as they remain above 50th percentile fitness
* **Mutation rate**: 0.8
  + 80% of each portion of genome slightly changing
    - This high rate would cause most GA’s to diverge, but neural networks hyperparameters are relatively robust – performance usually stays stable even if a hyperparameter is mutated, and elitism ensures a good individual is never lost to mutation
* **Recombination rate**: 0.8
  + 80% of the time, 2 individual’s genomes are randomly combined to produce 2 new individuals – 20% of the time, the new individuals are copies of the parents
* **Fitness function**: Validation MSE
  + Train a 2-fold neural networks, each of which uses half the training data and reserves the other half as validation
  + Calculate the validation MSE of each fold, and aggregate for both folds
  + When recombining, use random weighted selection to choose parents, with weights being standardized (lowest becomes 0, highest becomes 1) versions of 1 / validation MSE (such that better validation MSE results in a higher weight)

Each genome is the same as the above set of hyperparameters. Each hyperparameter is randomly initialized within reasonable ranges. On mutation, float-type hyperparameters are doubled or halved, integer-type hyperparameters are multiplied by 0.9 or 1.1 and rounded, and layer sizes (an integer list) either double a layer’s size, halve a layer’s size, copy the first layer, or copy the last layer. On recombination, each hyper-parameter is randomly taken from 1 of the 2 parents.

* To increase training speed, the epochs parameter only starts between 50 and 200, and only 2 folds are used, but the best individual is re-trained after evolution with 10 folds and 10x the number of epochs in its genome (an individual with 50 epochs in its genome trains 500 epochs at the end).

Best genome:

* **Layers**: 18 (input layer) -> 64 -> 64 -> 1 (output layer)
  + Slightly grew compared to default individual
  + In generation 15, this “species” closely outcompeted another with many more, larger layers
* **Epochs**: 311 (trained 3110 epochs during final re-training)
* **Patience**: 32
  + Stopping early to avoid over-fitting seems to have been a disadvantage
* **Adam Learning Rate**: 0.0752
  + Optimizer reacts very quickly to adjust to prioritize precision or quick progress
* **Regularization Lambda**: 0.00231
  + Minimizing parameter complexity was less important than expected

Resulted in a validation MSE of 1.31

## Visualization

R module utilizing ggplot2 to create a wide variety of visualizations. These include predicted/actual views with the results of each model’s forecast, in addition to various exploratory data analysis (EDA) plots such as anomaly since the last ice age and carbon dioxide since industrialization. These are used to present the results and gain a better understanding of the source data.

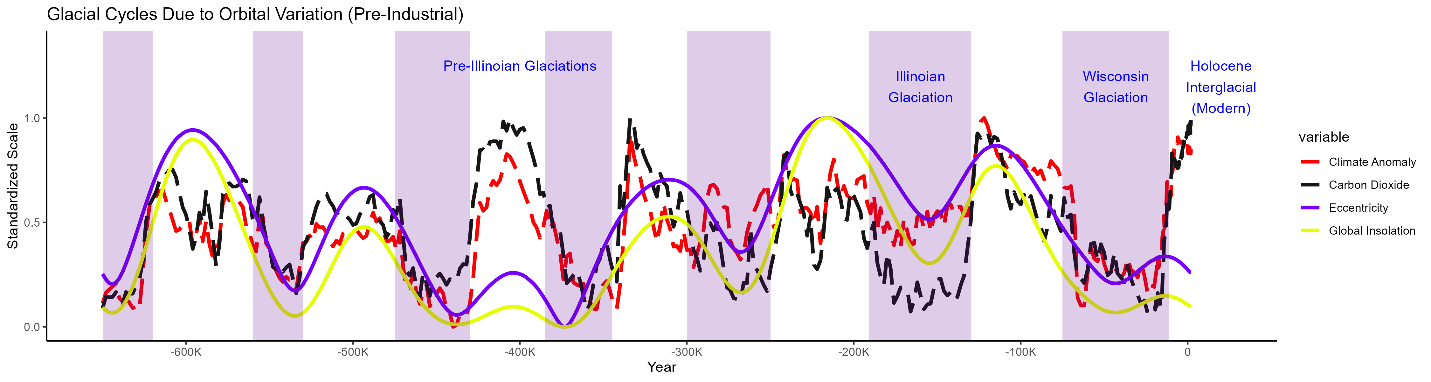


Figure 3: EDA plot showing the coinciding changes in orbital parameters, anomaly, and CO2. As atmospheric conditions likely can’t affect orbital parameters, this correlation suggests a causal relationship originating from the orbital parameters, driving both CO2 and anomaly fluctuation

# Results

The most promising results were provided by the genetically optimized neural regression, which provided an average MSE of 1.31 across all folds. This forecast suggests a severe ice age starting near 4,000 A.D. with temperatures dropping by about 1.5 °C every 2,000 years, which is geologically very fast but about 20x slower than our recent climate change.

Based on the forecast’s good performance in the test window, which was never accessed during training or evaluation, this forecast would be quite accurate given pre-industrial conditions. The ability to produce such an accurate out-of-sample anomaly forecast purely from orbital parameters and 40,000+ year lagged anomaly values supports Milankovitch’s proposition of orbital parameters as the underlying cause of glacial cycles.

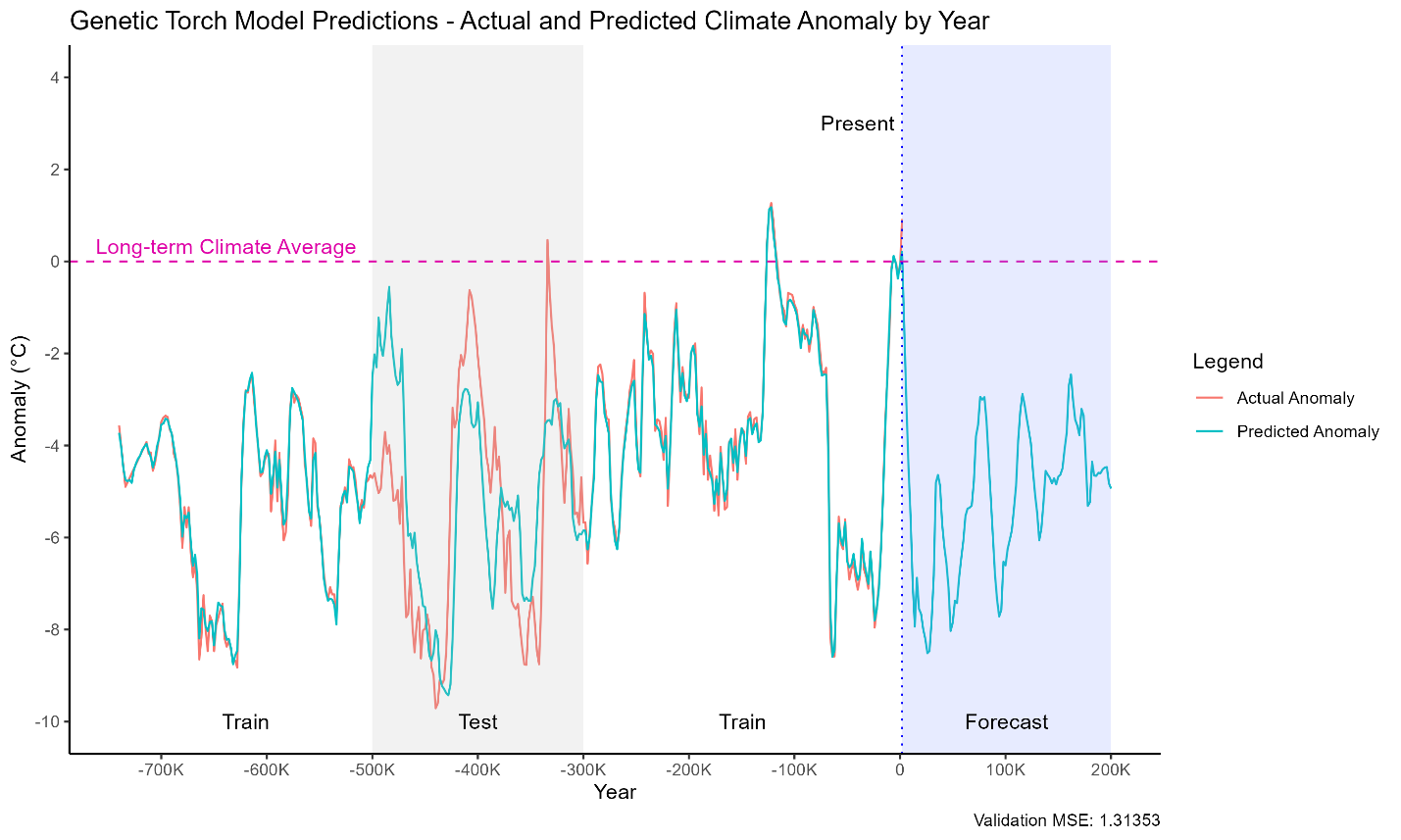


Figure 4: Average forecast produced by the 10 folds of the re-trained best evolved individual – highest-performance forecast obtained so far

# Next Steps

These forecasts are obviously limited by their lack of inclusion, for many reasons, of carbon dioxide. However, given this is the root cause of the current climate crisis, it would be of significant interest to find a valid way to incorporate carbon dioxide into the forecasts, perhaps using causal inference and adjustment methods for finding the effect of carbon dioxide on anomaly, given the confounding effects of orbital parameters. This could enable the valid usage of post-industrial data observations, and the creation of several forecasts based on different scenarios of human action and CO2 emissions in the near future.

Additionally, large portions of paleoclimate-related literature are interested in what exactly kicks off the feedback loop at the initiation and termination of each glacial cycle – likely solar modulation and its relationship with cosmic flux and low-cloud cover [2]. This could possibly merit a classification-based study in which we attempt to predict when glacial cycles will start and end, rather than their exact magnitudes in terms of anomaly regression.

In terms of methods utilized, there are myriad ML methods that could be applied to this regression/forecasting task for a more accurate or more valid forecast. This could include time series-native neural forecasting methods, evolutionary self-adaptation of the GA’s own parameters (e.g. mutation rates), deep learning architectures in place of a basic neural network, and a focus on more explainable methods over pure performance.

# Replication

Repository: [Viktor-Butkovich/Paleoclimate-Reconstruction-and-Forecasting](https://github.com/Viktor-Butkovich/Paleoclimate-Reconstruction-and-Forecasting)

1. Install Git, Python, Pip, and R
2. Run git clone <https://github.com/Viktor-Butkovich/Paleoclimate-Reconstruction-and-Forecasting.git> to pull the code locally
3. Run pip install -r requirements.txt to install the Python dependencies – complete interactive prompts
4. Run Rscript requirements.r to install the R dependencies – complete interactive prompts
5. Navigate to the src directory
6. To run the entire pipeline from the source data, run ./orchestrator.sh (.bat for Windows)
7. Alternatively, run just the prediction/visualization pipeline from the pre-computed, pre-processed datasets with ./orchestrator\_no\_etl.sh (.bat for Windows)
   1. The orchestrator commands run the prediction and visualization scripts, resulting in outputted predictions and visualizations to the Outputs directory

# References

[1] Berger, A.: Milankovitch, the father of paleoclimate modeling, Clim. Past, 17, 1727–1733, https://doi.org/10.5194/cp-17-1727-2021, 2021.

[2] Marsh, Gerald E. "Interglacials, Milankovitch cycles, solar activity, and carbon dioxide." *Journal of Climatology* 2014.1 (2014): 345482.

Find dataset citations in the README.md