A Study of In-Context Learning for Periodic Functions

Anonymous Author(s)
Affiliation
Address
email
Anonymous Author(s)

Abstract

We present a comprehensive investigation into the capacity of GPT-style transformers to learn and generalize periodic functions via in-context learning. Building on the findings of Garg et al. (2023) [1], we evaluate the the ability of small GPT-style transformers to fit sinusoidal superpositions within and beyond its training domain.

1 Introduction

Recent work has demonstrated remarkable in-context learning capabilities of transformers across diverse function classes [1]. However, the specific challenge of modeling periodic functions remains unaddressed. Periodic signals — ubiquitous in scientific and engineering applications — pose unique learning requirements, motivating our study. We claim that transformers are capable of interpolating between between points and slightly outside of a context given to them, but struggle to learn true periodic functions. To check our claim we aim to (1) assess whether standard GPT-derived architectures can implicitly learn periodicity via context alone and (2) evaluate whether augmenting these models with sinusoidal activations enables robust periodic learning.

2 Related Work

In-Context Learning on Synthetic Functions Garg et al. (2023) provide an extensive empirical study of Transformers' ability to perform in-context learning on synthetic function families. They train standard Transformer models from scratch so that, given a small set of input–output examples from an unseen function, the model can predict the function's value at new query points purely via its forward pass, without any weight updates. Notably, these Transformers match the analytic least-squares estimator on linear regression tasks and generalize to more complex families such as sparse linear models, two-layer MLPs, and decision trees [1]. This work establishes that Transformers can internalize learning algorithms; our study extends this paradigm to the challenging domain of periodic functions.

Neural Networks and Periodic Inductive Biases Ziyin et al. (2020) [5] demonstrate that standard feed-forward networks with ReLU, tanh, or sigmoid activations fail to extrapolate even simple periodic signals beyond their training range, owing to a lack of any built-in periodic bias. They introduce the activation $\phi(x) = x + \sin^2(x)$, which embeds a sinusoidal component into each layer and empirically recovers and extrapolates periodic patterns (e.g. temperature cycles, financial time series) that standard networks cannot . Their work highlights the necessity of architectural periodicity.

Fourier Feature Representations Tancik et al. (2020) [3] show that neural networks suffer from a spectral bias toward low-frequency functions , making it hard to learn fine-scale or high-frequency variation. To address this, they prepend inputs with *random Fourier features*—i.e. mappings of the form $\sin(\mathbf{B}x)$ and $\cos(\mathbf{B}x)$— thereby furnishing the network with a rich basis of periodic components.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

This transformation enables simple MLPs to fit and extrapolate high-frequency functions, proving particularly effective for coordinate-based neural representations of signals. These findings suggest that explicit sinusoidal embeddings can compensate for a network's native deficiencies in learning periodic structure.

Transformers for Continuous Time-Series Zhou. (2022) [4] propose the Frequency Enhanced Decomposed Transformer (FEDformer) for long-term time-series forecasting. FEDformer integrates a seasonal–trend decomposition step with sparse Fourier blocks that extract dominant frequency components before applying self-attention. This design captures periodic (seasonal) patterns more efficiently than vanilla Transformers, achieving state-of-the-art forecasting accuracy on benchmarks with clear periodic structure. The success of FEDformer underscores that combining Transformer architectures with frequency-domain inductive biases can markedly improve periodic signal modeling. In contrast, our work examines whether unmodified Transformers can learn periodic functions in-context, or whether similar frequency-aware mechanisms are essential.

3 Architecture and Codebase

We began our experiments extending the codebase from Garg et al. [1]. However the code was poorly optimized for execution on GPUs, requiring extensive data transfer between CPU and GPU and thus resulting in excessive training times (over 8 hours per train for the standard-sized model—even using an A100). We conducted initial experiments based on the 'tiny' model (3 layers, 2 attention heads, 64-dimensional embeddings, 0.2M parameters). However, since it yielded no results, we ended up rewriting the core data generation and training loops from scratch to minimize training time and allow for the training of larger models. Our GPU-optimized solution is contained in the Github [6] at the end of this file. The architecture used by our final version is discussed below.

We developed a regression model utilizing the GPT-2 architecture as the backbone. Our model architecture begins with an initial linear layer mapping input dimensions from 1 to 256, followed by a GPT-2-based transformer backbone composed of 12 transformer layers, each containing 8 attention heads with an embedding dimension of 256. The architecture concludes with a final linear layer projecting from 256 dimensions down to a single scalar output. We trained two distinct versions of this architecture: one variant employing the default activation functions (GeLU) and a second variant where each activation function was replaced by a custom activation defined as $\phi(x) = x + \sin^2(x)$.

4 Approach

4.1 Data Generation

Training and evaluation datasets consist of superpositions of sinusoidal functions of the form

$$f(x) = \sum_{n=1}^{k} a_n \sin\big(2\pi n x + c_n\big).$$

All the basis functions are periodic with period 1, and so the entire function is also periodic with period 1. The coefficients a_n are sampled from a standard normal distribution, and c_n is chosen to be either 0 or $\frac{\pi}{2}$ with equal probability. First, m points are sampled at uniform intervals sufficient for perfect reconstruction according to the Shannon-Nyquist theorem. Then, another set of m input values is randomly drawn from a standard normal distribution and appended to the grid values. Finally, the function f is evaluated at each of these points and the evaluations are interleaved with the t-values (see (1)).

4.2 Training Methodology

Our regression model was trained to approximate the value of a target function f at a specific test point t_{test}, given a contextual input comprising m pairs of points. The value of m is selected based on criteria derived from the Shannon-Nyquist reconstruction theorem. Specifically, the input to our model is structured as follows:

$$\left[\underbrace{t_0, f(t_0), t_1, f(t_1), \ldots, t_m, f(t_m)}_{\text{context points}}, \underbrace{t_{\text{random}_1}, f(t_{\text{random}_1}), \ldots, t_{\text{random}_m}, f(t_{\text{random}_m})}_{\text{randomly generated}}\right]. \quad (1)$$

In subsequent discussions, the initial set of $(t, f(t))$ pairs serve as contextual information, whereas the latter set of randomly sampled t values constitute the prediction targets.

During training, the model predicts the function values $f(t)$ for these randomly sampled points. Ground truth $f(t)$ values at these prediction points are masked (set to zero) to enforce predictive learning, and the loss is computed exclusively on these masked $f(t)$ values. Importantly, loss computations exclude all contextual information and input t-values.

We evaluated two distinct training methodologies: standard training and curriculum learning. Under standard training conditions, each training batch comprises the superposition of n sine waves, with n varying cyclically $(1 \to 2 \to \cdots \to n \to 1 \to 2 \to \ldots)$. In contrast, our curriculum learning strategy incrementally increases the complexity of training functions by progressively introducing additional sine-wave components. Specifically, the model undergoes extensive training (approximately 3,000 to 11,000 epochs) at each incremental step before advancing to functions composed of more sine waves.

As a scoring metric, we use MSE (mean squared error) with a perfect reconstruction as a baseline. The choice of a perfect reconstruction as a baseline is reasonable, since it is easy to perfectly recover the coefficients of each component wave of $f(t)$ [2]. We demonstrate this process in a separate notebook on our GitHub [6].

4.3 Resource Budget

Our resource budget for this project is $400. We have used $100 of it so far (various Google Colab subscriptions for each member of the team). All training was done using Google Colab on 40GB NVIDIA A100s.

4.4 Evaluation Methodology

To evaluate our models, we constructed several periodic functions and evaluated our model on a dense grids of points over (1) the same period as the context, (2) one period beyond the context, and (3) a large range that includes the range of the context.

5 Experiments & Results

5.1 Training

Across multiple runs, we found that, controlling for the number of epochs, curriculum learning converged to a much lower training loss than standard training, similar to the findings of Garg et al. [1]. See below for a comparison of loss curves for curriculum and non-curriculum learning.

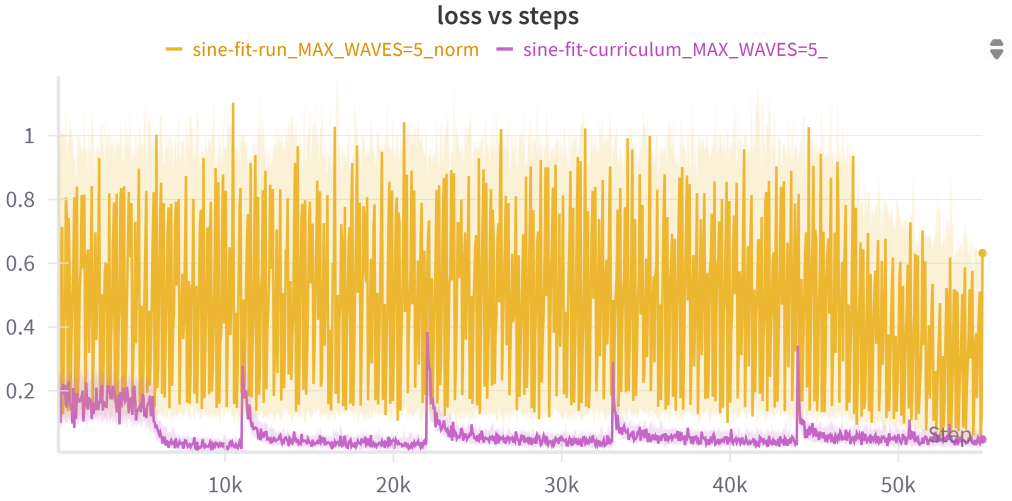

Figure 1: Loss vs. steps for curriculum learning (purple) and standard training (yellow)

5.2 Model Predictions

Upon evaluation, all of our models reconstruct signals quite well in the neighborhood that they were evaluated on during training (the period after the context).

After doing local runs we rewrote the code base so that it was optimized for GPU parallelization. With this optimization we were able to run the code base with the standard model. With these parameters. Standard 256 embedding size, 12 layers, 8 heads, and 9.5M total parameters. After these changes, we observed a massive difference in our experiments:

For our evaluation we ran 150 experiments where we predict 200 points between $t = 0$ and $t = 3$. We did this for a standard model with GeLU activation and a model with sin activation. We did this for each with 4, 5, and 6 superpositions for the sinusoidal waves. Table 5.2 contains a summary of the data.

Activation Type	# Of Sines	MSE Over 1st Period (0,1)	MSE Over 2nd Period (1,2)	MSE Over 3rd Period (2,3)	Overall
Standard	4	0.44893842	**0.31079719**	**0.2712146**	**0.34373317**
Sinusoidal	4	**0.34100499**	0.4239749	0.32062751	0.36196518
Standard	5	0.03727583	0.04325096	0.06601895	0.04886744
Sinusoidal	5	**0.03010587**	**0.0315977**	**0.04238979**	**0.03458119**
Standard	6	0.64249314	0.56375453	0.44534295	0.54915387
Sinusoidal	6	**0.32312029**	**0.33671242**	**0.36268511**	**0.33958891**

Table 1: Comparison of average MSE Between Standard and Sinusoidal Activations Across Periodic Intervals with Varying Numbers of Sine Components (lower is better).

Note that the sinusoidal activation model generally performs better than the standard activation model. The mechanism behind the dominance of the standard activation model under four waves is not fully understood. Also note that we have not evaluated the effectiveness of sinusoidal activations on other transformer tasks. The findings of Ziyin et al. [5] suggest that severe performance hits might not be expected.

Representative predictions are illustrated in Figures 2–7.

6 Figures

Model predictions and ground truth values: Standard Activations meaning GeLU.
An insight that we see is that the model actually improves the further it gets from t(0) until t(3) where the prediction flat lines and error shoots up. This is also the only tested superposition k = 4 where GeLU outperforms sine activation.

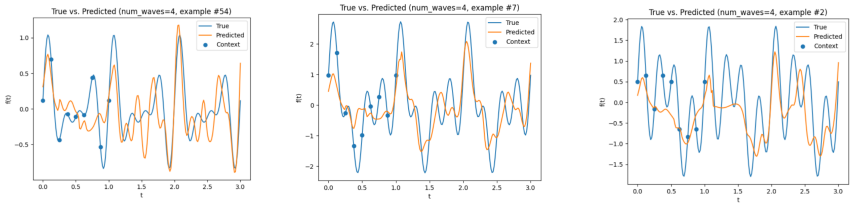

Figure 2: Standard Activations with 4 Superpositions

Unsurprisingly the model being trained on 5 superpositions does quite well. Sine activation outperforms this GeLU activation. Its mean squared error is a factor of ten less than the other superpositions.

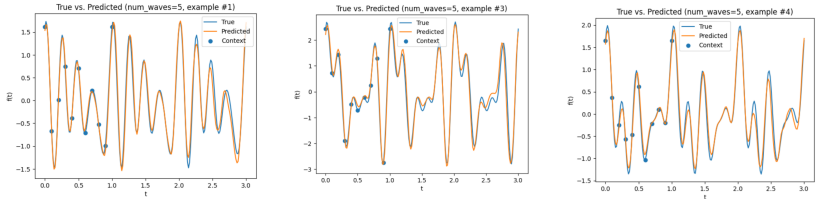

Figure 3: Standard Activations with 5 Superpositions

A similar insight to 4 superpositions is seen here. We observe that the model actually improves the further it gets from $t = 0$ until $t = 3$ where the prediction flat lines and error shoots up.

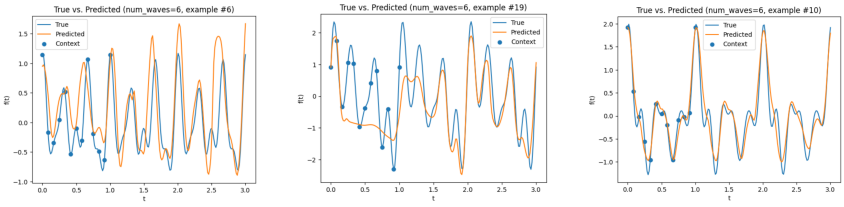

Figure 4: Standard Activations with 6 Superpositions

An interesting observation is that the MSE increases from $t = 1$ to $t = 2$ and then decreases from $t = 2$ to $t = 3$ to even beyond what it predicts within the context $t = 0$ to $t = 1$. Furthermore this is the only experimental setup where sine-based activation is outperformed by GeLU activation.

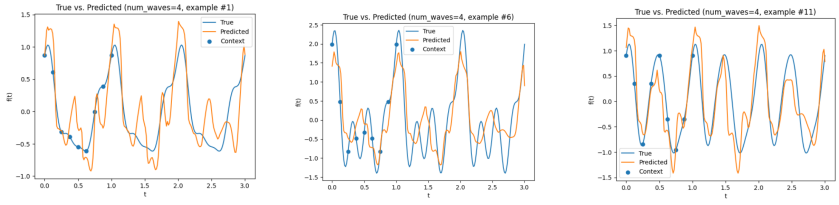

Figure 5: Sinusoidal Activations with 4 Superpositions

This is the best current model.

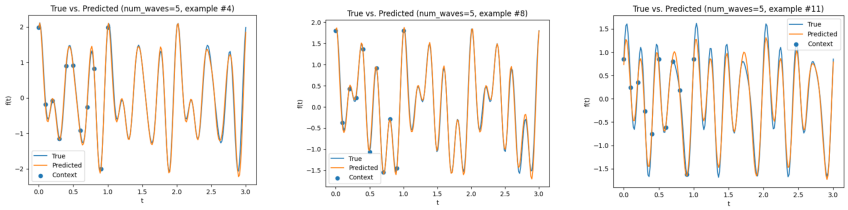

Figure 6: Sinusoidal Activations with 5 Superpositions (max number trained on)

Here we see what we expect with MSE increasing as predictions get further from the model's given context. It outperforms its GeLU counterpart.

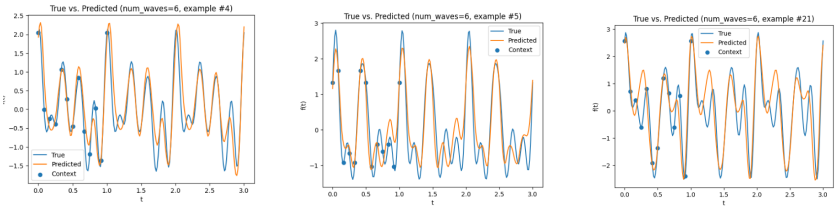

Figure 7: Sinusoidal Activations with 6 Superpositions

7 Next Steps, Current Problems

1. Evaluating the model's ability to generalize over a large interval by training on multiple smaller subintervals that collectively span the full range (e.g., training on intervals of size 10 covering the range $[0, 100]$).
2. Training the model on disjoint intervals and evaluating if it is able to interpolate between those intervals. (e.g. training on points from $[0, 1]$ & $[2, 3]$; evaluate its on points from $[1, 2]$).
3. Train using more robust curriculum schedules that incorporate small amounts of data using all previous numbers of sine waves during each step to ensure model generalization.
4. Try autoregressive prediction to see what effects this has on model generation at far away points.
5. Train on noisy data.
6. Experiment with real-life data, e.g., crop yield and weather.
7. Normalize the training data (divide sum of sine waves by total number of waves).
8. Shuffling and testing adding extra random points from the sine wave into the context.
9. Apply alternative regularization schemes (e.g., weight decay, dropout variations).
10. Selectively apply the sinusoidal activation function to certain parts of the transformer architecture instead of swapping all activations.

References

[1] S. Garg et al. What Can Transformers Learn in Context? (2023). `https://arxiv.org/pdf/2208.01066`.

[2] Shannon-Nyquist sampling theorem. `https://mathworld.wolfram.com/SamplingTheorem.html`

[3] Tancik et al. Spectral Bias and Fourier Features. `https://mathworld.wolfram.com/SamplingTheorem.html`

[4] Zhou et al. Frequency Enhanced Decomposed Transformer //arxiv.org/abs/2201.12740

[5] L. Ziyin et al. Neural Networks Fail to Learn Periodic Functions and How to Fix It. (2020). https://arxiv.org/abs/2006.08195.

[6] Our Code Base https://github.com/drnestedloop/periodic_icl

[7] RunPod / Lambda GPU Pricing. https://www.runpod.io/pricing; https://lambda.ai/service/gpu-cloud?matchtype=....

8 Self Review

8.1 Main Goal and Claims

The main goal of the project is to evaluate the ability of GPT-style transformers to learn and generalize periodic functions via in-context learning, and to determine whether architectural modifications, such as sinusoidal activations, improve their ability to do so.

We claim that transformers are capable of interpolating between between points and slightly outside of a context given to them, but struggle to learn true periodic functions (i.e. accurately model the periodic nature of a function over arbitrarily many periods).

8.2 Experimental Design and Evaluation Protocol

We trained two GPT-style transformers on synthetic datasets composed of sinusoidal superpositions. We began with a 'tiny' GPT-2 style model (not shown) (3 layers, 2 heads, 64-dim embeddings) and then progressed to a 'standard' model (12 layers, 8 heads, 256-dim embeddings). We trained one model with standard GeLu activations and the other with a sine-based activation function.

Models are evaluated using mean-squared error (MSE) measured on freshly-generated periodic function samples for both interpolation (within-domain) and extrapolation (outside-domain). Performance is compared against ground truth. We compared MSE loss across intervals and further visually evaluated the performance by plotting the predictions versus the ground truth.

8.3 Data and Task

The data used for training and evaluation are synthetic periodic functions generated as superpositions of sinusoids with randomly sampled parameters and Gaussian noise added to ensure robust training signals. Each training example presents a sequence of input/output pairs as context; the task is to predict the output of a function f at a new query t_{new}.

8.4 Support for Claims

The experiments support our claims. For the case of five superimposed sinusoids, the standard Transformer records an average mean squared error (MSE) of 0.0489 on held-out query points, showing that it can interpolate between examples with decent accuracy. The variant with sinusoidal activations reduces its MSE to 0.0346—a relative improvement of nearly 30% demonstrating that embedding a periodic bias enhances the model's ability to learn and generalize periodic patterns. The predictions gradually get worse as they move further from context thus not showing true growth.

8.5 Limitations, Strengths, and Weaknesses

Limitations are acknowledged, including the transformers' struggles with extrapolation tasks and the inefficiency of the initial code base. However, deeper theoretical reasons behind the transformer's difficulty with periodic extrapolation are not extensively discussed. Additionally, while our synthetic data regime enables precise control over signal complexity and noise, we do not evaluate real-world periodic signals (e.g. sensor or climate data), leaving questions of practical applicability open to future work.

Strengths of the paper include clearly defined objectives, well-structured experimentation, and careful step-by-step troubleshooting and optimization. We have transparently documented our implementation issues and systematically explored potential remedies.

Weaknesses include the absence of quantifying statistical significance in reported improvements, extensive justification for hyperparameter choices, and the limited interpretability of initial experiment outcomes. Additionally, a broader set of comparisons with other advanced regression techniques beyond 3-NN would strengthen the findings.

8.6 Suggestions for Improvement

To strengthen our project's impact, we believe we could (1) include experiments on real-world periodic datasets, (2) report variance estimates over multiple random seeds or cross-validation folds, and (3) provide a more systematic ablation of the sinusoidal activation parameters (e.g. frequency scaling factors). Additionally, we could implement more rigorous theoretical analysis or experimental comparisons with other function approximation methods, such as Fourier-based neural networks or state-of-the-art recurrent architectures, to better contextualize the transformers' performance.

8.7 Related Work and Reproducibility

Relevant related work is clearly identified, notably in-context learning of functions in Garg et al. (2023) [1] and periodic induction via custom activations in Ziyin et al. (2020) [5].

Reproducibility of our work is enabled by the clear description of model architectures, data generation procedures, and optimizer settings. We share the optimized code base via GitHub [6] with descriptions for each code cell. Rerunning the experiments and reproducing results is very feasible, assuming access to comparable computational resources.

8.8 Presentation and Readability

Most figures in the paper have well-labeled axes and clear legends, though some might benefit from axis explanations or error bars to convey variance. The methodology is explained in sufficient detail.

8.9 Quality of Writing

We have written this paper to the best of our abilities. Our process involved one contributor authoring a section, followed by a peer review from the other 3 contributors and the use of online tools such as Grammarly. We are therefore confident that the English prose is generally clear, with only minor stylistic improvements possible in terms of conciseness, brevity, and succinctness.

8.10 TODO

We have a strong understading of what we should do moving forward. I think robust testing for if our model can predict between periods, experimenting with real life data, and normalizing the training data are high priority. After that trying out curriculum learning while bringning in training from priorThe rest seem to be lower priority. Perhaps they can display their predictions against nearest-neighbors to display the improvement.