Appendix

Roadmap. In Appendix A, we introduce some formal definitions that used in our main content. In Appendix B, we show why we separate the Fourier components into the high-frequency part and the low-frequency part and why we choose τ to be 50. In Appendix C, we show our observation generalizes to another format of dataset, another arithmetic task and other models. In Appendix D, we provide more evidence that shows the Fourier features in the model when computing addition. In Appendix E, we provide more evidence that shows the GPT-2-XL trained from scratch does not use Fourier feature to solve the addition task. In Appendix F, we give the details of our experimental settings.

A Formal Definition of Transformer and Logits in Fourier Space

We first introduce the formal definition of the Transformer structure that we used in this paper.

Definition A.1 (Transformer)**.** *An autoregressive Transformer language model $G : \mathcal{X} \to \mathcal{Y}$ over vocabulary* Vocab *maps a token sequence $x = [x_1, \ldots, x_N] \in \mathcal{X}, x_t \in \text{Vocab}$ to a probability distribution $y \in \mathcal{Y} \subset \mathbb{R}^{|\text{Vocab}|}$ that predicts next-token continuations of x. Within the Transformer, the i-th token is embedded as a series of hidden state vectors $h_t^{(\ell)}$, beginning with $h_t^{(0)} = \text{emb}(x_t) + \text{pos}(i) \in \mathbb{R}^D$. Let $W^U \in \mathbb{R}^{|\text{Vocab}| \times D}$ denote the output embedding. The final output $y = \text{softmax}(W^U \left(h_N^{(L)}\right))$ is read from the last hidden state. In the autoregressive case, tokens only draw information from past tokens:*

$$h_t^{(\ell)} = h_t^{(\ell-1)} + \text{Attn}_t^{(\ell)} + \text{MLP}_t^{(\ell)}$$

where

$$\text{Attn}_t^{(\ell)} := \text{Attn}^{(\ell)}\left(h_1^{(\ell-1)}, h_2^{(\ell-1)}, \ldots, h_t^{(\ell-1)}\right) \quad \textit{and} \quad \text{MLP}_t^{(\ell)} := \text{MLP}_t^{(\ell)}(\text{Attn}_t^{(\ell)}, h_t^{(\ell-1)}).$$

In this paper, we only consider the output tokens to be numbers. Hence, we have the unembedding matrix $W^U \in \mathbb{R}^{p \times D}$, where p is the size of the number space. As we are given the length-N input sequences and predict the $(N+1)$-th, we only consider $h_N^{(\ell)} = h_N^{(\ell-1)} + \text{Attn}_N^{(\ell)} + \text{MLP}_N^{(\ell)}$. For simplicity, we ignore the subscript N in the following paper, so we get Eq. (1).

Definition A.2 (Intermediate Logits)**.** *Let $\mathcal{L}_{\text{Attn}}^{(\ell)} := W^U \text{Attn}^{(\ell)}$ denote the intermediate logits of the attention module at the ℓ-th layer. Let $\mathcal{L}_{\text{MLP}}^{(\ell)} := W^U \text{MLP}^{(\ell)}$ denote the intermediate logits of the MLP module at the ℓ-th layer. Let $\mathcal{L}^{(\ell)} := W^U h^{(\ell)}$ denote the logits on intermediate state $h^{(\ell)}$.*

Throughout the model, h undergoes only additive updates (Eq. (1)), creating a continuous residual stream [ENO+21], meaning that the token representation h accumulates all additive updates within the residual stream up to layer t.

To analyze the logits in Fourier space, we give the formal definition of the Fourier basis as follows:

Definition A.3 (Fourier Basis)**.** *Let p denote the size of the number space. Let $\overrightarrow{\mathbf{x}} := (0, 1, \ldots, (p-1))$.*

Let $\omega_k := \frac{2\pi k}{p-1}$. We denote the normalized Fourier basis F as the $p \times p$ matrix:

$$F := \begin{bmatrix} \sqrt{\frac{1}{p-1}} \cdot \overrightarrow{\mathbf{1}} \\ \sqrt{\frac{2}{p-1}} \cdot \sin\left(\omega_1 \overrightarrow{\mathbf{x}}\right) \\ \sqrt{\frac{2}{p-1}} \cdot \cos\left(\omega_1 \overrightarrow{\mathbf{x}}\right) \\ \sqrt{\frac{2}{p-1}} \cdot \sin\left(\omega_2 \overrightarrow{\mathbf{x}}\right) \\ \vdots \\ \sqrt{\frac{2}{p-1}} \cdot \cos\left(\omega_{(p-1)/2} \overrightarrow{\mathbf{x}}\right) \end{bmatrix} \in \mathbb{R}^{p \times p}$$

The first component $F[0]$ is defined as a constant component. For $i \in [0, p-1]$, $F[i]$ is defined as the k-th component in Fourier space, where $k = \lfloor \frac{i+1}{2} \rfloor$. The frequency of the k-th component is $f_k := \frac{k}{p-1}$. The period of the k-th component is $T_k := \frac{p-1}{k}$

We can compute the discrete Fourier transform under that Fourier basis as follows:

Remark A.4 (Discrete Fourier transformer (DFT) and inverse DFT)**.** *We can transform any logits $u \in \mathbb{R}^p$ to Fourier space by computing $\widehat{u} = F \cdot u$. We can transform $\widehat{u}$ back to u by $u = F^\top \cdot \widehat{u}$*

Next, we define the logits in Fourier space.

Definition A.5 (Logits in Fourier Space)**.** *Let $\mathcal{L}^{(L)}$, $\mathcal{L}^{(\ell)}_{\mathrm{Attn}}$ and $\mathcal{L}^{(\ell)}_{\mathrm{MLP}}$ denote the logits (Definition A.2). The output logits before softmax in Fourier space is defined as: $\widehat{\mathcal{L}}^{(L)} = F \cdot \mathcal{L}^{(L)}$. The logits of the MLP and attention modules in Fourier space are defined as:*

$$\widehat{\mathcal{L}}^{(\ell)}_{\mathrm{Attn}} = F \cdot \mathcal{L}^{(\ell)}_{\mathrm{Attn}} \quad \textit{and} \quad \widehat{\mathcal{L}}^{(\ell)}_{\mathrm{MLP}} = F \cdot \mathcal{L}^{(\ell)}_{\mathrm{MLP}}.$$

We ignore the first elements in $\widehat{\mathcal{L}}^{(L)}, \widehat{\mathcal{L}}^{(\ell)}_{\mathrm{Attn}}$ and $\widehat{\mathcal{L}}^{(\ell)}_{\mathrm{MLP}}$ for the Fourier analysis in this paper as they are the constant terms. Adding a constant to the logits will not change the prediction.

Let $\tau \in \mathbb{R}$ denote a constant threshold. The low-frequency components for the logits in Fourier space are defined as $\widehat{\mathcal{L}}^{(\ell)}[1 : 2\tau]$. The high-frequency components for the logits in Fourier space are defined as $\widehat{\mathcal{L}}^{(\ell)}[2\tau :]$. For the following analysis, we choose $\tau = 50$ (the specific choice of $\tau = 50$ is explained in Appendix B).

Next, we propose the formal definition of low-pass/high-pass filter that is used in the following ablation study.

Definition A.6 (Loss-pass / High-pass Filter)**.** *Let $x \in \mathbb{R}^D$ denote the output of MLP or attention modules. Let F denote the Fourier Basis (Definition A.3). Let $\tau \in R$ denote the frequency threshold. Let $W^U \in R^{p \times D}$ denote the output embedding. For low-pass filter, we define a diagonal binary matrix $B \in \{0,1\}^{p \times p}$ as $b_{ii} = \begin{cases} 1 & \text{if } i \geq \tau \\ 0 & \text{otherwise} \end{cases}$. For high-pass filter, we define a diagonal binary matrix $B \in \{0,1\}^{p \times p}$ as $b_{ii} = \begin{cases} 1 & \text{if } 1 \leq i < \tau \\ 0 & \text{otherwise} \end{cases}$. Note that we retain the constant component, so $b_{i,i} = 0$. The output of the filter $\mathcal{F}(x) : \mathbb{R}^D \to \mathbb{R}^D$ is defined by the following objective function:*

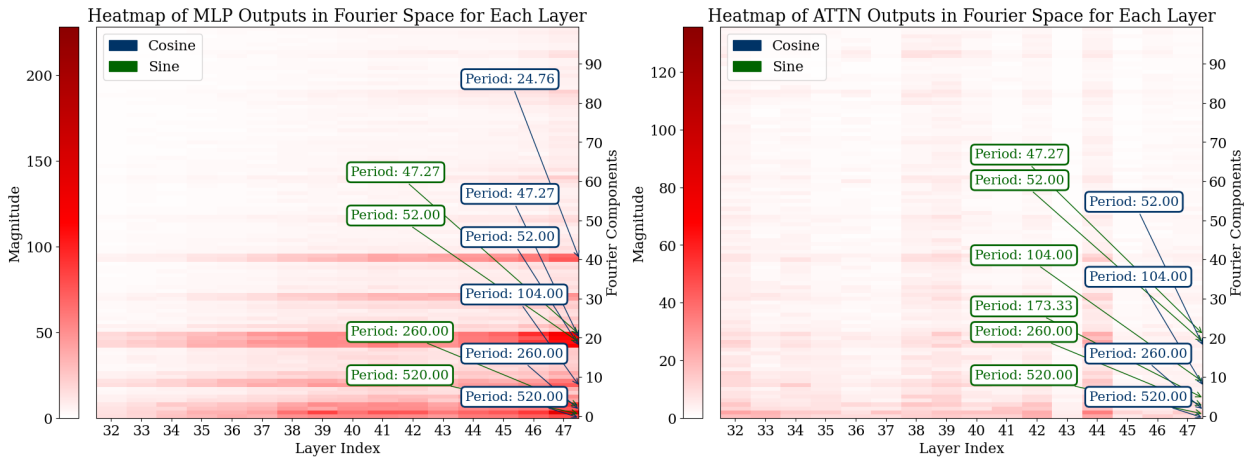
$$\begin{aligned} \min_y \quad & \|x - y\|_2^2 \\ \text{subject to} \quad & BFW^U y = 0 \end{aligned}$$

The solution to the above optimization problem is given by a linear projection.

Remark A.7. *The result of the optimization problem defined in Definition A.6 is the projection of x to the null space of BFW^U. Let $\mathcal{N}(BFW^U)$ denote the null space of BFW^U. We have*

$$\mathcal{F}(x) = \mathcal{N}(BFW^U) \cdot \mathcal{N}(BFW^U)^\top \cdot x^\top$$

B Fourier Components Separation and Selection of τ



(a) Logits for MLP output in Fourier space

(b) Logits for attention output in Fourier space

Figure 9: We analyzed the logits in Fourier space for all the test data across the last 15 layers. For both the MLP and attention modules. We only plot the first 50 Fourier components (a) The MLP exhibits some outlier low-frequency Fourier components. (b) The attention module's low-frequency Fourier components are not as obvious as the ones in MLP.

Following Definition A.6, we define single-pass filter as follows:

Definition B.1 (Single-Pass Filter). *Let $x \in \mathbb{R}^D$ denote the output of MLP or attention modules. Let F denote the Fourier Basis (Definition A.3). Let $\gamma \in R$ denote the γ-th Fourier component (Definition A.3) that we want to retain. Let $W^U \in R^{V \times D}$ denote the output embedding. We define a diagonal binary matrix $B \in \{0,1\}^{V \times V}$ as $b_{ii} = \begin{cases} 0 & \text{if } \lfloor \frac{i+1}{2} \rfloor = \gamma \text{ or } i = 0, \\ 1 & \text{otherwise.} \end{cases}$*

The output of the filter $\mathcal{F}_\gamma(x) : \mathbb{R}^D \to \mathbb{R}^D$ is defined as the following objective function:

$$\begin{aligned} \min_{y} \quad & \|x - y\|_2^2 \\ \text{subject to} \quad & BFW^U y = 0 \end{aligned}$$

Remark B.2. *The result of the optimization problem defined in Definition B.1 is the projection of x to the null space of BFW^U. Let $\mathcal{N}(BFW^U)$ denote the null space of BFW^U. We have*

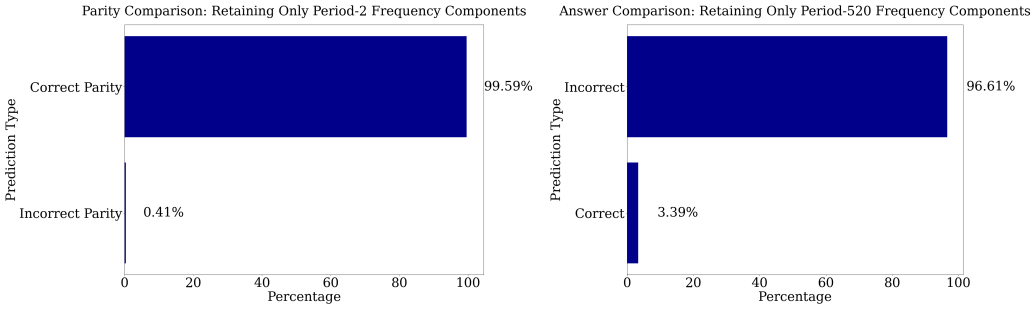
$$\mathcal{F}_\gamma(x) = \mathcal{N}(BFW^U) \cdot \mathcal{N}(BFW^U)^\top \cdot x^\top$$

For the single-pass filter, we only retrain one Fourier component and analyze how this component affects the model's prediction. The residual stream is then updated as follows:

$$h^{(\ell)} = h^{(\ell-1)} + \mathcal{F}_\gamma(\text{Attn}^{(\ell-1)}) + \mathcal{F}_\gamma(\text{MLP}^{(\ell-1)})$$

We evaluated the fine-tuned GPT-2-XL model on the addition dataset with the Fourier components period 520 and 2. Given that $T_k := \frac{V-1}{k}$ (Definition A.3), we retained only the Fourier components with $\gamma = 1$ and 260, respectively.

As shown in Figure 10a, with only one frequency component, whose period is 2, the model accurately predicts the parity with 99.59% accuracy. As depicted in Figure 10b, with a single frequency component of period 520, the model fails to accurately predict with 96.51% accuracy. We consider the frequency component with a period of 2 as the model's prediction for the `mod 2` task, and the frequency component with a period of 520 as its prediction for the `mod 520` task. Figures 10 and 11 suggest that the model effectively learns the `mod 2` task, as it involves a two-class classification, but struggles with the `mod 520` task, which requires classifying among 520 classes. As the model does not need to be trained to converge to the optimal for these low-frequency components as explained at the end of Section 3.2, predicting with the period-520 component leads to predictions that normally distributed around the correct answers.



(a) Retaining Period-2 Frequency Components

(b) Retaining Period-520 Frequency Components

Figure 10: The prediction analysis when predicting with only one Fourier component. (a) Retaining only the Period-2 Fourier component makes the prediction 99.59% accurate for `mod 2` task (b) Retaining only the Period-520 Fourier component makes the prediction 96.61% inaccurate for the `mod 520` task.

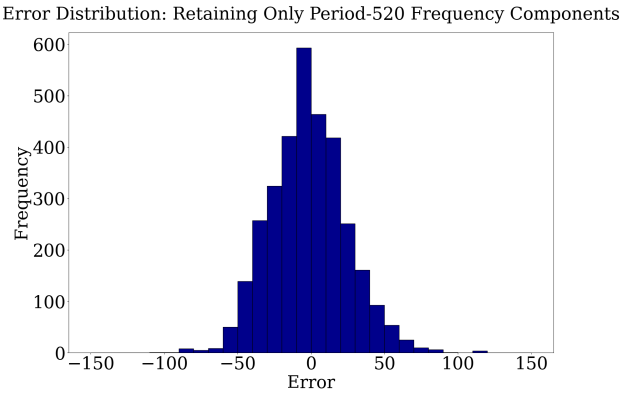
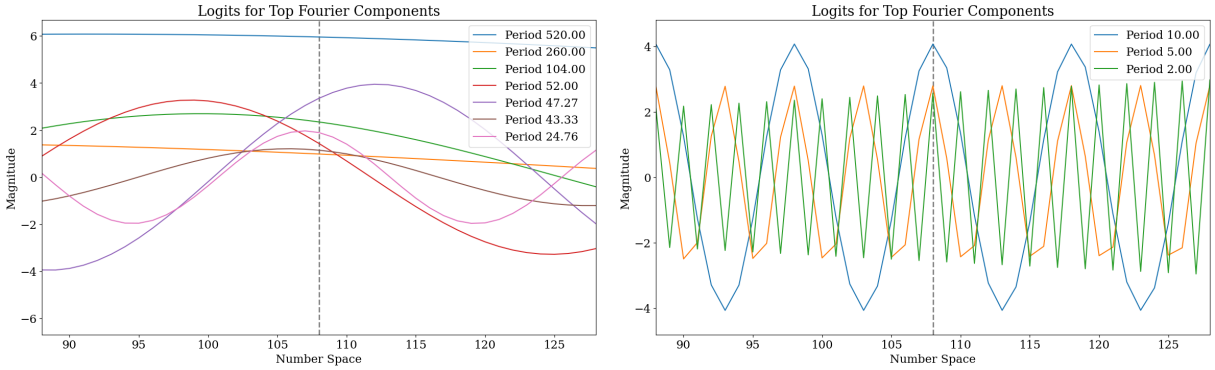


Figure 11: Retaining only the Period-520 Fourier component makes the model's predictions normally distributed around the correct answers.

The Fourier components with larger periods present greater difficulty in solving the corresponding modular addition task compared to those with smaller periods. As demonstrated in Figure 11, components with large periods serve primarily as approximations of the correct answer. Consequently, we categorize the Fourier components into low-frequency and high-frequency groups. The low-frequency components approximate the magnitude of the answer, whereas the high-

frequency components are employed to enhance the precision of the predictions.

In reference to Figure 4, to elucidate the contribution of these distinct Fourier components to our final prediction and the rationale behind their separation, consider the example: "Put together 15 and 93. Answer: 108". We selected the top-10 Fourier components of $\widehat{\mathcal{L}}^{(L)}$ based on their magnitudes and converted them back to logits in the numerical space by multiplying with $F^{\top}$. We plotted the components with components index less than 50 in Figure 12a and those with components index greater than 50 in Figure 12b. Leveraging the constructive and destructive inference for different waves, the components with low periods assign more weight to the correct answer, 108, and less weight to numbers close to 108. These high-frequency (low-period) components ensure the prediction's accuracy at the unit place. For the low-frequency (large-period) components, the model fails to precisely learn the magnitude of the factor between the cos and sin components, which results in failing to peak at the correct answer. Thus, the low-frequency (large-period) components are used to approximate the magnitude of the addition results.



(a) Final Logits for Components whose index Less than 50

(b) Final Logits for Components whose index Greater than 50

Figure 12: Visualization of tthe individual final logits for the top-10 Fourier components with example "Put together 15 and 93. Answer: 108' (a) The components index less than 50. (b) The components index greater than 50.

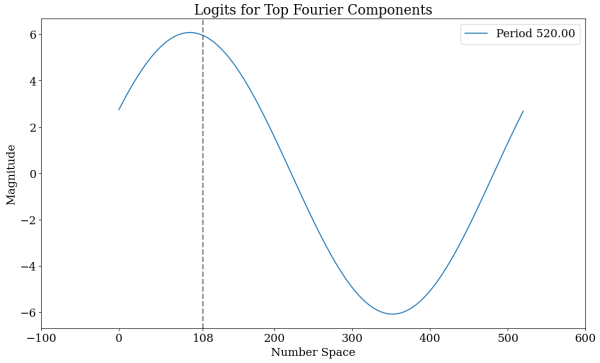
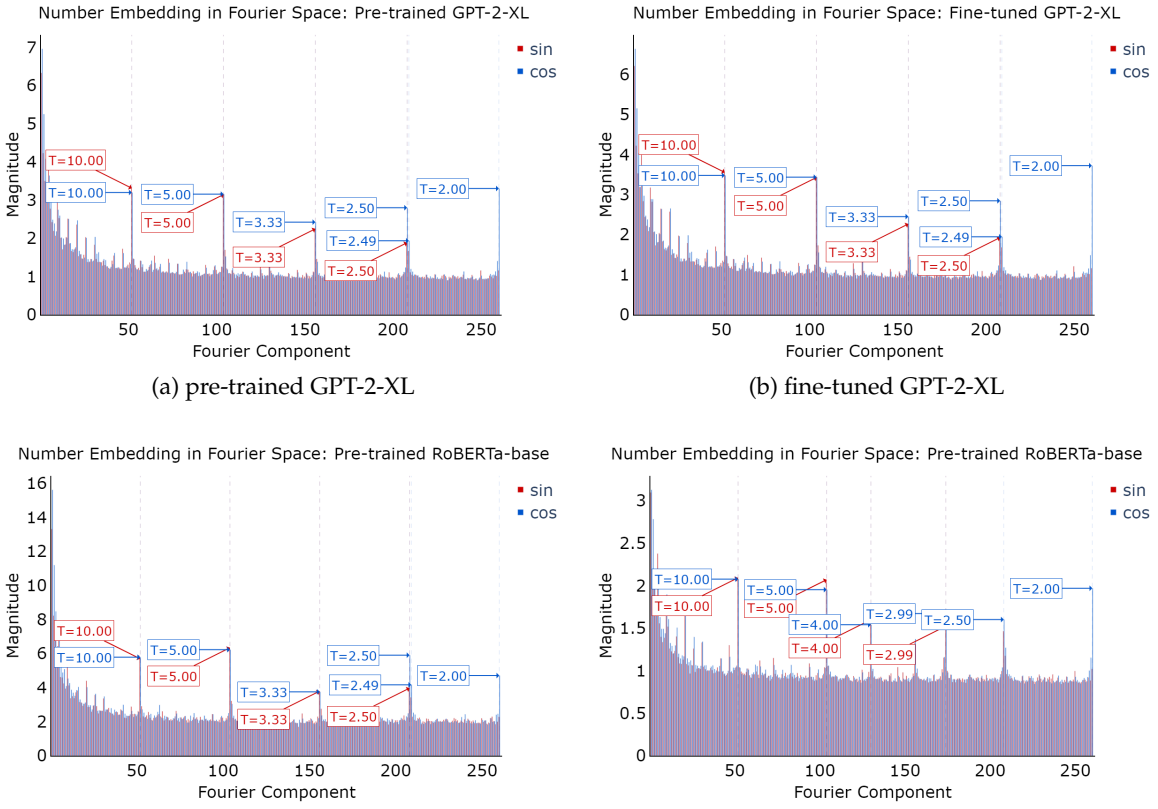


Figure 13: Visualization of the Fourier component whose period is 520 analysis for the final logits.

C Does Fourier Features Generalize?

C.1 Token Embedding for Other LMs

We first show that other pre-trained LMs also have Fourier features in their token embedding for the numbers $[0, 520]$.



(a) pre-trained GPT-2-XL

(b) fine-tuned GPT-2-XL

(c) pre-trained RoBERTa

(d) pre-trained Phi2

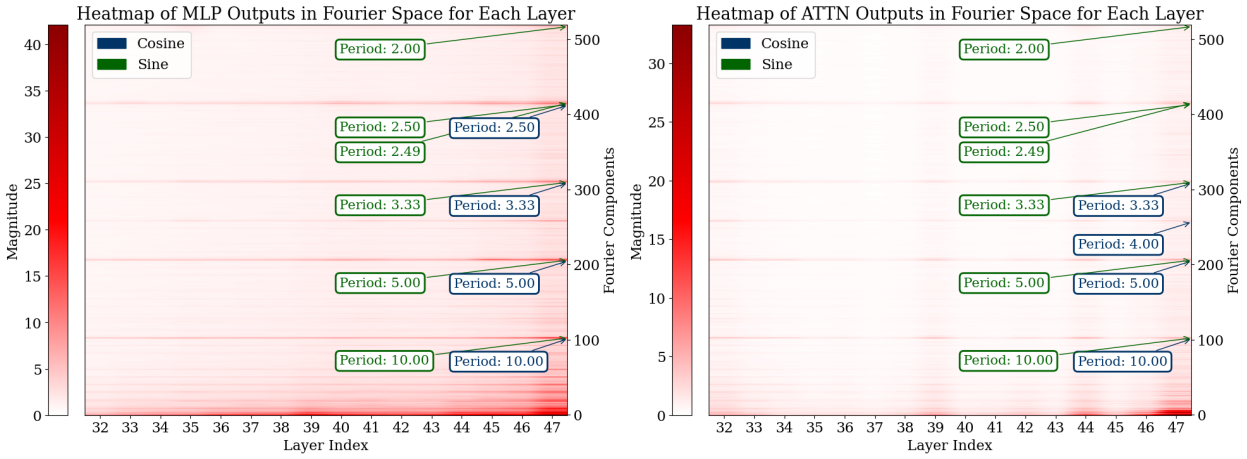
Figure 14: Number embedding in Fourier space for different pre-trained models.

C.2 Multiplication Task

A key question is whether pre-trained models utilize Fourier Features solely for solving addition tasks or if they generalize to other arithmetic tasks. We hypothesize the latter, knowing that numbers are represented by their Fourier features in the token embeddings after pre-training. Consequently, this Fourier representation should be leveraged in a variety of number-related tasks. To validate this hypothesis, we perform a Fourier analysis on the GPT-2-XL model fine-tuned for the multiplication task.

Considering a maximum number of 520 for multiplication would result in an insufficient dataset size. Therefore, we set the maximum allowable product to 10000. For each pair of numbers where the product does not exceed this limit, we generate various phrasings of multiplication questions and their corresponding answers in base 10. The different phrasings used include: "What is the product of num1 and num2?", "Find the product of num1 multiplied by num2.", "Calculate num1 times num2.", "num1 multiplied by num2 equals what?", and "Multiplication of num1 with num2." The dataset is then shuffled to ensure randomness and split into training (80%), validation (10%), and test (10%) sets. We finetune the model for 25 epochs with a learning rate of $1e-4$. Upon convergence, the validation accuracy reaches 74.58%.

As the primary objective is to determine whether the Fourier features are utilized in tasks other than addition, Figure 15 displays the logits in Fourier space for each layer, as in Figure 3. It is evident that the logits are sparse in Fourier space.



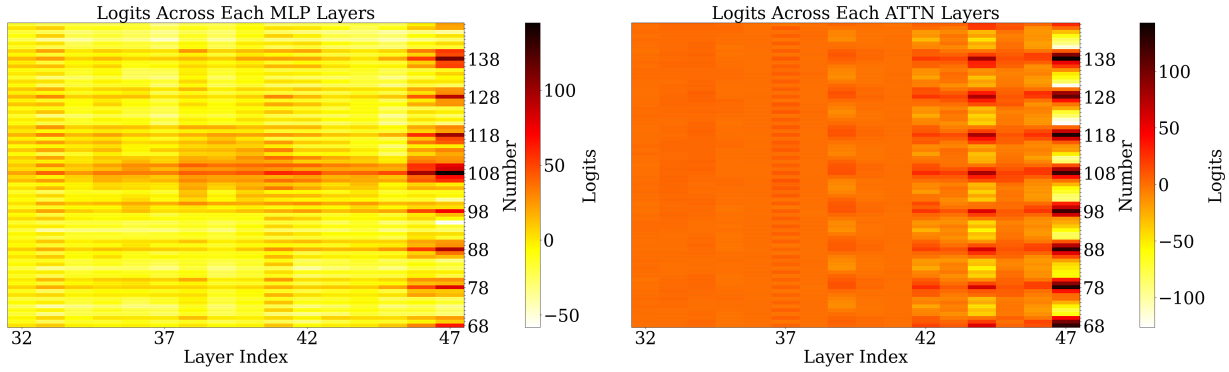
(a) Logits for MLP output in Fourier space

(b) Logits for attention output in Fourier space

Figure 15: We analyzed the logits in Fourier space for all the test data across the last 15 layers. For both the MLP and attention modules, the outlier Fourier components have periods around 2, 2.5, 3.3, 5, and 10.

C.3 Same Results for other format

To demonstrate that our observations are not confined to a specific description of the mathematical problem, we conducted experiments on another format of addition problem and obtained consistent results. From Figure 16, we can see that there are also periodic structures in the intermediate logits.

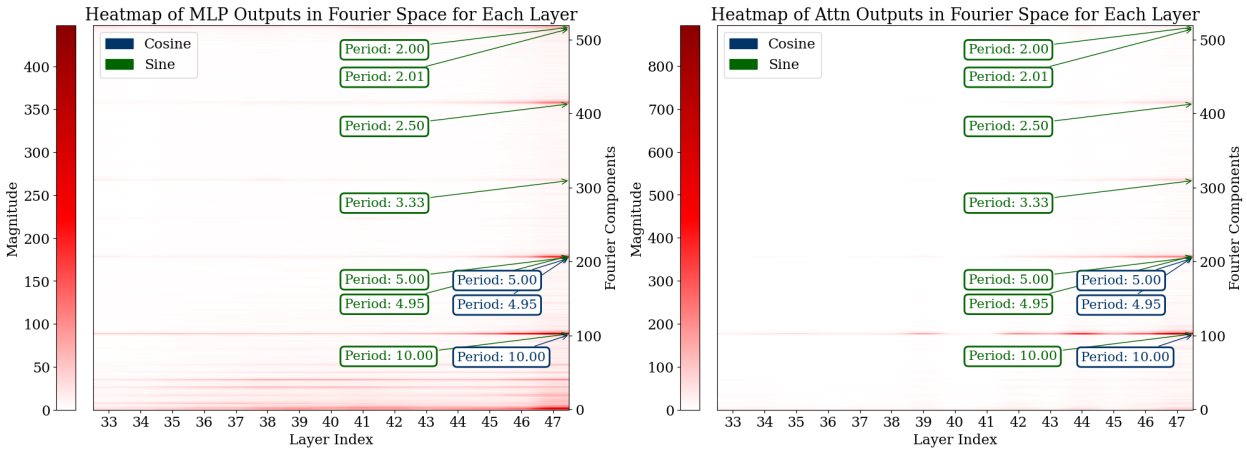


(a) MLP output logits for the last 15 layers

(b) Attention output logits for the last 15 layers

Figure 16: Heatmap of the logits across different layers. The y-axis represents the subset of the number space around the correct prediction, while the x-axis represents the layer index.

From Figure 17, we can also see the Fourier features for the MLP and attention output. These two experiments validate that our observations are not confined to a specific format of the addition problems.



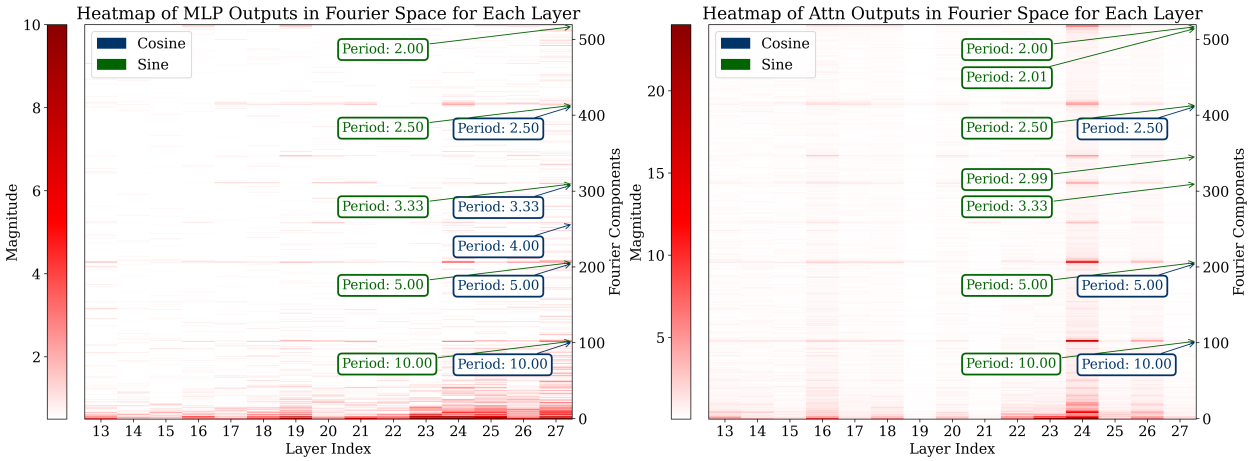
(a) Logits for MLP output in Fourier space

(b) Logits for attention output in Fourier space

Figure 17: We analyzed the logits in Fourier space for all the test data across the last 15 layers. For both the MLP and attention modules, the outlier Fourier components have periods around 2, 2.5, 5, and 10. (a) The MLP exhibits some outlier low-frequency Fourier components. (b) The attention module does not exhibit any outlier low-frequency Fourier components, but it has stronger high-frequency components.

C.4 Fourier Features in Other Pre-trained LM

Using the Fourier analysis framework proposed in Section 3.2, we demonstrate that for GPT-J, the outputs of MLP and attention modules exhibit approximate sparsity in Fourier space across the last 15 layers (Figure 18)



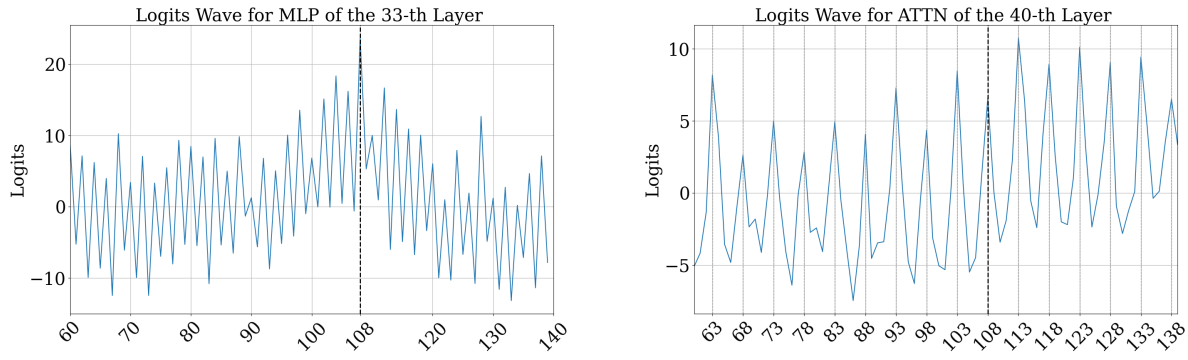
(a) Logits for MLP output in Fourier space

(b) Logits for attention output in Fourier space

Figure 18: For GPT-J (4-shot), we analyzed the logits in Fourier space for all the test data across the last 15 layers. For both the MLP and attention modules, the outlier Fourier components have periods around 2, 2.5, 5, and 10.

D Supporting Evidence For the Fourier Features

We selected the layers that clearly show the periodic pattern in Figure 1b and Figure 1c and plot their logits in Figure 19.

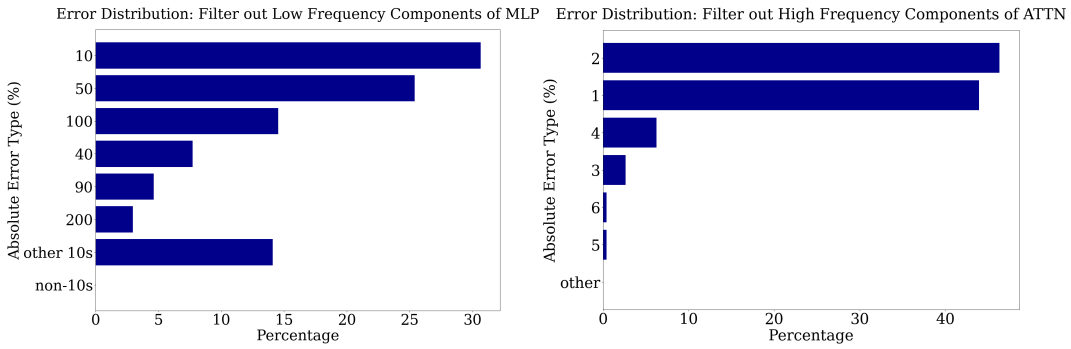


(a) Logits for the 33-th layer's MLP output

(b) Logits for the 40-th layer's attention output

Figure 19: The x-axis represents the number space, and the y-axis represents the logits value. (a) The logits wave for the MLP output of the 33^{rd} layer, $\mathcal{L}^{(33)}_{\text{MLP}}$. The MLP favors even numbers. The MLP module favors the answer to $15 + 93 \bmod 2$. (b) The logits wave for the attention output of the 40^{th} layer, $\mathcal{L}^{(40)}_{\text{Attn}}$. The attention module favors the answer to $15+93 \bmod 10$ and $15+93 \bmod 5$.

Figure 20 illustrates that the errors resulting from the ablation study (Section 3.3) correspond with our theoretical insights. Removing low-frequency parts from the MLP results in errors such as off-by 10, 50, and 100. Without these low-frequency components, the MLP is unable to accurately approximate, although it still correctly predicts the unit digit. In contrast, removing high-frequency components from the attention modules results in smaller errors, all less than 6 in magnitude. These findings support our statement that low-frequency components are essential for accurate approximation, whereas high-frequency components are key for precise classification tasks. Consequently, the primary function of MLP modules is to approximate numerical magnitudes using low-frequency components, and the essential function of attention modules is to facilitate precise classification by identifying the correct unit digit.



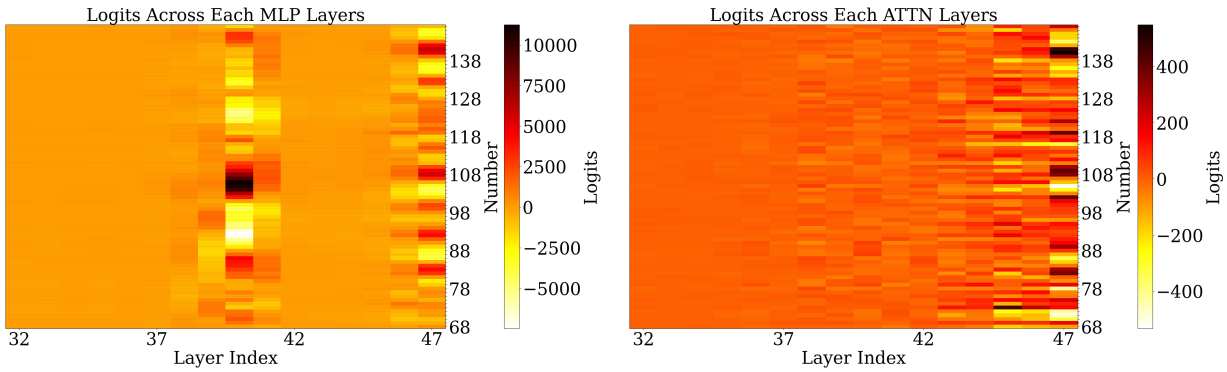
(a) Filtering out low-frequency components of MLP

(b) Filtering out high-frequency components of attention

Figure 20: (a) Across all the test data, the difference between predictions and labels are all the multiple of 10. (b) Across all the test data, the differences between predictions and labels are all below 6.

E More Experiments on GPT-2-XL Trained from Scratch

Following the methodology proposed in Section 3, we plotted the logits of the MLP and attention modules for each layer, as shown in Figure 21. The prediction is solely determined by the 40-th layer MLP. Unlike Figure 3, there is no observable periodic structure across all layers.



(a) MLP output logits for the last 15 layers

(b) Attention output logits for the last 15 layers

Figure 21: Heatmap of the logits across different layers. The y-axis represents the subset of the number space around the correct prediction, while the x-axis represents the layer index. The final prediction is solely decided by the 40-th layer MLP.

For the model trained from scratch on the created addition dataset, all of the predictions on the test dataset deviate from the correct answer within 2 as shown in Figure 22.

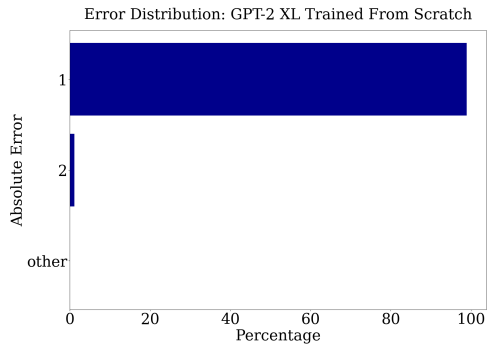


Figure 22: Error distribution for GPT-2-XL trained from scratch.

F Details of Experimental Settings

Fine-tuned GPT-2-XL We finetune GPT-2-XL on the "language-math-dataset" with 50 epochs and a batch size of 16. The dataset consists of $27,400$ training samples, $3,420$ validation samples, and $3,420$ test samples. We use the AdamW optimizer, scheduling the learning rate linearly from 1×10^{-5} to 0 without warmup.

Train GPT-2-XL from scratch We train GPT-2-XL on the "language-math-dataset" from scratch with 500 epochs and a batch size of 16. The dataset consists of $27,400$ training samples, $3,420$ validation samples, and $3,420$ test samples. We use the AdamW optimizer, scheduling the learning rate linearly from 1×10^{-4} to 0 without warmup.

Train GPT-2 from scratch For both with pre-trained token embedding and without token embedding, we train GPT-2 on the "language-math-dataset" with 700 epochs and a batch size of 16. The dataset consists of $27,400$ training samples, $3,420$ validation samples, and $3,420$ test samples. We use the AdamW optimizer, scheduling the learning rate linearly from 5×10^{-5} to 0 without warmup. In Figure 7b, we train the model with five different seeds and plot the mean and deviation for them.

Create the addition dataset in main content We consider numbers in base 10 up to a maximum value of 260. For each pair of numbers between 0 and 260, we generate various phrasings of addition questions and their corresponding answers. The different phrasings used are: "Total of num1 and num2.", "Add together num1 and num2.", "Calculate num1 + num2.", "What is the sum of num1 and num2?", and "Put together num1 and num2.". The dataset is shuffled to ensure randomness and then split into training (80%), validation (10%), and test (10%) sets.

Create the addition dataset in Appendix C.3 with different format We consider numbers in base 10 up to a maximum value of 260. We generate all possible pairs of numbers within this range using combinations with replacement. For each pair, we convert the numbers to the specified base and create questions formatted as "num1,num2+" with their corresponding answers. The dataset is then split into training (80%), validation (10%), and test (10%) sets.

Experiments Compute Resources All experiments involving fine-tuning and training from scratch in this paper were conducted on one NVIDIA A6000 GPU with 48GB of video memory. The fine-tuning process required less than 10 hours, while training from scratch took less than 3 days. Other experiments, such as those involving Logit Lens, were completed in less than 1 hour.

Licenses for Existing Assets & Open Access to Data and Code. For the following models, we use the checkpoints provided by Huggingface. For all the trained models, we use default hyperparameters during all the training but with different random seeds.

- GPT-2-XL: https://huggingface.co/openai-community/gpt2-xl, Modified MIT License

- GPT-2: https://huggingface.co/openai-community/gpt2, Modified MIT License
- GPT-J: https://huggingface.co/EleutherAI/gpt-j-6b, Apache-2.0 License
- Phi2: https://huggingface.co/microsoft/phi-2, MIT License
- GPT-3.5 and GPT-4: https://chatgpt.com/ or https://openai.com/index/openai-api/
- PaLM-2 https://ai.google/discover/palm2/