FOURIER HEAD: HELPING LARGE LANGUAGE MODELS LEARN COMPLEX PROBABILITY DISTRIBUTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

As the quality of large language models has improved, there has been increased interest in using them to model non-linguistic tokens. For example, the Decision Transformer recasts agentic decision making as a sequence modeling problem, using a decoder-only LLM to model the distribution over the discrete action space for an Atari agent. However, when adapting LLMs to non-linguistic domains, it remains unclear if softmax over discrete bins captures the continuous structure of the tokens and the potentially complex distributions needed for high quality token generation. We introduce a neural network layer, constructed using Fourier series, which we can easily substitute for any linear layer if we want the outputs to have a more continuous structure. We perform extensive analysis on synthetic datasets, as well as on large-scale decision making and time series forecasting tasks. We also provide theoretical evidence that this layer can better learn signal from data while ignoring high-frequency noise. All of our results support the effectiveness of our proposed Fourier head in scenarios where the underlying data distribution has a natural continuous structure. For example, the Fourier head improves a Decision Transformer agent's returns by 46% on the Atari Seaquest game, and increases a state-of-the-art times series foundation model's forecasting performance by 3.5% across 20 benchmarks unseen during training.

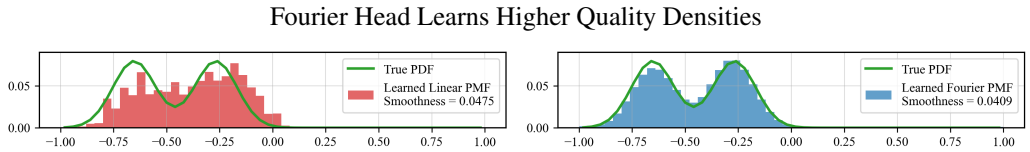

Figure 1: We task an MLP with learning to approximate a continuous bimodal density using a categorical distribution and a cross entropy objective. We observe that a standard linear classification head fails to distinguish between the two modes, and overfits to high-frequency noise in the training set. In contrast, our proposed Fourier head learns a smoother, more accurate categorical distribution.

1 INTRODUCTION

Human language can be viewed as a discretization for a *continuous*, often *probabilistic* representation of the world that is construed in our mind (Spivey, 2008). The continuous structure can be partially captured by language models with their token embeddings, where "nearby" tokens are embedded to have latent representations with high cosine similarities. The embeddings themselves are acquired as a result of the data-driven learning process. Can we, based on rich prior knowledge about the continuous world, inform the language model about the underlying continuity of its inputs, like the fact that the word "emerald" is more similar to "shamrock" than "pine" when they are used to describe different shades of green? As large language models (LLMs) have evolved into "foundation models" that are adapted to a diverse range of tasks, tokens that are *a priori* continuous are more essential than ever, for example for arithmetic computations (Liu et al., 2023), decision making with continuous or discrete actions (Chen et al., 2021), future anticipation and time-series forecasting (Ansari et al., 2024), or simply drawing random numbers given a probability distribution (Hopkins et al., 2023).

We view the problem of informing LLMs to utilize the continuity prior from the perspective of probability density estimation. For simplicity, we adopt the standard next token prediction framework whose training objective is softmax cross entropy. Assuming non-overlapping vocabulary, continuous values can be discretized via binning (Ansari et al., 2024). On one hand, the linear head adopted by LLMs independently projects each token into probabilities, and has the expressive power to flexibly approximate arbitrary probability density functions subject to the "quantization" errors. The linear head however does not consider any continuous structure that resides among the tokens (i.e. a random re-shuffle of the tokens in the vocabulary would not change the predictions). On the other hand, a head based on a parameterized distribution (e.g. Gaussian or Gaussian Mixtures) naturally incorporates the continuous structure, but is often too simple (and overly "smooth") to account for multi-modal distributions for future prediction or decision making. Can we design a head that is both expressive and incorporates continuous structures?

We introduce the Fourier head, motivated by Fourier series as universal function approximators. **The Fourier head learns a *continuous* probability density function, and returns a *discrete* approximation of it.** Intuitively, returning a discretization of a continuous density in this way allows the classification head to better model the low-frequency signals from the training data, because overfitting to high-frequency noise is explicitly penalized by the Fourier head's built-in regularization. At a high level, the Fourier head inputs $x \in \mathbb{R}^n$, uses a linear layer to learn the coefficients for a Fourier series with N frequencies over $[-1, 1]$, and quantizes the interval $[-1, 1]$ into m equal bins. Then, the Fourier head evaluates the learned Fourier PDF at those m bin center points, and returns those m likelihoods as a categorical distribution. The Fourier head is constructed using the Fourier Basis Density Model (De la Fuente et al., 2024).

Our first contribution is to reveal the underlying principle on the trade-off between the Fourier head's expressive power and the "smoothness" of the predicted distributions. We have proven a theorem which demonstrates a scaling law for the Fourier head. Namely, as we increase the quantity of Fourier coefficients that the Fourier head learns, the layer is able to model increasingly more complicated distributions; however, the Fourier head will necessarily fit to more high-frequency noise, thereby outputting categorical distributions which are less smooth.

Our second contribution is to propose a practical implementation of the Fourier head that allows us to handle sequential prediction tasks by modeling complex multi-modal distributions. Alongside our implementation, we propose strategies to improve the layer's performance, including Fourier coefficient norm regularization, weight initialization, and the choice of how many Fourier frequencies to use. We demonstrate the effectiveness of the Fourier head on two large scale tasks, where intuitively a continuity inductive bias over the output dimensions ought to help the model's generation performance. In the first task, an offline RL agent which uses a decoder-only transformer to model the next-action distribution for an Atari game, we improve returns by 46%. And in the second, we outperform a state-of-the-art time series foundation model on zero-shot forecasting by 3.5% across a benchmark of 20 datasets unseen during training. We commit to open source our models and code.

2 Fourier Head

2.1 Fourier Head: Motivation

When practitioners apply LLMs to model complex probability distributions, a standard technique is to quantize the latent space into m tokens and learn a conditional categorical distribution over those tokens. We share two examples here:

- The Decision Transformer (Chen et al., 2021) models an Atari agent's behavior in the Seaquest game by learning a categorical distribution over the 18 possible actions (move left, move right, shoot left, etc.). They use an encoder-only transformer architecture.
- The Chronos time series foundation model (Ansari et al., 2024) models the distribution of next numerical values by quantizing the closed interval $[-15, 15]$ into 4096 bins, and learning a categorical distribution over those bins. They use an encoder-decoder transformer.

In a pure language modeling task, token ID 1000 and token ID 1001 likely represent unrelated words. However, in a task where the token IDs represent numerical values, the token ID 1000 and 1001 would represent numbers that are close together.

The final layers of an LLM for such a task are generally a linear layer, followed by softmax, followed by cross entropy loss. We hypothesize that in scenarios where nearby token IDs encode similar items, an inductive bias that encourages them to have similar probabilities will improve performance. A generic linear layer learns an unstructured categorical distribution and thereby allows more arbitrary probabilities. **In this work, we propose to give the model this inductive bias by letting the classification head learn a categorical distribution as the discretization of a continuous learned function from a suitably flexible class.** In this paper, we consider the very flexible class of truncated Fourier series with N frequencies. These are functions of the form

$$f(x) = a_0 + \sum_{k=1}^{N} \big(a_k \cos(k\pi x) + b_k \sin(k\pi x)\big). \tag{2.1}$$

Fourier series are a classical tool for solving quantitative problems (Stein & Shakarchi, 2003) because functions like Equation 2.1 are universal function approximators, with the approximation improving as N increases.

2.2 Fourier Head: Definition

We now propose a replacement for the generic linear layer token classification head, built using Fourier series. We call our replacement the **Fourier Series Classification Head**, or the **Fourier head** for short. The Fourier head inputs any vector $x \in \mathbb{R}^n$, and outputs a categorical distribution in $\mathbb{R}^m$. For a high level summary of how it works–*the Fourier head inputs $x \in \mathbb{R}^m$, uses a linear layer to extract the coefficients for a Fourier series over $[-1, 1]$, quantizes the interval $[-1, 1]$ into m equal bins, evaluates the learned Fourier PDF at those m bin centerpoints, and returns those m likelihoods as a categorical distribution.* We formally define this layer in Algorithm 1, and we present a concrete low-dimensional demonstration of the Fourier head in action in Section 2.3. The Fourier head is constructed using the Fourier Basis Density Model from (De la Fuente et al., 2024). For more details on the original method (e.g. justification for how learning the autocorrelation coefficients guarantees that the Fourier series has integral 1, and justification for normalizing the Fourier coefficients by $\Re(c_0)$) we refer the author to (De la Fuente et al., 2024).

Algorithm 1 Fourier head

Hyperparameters: the input dimension n, output dimension m, number of frequencies N
Initialization: define a linear layer $A : \mathbb{R}^n \to \mathbb{R}^{2(N+1)}$ // maps input to autocorrelation coefficients
Step 1: INPUT $x = (x_1, \dots, x_n) \in \mathbb{R}^n$
Step 2: $(\alpha_0, \beta_0, \dots, \alpha_N, \beta_N) \leftarrow Ax$
Step 3: $a_k \leftarrow \alpha_k + i\beta_k \in \mathbb{C}$, for every $k = 0, \dots, N$ // compute autocorrelation coefficients
Step 4: $c_k \leftarrow \sum_{\ell=0}^{N-k} a_\ell a_{\ell+k}^* \in \mathbb{C}$, for every $k = 0, \dots, N$ // compute Fourier coefficients
Step 5: $p(z) = \frac{1}{2} + \Re\left(\sum_{k=1}^{N} \frac{c_k}{\Re(c_0)} \exp(ik\pi z)\right)$ // define Fourier PDF over $[-1, 1]$
Step 6: $\omega_k \leftarrow (-m + 1 + 2k)/m$, for every $k = 0, \dots, m-1$ // define m bin centerpoints
Step 7: $y_k \leftarrow \frac{p(\omega_k)}{\sum_{j=0}^{m-1} p(\omega_j)}$, for every $k = 0, \dots, m-1$ // evaluate PDF at m bin centerpoints
Step 8: OUTPUT $(y_1, \dots y_m) \in \mathbb{R}^m$ // by design, we know that each $y_i \geq 0$, and $\sum_{k=1}^{m} y_k = 1$

2.3 Fourier head: motivating example

To illustrate a simple problem setting where the design of the Fourier head is appropriate, we use it as a drop in replacement for a linear classification head in the Audio Spectrogram Transformer (Gong et al., 2021). We consider the task of beats per minute (BPM) classification for metronome-like audio samples within the tempo range $\{50, 51, \dots, 210\}$. While this task is not difficult, we use this audio classification task to illustrate some of the design choices one can make when using the Fourier head. In this case, it is natural to group the BPMs into contiguous bins $\{[50, 54], [55, 59], \dots\}$ and use the Fourier head to classify them. These bins have a natural continuous structure, which is where the Fourier head performs well. We also expect that the categorical distribution over possible BPMs for a given audio clip ought to be unimodal and therefore require few frequencies to approximate. In fact, our best performing model for this example uses only one frequency.

We initialize the Audio Spectrogram Transformer with pretrained weights from AudioSet (Gemmeke et al., 2017), and we train two different models–one with a standard linear classification head, and one with the Fourier head. The Fourier head outperforms the linear classification head by an F1 score improvement of $+118\%$. We attribute this success to inductive bias of continuity that the Fourier head imparts. In Figure 2 we present the learned probability masses of both heads on the same input sample. This graph illustrates that the Fourier head learns smoother PMFs than the linear head, a concept which we will later formalize and explore.

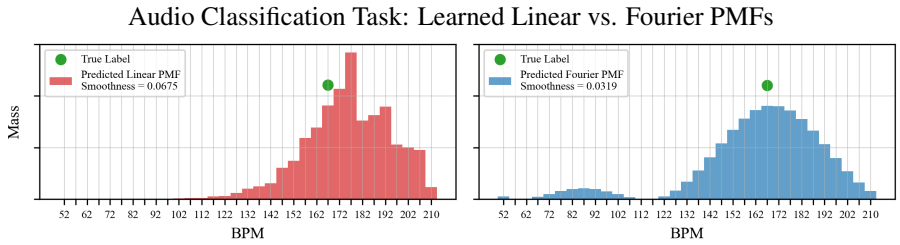

Figure 2: Comparison between the PMF learned by the linear head, and the Fourier head with 2 frequencies, for the toy BPM classification task, on a single audio example. We observe that the Fourier head learns a smoother categorical distribution over its predicted values, and is better centered around the ground truth label. We also note the small mini-sine wave artifacting on the left side of the Fourier model, which tends to occur when using few frequencies.

2.4 Fourier head: Details for using it during training

We highlight the main design choices for a user when applying the Fourier head in practice.

Training objective: The Fourier head inputs a signal $x \in \mathbb{R}^n$ and extracts from that signal an intermediate representation of a probability distribution $p(x)$ defined over $[-1, 1]$. This probability distribution has a closed formula equal to a Fourier series. In our experiments, we optimize the parameters of the Fourier PDF by discretizing it over the latent space and training using cross entropy loss. However, we should note that the Fourier layer allows MLE training directly on continuous values, by evaluating the Fourier PDF directly on the ground truth value in the latent space. But for consistency of comparison, and to demonstrate how easy it is to swap the Fourier head with a linear layer, we use softmax cross-entropy loss as the objective.

Choice of hyperparameter N: The Fourier head has one crucial hyperparameter–namely, the number of frequencies. How should one choose this in practice? We offer Theorem 3.3 as guiding principle beyond simple trial and error. This result provides a scaling law which formalizes the smoothness-expressive power trade-off in choosing the number of frequencies. In general, using more frequencies leads to more expressive power, and generally better success metrics, but at the cost of a learning less smooth densities, as well as more model parameters.

Fourier regularization: For a given number of frequencies N, there could be many learned Fourier models that fit the given data equally well. To encourage a smoother learned model and penalize unnecessary high frequency content, we follow (De la Fuente et al., 2024) and add a regularization term that measures the total squared variation for the Fourier model, to prevent higher order Fourier coefficients from growing too large during training. This helps ensure that the learned Fourier PDF doesn't overfit to noise in the data, and therefore has a bias towards learning smoother densities. In the notation from Algorithm 1, this means adding a regularization term of $\gamma \cdot \frac{2\pi^2}{m} \sum_{k=1}^{m} k^2 |c_k|^2$ to the loss function, where γ is a hyperparameter. When picking regularization strength, we find that in the low-frequency domain (e.g. frequencies in the single digits) using $\gamma = 0$ works best, and in the high-frequency domain (e.g. greater than 10 frequencies), using $\gamma = 10^{-6}$ works best.

Binning strategy: The choice of how we bin the data can affect performance significantly. As we already discussed, we should only apply the Fourier head when nearby bins are "similar" in some sense. This means we should order our bins in a semantically meaningful ordering. Further, in the case where the bins represent quantized numerical values over a continuous latent space, it can be helpful to use a "mixed-precision" binning strategy. For instance, if we want to model all values

from $[-15, 15]$, but we find that most values lie in the range $[-1, 10]$, then we should allocate a higher proportion of bins to the dense data interval. Specifically, if we would like to use m total bins to quantize the data, then we control the allocation of bins using a hyperparameter $d \in [0, 1)$, where $\lfloor d \cdot m \rfloor$ uniformly spaced bins are allocated to the sparse data interval while the remaining $m - \lfloor d \cdot m \rfloor$ bins are allocated to the dense range (estimated from training data). This is motivated and supported by the Fourier theory as well, since by increasing precision in the dense data range we are effectively de-localizing the quantized data distribution, which leads to a more localized Fourier spectrum. This lets us obtain a quicker decay of higher frequency content, which ensures that we can more effectively learn the same distribution with lower-frequency Fourier heads. We also note that (De la Fuente et al., 2024) proposes an optional re-parameterization step that replaces the periodic domain $[-1, 1]$ with the real line, although we don't use that in this work.

Weight initialization: The learned parameters for the Fourier head consist of the learned linear layer which extracts autocorrelation parameters. In PyTorch, the linear layers uses the He initialization (He et al., 2015) by default, which ensures that the linear layer outputs values close to zero in expectation. Similarly, it's better for the learning dynamics for the Fourier densities to be initialized to uniform $p(z) \approx 1/2$. We accomplish this by dividing the weights and biases by a large number, such as 1000, after He initialization; this guarantees that the linear layer outputs very small values, so that Fourier coefficients output from the autocorrelation step are very small as well.

3 Theory

3.1 "Smoothness": A Metric for High Frequency Content

In this subsection, we propose a smoothness metric which inputs a categorical disribution $y = (y_1, \ldots, y_m) \in \mathbb{R}^m$, and assigns a numerical value depending on how smooth it is. The score will output 0 if y is the smoothest possible categorical distribution, and larger values if y is less smooth. We will first specify what we mean by "smooth":

Heuristic 3.1. *We say a function is **smooth** if it contains **very little high-frequency information**.*

For example, the uniform categorical distribution contains no high-frequency information, so it is the smoothest possible function, and should get a smoothness score of 0. In contrast, a categorical distribution containing samples from $\sin(100\pi x)$ contains lots of high frequency information, so it should get a smoothness score greater than 0. We seek to define a metric which measures smoothness according to Heuristic 3.1.

We will first develop a smoothness metric in the general case of a function $f : [a, b] \to \mathbb{R}$, then specialize to case of the discrete categorical distribution that we consider in the paper. If we let $\alpha_\sigma \in \mathbb{R}$ be weights satisfying $\int_0^\infty \alpha_\sigma d\sigma = 1$, and D be some measure of discrepancy such as L^2, and let $g_\sigma(x) * f(x)$ denote the convolution of $f(x)$ with a Gaussian kernel of standard deviation σ, then it is reasonable to define the smoothness of f to be the quantity

$$s(f) := \int_0^\infty \int_a^b \alpha_\sigma D[f(x), g_\sigma(x) * f(x)] dx d\sigma. \tag{3.1}$$

In this expression, the discrepancy $D[f(x), g_\sigma(x) * f(x)]$ measures how different $f(x)$ is from a Gaussian-smoothed version of itself. Because the Gaussian is a low-pass filter, we can interpret Equation 3.1 as saying, at a high level, that a function is "smooth" if it doesn't change that much when you remove high frequency content from it.

In our experiments, we consider discrete categorical distributions, and wish to evaluate how smooth they are in a numerically tractable way. Accordingly, we define a specific case of this as follows.

Definition 3.2 (Smoothness metric for categorical distributions)**.** *Suppose $y = (y_1, \ldots, y_m) \in \mathbb{R}^n$ is a categorical distribution, so every $y_k \geq 0$ and $\sum_{k=1}^m y_k = 1$. Denote by $g_\sigma(x)$ the discrete Gaussian kernel with zero-padding of standard deviation σ. Define the weights $\alpha_\sigma = 6/\pi^2\sigma^2$. Then we define the **smoothness** of y to be the constant*

$$s(y) := \sum_{\sigma=1}^{\infty} \alpha_\sigma \| f(x) - (g_\sigma * f)(x) \|_2 \tag{3.2}$$

We direct the curious reader to Appendix B, where we conduct additional experiments to justify this choice of smoothness metric for our experiments.

3.2 A Scaling Law for the Fourier Head, in Frequency-aspect

In this subsection, we share a theorem that analyzes the quality of the Fourier head as the quantity of frequencies changes. We refer to this as the **Fourier head scaling law** as it quantifies the trade-off between modeling capacity and smoothness as the number of frequencies increases. On one hand, it is a celebrated result from Fourier analysis that a Fourier series with a greater number of frequencies models a larger class of functions; but on the other hand, we show that increasing frequencies also incurs loss in smoothness. This is to be expected, as we designed our smoothness metric with the intention of identifying a distribution as less smooth if it contains more high-frequency information.

Theorem 3.3. *(Fourier head scaling law.) Consider a Fourier head with input dimension n, output dimension m, and N frequencies. Suppose that $1 \ll N < \frac{m}{2}$. Then the following are true:*

1. *(**Increasing N improves modeling power**.) As N increases, the Fourier head is capable of learning a larger class of densities.*
2. *(**Increasing N degrades smoothness**.) Consider an input to the Fourier head $x \in \mathbb{R}^n$, and denote by $f_x : [-1, 1] \to \mathbb{R}$ the optimal conditional distribution that we would like the Fourier head to approximate for this input. We assume that f_x is twice continuously differentiable, and that its Fourier coefficients decay on the order of $1/k^2$. Denote by $f_{x,N}$ the truncation of f_x to its first N frequencies, and denote by $y^{(N)}$ a discretized version of $f_{x,N}$ into m bins. Then, there exist constants $C_1, C_2 > 0$ such that*

$$s(y^{(N)}) \approx \sqrt{m}C_1 - \frac{\sqrt{m}C_2}{N^3} + O(1/N^4). \tag{3.3}$$

Note that the smoothness scaling law asymptotic in Equation 3.3 shows that as N increases, so does $s(y^{(N)})$. In part (2), since f_x is at least twice continuously differentiable, we already know its Fourier coefficients corresponding to the k-th frequency are in $O(1/k^2)$ (Stein & Shakarchi, 2003, Ch.2, Cor. 2.4). Thus, our assumption that the Fourier coefficients decay quadratically is reasonable and our Fourier weight decay regularization helps toward ensuring that this condition is met in practice as well. We include a full proof of this result in Appendix A.

4 Toy Example: Learning A Continuous Conditional Distribution

We demonstrate the advantage of using the Fourier head to learn a probability distribution for a simple task: learning the conditional distribution of the third number in the sequence given the first two. Here we will use $q(z)$ to denote the quantization of z.

Dataset: We create 3 synthetic datasets, which we name **Gaussian**, **GMM-2**, and **Beta**. Each dataset consists of 5000 quantized triples $\{(q(x), q(y), q(z))\} \subseteq [-1, 1]^3$. Crucially, z is sampled from a distribution which is conditioned on x and y, and we have an explicit closed formula for this distribution. By design, the Gaussian dataset is unimodal in z, whereas the more challenging GMM-2 and Beta datasets are not unimodal. Full details about the datasets can be found in Appendix C.

Task: Predict the conditional distribution of $q(z)$ given the quantized tuple $(q(x), q(y))$.

Model architecture: Our model is an MLP with ReLU activations and one hidden layer, which maps $\mathbb{R}^2 \to \mathbb{R}^{64} \to \mathbb{R}^{32} \to \mathbb{R}^{50}$. The output of the model has dimension 50 because we quantize into 50 bins. We consider two baselines alongside the Fourier model. For the first baseline, the classification head is a linear layer; for the second baseline, the classification head is a Gaussian model mixture classification layer with two Gaussians, where the means and standard deviations are learned; for the Fourier model, the classification head is the Fourier head. We sweep over frequencies $N = 2, 4, \ldots, 20$, and consider regularization $\gamma \in \{0, 10^{-6}\}$. We train those models via cross entropy loss. We also consider a regression-based model, trained using MSE.

Model evaluation: We use three metrics for evaluation. Our first metric is the average KL divergence $D_{\text{KL}}(q(\mathcal{P}(x, y))||M(q(x), q(y)))$, where $\mathcal{P}(x, y)$ is the fixed conditional distribution of z

given (x, y); $q(\mathcal{P}(x, y))$ is the quantized approximation of $\mathcal{P}(x, y)$, obtained by evaluating the density function of $\mathcal{P}(x, y)$ at the bin centers, multiplying by the bin width, and finally scaling by the sum of the likelihoods; and $M(q(x), q(y))$ denotes the predicted categorical conditional distribution of $q(z)$. Our second metric is smoothness. And our third metric is MSE, where we consider the expected value of $q(z)$ under the learned categorical distribution as a prediction for $q(z)$.

Results: The metrics for the best performing model on each dataset are reported in Table 1. Figure 3 presents sample visualizations of the learned conditional distributions alongside the true densities. And in Appendix C, we present the results of a study on the impact of number of frequencies and Fourier regularization. Notably, this study provides empirical evidence for the Fourier head scaling law in Theorem 3.3, as it demonstrates that for all datasets, as frequency decreases, the smoothness degrades, and model performance improves until it reaches a saturation point. Crucially, we observe that the Fourier head flexibly learns all three distributions better than the linear baseline does. We note that the Fourier head outperforms the linear head on MSE as well; we include a complete comparison with both Linear and GMM head baselines in Appendix C.

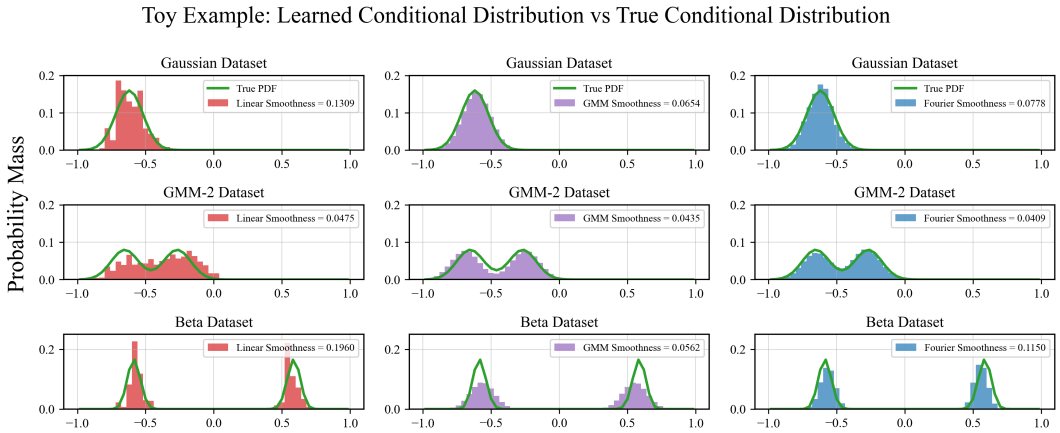

Figure 3: Comparison between the PMFs learned by the linear head, GMM head, and the Fourier head, for each of the datasets in the toy example. We observe that the Fourier head learns a smoother categorical distribution than the linear head over its predicted values. Furthermore, the Fourier head better fits the true conditional PDF; this is reflected in the KL divergence and smoothness metrics.

	KL Divergence (↓)		Smoothness (↓)	
Dataset	Linear	Fourier	Linear	Fourier
Gaussian	0.170 ± 0.052	$\mathbf{0.116} \pm 0.043$	0.116 ± 0.049	$\mathbf{0.057} \pm 0.011$
GMM-2	0.238 ± 0.032	$\mathbf{0.146} \pm 0.033$	0.068 ± 0.022	$\mathbf{0.038} \pm 0.007$
Beta	0.234 ± 0.032	$\mathbf{0.191} \pm 0.016$	0.127 ± 0.044	$\mathbf{0.076} \pm 0.021$

Table 1: We compare metrics between the linear head, and the Fourier head with 12 frequencies and no regularization, for every dataset in our toy example. We observe that the Fourier head outperforms the linear head across all metrics. Notably, using Fourier head improves the KL divergence (the primary success metric) on average by approximately 40%. We aggregate metrics over 4 different seeds and report the standard deviation.

5 Large-Scale Study: Offline Reinforcement Learning

The Decision Transformer (Chen et al., 2021) casts the problem of reinforcement learning as sequentially modeling rewards, states, and actions. Here, we study the performance of the Decision Transformer on the Seaquest game in the Atari (Bellemare et al., 2013) benchmark. The Seaquest game contains 18 actions, with two groups of eight actions that have a natural "closeness" metric defined on them: move left, up left, up, up right, right, down right, down, down left; as well as shooting in those eight directions. In their architecture, a decoder-only language model (Radford

et al., 2018) encodes the context and then maps it through a linear layer, outputting a categorical distribution over the 18 possible actions. At test time, the agent chooses its next action by sampling from the learned next-action distribution. In our study, we replace that linear classification head with a Fourier head. Intuitively, this ought to give the model the prior that actions like "move left" and "move up left" are semantically similar, and therefore should have similar likelihoods. Our study confirms that the Fourier head outperforms the linear head in returns obtained by as much as 46%, in the reward conditioned setting considered in the paper, using identical training hyperparameters.

Task: In the Seaquest game, the agent moves a submarine to avoid enemies, shoot at enemies, and rescue divers. The Seaquest game contains 18 actions: move left, up left, up, up right, right, down right, down, down left; as well as shooting in those eight directions; as well as no move, and a generic fire move. We consider this task in the Offline RL setting. The agent observes the past states, actions, and rewards, as well as the return-to-go, and attempts to predict the action that matches what an agent operating like the dataset would likely do. We also consider three other Atari games with the same action space: BankHeist, DoubleDunk, and Gravitar.

Dataset: We use the same dataset from the original Decision Transformer implementation (Chen et al., 2021). This dataset consists of 500k transitions experienced by an online deep Q-network agent (Mnih et al., 2015) during training on each of the games.

Model architecture: (Chen et al., 2021) used the GPT-1 model (Radford et al., 2018) to autoregressively encode the context, which is then fed through a linear layer of dimension 18, and the model ultimately optimizes the cross entropy loss between the action logits and the ground truth action from the dataset. We refer to this model as the linear baseline. To create our Fourier-n version, we simply replace the linear head with a Fourier head.

Normalized Returns for Decision Transformer Agent

	Atari Game			
Classification Head	BankHeist	DoubleDunk	Gravitar	Seaquest
Linear head	-0.09 ± 0.05	-72.72 ± 33.08	1.32 ± 0.17	2.53 ± 0.63
Fourier head	$\mathbf{0.92 \pm 0.33}$	$\mathbf{45.45 \pm 36.36}$	$\mathbf{4.98 \pm 0.93}$	$\mathbf{3.70 \pm 0.47}$

Table 2: We present returns obtained by the Decision Transformer agent using the linear baseline, and the Fourier head, across the four Atari games. We compute the returns (mean and standard deviation) by averaging over four seeds. Across all these games, the Fourier head significantly improves the normalized returns obtained by the agent.

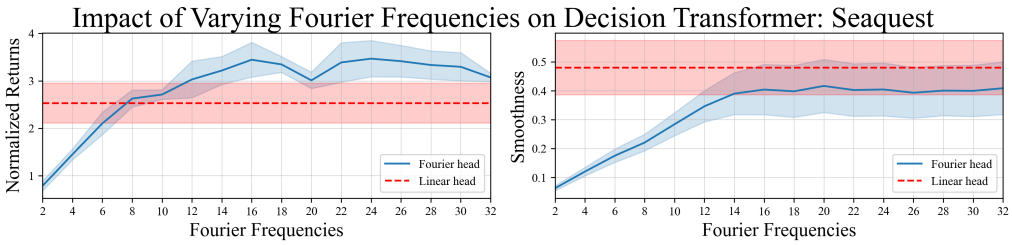

Figure 4: We present empirical results for how the quantity of Fourier frequencies impacts returns and smoothness for the imitation learning task. For normalized returns, higher is better; for smoothness, lower is better. We can see that the Fourier agent achieves higher normalized returns than the linear baseline agent when sufficiently many Fourier frequencies are used, while still learning smoother next-action distributions.

Model evaluation: We present results for the linear baseline, as well as the Fourier-n head, for $n = \{2, 4, 6, 8, \ldots, 30, 32\}$, across the four Atari games. We present mean reward totals for rollouts for each of these for the best epoch across 4 seeds. In Table 2, our results demonstrate that the Fourier head increases agent returns significantly. For example, for the Seaquest game, normalized returns increase by as much as 46.2%, and for Gravitar, normalized returns increase by as much as 300%. In Figure 4, we can see that the Fourier head is able to perform better in the Seaquest game

as the number of frequencies grows. We can also see that as we increase the quantity of frequencies, learned PMFs become less smooth, in accordance with Theorem 3.3. Qualitatively, we can also see that in Figure 12 (Appendix) the PMFs learned by the Fourier head are smoother. In Figure 13 (Appendix), we also include results for the remaining three games, BankHeist, DoubleDunk, and Gravitar. Across these games, the results show that the Fourier agent consistently achieves higher normalized returns than the linear baseline agent, while still learning smoother next-action distributions. And in Figure 9 (Appendix), we demonstrate that the regression model simply regresses to the mean of the conditional distribution. Accordingly, the regression model performs well for the unimodal Gaussian dataset, and it performs poorly for the bimodal datasets GMM-2 and Beta.

Ablations: We analyze whether model size has any effect on the relative performance of the Linear head and the Fourier head. The results in Figure 10 (Appendix) demonstrate that, across model sizes, the Decision Transformer with a Fourier head is better at learning high-quality next action distributions than the Decision Transformer with a Linear head. We also analyze whether dataset size has any effect on the relative performance of the Linear head and the Fourier head, and obtain a similar result. In Figure 11 (Appendix) we show that, across dataset sizes, the Decision Transformer agent with the Fourier head achieves larger returns than the agent with a linear head.

6 Large-Scale Study: Probabilistic Time Series Forecasting

The Chronos time series foundation models (Ansari et al., 2024) "learn the language of time series". They do this by approaching time series forecasting as language modeling–tokenizing the quantized number line, learning token embeddings for each of those quantized values, and finally learning a categorical distribution to decide what the next value ought to be. This model is built on top of the encoder-decoder T5 model (Raffel et al., 2020). In particular, this model normalizes time series values to the range $[-15, 15]$ and quantizes this interval into 4096 tokens. As usual for language modeling, the final layer is a linear map which learns a categorical distribution over next tokens. In particular, we observe that token i represents a number very close to tokens $i-1$ and $i+1$. However, we note that there is no inductive bias in the T5 architecture which pushes their likelihoods to be similar. This is not a hypothetical problem; in Figure 14 (Appendix), we can see that the linear next-token prediction PMFs fit to the noise, and appear very jagged. **Here, the motivation for replacing the linear head with the Fourier head is to "smooth" out this distribution, to help the forecasting model better learn the signal, and ignore the noise.** In Figure 14, we can see that the Fourier head accomplishes this successfully.

In this section, we study how performance of the Chronos time series foundation model changes when we pre-train using the Fourier head, instead of the linear head. For all of the frequencies that we consider, the Fourier head outperform the Chronos linear baseline on the MASE metric, while learning next token multinomials which are 8x smoother, with fewer parameters than the baseline.

Dataset: We use the same training dataset for large-scale pretraining that Ansari et al. (2024) used. We gather an evaluation benchmark of 20 time series datasets which were not seen during training. These 20 come from the zero-shot eval from (Ansari et al., 2024). The reader can check Appendix E for details on the training and evaluation datasets we used.

Model architecture: We use the Chronos model, which is built using the T5 architecture (Raffel et al., 2020). The original model has a linear classification head. For our study, we will replace this with a Fourier head with frequencies $N = 64, 128, 256, 550$. We use mixed precision binning; this is informed by an analysis of the Fourier spectrum of the next-token distribution, as described in Section 2.4). We also use Fourier weight decay regularization. For the task, the model learns to input time series context of length 512, and output a probabilistic forecast of length 64. At test time, the model chooses the next numerical token by sampling from the next-token distribution.

Model evaluation: We have two sets of metrics: model performance from (Ansari et al., 2024) (MASE measures the accuracy of median forecast, and WQL measures the quality of the probabilistic forecast), as well as our smoothness metric. Our Fourier metrics in Table 3 demonstrate that every Fourier model outperforms the linear baseline for MASE and Smoothness. Furthermore, for the largest Fourier model that we consider, Fourier outperforms linear on WQL as well.

Ablations: The results in Table 7 (Appendix) show that mixed precision binning and regularization improve the MASE and smoothness for the Fourier head.

Chronos Time Series Model	MASE ↓	WQL ↓	Smoothness ↓
Linear	0.883	0.750	0.1236 ± 0.0712
Fourier-64	0.875	0.798	$\mathbf{0.0027 \pm 0.0012}$
Fourier-128	0.872	0.767	0.0053 ± 0.0030
Fourier-256	0.859	0.755	0.0101 ± 0.0072
Fourier-550	**0.852**	**0.749**	0.0203 ± 0.0176

Table 3: We present large-scale experiments on Chronos time series forecasting. Notably, every Fourier model outperforms the linear baseline on MASE and smoothness metrics. Within the Fourier model class, decreasing the number of frequencies lets you trade off the continuity of the learned probability mass functions (smoothness) for the quality of the forecasts (MASE, WQL).

7 Related work

LLMs outside of natural language domains: LLMs are often adapted to domains beyond natural language, as general purpose sequence models. For example, they have been used in protein synthesis (Madani et al., 2023), time series forecasting (Ansari et al., 2024; Das et al., 2024; Jin et al., 2024; Nate Gruver & Wilson, 2023; Requeima et al., 2024; Jia et al., 2024; Zhou et al., 2023; Wang et al., 2024), music generation (Dhariwal et al., 2020; Agostinelli et al., 2023; Copet et al., 2023; Yuan et al., 2024), and as well as in decision making (Li et al., 2022; Chen et al., 2021).

We consider three categories to adapt LLMs to non-language domains: when the output of a language-trained LLM is used as a feature for some out-of-domain task; when a language-pretrained LLM is fine-tuned on a domain-specific task; and when an LLM architecture is trained on a domain-specific dataset from scratch. Our work directly considers the latter method of LLM adaptation, particularly in settings where the outputs approximate continuous values. We note that using LLMs to model numerical functions has seen success in continuing sequences (Mirchandani et al., 2023) but has been challenging for modeling samplers for probability distributions (Hopkins et al., 2023). In a related direction, Razeghi et al. (2022) found that model performance on numerical reasoning tasks is correlated with the frequency of specific numbers in its corpus. Further, some have re-framed continuous regression as a descretized classification problem to leverage LLMs in numerical modeling contexts (Song et al., 2024). While even frozen LLMs with no further training show interesting empirical results as regressors (Vacareanu et al., 2024), there is a conceptual mismatch between the downstream task and model construction because tokenized numerical values trained using cross-entropy loss does not explicitly enforce numerical relationships between the tokens.

Fourier series in neural networks: Many works leverage the Fourier transform as a data preprocessing step or a deterministic transformation within the network, or use Fourier analysis to motivate design choices. It is far less common to learn the Fourier series directly. De la Fuente et al. (2024) learned marginal univariate densities parameterized using a Fourier basis; our work extends their Fourier Basis Density model to multivariate settings with an autoregressive scheme. Our method learns conditional univariate densities using a Fourier basis, where the coefficients of the Fourier density model are input dependent. Sitzmann et al. (2020) proposed sinusoidal activation functions, which can be seen as learning the *frequencies* of a Fourier series; in contrast, we seek to fix the frequencies to the canonoical choice $\{1, 2, \ldots, N\}$, and learn the *amplitudes*. This allows the Fourier head to more directly benefit from approximation results from Fourier analysis.

8 Conclusion

We propose the Fourier head and demonstrate its positive impact on performance on several tasks. We prove scaling laws that characterize the trade-off between the model's expressivity and the smoothness of its output distribution. The Fourier head is a modular architecture that can be easily added to existing models that would benefit from the continuity inductive bias that the head imparts. The Fourier head extends the already extensive reach of LLMs into more diverse, numerical, and probabilistic domains. Future work includes exploring alternative training objectives that do not depend on discretizing probability density functions, and incorporating the Fourier head in general-purpose LLM training, where the head can be adaptively employed when needed.

9 Reproducibility Statement

We have made efforts to ensure reproducibility. In Algorithm 1 we provide all the mathematical details that one needs to reproduce the Fourier head. In Appendix E we prove our scaling law, Theorem 3.3, in full detail, and we list all assumptions in the statement of the theorem. Additionally, we release the research code in the supplemental section on OpenReview.

References

Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning*, 2024.

Alfredo De la Fuente, Saurabh Singh, and Johannes Ballé. Fourier basis density model. *arXiv preprint arXiv:2402.15345*, 2024.

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

Yuan Gong, Yu-An Chung, and James Glass. Psla: Improving audio tagging with pretraining, sampling, labeling, and aggregation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021. doi: 10.1109/TASLP.2021.3120633.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Aspen K Hopkins, Alex Renda, and Michael Carbin. Can llms generate random numbers? evaluating llm sampling in controlled domains. In *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*, 2023.

Tsuyoshi Inouye, Kazuhiro Shinosaki, H. Sakamoto, Seigo Toi, Satoshi Ukai, Akinori Iyama, Y Katsuda, and Makiko Hirano. Quantification of eeg irregularity by use of the entropy of the power spectrum. *Electroencephalography and Clinical Neurophysiology*, 79(3):204–210, 1991. ISSN 0013-4694. doi: https://doi.org/10.1016/0013-4694(91)90138-T. URL `https://www.sciencedirect.com/science/article/pii/001346949190138T`.

Furong Jia, Kevin Wang, Yixiang Zheng, Defu Cao, and Yan Liu. Gpt4mts: Prompt-based large language model for multimodal time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(21):23343–23351, Mar. 2024. doi: 10.1609/aaai.v38i21.30383. URL `https://ojs.aaai.org/index.php/AAAI/article/view/30383`.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212, 2022.

Yixin Liu, Avi Singh, C Daniel Freeman, John D Co-Reyes, and Peter J Liu. Improving large language model fine-tuning for solving math problems. *arXiv preprint arXiv:2310.10047*, 2023.

Madani, Krause, and et al. Greene. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41:1099–1106, 2023. doi: 10.1038/s41587-022-01618-2.

Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. Large language models as general pattern machines. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Shikai Qiu Nate Gruver, Marc Finzi and Andrew Gordon Wilson. Large Language Models Are Zero Shot Time Series Forecasters. In *Advances in Neural Information Processing Systems*, 2023.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI website*, 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. Impact of pretraining term frequencies on few-shot numerical reasoning. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 840–854, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.59.

James Requeima, John Bronskill, Dami Choi, Richard E Turner, and David Duvenaud. Llm processes: Numerical predictive distributions conditioned on natural language. *arXiv preprint arXiv:2405.12856*, 2024.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Xingyou Song, Oscar Li, Chansoo Lee, Bangding Yang, Daiyi Peng, Sagi Perel, and Yutian Chen. Omnipred: Language models as universal regressors. *CoRR*, abs/2402.14547, 2024. doi: 10.48550/ARXIV.2402.14547. URL `https://doi.org/10.48550/arXiv.2402.14547`.

Michael Spivey. *The continuity of mind*. Oxford University Press, 2008.

Elias M Stein and Rami Shakarchi. *Fourier analysis: an introduction*, volume 1. Princeton University Press, 2003.

Robert Vacareanu, Vlad Andrei Negru, Vasile Suciu, and Mihai Surdeanu. From words to numbers: Your large language model is secretly a capable regressor when given in-context examples. In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=LzpaUxcNFK`.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024.

Eric W. Weisstein. Square wave. From MathWorld–A Wolfram Web Resource, 2024. URL https://mathworld.wolfram.com/SquareWave.html. Accessed: September 16, 2024.

Ruibin Yuan, Hanfeng Lin, Yi Wang, Zeyue Tian, Shangda Wu, Tianhao Shen, Ge Zhang, Yuhang Wu, Cong Liu, Ziya Zhou, Ziyang Ma, Liumeng Xue, Ziyu Wang, Qin Liu, Tianyu Zheng, Yizhi Li, Yinghao Ma, Yiming Liang, Xiaowei Chi, Ruibo Liu, Zili Wang, Pengfei Li, Jingcheng Wu, Chenghua Lin, Qifeng Liu, Tao Jiang, Wenhao Huang, Wenhu Chen, Emmanouil Benetos, Jie Fu, Gus Xia, Roger Dannenberg, Wei Xue, Shiyin Kang, and Yike Guo. Chatmusician: Understanding and generating music intrinsically with llm. *arXiv preprint arXiv:2307.07443*, 2024.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One Fits All: Power general time series analysis by pretrained lm. In *NeurIPS*, 2023.

A MATHEMATICAL DETAILS

In this section we prove Theorem 3.3, the Fourier head scaling law. To do this, we must first discuss the Nyquist-Shannon Sampling Theorem. This result states that in order to avoid distortion of a signal (such as aliasing) the sampling rate must be at least twice the bandwidth of the signal. In the setting of the Fourier head, our sampling rate is $m/2$ because we have m bins uniformly spaced in $(-1, 1)$, and the bandwidth is $N/2$ because the frequency of $\sin(\pi N x)$ is $N/2$. Thus the Nyquist Theorem requires us to have

$$m/2 \geq 2 \cdot (N/2) = N$$

in order for the higher order frequency content learned by our model to not be fallacious when we are learning from only m bins.

We now present a theorem that provides a scaling law for the Fourier head. This result quantifies the trade-off between modeling capacity and smoothness as the number of frequencies increases. In order to prove this, we assume that the underlying function being learned is at least twice continuously differentiable, which implies that the Fourier coefficients corresponding to the k-th frequency of f_x are in $O(1/k^2)$ (Stein & Shakarchi, 2003, Ch.2, Cor. 2.4). Thus, our assumption that the Fourier coefficients decay quadratically is reasonable, and our Fourier quadratic weight decay regularization helps ensure that this condition is met in practice as well.

Theorem 3.3. *(Fourier head scaling law.) Consider a Fourier head with input dimension n, output dimension m, and N frequencies. Suppose that $1 \ll N < \frac{m}{2}$. Then the following are true:*

1. ***(Increasing N improves modeling power.)*** *As N increases, the Fourier head is capable of learning a larger class of densities.*
2. ***(Increasing N degrades smoothness.)*** *Consider an input to the Fourier head $x \in \mathbb{R}^n$, and denote by $f_x : [-1, 1] \to \mathbb{R}$ the optimal conditional distribution that we would like the Fourier head to approximate for this input. We assume that f_x is twice continuously differentiable, and that its Fourier coefficients decay on the order of $1/k^2$. Denote by $f_{x,N}$ the truncation of f_x to its first N frequencies, and denote by $y^{(N)}$ a discretized version of $f_{x,N}$ into m bins. Then, there exist constants $C_1, C_2 > 0$ such that*

$$s(y^{(N)}) \approx \sqrt{m} C_1 - \frac{\sqrt{m} C_2}{N^3} + O(1/N^4). \tag{3.3}$$

Proof. We first prove part (2). Let $b_j = -1 + \frac{2j+1}{m}, 0 \leq j < m$ be the center points of the m bins in $(-1, 1)$. Let $a_j(x) \in \mathbb{C}, -N \leq j \leq N$ denote the the Fourier coefficients of f_x corresponding to the first N frequencies. In other words,

$$f_{x,N}(y) = \sum_{j=-N}^{N} a_j(x) e^{\pi i j y} \tag{A.1}$$

is the function the Fourier head is learning. By our assumption, there exists a constant c_x such that $|a_k(x)| \approx c_x/k^2$. We want to study

$$s(y^{(N)}) = \sum_{\sigma=1}^{\infty} \alpha_\sigma \left(\sum_{j=0}^{m-1} \left| (f_{x,N} - g_\sigma * f_{x,N})(b_j) \right|^2 \right)^{1/2}. \tag{A.2}$$

Let $d_j(x)$ be the Discrete Fourier transform of $(f_{x,N} - g_\sigma * f_{x,N})(b_j)$ for $0 \leq j < m$. By Parseval's Theorem for the DFT, we have

$$\sum_{j=0}^{m-1} \left| (f_{x,N} - g_\sigma * f_{x,N})(b_j) \right|^2 = \frac{1}{m} \sum_{k=0}^{m-1} \left| d_k(x) \right|^2. \tag{A.3}$$

Since f_x is supported only on $(-1, 1)$, the convolution $g_\sigma * f_{x,N}$ is the same as $(g_\sigma I_{(-1,1)}) * f_{x,N}$, where $I_{(-1,1)}$ is the indicator function for $(-1, 1)$. Treating $g_\sigma I_{(-1,1)}$ as a function on $(-1, 1)$, let $h_{\sigma,n}$ be the Fourier coefficients of $g_\sigma I_{(-1,1)}$. Note that by the Convolution Theorem, we have

$$(f_{x,N} - g_\sigma * f_{x,N})(b_n) = \sum_{j=-N}^{N} a_j(x)(1 - h_{\sigma,j}) \cdot e^{\pi i j b_n}. \tag{A.4}$$

Thus, using the definition of DFT along with Equation A.4, we get

$$d_k(x) = \sum_{n=0}^{m-1} (f_{x,N} - g_\sigma * f_{x,N})(b_n) \cdot e^{-2\pi ikn/m} \tag{A.5}$$

$$= \sum_{n=0}^{m-1} \sum_{j=-N}^{N} a_j(x)(1 - h_{\sigma,j}) e^{\pi ijb_n} \cdot e^{-2\pi ikn/m} \tag{A.6}$$

$$= \sum_{j=-N}^{N} a_j(x)(1 - h_{\sigma,j}) \sum_{n=0}^{m-1} e^{\pi ij(-1+\frac{2n+1}{m})} \cdot e^{-2\pi ikn/m} \tag{A.7}$$

$$= \sum_{j=-N}^{N} a_j(x)(1 - h_{\sigma,j}) e^{\pi ij(1-1/m)} \sum_{n=0}^{m-1} e^{2\pi i(j-k)n/m}. \tag{A.8}$$

Note that

$$\sum_{n=0}^{m-1} e^{2\pi i(j-k)n/m} = \begin{cases} 0 & \text{if } j \not\equiv k \pmod m \\ m & \text{else.} \end{cases} \tag{A.9}$$

We therefore have

$$d_k(x) = \begin{cases} m \cdot a_k(x)(1 - h_{\sigma,k}) e^{\pi ik(1-1/m)} & \text{if } 0 \le k \le N \\ m \cdot a_{k-m}(1 - h_{\sigma,k-m}) e^{\pi i(k-m)(1-1/m)} & \text{if } m - N \le k \le m-1 \\ 0 & \text{otherwise.} \end{cases} \tag{A.10}$$

Using this in A.3, we obtain

$$\sum_{j=0}^{m-1} |(f_{x,N} - g_\sigma * f_{x,N})(b_j)|^2 = \frac{1}{m} \sum_{k=0}^{N} \left| m \cdot a_k(x)(1 - h_{\sigma,k}) e^{\pi ik(1-1/m)} \right|^2 \tag{A.11}$$

$$+ \frac{1}{m} \sum_{k=m-N}^{m-1} \left| m \cdot a_{k-m}(1 - h_{\sigma,k-m}) e^{\pi i(k-m)(1-1/m)} \right|^2 \tag{A.12}$$

$$= m \sum_{k=0}^{N} |a_k(x)(1 - h_{\sigma,k})|^2 + m \sum_{k=m-N}^{m-1} |a_{k-m}(1 - h_{\sigma,k-m})|^2, \tag{A.13}$$

where in the last step we used that $\left|e^{\pi ik(1-1/m)}\right| = 1 = \left|e^{\pi i(k-m)(1-1/m)}\right|$ since they are both complex exponentials. Now, since $g_\sigma I_{(-1,1)}$ is a real and even function, we know that $h_{\sigma,k}$ is real. Further, since the truncated Gaussian $g_\sigma I_{(-1,1)}$ is infinitely differentiable, we also know that $h_{\sigma,k} = O(1/k^2)$. Thus, using that $|a_k(x)| \sim c_x/k^2$, we see

$$|a_k(x)(1 - h_{\sigma,k})|^2 = |a_k(x)|^2 (1 - 2h_{\sigma,k}(x) + h_{\sigma,k}^2) \sim \frac{c_x^2}{k^4} + O(1/k^6) + O(1/k^8). \tag{A.14}$$

From A.14, it is clear that since we are interested in only the dominant asymptotic, we can safely ignore the higher order terms coming from the $h_{\sigma,k}$. As a result,

$$\sum_{j=0}^{m-1} |(f_{x,N} - g_\sigma * f_{x,N})(b_j)|^2 \approx m a_0(x)^2 + m \sum_{k=1}^{N} \frac{c_x^2}{k^4} + m \sum_{k=m-N}^{m-1} \frac{c_x^2}{(k-m)^4} \tag{A.15}$$

$$= m a_0(x)^2 + m \sum_{k=1}^{N} \frac{c_x^2}{k^4} + m \sum_{k=-N}^{-1} \frac{c_x^2}{k^4} \tag{A.16}$$

$$= m a_0(x)^2 + m \sum_{k=1}^{N} \frac{c_x^2}{k^4} + m \sum_{k=1}^{N} \frac{c_x^2}{k^4} \tag{A.17}$$

$$= m a_0(x)^2 + 2m \sum_{k=1}^{N} \frac{c_x^2}{k^4}. \tag{A.18}$$

We can approximate the dominant terms of the sum using an integral:

$$\sum_{k=1}^{N} \frac{1}{k^4} = \int_1^N \frac{1}{x^4}\,dx + \frac{1}{2}\left(1 + \frac{1}{N^4}\right) + O(1/N^4) = \frac{1}{3}\left(1 - \frac{1}{N^3}\right) + \frac{1}{2}\left(1 + \frac{1}{N^4}\right) + O(1/N^4). \tag{A.19}$$

Substituting the estimate into A.18, we obtain

$$\sum_{j=0}^{m-1} |(f_{x,N} - g_\sigma * f_{x,N})(b_j)|^2 \approx m\left(C^2(x) - \frac{2c_x^2}{3N^3} + O(1/N^4)\right), \tag{A.20}$$

where $C(x) = \sqrt{a_0(x)^2 + \frac{5}{3}c_x^2}$ is a constant depending upon x.

Using the Taylor expansion $(1+x)^{1/2} = 1 + \frac{x}{2} + O(x^2)$ about 0 and using that $N \gg 1$,

$$\left(mC^2(x) - \frac{2m}{3N^3} + mO(1/N^4)\right)^{1/2} \approx \sqrt{m}C(x)\left(1 - \frac{2c_x^2}{3C(x)N^3} + O(1/N^4)\right)^{1/2} \tag{A.21}$$

$$= mC(x)\left(1 - \frac{1}{2}\cdot\frac{2c_x^2}{3C(x)N^3} + O(1/N^4)\right) \tag{A.22}$$

$$= \sqrt{m}\left(C(x) - \frac{c_x^2}{3N^3} + O(1/N^4)\right). \tag{A.23}$$

Putting it all together in A.2, we get

$$s(y^{(N)}) \approx \sqrt{m}\left(C(x) - \frac{c_x^2}{3N^3} + O(1/N^4)\right)\sum_{\sigma=1}^{\infty}\alpha_\sigma \tag{A.24}$$

$$= \sqrt{m}\left(C(x) - \frac{c_x^2}{3N^3} + O(1/N^4)\right), \tag{A.25}$$

as claimed. This completes the proof of part (2).

The proof of part (1) is more straightforward. For any function f on $[-1, 1]$ that is at least twice continuously differentiable, we know that the Fourier series of f converges uniformly and absolutely to f (Stein & Shakarchi, 2003, Ch. 2, Cor. 2.4). In other words, the function f_N being learnt by the Fourier head converges uniformly and absolutely to f, which is precisely the statement of part (1). □

B Smoothness Metric

We will examine how the proposed smoothness metric Equation 3.1 behaves in a toy example setting to gain intuition for its behavior. Consider a square wave, which can be expressed as an infinite sum of odd integer harmonics that decay in amplitude proportional to their frequency:

$$f(x) = \frac{4}{\pi}\sum_{n=1,3,5,\ldots}^{\infty}\frac{1}{n}\sin\left(\frac{n\pi x}{L}\right). \tag{B.1}$$

Here, the wavelength is $2L$ (Weisstein, 2024).

We construct a truncated version of the square wave with a finite and fixed number of frequencies. The waveform will slowly approach its jagged, square shape as more sine waves are added. We frame these increasingly jagged waves as discretized multinomial densities to simulate the output of the Fourier head. To do this, we simply set the height to zero when the wave crest becomes negative and normalize the sum to 1. The output of this transformation for a few representative waveforms is pictured in Figure 5.

Intuitively, the truncated square wave with a single sine wave ought to be the smoothest. Thus our metric in this context should be smallest at that point, and increase monotonically as we add more sine waves. The plot in 6 demonstrates that this is indeed the case.

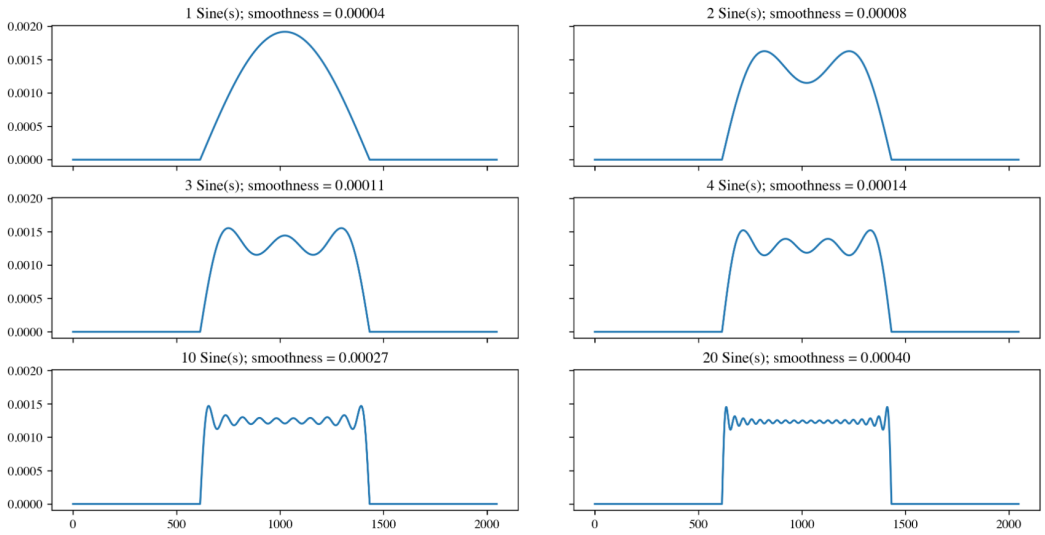

Figure 5: Truncated square waves framed as densities and their smoothness.

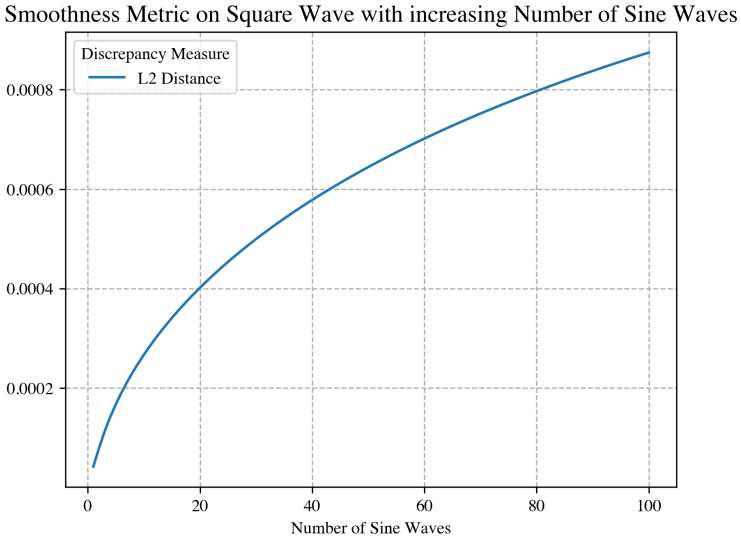

Figure 6: Values of the smoothness metric Equation 3.2 on our square-wave-like multinomials as we increase the number of sine waves. We desire the value of this metric to be close to zero when there are few sine waves, and be monotonically increasing with each additional wave, indicating that adding more high frequency content results in a less smooth distribution.

Choice of L2 Distance over L1 Distance: The proposed smoothness metric Equation 3.1 permits a general measure of discrepancy D, and we've chosen D to be L^2 distance as indicated in 3.2. We empirically observe that L^2 distance better preserves monotonicity than the L^1 for higher frequency content, thus motivating this choice. With a sample rate of 2048Hz, the L^1 distance exhibits some undesirable warping when our square-wave multinomial uses over 80 sine waves (see Figure 7). A Fourier head in a practical setting may possess several more than 80 frequencies; accordingly, we favor the L^2 distance as our discrepancy measure.

Alternative Notions of Smoothness: In validating our choice of smoothness metric, we compare it to the *spectral entropy* (Inouye et al., 1991), which has a similar purpose in quantifying the "smooth-

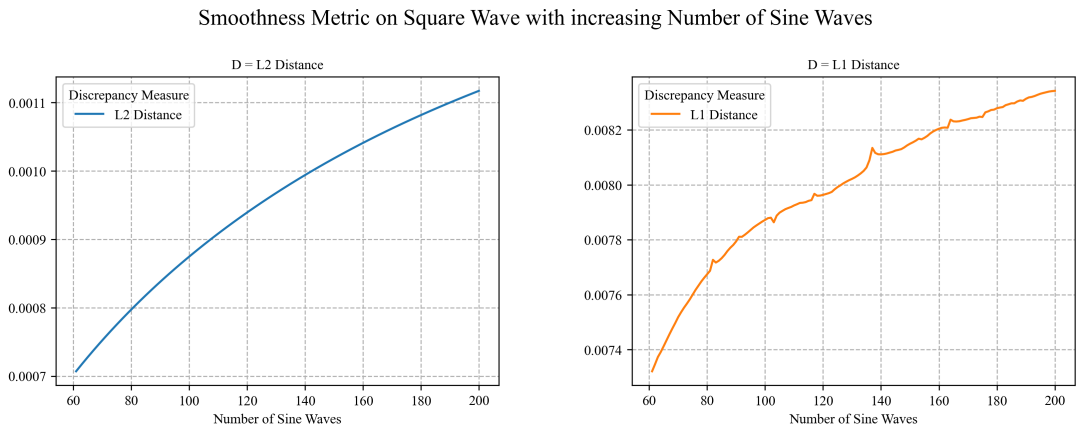

Figure 7: Values of the smoothness metric 3.2 on our square-wave-like multinomials as we increase the number of sine waves. On the right, we can see that L^1 as a discrepancy measure leads to non-monotonicity, motivating our choice of L^2 distance in measuring our results.

ness" of the frequency content of a signal. Spectral entropy is defined as the Shannon entropy of the *power spectral density* of a sampled signal f, which is defined as follows:

$$H(f; N) = \sum_{n \in N} p(n) \log_2 \left(\frac{1}{p(n)} \right) = - \sum_{n \in N} \frac{S_n}{S_{total}} \log_2 \left(\frac{S_n}{S_{total}} \right) \tag{B.2}$$

Here, N is the number of Fourier frequencies and S is the power of a frequency $n \in N$; S_n is the power spectrum of the nth frequency, and S_{total} is the power of the signal using all N frequencies. For some frequency at index n, S_n/S_{total} is called its relative power and $\sum_{n \in N} \frac{S_n}{S_{total}} = 1$ enables us to consider each frequency's power as a probability.

In the discrete case, the maximum entropy distribution is the uniform distribution. Thus, white noise will have the highest spectral entropy. This has the consequence that power spectral densities have more high frequency information will have lower entropy than that of white noise, provided that there is a relationship between amplitude and frequency. More concretely, blue noise, which is defined by the amplitude increasing proportionally to the frequency, will have lower spectral entropy than white noise. We sought a metric that always quantified 'sharper' signals like blue noise as less smooth. In Table 4, we frame sampled noises of different types as multinomial distributions to match our model setting by normalizing their amplitudes to be in $[0, 1]$ and normalizing their sum to 1. Our noise types are defined before normalization, in order of smoothest to sharpest:

- Brown: $S \propto \frac{1}{F^2}$
- Pink: $S \propto \frac{1}{F}$
- White: $S \sim \mathcal{N}(0, 1)$
- Blue: $S \propto F$

where S is the power density and F is the frequency. To obtain samples of each type, we first generate white noise. We do this by sampling a Gaussian with mean 0 and standard deviation 1 to obtain amplitudes for t samples. We then apply the Fourier transform, and multiply (or divide) the amplitudes of each component by their frequency, and apply the inverse Fourier transform to recover the waveform. Finally we adjust the range of amplitudes of the signal to be within $[0, 1]$ and normalize the sum to 1.

C TOY EXAMPLE DETAILS

Here we provide full details of the datasets used in our toy example of learning a known conditional distribution.

Discrepancy	Noise	Mean ± Std. Deviation	Diff	Delta	Desired Delta
L2	Brown	0.0003 ± 0.0001	n/a	n/a	n/a
L2	Pink	0.0017 ± 0.0002	0.0014	+	+
L2	White	0.0034 ± 0.0003	0.0016	+	+
L2	Blue	0.0038 ± 0.0003	0.0005	+	+
Spectral Entropy	Brown	0.4516 ± 0.0894	n/a	n/a	n/a
Spectral Entropy	Pink	0.3878 ± 0.0603	-0.0638	-	+
Spectral Entropy	White	0.4266 ± 0.0614	0.0388	+	+
Spectral Entropy	Blue	0.4191 ± 0.0583	-0.0076	-	+

Table 4: Smoothness measurements for four types of noise bootstrap aggregated over 1,000 trials. The color red emphasizes how the value of Spectral Entropy is undesirably not monotonic increasing for what we consider increasingly "sharp" noise types.

Dataset: We create a synthetic dataset $\mathcal{D} = \{(q(x), q(y), q(z))\} \subset \mathbb{R}^3$ as follows. Fix a probability distribution $\mathcal{P}_1 = \mathcal{P}_1(x)$ that is parameterized by one variable and a second distribution $\mathcal{P}_2 = \mathcal{P}_2(x, y)$ parameterized by two variables. Fix an interval $I \subset \mathbb{R}$. Sample x uniformly from I, sample $y \sim \mathcal{P}_1(x)$, and finally sample $z \sim \mathcal{P}_2(x, y)$. We can repeat this sampling procedure N times to obtain a set of N triples for which we know the conditional distribution of z given x and y. Finally, we quantize this set to a fixed number of uniformly spaced bins in the range $[-1, 1]$ to obtain the dataset $\mathcal{D}_{\mathcal{P}_1,\mathcal{P}_2}$. We will denote the quantization of z by $q(z)$. We quantize into 50 bins and our dataset has size 5000, with a 80-20 split between the train and test set. We describe three choices for the distributions we used to create our datasets. We fix $I = [-0.8, 0.8]$ and $\sigma^2 = 0.01$ in all of them.

1. *Gaussian dataset:* $\mathcal{P}_1(x) = \mathcal{N}(x, \sigma^2)$, and $\mathcal{P}_2(x, y) = \mathcal{N}(y, \sigma^2)$.
2. *GMM-2 dataset:* $\mathcal{P}_1 = \text{Uniform}(I)$, and $\mathcal{P}_2(x, y)$ is a GMM centered at x and y with variance σ^2.
3. *Beta dataset:* $\mathcal{P}_1(x) = \mathcal{N}(x, \sigma^2)$, and $\mathcal{P}_2(x, y) \sim U(\{\pm 1\}) \times \text{Beta}(100\,|x|, 100\,|y|)$, where $U(\{\pm 1\})$ denotes the Rademacher distribution supported on $\{\pm 1\}$ with probability $1/2$ each.

Additional results: In Figure 8, we present results from training over a range of frequencies, and for each frequency we ran experiments with and without Fourier regularization. In Table 6 we present results on the MSE metric, that show that the Fourier head outperforms the linear classification head.

	KL Divergence (↓)		
Dataset	Linear	GMM	Fourier
Gaussian	0.170 ± 0.052	**0.026** ± 0.011	0.116 ± 0.043
GMM-2	0.238 ± 0.032	**0.030** ± 0.006	0.146 ± 0.033
Beta	0.234 ± 0.032	0.407 ± 0.012	**0.191** ± 0.016

	Smoothness (↓)		
Dataset	Linear	GMM	Fourier
Gaussian	0.116 ± 0.049	0.068 ± 0.012	**0.057** ± 0.011
GMM-2	0.068 ± 0.022	0.043 ± 0.009	**0.038** ± 0.007
Beta	0.127 ± 0.044	**0.061** ± 0.003	0.076 ± 0.021

Table 5: KL divergence and Smoothness for the three classification heads (Linear, GMM, and Fourier) on each of the three synthetic datasets (Gaussian, GMM-2, Beta). As expected, the GMM head achieves the best KL divergence on the Gaussian and GMM-2 datasets, as their conditional distributions are Gaussian. However, the Fourier head has the best KL divergence on the Beta dataset. This demonstrates the flexibility of the Fourier head in modeling non-Gaussian distributions as well.

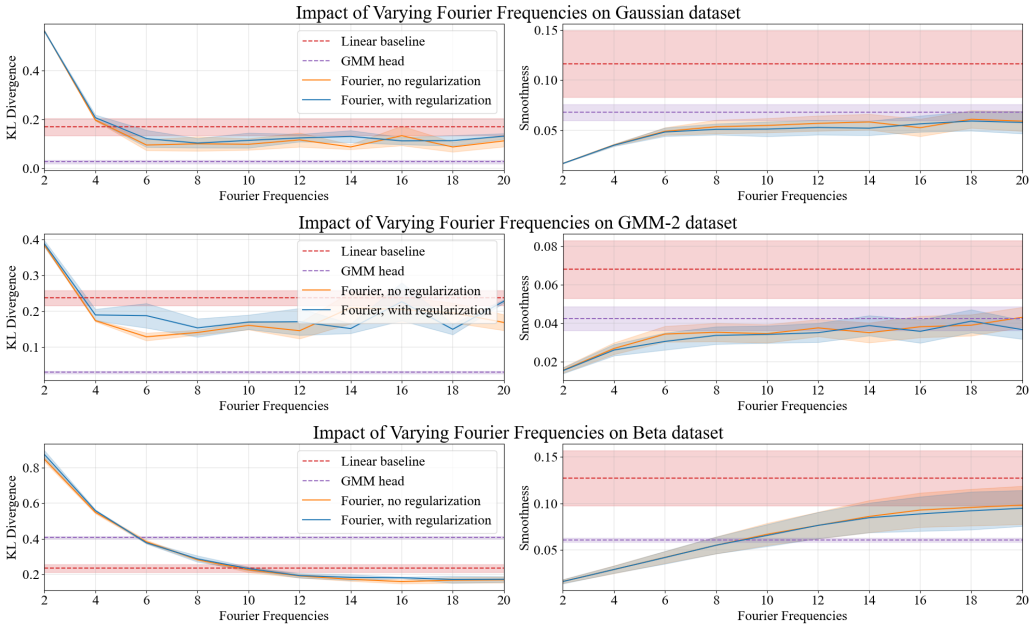

Figure 8: We study how the quantity of Fourier frequencies impacts KL divergence and smoothness for the toy example on each dataset. For both KL divergence and smoothness, lower is better. We observe that the Fourier models with and without regularization performed similarly to each other, and outperformed the linear baseline. We also note that the 50% error bars are larger for the linear baseline model; this indicates that the Fourier models (both with and without regularization) are in general more stable. This is in contrast to our large scale time series forecasting experiments, where we find that regularization helps; this is likely because those experiments use an order of magnitude more frequencies, and their conditional distributions are more complicated. While the GMM head has better KL divergence on the Gaussian and GMM-2 datasets, which is to be expected, the Fourier model (both with and without regularization) eventually has the best KL divergence on the Beta dataset, since it is non-Gaussian. Notice also how on each of the datasets, the smoothness degrades as frequency increases, in a fashion that follows the asymptotic from our Theorem 3.3.

Toy Example: MSE (↓)

		Classification Head		
Dataset	Pointwise Regression	Linear	GMM	Fourier
Gaussian	0.010 ± 0.001	0.013 ± 0.001	0.010 ± 0.001	0.012 ± 0.001
GMM-2	0.121 ± 0.004	0.126 ± 0.004	0.120 ± 0.004	0.123 ± 0.005
Beta	0.275 ± 0.009	0.276 ± 0.008	0.273 ± 0.009	0.275 ± 0.008

Table 6: We compare the MSE between the linear head, GMM head, and the Fourier head with 12 frequencies and no regularization, for every dataset in the toy example. We also include a Pointwise Regression model baseline, whose base architecture is same as the classification heads, except the last classification layer is replaced with a dense layer having output dimension 1. We train the Pointwise Regression model using MSE. For a given dataset, the MSE values across all of the models is roughly similar. This is because the pointwise regression model tends to regress to the mean, as does the expected value of each of the classification heads.

D Additional Decision Transformer Experiment Details

Following the original Decision Transformer implementation, we trained on 500k transitions observed by a DQN agent during training, for 5 epochs. We trained on the same model size as the original implementation (a GPT-1 model with approximately 2.012M parameters) which takes about 4 hours on a single GPU. We can see that in Figure 12 that the PMFs learned by the Fourier head

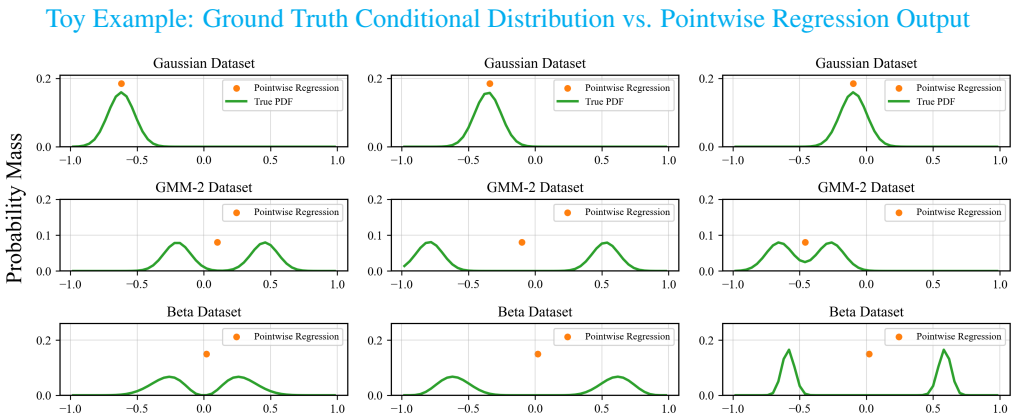

Figure 9: We present some examples of the ground truth conditional distribution versus the point predicted by the Pointwise Regression model. The regression model simply regresses to the mean of the conditional distribution. Accordingly, the regression model performs extremely well for the unimodal Gaussian dataset, and it performs poorly for the bimodal datasets GMM-2 and Beta.

are smoother. In Figure 13 we present results for more Atari games. In Figure 10, we present results from an ablation study of the model size. The results demonstrate that, across model sizes, the Decision Transformer with a Fourier head is better at learning high-quality next action distributions than the Decision Transformer with a linear head. And in Figure 11, we present results from an ablation study of the dataset size, which show that the Fourier head obtains larger returns than the Linear classification head across dataset sizes.

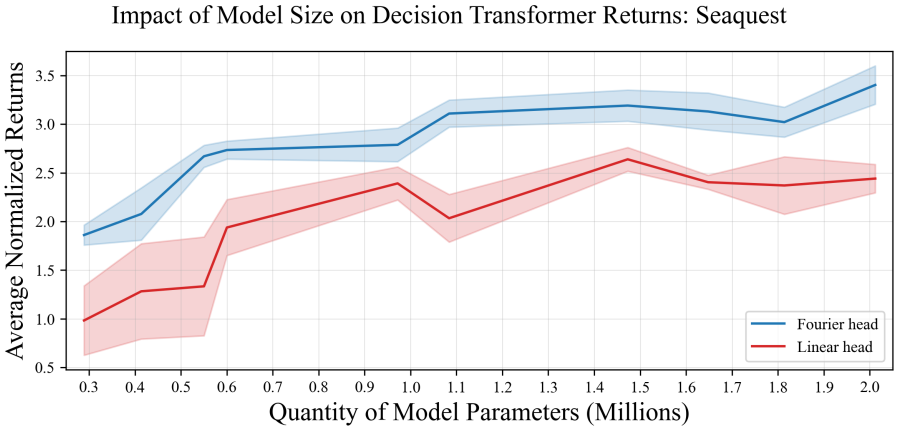

Figure 10: We present an ablation study on the effect of the model size on the relative performance of the Fourier head and the Linear head. The results demonstrate that, across model sizes, the Decision Transformer with a Fourier head is better at learning high-quality next action distributions than the Decision Transformer with a linear head.

E Additional Chronos experiment details

In Figure 14 we present a learned next-token PMF from a linear Chronos model, and a next-token PMF from a Chronos model which uses the linear head. The Fourier head is about 4x smoother. In Table 7 we present results from an ablation study on the choice of regularization, and binning strategy. We followed the original Chronos implementation, keeping all hyperparameters the same. In particular, we trained for 200k steps, on the same model size as the original implementation

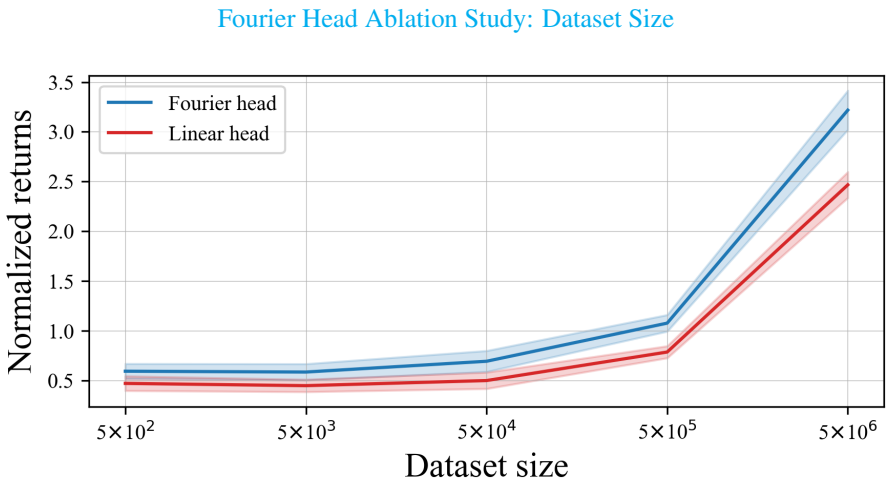

Figure 11: In this ablation study, we analyze whether dataset size has any effect on the relative performance of the Linear head and the Fourier head. Our results show that, across dataset sizes, the Decision Transformer agent with a Fourier head achieves larger returns than the linear head on the Seaquest game.

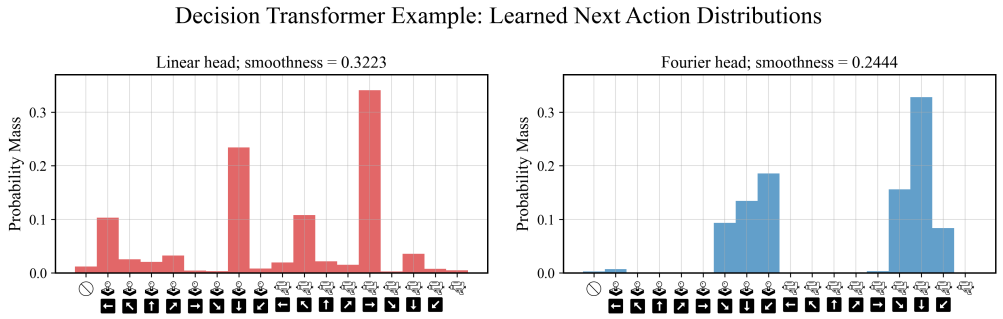

Figure 12: We present example next action distributions for a single step in the Decision Transformer. We can see that the Fourier agent with 14 frequenices produces "clumps" of actions that are semantically meaningful; for example, this agent either wants to move down right, down, or down left, or else the agent wants to shoot down right, down, or down left, presumably because they need to move in that general direction to rescue a diver in that direction but there are submarines in the way. In contrast, there is no indication that the linear agent has learned a cohestive strategy which relates moving and shooting.

(the T5 model with approximately 20M parameters) and this takes about 48 hours on 8 GPUs. See Table 8 for the datasets we used to train and evaluate Chronos.

Chronos Time Series Model	MASE ↓	WQL ↓	Smoothness ↓
Fourier-550	**0.852**	**0.749**	0.0203 ± 0.0176
Fourier-550 (no regularization)	0.861	0.753	0.0204 ± 0.0172
Fourier-550 (uniform precision binning)	0.873	0.747	0.0292 ± 0.0205

Table 7: We present large-scale ablations on Chronos time series forecasting. The best overall performing Fourier-550 model uses Fourier regularization and mixed precision binning, which are both techniques informed by Fourier analysis. We observe that both of these interventions improve the MASE, but have minimal effect on the WQL. We note that the choice of binning strategy doesn't affect the performance of the linear baseline.

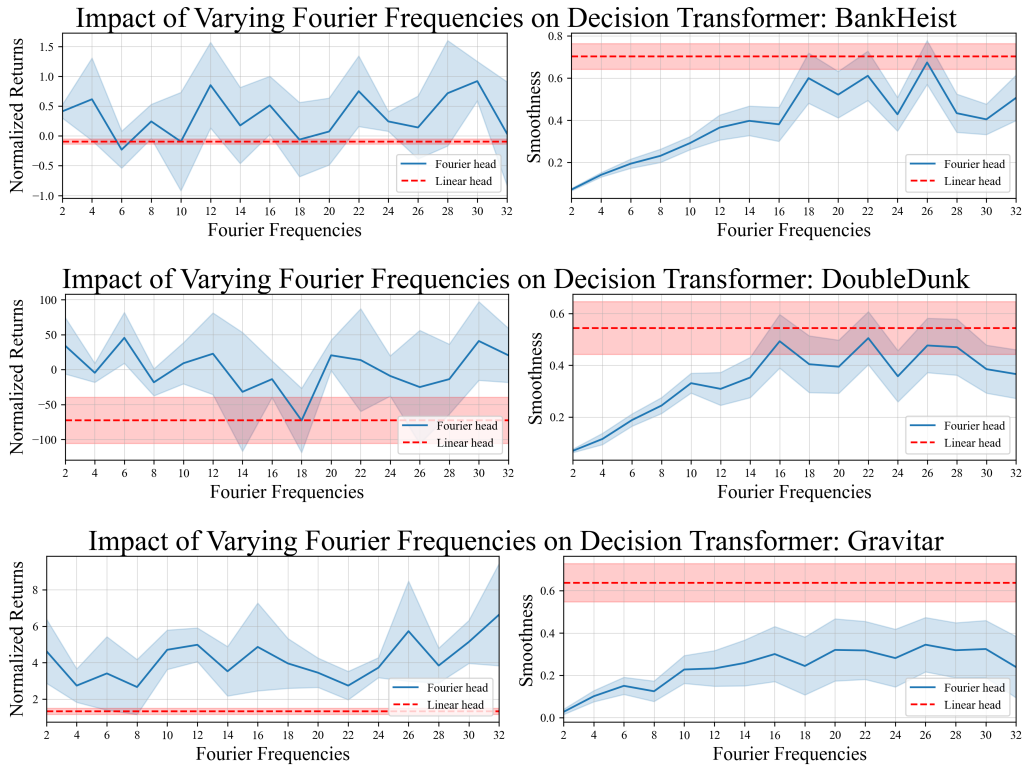

Figure 13: We present empirical results for how the quantity of Fourier frequencies impacts returns and smoothness for additional imitation learning games. For normalized returns, higher is better; for smoothness, lower is better. We can see that for the BankHeist, DoubleDunk, and Gravitar games, the Fourier agent consistently achieves higher normalized returns than the linear baseline agent, while still learning smoother next-action distributions.

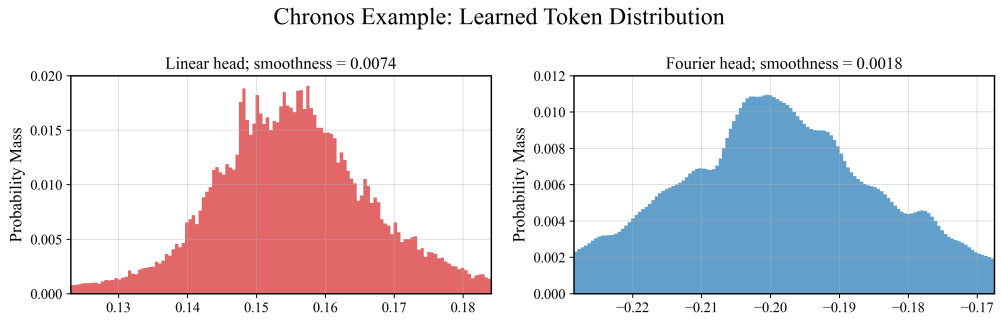

Figure 14: We present the next token value distribution for a single forecasted timestep on the Tourism Monthly dataset. We observe that the Fourier head's learned conditional distribution is smoother, fitting signal more robustly, whereas the linear head overfits to the noise, and is therefore more jagged. We note that the x-axis represents the bins in the latent space $[-1, 1]$; the x-axis values for the Fourier head are lower because the linear head uses uniform binning, and the Fourier head uses mixed precision binning.

Table 8: All datasets that are used for our time series forecasting experiments. We built our time series forecasting experiments on top of Chronos (Ansari et al., 2024), and this table is mostly copied from their paper. The datasets are partitioned according to how they are used for training and evaluation of models: *pretraining-only* data is only used for training; *evaluation* data is *not* used in training models, but only for evaluation (final H observations). All of our evaluation datasets came from the zero-shot evaluation set from Chronos.

Dataset	**Domain**	**Freq.**	**# Series**	**Series Length**			**Prediction**
				min	**avg**	**max**	**Length (H)**
Pretraining							
Brazilian Cities Temperature	nature	M	12	492	757	1320	-
Mexico City Bikes	transport	1H	494	780	78313	104449	-
Solar (5 Min.)	energy	5min	5166	105120	105120	105120	-
Solar (Hourly)	energy	1H	5166	8760	8760	8760	-
Spanish Energy and Weather	energy	1H	66	35064	35064	35064	-
Taxi (Hourly)	transport	1H	2428	734	739	744	-
USHCN	nature	1D	6090	5906	38653	59283	-
Weatherbench (Daily)	nature	1D	225280	14609	14609	14610	-
Weatherbench (Hourly)	nature	1H	225280	350633	350639	350640	-
Weatherbench (Weekly)	nature	1W	225280	2087	2087	2087	-
Wiki Daily (100k)	web	1D	100000	2741	2741	2741	-
Wind Farms (Daily)	energy	1D	337	71	354	366	-
Wind Farms (Hourly)	energy	1H	337	1715	8514	8784	-
Evaluation							
Australian Electricity	energy	30min	5	230736	231052	232272	48
CIF 2016	banking	1M	72	28	98	120	12
Car Parts	retail	1M	2674	51	51	51	12
Hospital	healthcare	1M	767	84	84	84	12
M1 (Monthly)	various	1M	617	48	90	150	18
M1 (Quarterly)	various	3M	203	18	48	114	8
M1 (Yearly)	various	1Y	181	15	24	58	6
M3 (Monthly)	various	1M	1428	66	117	144	18
M3 (Quarterly)	various	3M	756	24	48	72	8
M3 (Yearly)	various	1Y	645	20	28	47	6
M4 (Quarterly)	various	3M	24000	24	100	874	8
M4 (Yearly)	various	1Y	23000	19	37	841	6
M5	retail	1D	30490	124	1562	1969	28
NN5 (Daily)	finance	1D	111	791	791	791	56
NN5 (Weekly)	finance	1W	111	113	113	113	8
Tourism (Monthly)	various	1M	366	91	298	333	24
Tourism (Quarterly)	various	1Q	427	30	99	130	8
Tourism (Yearly)	various	1Y	518	11	24	47	4
Traffic	transport	1H	862	17544	17544	17544	24
Weather	nature	1D	3010	1332	14296	65981	30