Fourier Controller Networks for Real-Time Decision-Making in Embodied Learning

Hengkai Tan [1] **Songming Liu** [1] **Kai Ma** [1] **Chengyang Ying** [1] **Xingxing Zhang** [1] **Hang Su** [1] **Jun Zhu** [1]

Abstract

Transformer has shown promise in reinforcement learning to model time-varying features for obtaining generalized low-level robot policies on diverse robotics datasets in embodied learning. However, it still suffers from the issues of low data efficiency and high inference latency. In this paper, we propose to investigate the task from a new perspective of the frequency domain. We first observe that the energy density in the frequency domain of a robot's trajectory is mainly concentrated in the low-frequency part. Then, we present the Fourier Controller Network (FCNet), a new network that uses Short-Time Fourier Transform (STFT) to extract and encode time-varying features through frequency domain interpolation. In order to do real-time decision-making, we further adopt FFT and Sliding DFT methods in the model architecture to achieve parallel training and efficient recurrent inference. Extensive results in both simulated (e.g., D4RL) and real-world environments (e.g., robot locomotion) demonstrate FCNet's substantial efficiency and effectiveness over existing methods such as Transformer, e.g., FCNet outperforms Transformer on multi-environmental robotics datasets of all types of sizes (from 1.9M to 120M). The project page and code can be found https://thkkk.github.io/fcnet.

1. Introduction

Reinforcement Learning (RL) has been widely used in embodied learning scenarios (Lee et al., 2020a; Brohan et al., 2023), where agents interact with and learn from complex, dynamic physical environments. Representative examples include robotic arm manipulation (Gu et al., 2017; Kalashnikov et al., 2018; Lee et al., 2021), legged robot locomotion in various challenging terrains (Lee et al., 2020a; Rudin et al., 2022; Agarwal et al., 2023), dexterous manipulation (Gupta et al., 2016; Rajeswaran et al., 2017; Andrychowicz et al., 2020) and multi-task policies learning (Kumar et al., 2022; Kalashnikov et al., 2021; Yu et al., 2020). Most of these RL methods use MLPs or RNNs to learn robotic control policies. Recently, a growing body of efforts has been devoted to pre-training on large-scale robotic datasets to obtain more generalized policies that can apply to various tasks (Reed et al., 2022; Jiang et al., 2022; Brohan et al., 2022; 2023; Chebotar et al., 2023; Fu et al., 2024). Among various architectures, Transformer has been widely adopted because of its ability to model time-

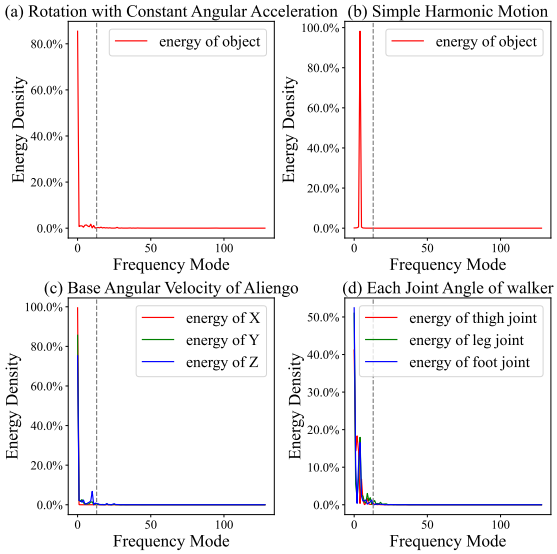

Figure 1. Energy density (normalized to $0\% \sim 100\%$ for all modes) in the frequency domain of different physical quantities across various motions. We choose $n = 256$ as the context length in the time domain. The time domain data represented in each of the four plots include: (a) Rotational motion with constant angular acceleration. (b) Simple harmonic motion. (c) Body angular velocity during a quadrupedal robot's run. (d) Joint angle of walker2d-expert-v2 in D4RL dataset.

[1]Department of Computer Science and Technology, Institute for AI, Tsinghua-Bosch Joint ML Center, THBI Lab, BNRist Center, Tsinghua University, Beijing, 100084, China. Correspondence to: Hang Su <suhangss@tsinghua.edu.cn>, Jun Zhu <dcszj@tsinghua.edu.cn>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Table 1. Comparision of Different Architectures. The energy density distribution of states in the frequency domain is often concentrated in the lowest m modes, where $m \ll n$. The term b in the RetNet line represents the chunk size, which is set to 512 in the paper(Sun et al., 2023).

	INFERENCE COST	TRAINING COST	MEMORY	PARALLEL TRAINING	PERFORMANCE
MLP (HISTORY)	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✔	✘
RNN	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✘	✔
TRANSFORMER	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	✔	✔
RETNET	$\mathcal{O}(1)$	$\mathcal{O}(nb)$	$\mathcal{O}(n)$	✔	✔
FCNET	$\mathcal{O}(m)$	$\mathcal{O}(mn(\log n + m))$	$\mathcal{O}(nm)$	✔	✔✔

varying features and handle large-scale sequential robotics data. In particular, DT (Chen et al., 2021) is a pioneering and simple method that uses Transformer to directly model the trajectory sequence in RL auto-regressively, compared to traditional offline RL algorithms (Kumar et al., 2020) which use many mathematical tricks to learn from offline data. The simplicity of the DT structure inspires follow-up work to use the attention mechanism to model the temporal features of diverse robotic trajectories. For instance, Gato (Reed et al., 2022) serializes the Atari, Gym, chat, and image datasets (1.5T tokens) all into flat sequences for input to a Transformer, by utilizing its high capacity. RT-1 (Brohan et al., 2022) also utilizes Transformer for its multi-task integration capability for imitation learning on 130K expert trajectories of robotic control.

One common feature of the existing works on Transformer is that they mainly focus on modeling the time-varying features of robotic trajectories in *time domain* by drawing a direct analogy with that in modeling natural language sentences. We argue that this is insufficient for embodied learning, and instead we explore its unique features from the new angle of *frequency domain*. Specifically, we take a close examination on embodied learning in the frequency domain and observe that the energy density distribution of a robot's state sequence is mainly concentrated in the low-frequency part, as shown in Fig. 1. This is due to the inherent continuity and smoothness in natural physical phenomena and robot motor motion (Kashiri et al., 2018). However, existing works on Transformer and its variants directly model trajectories in the time domain, resulting in various issues. First, the data efficiency of existing Transformer architectures is low. They typically rely on large-scale data for good performance, while real-world physical data collection can be costly and time-consuming (Brohan et al., 2022; Padalkar et al., 2023; Fu et al., 2024). For instance, the dataset in RT-1 (Brohan et al., 2022) and subsequent work (Brohan et al., 2023) requires 17 months to gather 130k episodes, out of reach for most research labs. Moreover, Transformer-based models cannot intrinsically reduce the computation complexity (e.g., $\mathcal{O}(n)$ for inference) and often struggle with the real-time processing requirements (i.e., low inference latency). For example, the inference frequency of such models is often around $3Hz$ (Brohan et al., 2022). However, a typical frequency of real-world legged robot control is above $50Hz$ (Lee et al., 2020a). Therefore, it is crucial to reduce the model's time complexity through algorithmic optimization, especially considering the constraints of robot hardware.

To address the aforementioned issues, we propose a new architecture of Fourier Controller Network (FCNet) based on the key observation in the frequency domain. FCNet grounds the inductive bias in robotic control inspired by the Fourier transform. We conceptualize low-level continuous control as a sequential decision-making problem. Our neural model is adept at predicting subsequent actions by analyzing a historical window of state data, as depicted in Fig. 3. Guided by the observation in the frequency domain and the inductive reasoning that differential dynamics are simplified in the frequency domain (as suggested by (Trefethen, 1996)), FCNet introduces a causal spectral convolution (CSC) block. It employs the Short-Time Fourier Transform (STFT) and linear transform for efficient feature extraction in the *frequency domain*, distinct from Transformer and other prevalent architectures. As shown in Fig. 2, we focus on the m lowest modes, with m strategically selected to be $\ll n$, where n is the length of the state window. Consequently, the high-frequency part in the frequency domain is filtered, allowing us to focus solely on these m lowest modes. The CSC makes efficient training and inference possible, and has also been shown to have good performance in experiments.

Furthermore, to achieve efficient parallel training and inference, which necessitates causality in the model's sequential outputs (dependent only on previous inputs) and the rapid generation of each output token for real-time response, we introduce parallel training based on Fast Fourier transform (FFT), and recurrent inference based on sliding discrete Fourier transform (Sliding DFT) in FCNet. As outlined in Table 1, the FCNet demonstrates a computational complexity of $\mathcal{O}(mn \log n + m^2 n)$ in parallel training setups, and $\mathcal{O}(m)$ for single-step inference. This efficiency marks a significant speed advantage over traditional Transformer models, enabling handling the complexities of real-time

continuous control in dynamic environments.

We extensively evaluate FCNet in various settings. First, in the classic offline RL environments such as D4RL (Fu et al., 2020), FCNet outperforms Transformer, MLP-based methods as well as RetNet (Sun et al., 2023) which is a representative State Space Model (SSMs) and potential successor to Transformer. This shows strong feature extraction capabilities of FCNet from series of state data. Second, we evaluate Transformer and FCNet on a multi-environment robotics dataset. The results show that FCNet significantly outperforms Transformer with limited data, and also has lower inference latency and good scalability. We also verify the robustness of FCNet in real-world robots. Finally, we test the inference latency of the Transformer (with KV cache) and FCNet under different hyperparameter settings related to model structure. The results show that the upward curve of the inference latency of FCNet is significantly slower than that of Transformer as the context length, number of layers, and hidden size are improved. This demonstrates the efficiency of the inference of FCNet.

2. Background

2.1. Preliminary

Real-world robotics control is often formulated as a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^{d_s}$ and $\mathcal{A} \subseteq \mathbb{R}^{d_a}$ denote the state space and action space, respectively; $\mathcal{T}$ is the transition probability; r is the reward function; and $\gamma \in (0, 1)$ is a discount factor. The goal of robot control is to get a parameterized policy π to take actions for interacting with the environment. Even in the fully-observed setting, the consideration of multi-task will introduce partial-observability (Lee et al., 2020b; Ghosh et al., 2021; Ying et al., 2023), which demands encoding enough history information in the model. Following previous work (Chen et al., 2021; Agarwal et al., 2023), we consider policies that map historical trajectories into action spaces $a_t = \pi(s_{\leq t}, a_{<t})$ and maximize the accumulated expected rewards, i.e.,

$$\mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{i=0}^{\infty}\gamma^i r(s_i, a_i)\right], \tag{1}$$

where τ represents the trajectory of states and actions, defined as $\tau = \{s_0, a_0, s_1, a_1, \dots\}$. As acquiring the robotics dataset in the real world can be expensive, a well-used alternative method is to utilize expert datasets $D = \{\tau | \tau \sim \pi_\beta, \tau = (s_0, a_0, \dots, s_T, a_T)\}$ for imitation learning, or offline datasets $D = \{\tau | \tau \sim \pi_\beta, \tau = (s_0, a_0, r_0, \dots, s_T, a_T, r_T)\}$ for offline RL. Here π_β is an unknown behavior policy. We aim to train a policy π_θ using the dataset, with the goal of maximizing the expected return $\mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$.

For expert datasets that do not include reward information, our objective is to ensure that the behavior of the trained policy π_θ closely aligns with the expert behavior policy π_β. Thus, the optimization objective can be summarized as

$$\mathcal{L}(\theta) = \frac{1}{T|D|}\sum_{\tau\in D}\sum_{i=0}^{T} d(a_i, \hat{a}_i), \tag{2}$$

where $\hat{a}_i$ is the action derived from π_θ, a_i is the action in τ, and d is the distance function. Typically, when π_θ takes into account historical trajectories, it employs a context window of length n representing the extent of historical dependencies, i.e., $a_t = \pi_\theta(s_{t-n+1:t}, a_{t-n+1:t-1})$. In practice, we can choose d in Eq. (2) as the mean-squared error (MSE) for deterministic actions, and the Kullback-Leibler (KL) Divergence for stochastic actions.

For offline datasets that include reward information, one method is to incorporate the return-to-go, i.e., the suffix of the reward sequence, into the policy's sequence input. This often results in a policy with higher performance than the original behavior policy in the dataset, as the results of DT (Chen et al., 2021) show.

Time Complexity of Transformer In addition to the DT just mentioned, there is a lot of work on pre-training robotics datasets using Transformer (Reed et al., 2022; Jiang et al., 2022; Brohan et al., 2022; 2023; Chebotar et al., 2023), and they all use trajectories analogous to language as input to Transformer. The training time complexity of the Transformer model is $\mathcal{O}(n^2 d_h + n d_h^2)$, where n is the length of both the input and output sequences and d_h is the hidden dimension. The inference time complexity of the Transformer with KV cache is $\mathcal{O}(n d_h + d_h^2)$. In scenarios where d_h is relatively stable, we primarily consider the impact of n on time complexity. Consequently, the training and inference time complexities of the Transformer can be simplified to $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$, respectively.

2.2. Motivation

As previously mentioned, existing architectures such as Transformer mainly focus on modeling the time-varying features of robotic trajectories in the time domain, through mechanisms such as attention. However, they do not take into account the special properties embedded in robotic control in the frequency domain. Below, we start with simple physical motions to illustrate the significance of modeling temporal features in the frequency domain.

Consider the motion of a mass along the x-axis as a simple example. We assume knowledge of the mass's state at n distinct moments. The complexity of describing its motion depends on the nature of the movement. For a particle at rest, only the initial position x_0 is needed. In uniform linear motion, both the initial position x_0 and velocity v are required.

For uniformly accelerated linear motion, three quantities are essential: initial position x_0, initial velocity v, and acceleration a, resulting in the equation $x = x_0 + v_0 t + at^2/2$. Similarly, simple harmonic motion, which is common in nature, requires amplitude A, angular frequency ω, and phase ϕ for the equation $x = A\sin(\omega t + \phi)$. In fact, real motor movements are typically smooth to minimize energy losses and to facilitate their application in robotic arms (Kashiri et al., 2018). For instance, uniformly accelerated motion is employed in the trapezoidal acceleration and deceleration curves of motors, and simple harmonic motion (or sinusoidal function) is used in the S-curves of motors, both common in motor motion.

This insight leads us to recognize that natural object motion often adheres to physical principles, like energy conservation, and optimal trajectories are typically smooth, possibly with minor irregularities. Viewing this from a data-driven standpoint, if we have details on a mass's position, velocity, and other states at n moments, the quantity of information embedded in this does not need to be described using $\mathcal{O}(n)$ level parameters. In fact, a significantly smaller set of parameters might be adequate for accurate representation. Similarly, complex rigid structures such as robots can also be approximated in this manner, given that robots are composed of rigid components like joints and links. Typically, a robot's state includes various parameters such as motor position, motor velocity, and body angular velocity, among others. The flexible motion of the robot can be characterized by a limited set of physical quantities. However, the measured state variables often contain a significant amount of high-frequency noise, due to sensor uncertainties. Therefore, we propose to transform the state representation of the object on n moments into the frequency domain, rather than modeling the trajectories in the time domain. By filtering out high-frequency components and retaining low-frequency ones, we can more accurately capture the essential motion information of the object.

Indeed, as demonstrated in Fig. 1, both the common mechanical motions and the motions of real robots show a consistent pattern in the frequency domain, namely, their energy density distributions predominantly reside in a low-frequency region. This observation aligns with our hypothesis. Additionally, our empirical analyses reveal that in various scenarios, the number m of significant low-frequency modes for robot states is much smaller than n. We empirically set the value $m \ll n$ in the experiment. Consequently, this allows us to model state sequences over time with reduced complexity compared to the Transformer, thanks to frequency domain interpolation.

3. Fourier Controllers

We now formally introduce Fourier Controller Network (FCNet) for efficient robotic control. FCNet incorporates an inductive bias about frequency domain features and aims to minimize the time complexity of both training and inference.

3.1. Overall Architecture

As shown in Fig. 2, FCNet comprises a position-wise encoder $P\colon \mathbb{R}^{d_s} \to \mathbb{R}^{d_h}$, L identical stacked Fourier layers, and a position-wise decoder $Q\colon \mathbb{R}^{d_h} \to \mathbb{R}^{d_a}$, where both P and Q are parameterized by feed-forward networks (FFNs) and d_h denotes the hidden dimension. Each Fourier layer consists of two primary modules: a causal spectral convolution (CSC) block and a position-wise FFN block. Given an input window of historical states, denoted as $\boldsymbol{X}^0 := [\boldsymbol{x}_0^0, \ldots, \boldsymbol{x}_{n-1}^0]^\top \in \mathbb{R}^{n\times d_s}$ where $\boldsymbol{x}_i^0 := \boldsymbol{s}_i$ for $\forall 0 \le i < n$, our model outputs the predicted actions for corresponding time steps: $\boldsymbol{X}^{L+2} := [\boldsymbol{x}_0^{L+2}, \ldots, \boldsymbol{x}_{n-1}^{L+2}]^\top \in \mathbb{R}^{n\times d_a}$ with $\boldsymbol{x}_i^{L+2} := (\hat{\boldsymbol{a}}_i)$ for $\forall 0 \le i < n$. The computation process of the FCNet is:

$$
\begin{aligned}
\boldsymbol{X}^1 \in \mathbb{R}^{n\times d_h} &= P(\boldsymbol{X}^0), \\
\boldsymbol{Y}^l &= \mathrm{gelu}(\mathrm{CSC}(\mathrm{LN}(\boldsymbol{X}^l))) + \boldsymbol{X}^l, \\
\boldsymbol{X}^{l+1} &= \mathrm{FFN}(\mathrm{LN}(\boldsymbol{Y}^l)) + \boldsymbol{Y}^l, \\
\boldsymbol{X}^{L+2} \in \mathbb{R}^{n\times d_a} &= Q(\boldsymbol{X}^L),
\end{aligned}
\tag{3}
$$

where $l \in \{1, \ldots, L\}$, $\mathrm{LN}(\cdot)$ is the LayerNorm (Ba et al., 2016), and P is a single-layer FFN, while $\mathrm{FFN}(\cdot)$ and Q are two-layer FFNs. $\mathrm{gelu}(\cdot)$ refers to the Gaussian Error Linear Unit (GELU) activation (Hendrycks & Gimpel, 2016). Of note, the parameters for each layer's $\mathrm{FFN}(\cdot)$ are not shared.

In order to leverage good scalability of sequence modeling (Brown et al., 2020), we can supervise the action of the policy output X^{L+2} in a supervised learning manner(Chen et al., 2021). Based on Eq. (2), the training objective can be formulated as:

$$
\mathcal{L}(\theta) = \frac{1}{|D|}\sum_{\tau\in D} d(\boldsymbol{a}_{0:n}, X^{L+2}). \tag{4}
$$

3.2. Causal Spectral Convolution

In order to model time-varying features in the frequency domain, in this subsection, we delve into the mechanism of causal spectral convolution (CSC) block, which is a trainable block and utilizes the Short-Time Fourier Transform (STFT) to efficiently extract and encode time-varying features. It is nontrivial to encode because it is necessary to ensure that the output is causal. Given an input sequence $\boldsymbol{X} := [\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}]^\top \in \mathbb{R}^{n\times d_h}$, the output of the CSC

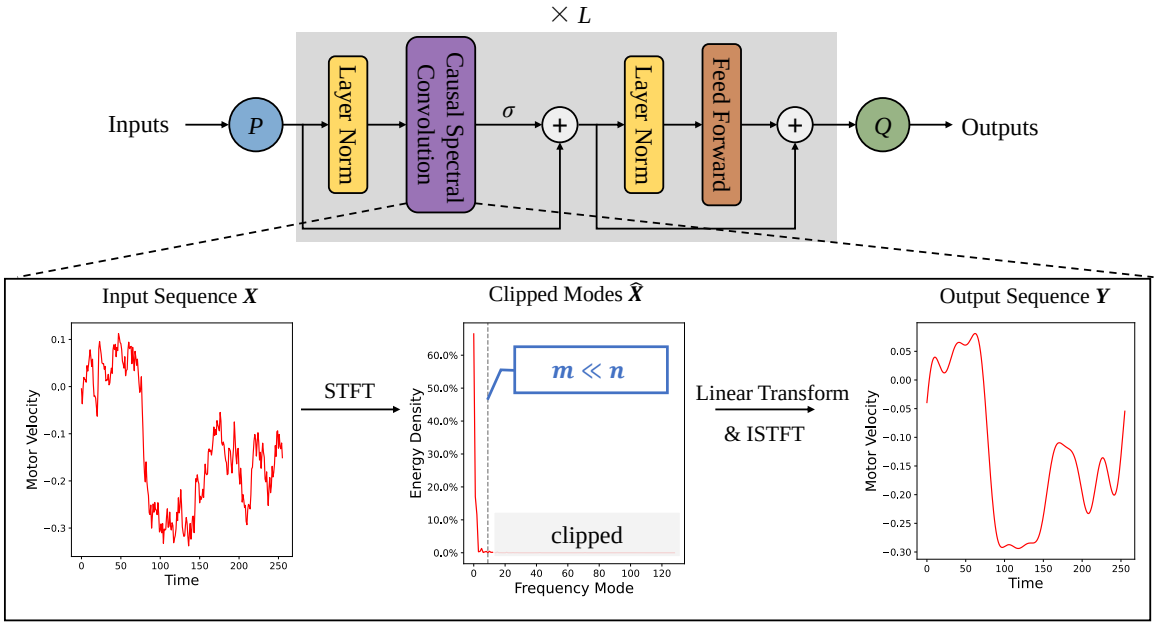

Figure 2. The overall model architecture of FCNet. In order to ensure efficient training and inference, we cannot apply Fourier transform to all history trajectories, but instead apply STFT to a window of historical data of length n to filter high-frequency part in the frequency domain, and then apply linear transform and inverse STFT back to the time domain. That is, the CSC block comes in the frequency domain to model temporal features, while the FFN is used to model features on the hidden dimension. P is point-wise encoder and Q is point-wise decoder mentioned in Sec. 3.1. σ is the activation function.

$\boldsymbol{y}_{n-1} \in \mathbb{R}^{d_h}$ can be formalized as:

$$\begin{aligned} \hat{\boldsymbol{x}}_k &= \mathcal{F}(\boldsymbol{X})(k), & \hat{\boldsymbol{x}}_k &\in \mathbb{C}^{d_h}, 0 \le k < m, \\ \hat{\boldsymbol{Y}} &= \boldsymbol{W}\hat{\boldsymbol{X}}, & \hat{\boldsymbol{X}} &= [\hat{\boldsymbol{x}}_0, \dots, \hat{\boldsymbol{x}}_{m-1}]^\top, \\ & & \hat{\boldsymbol{Y}} &= [\hat{\boldsymbol{y}}_0, \dots, \hat{\boldsymbol{y}}_{m-1}]^\top, \\ \boldsymbol{y}_{n-1} &= \mathcal{F}^{-1}(\hat{\boldsymbol{Z}})(n-1), & \hat{\boldsymbol{Z}} &= [\hat{\boldsymbol{z}}_0, \dots, \hat{\boldsymbol{z}}_{n-1}]^\top, \end{aligned} \tag{5}$$

where $\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}} \in \mathbb{C}^{m \times d_h}$ and $\hat{\boldsymbol{Z}} \in \mathbb{C}^{n \times d_h}$. $1 \le m \le 1 + \lfloor n/2 \rfloor$ is a hyper-parameter denoting the number of modes preserved in the frequency domain, $\boldsymbol{W} \in \mathbb{C}^{m \times m}$ is the weight matrix, and $\boldsymbol{Z}$ is obtained by extending $\boldsymbol{Y}$ according to the conjugate symmetry of real series:

$$\hat{\boldsymbol{z}}_k = \begin{cases} \hat{\boldsymbol{y}}_k, & 0 \le k < m, \\ \overline{\hat{\boldsymbol{y}}_{n-k}}, & n-m+1 \le k < n, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Here, $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Discrete Fourier Transform (DFT) and its inverse (IDFT):

$$\begin{aligned} \mathcal{F}(\boldsymbol{X})(k) &= \sum_{i=0}^{n-1} \boldsymbol{x}_i \exp\left(-\frac{j2\pi}{n} ki\right), \\ \mathcal{F}^{-1}(\hat{\boldsymbol{Z}})(i) &= \frac{1}{n} \sum_{k=0}^{n-1} \hat{\boldsymbol{z}}_k \exp\left(\frac{j2\pi}{n} ki\right), \end{aligned} \tag{7}$$

where $j = \sqrt{-1}$ is the imaginary unit.

In Eq. (5), we start by encoding the input sequence via the DFT, where only low-frequency components are preserved while high-frequency noises are filtered by using only m DFTs, according to the smoothness of the physical states. Then, a linear transform is applied in the frequency domain, followed by an IDFT back to the physical domain. It is noted that the output $\boldsymbol{y}_{n-1}$ only depends on the history $\boldsymbol{x}_i, i \le n-1$, ensuring the causality of the prediction.

3.3. Parallel Training

Parallelization significantly speeds up model training by distributing tasks across multiple processors, especially for large datasets. Here we fully parallelize the FCNet training to improve efficiency. The FFN can be easily parallelized by matrix partitioning, and thus we mainly discuss the parallelization of the CSC block (see Appendix A for its schematic diagram). The CSC block involves operations that are inherently sequential, and it is nontrivial to parallelize due to dependencies between operations.

Representation of computation process. Adopting this approach, for a given input sequence $\boldsymbol{X} := [\boldsymbol{x}_0, \dots, \boldsymbol{x}_{n-1}]^\top \in \mathbb{R}^{n \times d_h}$, the CSC block outputs in parallel $\boldsymbol{Y} := [\boldsymbol{y}_0, \dots, \boldsymbol{y}_{n-1}]^\top \in \mathbb{R}^{n \times d_h}$, where $\boldsymbol{y}_k$ is solely

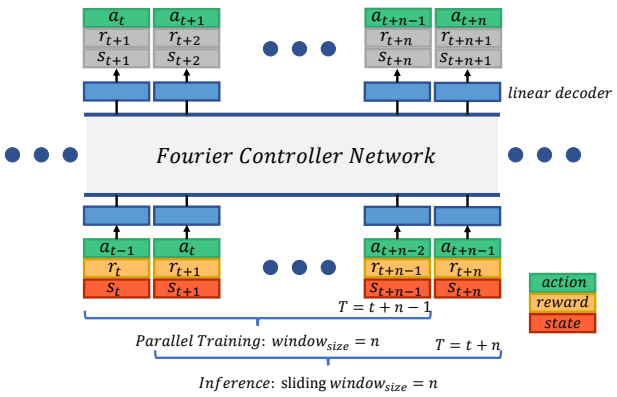

Figure 3. Consider MDP as a sequence modeling problem. Especially in a continuous state space like embodied learning, it makes sense to perform Fourier-based modeling and training in the frequency domain of time-domain sequences within a window $[t, t+n-1]$. In the inference phase $T = t+n-1 \to T = t+n$, the sliding window in Sec. 3.4 is utilized for efficient inference.

dependent on $\boldsymbol{x}_i$ for $i \leq k$. Our goal is to compute each $\boldsymbol{y}_t$ $(0 \leq t < n)$ quickly and in parallel. Consider that we are computing the output $\boldsymbol{y}_t$ $(0 \leq t < n)$ at moment t. $\boldsymbol{y}_t$ can only depend on $\boldsymbol{x}_i$ $(i \leq t)$ for causality. Therefore, the $\hat{\boldsymbol{X}}$ in the frequency domain for each $\boldsymbol{y}_t$ is distinct. We use $\hat{\boldsymbol{X}}^{(t)}$ to denote the $\hat{\boldsymbol{X}}$ corresponding to $\boldsymbol{y}_t$. From Eq. (5) and Eq. (7), we have:

$$\hat{\boldsymbol{x}}_k^{(t)} = \sum_{i=t-n+1}^{t} \boldsymbol{x}_i \exp\left(-\frac{j2\pi}{n} ki\right), \tag{8}$$

To simplify the expression, we let $u_k = \exp\left(\frac{j2\pi}{n}k\right)$ and $\boldsymbol{u} = [u_0, \ldots, u_{m-1}]^\top \in \mathbb{C}^m$. Each $\hat{\boldsymbol{x}}_k^{(t)}$ can be expressed in terms of its previous $\hat{\boldsymbol{x}}_k$:

$$\begin{aligned} \hat{\boldsymbol{x}}_k^{(t)} &= u_k(\hat{\boldsymbol{x}}_k^{(t-1)} + x_t - x_{t-n}) = u_k(\hat{\boldsymbol{x}}_k^{(t-1)} + f_t), \\ \Rightarrow \hat{\boldsymbol{X}}^{(t)} &= \boldsymbol{u} \odot (\hat{\boldsymbol{X}}^{(t-1)} + \mathbf{1}_m f_t^\top), \end{aligned} \tag{9}$$

where $f_t = x_t - x_{t-n} \in \mathbb{R}^{d_h}$ can be computed in parallel, $\hat{\boldsymbol{X}}^{(t)} = [\hat{\boldsymbol{x}}_0^{(t)}, \ldots, \hat{\boldsymbol{x}}_{m-1}^{(t)}]^\top \in \mathbb{C}^{m \times d_h}$ and $\odot$ means Hadamard product or element-wise product. The above equation can be computed recursively for each $\hat{\boldsymbol{X}}^{(t)}$.

After determining $\hat{\boldsymbol{X}}^{(t)}$ for $0 \leq t < n$, we can obtain $\hat{\boldsymbol{Y}}^{(t)}, \hat{\boldsymbol{Z}}^{(t)}, y_t, 0 \leq t < n$ using Eq. (5) and Eq. (6).

Parallel training. The representation in Eq. (8) has time complexity $\mathcal{O}(md_h n^2)$ in parallel and Eq. (9) has a **serial** $\mathcal{O}(md_h n)$ time complexity. Here, we seek to find a training process that can be parallelized with low time complexity. Let us dissect the recursive formula given in Eq. (9) for $\hat{\boldsymbol{X}}^{(t)}$ where $0 \leq t < n$. We have

$$\hat{\boldsymbol{X}}^{(t)} = \boldsymbol{u}^{\circ t} \odot \hat{\boldsymbol{X}}^{(0)} + \sum_{i=0}^{t-1} \boldsymbol{u}^{\circ(i+1)} f_{t-i}^\top, \tag{10}$$

where $\boldsymbol{u}^{\circ t}$ means $\underbrace{\boldsymbol{u} \odot \boldsymbol{u} \odot \cdots \odot \boldsymbol{u}}_{t \text{ elements}}$. Let

$$\begin{aligned} \boldsymbol{A}_i &= [(\boldsymbol{u}^{\circ(i+1)})_{\times d_h}] \in \mathbb{C}^{m \times d_h}, 0 \leq i < n-1, \\ \boldsymbol{B}_i &= [(f_i^\top)_{\times m}] \in \mathbb{C}^{m \times d_h}, 1 \leq i < n, \end{aligned} \tag{11}$$

and for special case $\boldsymbol{A}_{n-1} = \boldsymbol{A}_{n-2}$, $\boldsymbol{B}_0 = \hat{X}^{(0)} \in \mathbb{C}^{m \times d_h}$. From Eq. (10) we get:

$$\hat{\boldsymbol{X}}^{(t)} = \sum_{i=0}^{t} A_i \odot B_{t-i}, \quad 0 \leq t < n, \tag{12}$$

which is a typical form of FFT convolution $\{\hat{\boldsymbol{X}}^{(t)}\} = \{A_i\} * \{B_i\}$ where $*$ represents linear convolution. We can compute all $\hat{\boldsymbol{X}}^{(t)}, 0 \leq t < n$ by FFT with time complexity $\mathcal{O}(md_h n \log n)$ **in parallel**. The fast parallel computation of the IDFT is homogeneous. Thus the training time complexity of the whole CSC block is $\mathcal{O}(md_h n \log n + m^2 d_h n)$ because of the computation of $\hat{\boldsymbol{Y}}^{(t)} = \boldsymbol{W}\hat{\boldsymbol{X}}^{(t)}, 0 \leq t < n$. Therefore, the total training time complexity is $\mathcal{O}(md_h n \log n + m^2 d_h n + nd_h^2)$ or reduced to $\mathcal{O}(mn \log n + m^2 n)$.

Furthermore, FCNet exhibits a memory complexity of $\mathcal{O}(mnd_h)$ (can be reduced to $\mathcal{O}(mn)$) during training due to the storage for $\hat{x}_k^{(t)}$ (where $0 \leq t < n$ and $0 \leq k < m$).

3.4. Reprensentation of Recurrent Inference

Efficient real-time inference is important for the deployment of robots (Sandha et al., 2021). It is difficult to utilize time-domain information to achieve efficient inference. Therefore we utilize the cache of frequency domain information for fast inference. During inference, supposing that we cache the results computed in the last time step, the CSC can recurrently compute the current output $\boldsymbol{y}_{n-1}$ given a newly coming $\boldsymbol{x}_{n-1}$ using Sliding DFT (as well as Eq. (9)):

$$\hat{\boldsymbol{x}}_k = \exp\left(\frac{j2\pi}{n}k\right)(\hat{\boldsymbol{x}}'_k - \boldsymbol{x}_{-1} + \boldsymbol{x}_{n-1}), \tag{13}$$

where $0 \leq k < m$. After obtaining $\hat{\boldsymbol{x}}_0, \ldots, \hat{\boldsymbol{x}}_{m-1}$, we compute $\boldsymbol{y}_{n-1}$ as in Eq. (5). Here, $\hat{\boldsymbol{x}}'_k$ is the result cached in the last time step, which is equal to $\mathcal{F}(\boldsymbol{X}')(k), \boldsymbol{X}' := [\boldsymbol{x}_{-1}, \ldots, \boldsymbol{x}_{n-2}]^\top$. Because we cached the $\hat{\boldsymbol{x}}'_k$ in the last time step, the time complexity of Eq. (13) is $\mathcal{O}(md_h)$ instead of $\mathcal{O}(md_h n)$.

After computing $\hat{\boldsymbol{X}} = [\hat{\boldsymbol{x}}_0, \ldots, \hat{\boldsymbol{x}}_{m-1}]^\top$, the $\hat{\boldsymbol{Y}} = \boldsymbol{W}\hat{\boldsymbol{X}}$ and $\mathcal{F}^{-1}(\hat{\boldsymbol{Z}})(n-1)$ part in Eq. (5) can be preprocessed by

calculating the result of multiplying $\boldsymbol{W}$ by the corresponding vector of $\mathcal{F}^{-1}(\hat{\boldsymbol{Z}})(n-1)$ in advance. Therefore, the total inference time complexity is $\mathcal{O}(md_h + d_h^2)$ (where $\mathcal{O}(d_h^2)$ corresponds to FFN), further reduced to $\mathcal{O}(m)$. This mechanism is advantageous for rapid inference since it allows for the generation of actions sequentially during the evaluation of an embodied agent.

4. Experiments

In this section, we conduct extensive experiments to evaluate the performance and efficiency of FCNet. Initially, FCNet's effectiveness in fine continuous control is compared against Transformer and RetNet (a current SSM architecture) using the D4RL offline RL benchmark. Next, FCNet and Transformer are assessed on a multi-environment legged robot locomotion dataset via imitation learning, highlighting FCNet's adaptability across various environments, particularly with scarce data. Finally, we investigate the inference efficiency of FCNet compared to Transformer under diverse model structure hyperparameter settings.

4.1. Evaluation in Offline RL

To evaluate FCNet in the D4RL (Fu et al., 2020) offline RL tasks, we employ a broad range of baselines featuring various network architectures such as MLP, Transformer, and RetNet. We start with classic MLP-based methods like Behavior Cloning (BC), CQL (Kumar et al., 2020), BEAR (Kumar et al., 2019), BRAC-v (Wu et al., 2019), and AWR (Peng et al., 2019). Additionally, we incorporate the transformer-based method DT (Chen et al., 2021) and the RetNet-based method DT-RetNet, which replaces the Transformer architecture in DT into RetNet. In D4RL evaluations, FCNet follows some design strategies of DT, i.e., concatenating action, return-to-go (normalized, with 1.0 indicating an expert policy), and state into a single token $(a^\top, R^\top, s^\top)^\top$, and then predicts the next token autoregressively. As shown in Table 2, our FCNet is competitive with DT, and significantly surpasses other methods. Particularly, FCNet outperforms DT-RetNet, suggesting that FCNet's inductive bias for robotic control has more powerful feature extraction capability and performance compared to SSMs.

In addition to locomotion tasks, we expand our investigation to encompass manipulation tasks, specifically the Adroit suite in the D4RL dataset, which comprises various robotic arm manipulation challenges and are not harmonic, distinct from locomotion tasks. Our results, as detailed in the Table 3, demonstrate the performance of FCNet in comparison with DT, BC, and CQL. FCNet beats DT on 9 of the 12 manipulation tasks in D4RL-Adroit. These results highlight FCNet's robust performance across a range of tasks, outperforming DT, BC, and CQL in the domain of manipulation. This demonstrates the broad applicability of our approach beyond locomotion tasks.

4.2. Evaluation in Legged Robot Locomotion

To test FCNet on the more challenging multi-environment legged robot locomotion, we introduce a new, substantially larger dataset than that in D4RL. Compared with Transformer-based methods in this dataset, our focus is to showcase FCNet's strong fitting ability even with limited data, and further real-time inference on a real-world robot.

Multi-Environment Legged Robotics Dataset. In order to evaluate the performance of the policy in modeling historical information sequences in a complex multi-task environment, which is rarely available in previous d4rl datasets, We develop a multi-environment legged robot locomotion dataset, which covers various skills (e.g., standing, rushing, crawling, squeezing, tilting, and running), and different terrains (e.g., rough terrain, stairs, slopes, and obstacles), following Rudin et al. (2022). The dataset is collected in Isaacgym (Makoviychuk et al., 2021) by some expert-performance RL policies which have demonstrated strong performance in the real world. It is noteworthy that this dataset is designed for practical robotics applications. Models trained using this dataset can be directly deployed on real-world legged robots (e.g., Unitree Aliengo) to perform a variety of locomotive skills across diverse terrains. The test version of the dataset contains the aforementioned skills and terrains, with 320,000 trajectories and 60M steps.[1]

Evaluation on Simulator. To assess FCNet's effectiveness on multi-environment robotic datasets, we first perform imitation learning on this dataset. Our comparison focuses on Transformer, as other model architectures like RetNet and MLP have been demonstrated to be less effective in Sec. 4.1. We report the performances of FCNet and Transformer across various dataset sizes in Fig. 4, where FCNet consistently outperforms Transformer. Notably, when the dataset size is comparable to that of D4RL (1M$\sim$ 3M), Transformer's performance is markedly low, whereas FCNet still delivers promising results even with limited data. This underscores that FCNet, with its inductive bias tailored for robotic control, is more adept at modeling real-world robot trajectories than Transformer.

Given the constraints of end-side device performance and the real-time control requirements for robots, it's crucial that the inference latency of an embodied model is kept minimal when deployed in real-world scenarios (e.g., $< 20ms$ for quadruped robot).

Real-World Applications. We further deploy FCNet on real-world robots to evaluate its real-time inference ability

[1]We make some of the data public on the project page.

Table 2. **Results in D4RL-Mujoco**. For all algorithms, we report the performance with mean and variance under three random seeds.

Task Name	FCNet(Ours)	DT	DT-RetNet	BC	CQL	BEAR	BRAC-v	AWR
HALFCHEETAH-MEDIUM-EXPERT	$\mathbf{91.2 \pm 0.3}$	86.8 ± 1.3	$\mathbf{91.4 \pm 1.2}$	35.8	62.4	53.4	41.9	52.7
WALKER2D-MEDIUM-EXPERT	$\mathbf{108.8 \pm 0.1}$	$\mathbf{108.1 \pm 0.2}$	79.1 ± 1.8	6.4	98.7	40.1	81.6	53.8
HOPPER-MEDIUM-EXPERT	$\mathbf{110.5 \pm 0.5}$	107.6 ± 1.8	102.9 ± 5.6	**111.9**	**111.0**	96.3	0.8	27.1
HALFCHEETAH-MEDIUM	42.9 ± 0.4	42.6 ± 0.1	43.2 ± 0.4	36.1	44.4	41.7	**46.3**	37.4
WALKER2D-MEDIUM	75.2 ± 0.5	74.0 ± 1.4	73.3 ± 4.3	6.6	**79.2**	59.1	**81.1**	17.4
HOPPER-MEDIUM	57.8 ± 6.0	67.6 ± 1.0	$\mathbf{74.1 \pm 5.3}$	29.0	58.0	52.1	31.1	35.9
HALFCHEETAH-MEDIUM-REPLAY	39.8 ± 0.8	36.6 ± 0.8	17.2 ± 4.3	38.4	46.2	38.6	**47.7**	40.3
WALKER2D-MEDIUM-REPLAY	63.5 ± 7.5	$\mathbf{66.6 \pm 3.0}$	32.4 ± 6.6	11.3	26.7	19.2	0.9	15.5
HOPPER-MEDIUM-REPLAY	$\mathbf{85.8 \pm 1.7}$	82.7 ± 7.0	58.4 ± 2.8	11.8	48.6	33.7	0.6	28.4
Average	**75.1**	**74.7**	63.6	46.4	63.9	48.2	36.9	34.3

Table 3. **Results in D4RL-Adroit**. For FCNet and DT, we report the performance with mean and variance under five random seeds. Other scores are reported by the D4RL paper.

	FCNet (Ours)	DT	BC	CQL
PEN-HUMAN	57.7 ± 11.1	-0.2 ± 1.8	34.4	37.5
HAMMER-HUMAN	1.2 ± 0.0	0.3 ± 0.0	1.5	4.4
DOOR-HUMAN	0.4 ± 0.5	0.1 ± 0.0	0.5	9.9
RELOCATE-HUMAN	0.2 ± 0.2	0.0 ± 0.0	0.0	0.2
PEN-CLONED	50.4 ± 24.1	22.7 ± 17.1	56.9	39.2
HAMMER-CLONED	0.2 ± 0.0	0.3 ± 0.0	0.8	2.1
DOOR-CLONED	-0.2 ± 0.0	0.1 ± 0.0	-0.1	0.4
RELOCATE-CLONED	-0.2 ± 0.0	-0.3 ± 0.0	-0.1	-0.1
PEN-EXPERT	108.0 ± 11.3	110.4 ± 20.9	85.1	107.0
HAMMER-EXPERT	121.1 ± 6.1	89.7 ± 24.6	125.6	86.7
DOOR-EXPERT	102.9 ± 2.9	95.5 ± 5.7	34.9	101.5
RELOCATE-EXPERT	50.0 ± 6.0	15.3 ± 3.6	101.3	95.0
Average	**41.0**	27.8	36.7	40.3

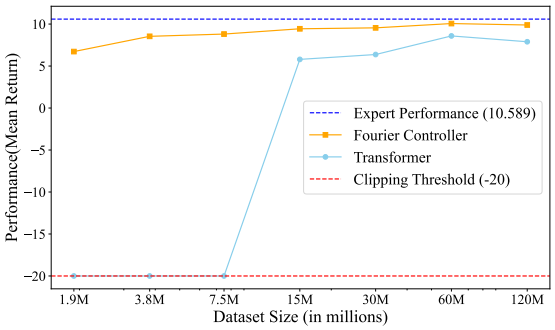

Figure 4. The performance of each model on the legged robotics dataset (measured by mean return, averaging the results across 1500*3 trajectories). For all future experiments, the 60M-step dataset is utilized as the standard reference.

Figure 5. Deploying FCNet to real-world legged robots.

and generalizability. Training on the 60M-step dataset mentioned in Sec. 4.2, our FCNet shows great adaptability on a variety of terrains not seen in the dataset, such as ice, deep snow, and steep slopes. The FCNet-controlled robot moves fluidly, likely due to the filtering of high-frequency noise. In terms of computational efficiency, FCNet achieves a low inference latency (∼2ms) on less powerful end-side devices, a critical factor for practical deployment. Conversely, the Transformer model, in our tests, does not perform as well in real-world settings due to issues like lower returns, less smooth output, and longer inference times, leading to less fluid motion and suboptimal performance in indoor scenarios such as stair climbing, turning, and navigating grassy terrains. More images and videos of these tests are available in the Appendix D and supplementary materials.

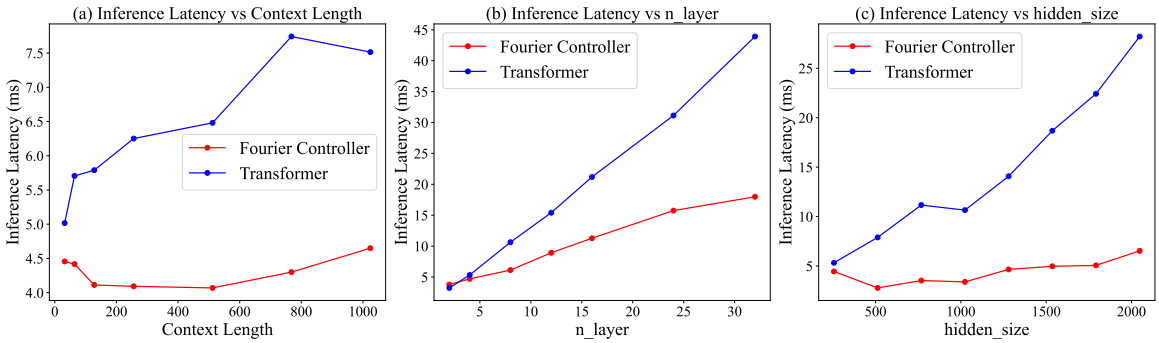

Figure 6. The CPU inference latency of FCNet and Transformer under various hyperparameter settings about structure.

4.3. Low Inference Latency

To assess FCNet's advantage in inference efficiency, particularly its capability to extract and encode time-varying features via frequency domain interpolation, we conduct a comparative analysis with Transformer under various hyperparameter settings. This is done by randomizing input data and initialized weights. We use the empirical value of m in FCNet as $\min\{2.5 \log n, \lfloor \frac{n}{2} \rfloor + 1\}$, where n is the context length. In Fig. 6 (a), as the context length increases, FCNet's inference latency remains relatively stable, whereas Transformer's latency continues to grow significantly, even with a KV cache. This trend can be attributed to Transformer's inherent $\mathcal{O}(n)$ inference complexity.

In Fig. 6(b) and (c), the inference latencies of FCNet and Transformer are examined as the number of layers and hidden sizes are increased. The results demonstrate that despite the growth in these hyperparameters, the increase in FCNet's inference latency is substantially smaller compared to that of the Transformer. When the model has as many as 32 layers or a hidden size of 2048, FCNet maintains a significant speed advantage, being 3 to 5 times faster than the Transformer. Importantly, even under these conditions, FCNet's inference latency remains below 20ms. This suggests that FCNet, even with an increased number of model parameters, can still meet the requirements for low inference latency. This is a crucial factor for real-time applications in robotics where quick and responsive control is essential.

More comprehensive experiments on ablation study on FCNet and Transformer are included in Appendix B, varying the parameter count (controlled by the number of layers and hidden size) and context length.

5. Conclusion and Limitations

In this paper, we present the Fourier Controller Network (FCNet), an innovative model reshaping embodied learning with a focus on frequency domain analysis. FCNet emerges as a distinctive Fourier-based structure, ingeniously incorporating inductive biases for robotic control, and is adeptly designed to minimize time complexity in embodied learning architectures. This design empowers FCNet to extract features with high accuracy, facilitating decision-making in embodied learning through frequency domain interpolation. FCNet excels in training efficiency, showcasing a training complexity of $\mathcal{O}(mn \log n + m^2 n)$, and remarkable inference efficiency at $\mathcal{O}(m)$. These developments not only boost the model's effectiveness but also greatly reduce computational requirements. Our results highlight the promise of frequency domain analysis in embodied learning and pave the way for further investigations into pre-training FCNet on extensive robotics datasets, potentially enhancing its generalization abilities significantly.

However, our work has the following limitations: First, we did not validate the scalability of FCNet on a larger embodied dataset. We hope that future scalable methods, such as the attention mechanism, can be adapted in the frequency domain. Second, although we validate the utility of FCNet on complex terrain locomotion and manipulation, we do not consider those scenarios where high-frequency information may be important. Also, we have not yet extended to multimodal inputs. We hope that combining FCNet for online RL training, incorporating multimodal inputs, and scaling FCNet are directions for future exploration.

Acknowledgements

This work was supported by the National Key Research and Development Program of China (No. 2020AAA0106302), NSFC Projects (Nos. 92248303, 92370124, 62350080, 62276149, U2341228, 62061136001, 62076147), BNRist (BNR2022RC01006), Tsinghua Institute for Guo Qiang, and the High Performance Computing Center, Tsinghua University. J. Zhu was also supported by the XPlorer Prize. We would like to thank Ronghua Hu and Zhuang Zhang for their help in real-world legged robot experiments. We would also like to thank Hao Guo, Muyan Hu, Weilin Zhao, Xinning Zhou, and Huayu Chen for their useful comments.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. The development and implementation of Fourier Controller Networks (FCNet) for real-time decision-making in embodied learning have the potential to significantly enhance the efficiency and effectiveness of robotic systems. The application of FCNet in various fields could lead to advancements in automation and intelligent systems, influencing sectors like healthcare, manufacturing, and service industries.

References

Agarwal, A., Kumar, A., Malik, J., and Pathak, D. Legged locomotion in challenging terrains using egocentric vision. In *Conference on Robot Learning*, pp. 403–415. PMLR, 2023.

Ahn, M., Dwibedi, D., Finn, C., Arenas, M. G., Gopalakrishnan, K., Hausman, K., Ichter, B., Irpan, A., Joshi, N., Julian, R., et al. Autort: Embodied foundation models for large scale orchestration of robotic agents. *arXiv preprint arXiv:2401.12963*, 2024.

Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bousmalis, K., Vezzani, G., Rao, D., Devin, C., Lee, A. X., Bauza, M., Davchev, T., Zhou, Y., Gupta, A., Raju, A., et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33: 1877–1901, 2020.

Chebotar, Y., Vuong, Q., Hausman, K., Xia, F., Lu, Y., Irpan, A., Kumar, A., Yu, T., Herzog, A., Pertsch, K., et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pp. 3909–3928. PMLR, 2023.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*, 34:15084–15097, 2021.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Fellows, M., Ciosek, K., and Whiteson, S. Fourier policy gradients. In *International Conference on Machine Learning*, pp. 1486–1495. PMLR, 2018.

Feng, G., Zhang, H., Li, Z., Peng, X. B., Basireddy, B., Yue, L., Song, Z., Yang, L., Liu, Y., Sreenath, K., et al. Genloco: Generalized locomotion controllers for quadrupedal robots. In *Conference on Robot Learning*, pp. 1893–1903. PMLR, 2023.

Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A., and Ré, C. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fu, Z., Zhao, T. Z., and Finn, C. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

Ghosh, D., Rahme, J., Kumar, A., Zhang, A., Adams, R. P., and Levine, S. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability. *Advances in Neural Information Processing Systems*, 34:25502–25515, 2021.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Gu, J., Kirmani, S., Wohlhart, P., Lu, Y., Arenas, M. G., Rao, K., Yu, W., Fu, C., Gopalakrishnan, K., Xu, Z., et al. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches. *arXiv preprint arXiv:2311.01977*, 2023.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396. IEEE, 2017.

Gupta, A., Eppner, C., Levine, S., and Abbeel, P. Learning dexterous manipulation for a soft robotic hand from human demonstrations. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3786–3793. IEEE, 2016.

Hao, Z., Liu, S., Zhang, Y., Ying, C., Feng, Y., Su, H., and Zhu, J. Physics-informed machine learning: A survey on problems, methods and applications. *arXiv preprint arXiv:2211.08064*, 2022.

He, Z., Yang, M., Feng, M., Yin, J., Wang, X., Leng, J., and Lin, Z. Fourier transformer: Fast long range modeling by removing sequence redundancy with fft operator. *arXiv preprint arXiv:2305.15099*, 2023.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in Neural Information Processing Systems*, 34: 1273–1286, 2021.

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L. Vima: General robot manipulation with multimodal prompts. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pp. 651–673. PMLR, 2018.

Kalashnikov, D., Varley, J., Chebotar, Y., Swanson, B., Jonschkowski, R., Finn, C., Levine, S., and Hausman, K. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Kashiri, N., Abate, A., Abram, S. J., Albu-Schaffer, A., Clary, P. J., Daley, M., Faraji, S., Furnemont, R., Garabini, M., Geyer, H., et al. An overview on principles for energy efficient robot locomotion. *Frontiers in Robotics and AI*, 5:129, 2018.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Kumar, A., Singh, A., Ebert, F., Nakamoto, M., Yang, Y., Finn, C., and Levine, S. Pre-training for robots: Offline rl enables learning new tasks from a handful of trials. *arXiv preprint arXiv:2210.05178*, 2022.

Lai, H., Zhang, W., He, X., Yu, C., Tian, Z., Yu, Y., and Wang, J. Sim-to-real transfer for quadrupedal locomotion via terrain transformer. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5141–5147. IEEE, 2023.

Lee, A. X., Devin, C. M., Zhou, Y., Lampe, T., Bousmalis, K., Springenberg, J. T., Byravan, A., Abdolmaleki, A., Gileadi, N., Khosid, D., et al. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. In *5th Annual Conference on Robot Learning*, 2021.

Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020a.

Lee, K., Seo, Y., Lee, S., Lee, H., and Shin, J. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 5757–5766. PMLR, 2020b.

Lee-Thorp, J., Ainslie, J., Eckstein, I., and Ontanon, S. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*, 2021.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Margolis, G. B. and Agrawal, P. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *Conference on Robot Learning*, pp. 22–31. PMLR, 2023.

Nguyen, T., Pham, M., Nguyen, T., Nguyen, K., Osher, S., and Ho, N. Fourierformer: Transformer meets generalized fourier integral theorem. *Advances in Neural Information Processing Systems*, 35:29319–29335, 2022.

Padalkar, A., Pooley, A., Jain, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Singh, A., Brohan, A., et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Cao, H., Cheng, X., Chung, M., Grella, M., GV, K. K., et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866*, 2023.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Radosavovic, I., Xiao, T., Zhang, B., Darrell, T., Malik, J., and Sreenath, K. Learning humanoid locomotion with transformers. *arXiv preprint arXiv:2303.03381*, 2023.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Rudin, N., Hoeller, D., Reist, P., and Hutter, M. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pp. 91–100. PMLR, 2022.

Sandha, S. S., Garcia, L., Balaji, B., Anwar, F., and Srivastava, M. Sim2real transfer for deep reinforcement learning with stochastic state transition delays. In *Conference on Robot Learning*, pp. 1066–1083. PMLR, 2021.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Trefethen, L. N. Finite difference and spectral methods for ordinary and partial differential equations. 1996.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Wen, Y., Wan, Z., Zhou, M., Hou, S., Cao, Z., Le, C., Chen, J., Tian, Z., Zhang, W., and Wang, J. On realization of intelligent decision-making in the real world: A foundation decision model perspective. *arXiv preprint arXiv:2212.12669*, 2022.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. Sep 2023.

Xiong, H., Mendonca, R., Shaw, K., and Pathak, D. Adaptive mobile manipulation for articulated objects in the open world. *arXiv preprint arXiv:2401.14403*, 2024.

Xu, Z., Zeng, A., and Xu, Q. Fits: Modeling time series with $10k$ parameters, 2024.

Ye, M., Kuang, Y., Wang, J., Rui, Y., Zhou, W., Li, H., and Wu, F. State sequences prediction via fourier transform for representation learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Ying, C., Hao, Z., Zhou, X., Su, H., Liu, S., Li, J., Yan, D., and Zhu, J. Reward informed dreamer for task generalization in reinforcement learning. *arXiv preprint arXiv:2303.05092*, 2023.

Yu, C., Zhang, W., Lai, H., Tian, Z., Kneip, L., and Wang, J. Multi-embodiment legged robot control as a sequence modeling problem. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7250–7257. IEEE, 2023.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33: 5824–5836, 2020.

Zhang, S., Tong, H., Xu, J., and Maciejewski, R. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

A. Related Work

RL in Robotic Control. Designing general algorithms and models to control real-world robots, like robotic arms and legged robots, has always been a focus of research. With the development of Deep Reinforcement Learning, a promising method is to train a general policy via interacting with different dynamics, of which the trained agent can adapt to changes in physical properties, even in some extreme conditions (Schulman et al., 2017; Lee et al., 2020a; Rudin et al., 2022; Margolis & Agrawal, 2023; Feng et al., 2023; Agarwal et al., 2023; Lai et al., 2023; Yu et al., 2023; Radosavovic et al., 2023; Xiong et al., 2024). More recently, pretraining on diverse datasets has shown promise in designing general models. DT (Chen et al., 2021) and TT (Janner et al., 2021) treat offline RL as a sequence modeling problem by treating the rewards, states and actions as a sequence so that the Transformer autoregressively generates the next token such as action token. Borrowing these ideas, an array of works (Reed et al., 2022; Brohan et al., 2022; 2023; Jiang et al., 2022; Bousmalis et al., 2023; Wen et al., 2022; Padalkar et al., 2023; Chebotar et al., 2023; Fu et al., 2024; Ahn et al., 2024; Gu et al., 2023) utilize Transformer (Vaswani et al., 2017) for pretraining on a larger dataset with multiple decision-making and robotics control tasks, demonstrating strong performance and revealing the generalization that comes with increased model parameters and data size. Among them, RT-2 (Brohan et al., 2023) uses Internet-scale data to pre-train a Robotics Transformer model with imitation learning methods, and deploys RT-2 on real-world robots for tabletop tasks, which can perform well even in unseen environments. However, the large scale of parameters in these models results in high inference latency, which limits their application in real-world robots, thus designing scalable and efficient inference network architecture is significant in this field.

Efficient Variants of Transformer Architecture. Transformer (Vaswani et al., 2017) has been heavily used in Natural Language Processing (NLP) tasks since it was proposed. It models the relationship between two by two elements in the context window through the attention mechanism. Due to its high inference time complexity, some works trying to speed up the inference of Transformer such as KV cache, FlashAttention(Dao et al., 2022; Dao, 2023) and StreamingLLM (Xiao et al., 2023) are proposed to make Transformer have higher speed in inference. However, these works are difficult to achieve theoretically lower complexity while maintaining the same performance. (Child et al., 2019; Kitaev et al., 2020) improves the efficiency of Transformer when dealing with long sequence tasks by improving the attention mechanism, but they do not make improvements in terms of feature extraction capability and inference time complexity. Similarly, Dai et al. makes efforts for the efficiency of the Transformer on long sequences of text. Meanwhile, many variants of Transformer such as (Katharopoulos et al., 2020; Peng et al., 2023; Fu et al., 2022; Poli et al., 2023; Sun et al., 2023; Gu & Dao, 2023) have been proposed in order to achieve lower time complexity of inference. However, the inductive bias of these model architectures is not appropriate for embodied learning tasks, and it is difficult to achieve strong performance using limited data training. We choose one of the representative State Space Models (SSMs) for experimental validation to demonstrate this. FNet (Lee-Thorp et al., 2021), Fourier Transformer (He et al., 2023), and FourierFormer (Nguyen et al., 2022) explore combining Fourier transforms and Transformer, but do not incorporate feature extraction with low time complexity in the frequency domain, as well as a lack of real-time inference validation on real-world robots.

Introducing Physical Prior in Machine Learning. When dealing with real-world physics, including robotics, it is essential to introduce physical priors into the machine-learning process to decrease data demand and improve models' performance (Karniadakis et al., 2021; Hao et al., 2022). For example, PointNet (Qi et al., 2017) and Graph Convolutional Networks (GCN) (Zhang et al., 2019) make the network architecture permutation-invariant utilizing permutation-invariant operation like summation. And NeuralODE (Chen et al., 2018) designs an architecture that naturally conforms to ordinary differential equations (ODEs). Fourier neural operator (Li et al., 2020) takes advantage of the symmetry of the differentiation in the time domain and the multiplication in the frequency domain to design a network structure that can efficiently solve partial differential equations (PDEs). Also, FITS (Xu et al., 2024) discards the high-frequency portion of the time series that has little effect on the time series and fits the time series through a linear layer in the frequency domain. Due to the smoothing properties of most physical phenomena, such structures are also ideal for filtering noise in data. Ye et al.; Fellows et al. introduce Fourier analysis in reinforcement learning, but do not introduce multi-layer stacked structure for complex pattern learning.

Schematic of the CSC block A detailed schematic of the CSC block, illustrating the computation of $y_0, y_1, \ldots, y_{n-1}$, is shown in Fig. 7.

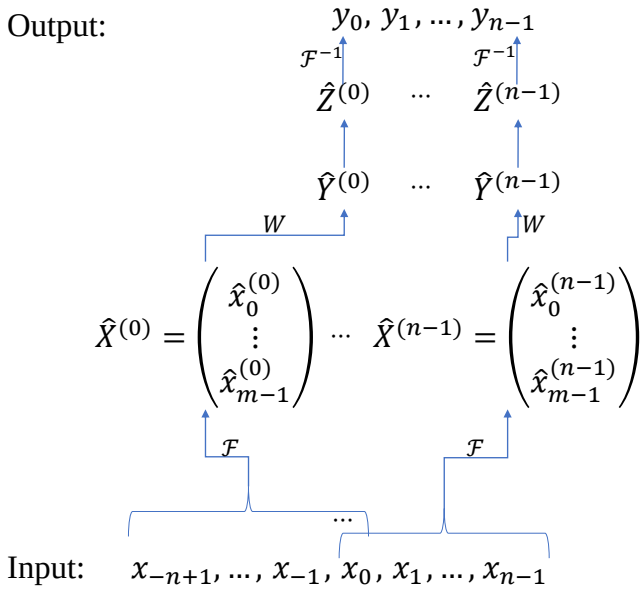

Figure 7. CSC block diagram for computing $y_0, y_1, \dots, y_{n-1}$.

B. Ablation Study

Varying Model Structures In Fig. 8, FCNet shows consistent superiority in both performance and inference latency over the Transformer across various structures, encompassing different numbers of layers and hidden sizes. Notably, with fewer parameters, FCNet exhibits a robust fitting ability, significantly outperforming Transformer. This implies its effectiveness even in scenarios demanding compact models. As the model scales up, FCNet maintains greater stability.

Varying Context Lengths Fig. 9 illustrates how FCNet and Transformer perform under various context lengths. With increasing context length, FCNet shows improved performance with relatively stable inference times. In contrast, Transformer does not display the same level of efficiency. Moreover, considering the diverse terrains and historical data in the dataset, context is crucial for policy learning. Experiments reducing Transformer's context length (e.g., to 8) result in completely ineffective policies.

C. Experimental Details

C.1. D4RL

The hyperparameters of FCNet for D4RL are shown in Table 4.

C.2. Multi-Environment Legged Robot Locomotion Dataset Generation

The multi-environment legged robot locomotion dataset is an expert dataset, collected in Isaacgym (Makoviychuk et al., 2021) by some expert-performance RL policies which have demonstrated strong performance in the real-world legged robot (i.e., Unitree Aliengo).

It covers various skills (commands from the remote control, e.g., standing, rushing, crawling, squeezing, tilting, and running), and different terrains (e.g., rough terrain, stairs, slopes, and obstacles), following Rudin et al. (2022). The vector of commands issued by the remote control is concated in the state for each time step. Terrain-related information is privileged information, i.e., inaccessible, and thus needs to be approximated by historical information to learn an optimal policy. The components of the state include motor velocity, motor position, body angular velocity, and remote control commands. The action in the dataset corresponds to the motor position control of the robot.

The test version of the dataset contains the aforementioned skills and terrains, with 320,000 trajectories and totaling 60M steps, generated by expert policy. Each trajectory includes the robot's standing and walking on each terrain. To maintain

Table 4. Hyperparameters of FCNet for OpenAI Gym (D4RL) experiments.

Hyperparameter	**Value**
Number of layers	4
Number of modes m	10
hidden dimension	128 for the last decoder Q
	512, otherwise
Nonlinearity function	GeLU
Batch size	128
Context length K	64 HalfCheetah, Hopper, Walker
Return-to-go conditioning	1.15 medium-expert
	1.0 medium, medium-replay
Adam β	$(0.9, 0.98)$
Adam ϵ	10^{-9}
LR scheduler	get_cosine_schedule_with_warmup
Learning rate	5×10^{-3}
Weight decay	10^{-4}
Epoch	50
Learning rate decay	Linear warmup for first 20% training steps

both the quantity and diversity of trajectories, we have set an average length of 192 for each trajectory. This length is chosen to comprehensively capture the robot's range of motions, including standing and walking, across various terrains.

C.3. Multi-Environment Legged Robot Locomotion

In this experiment, we keep the number of parameters of FCNet and Transformer roughly equal (790k parameters).

The hyperparameters of FCNet for the multi-environment legged robot locomotion dataset are shown in Table 5.

Table 5. Hyperparameters of FCNet for multi-environment legged robot locomotion experiments.

Hyperparameter	**Value**
Number of layers	4
Number of modes m	10
hidden dimension	128 for the last decoder Q
	256, otherwise
Nonlinearity function	GeLU
Batch size	1024
Context length K	64
Adam β	$(0.9, 0.98)$
Adam ϵ	10^{-9}
LR scheduler	get_cosine_schedule_with_warmup
Learning rate	5×10^{-3}
Epoch	50
Learning rate decay	Linear warmup for first 20% training steps

The hyperparameters of Transformer for the multi-environment legged robot locomotion dataset are shown in Table 6.

Other ablation experiments are modified from these hyperparameters. It is also guaranteed that the number of FCNet and Transformer model parameters is close to equal.

A robust sim-to-real FCNet model which we deploy in the real-world legged robot (i.e., Unitree Aliengo for video capture) is trained from the 60M-step dataset with the metrics shown in Table 7. We choose $m = 10$ and $n = 64$ here (n denotes the context length). In fact, better results might be obtained with larger n.

Table 6. Hyperparameters of Transformer for multi-environment legged robot locomotion experiments.

Hyperparameter	**Value**
Number of layers	4
Number of modes m	10
hidden dimension	128 for the last decoder Q 256, otherwise
Nonlinearity function	GeLU
Batch size	1024
Context length K	64
LR scheduler	get_cosine_schedule_with_warmup
optimizer	Lion
Learning rate	5×10^{-3}
Epoch	50
Weight decay	10^{-4}
Learning rate decay	Linear warmup for first 20% training steps

Table 7. Metrics of sim-to-real FCNet model.

Metrics	**Value**
Number of model trainable parameters	$787,852$
Loss of training	0.000408
Loss of test	0.000420
Mean return in simulator	$10.060/10.589$

C.3.1. Details of observation and action

We provide detailed information about the observations and actions used in FCNet for the multi-environment legged robotics locomotion task:

- Observations:
  - Projected gravity (3-dimensional),
  - Joint velocity (12-dimensional),
  - Sine values of joint position (12-dimensional),
  - Cosine values of joint position (12-dimensional),
  - Body angular velocity (12-dimensional),
  - Last action (12-dimensional),
  - Command (16-dimensional)
- Actions
  - Expected joint position (12-dimensional).

Of note, each '12-dimensional' parameter corresponds to the 12 motors that drive the quadruped robot, including the motors for the hips, thighs, and calves of each of the four legs.

C.3.2. Sim-to-real gap

FCNet is pre-trained solely on simulator data before being directly deployed on a real robot to navigate a variety of terrains. This zero-shot transfer leads to robust real-world locomotion, despite the inherent sim-to-real gap.

Sim-to-real gap The discrepancies between simulators and real-world environments can be seen in areas like state space (differences in robot mechanics), action space (issues like overshooting due to high torque), and transition dynamics

(terrain inconsistencies). Additionally, real-world scenarios demand low inference latency to ensure smooth operation - a requirement often not necessary in simulations.

To bridge this gap, we focus on two main areas: data collection and inference.

Data collection High-quality data is vital. We use reward shaping to guide data generation, avoiding illegal states or actions. The data also needs to encompass a wide range of commands and terrains, achievable through domain randomization, to ensure the model closely aligns with real-world scenarios.

Inference Post-data collection, engineering efforts such as integrating sensor data into the state and relaying neural network outputs to motor actions become critical. The key to successful sim-to-real transition is the neural network's inference latency. Given FCNet's low complexity in this regard, we efficiently address the sim-to-real challenge during deployment.

C.3.3. HIGH-LEVEL CONTROLLER

The robot's actions, such as moving backward and forward, turning left and right, spinning, standing, crouching, and sprinting, are directed by human operators via remote control. These commands, constituting part of the observation, are interpreted by the policy network, which then translates them into low-level actions executed by the robot.

C.4. Low Inference Latency

All our tests for inference latency only target the last action in each context length window, as this ensures that the maximum inference latency of the robot does not exceed a certain hardware threshold when deployed. The testing procedure incorporates three random number seeds, with each seed conducting 10 episodes and subsequently calculating the average of CPU inference latency using `torch.profiler`, to ensure a small variance.

In the experiments of Fig. 6, when we change a hyperparameter, other hyperparameters remain at their default values: `Context Length`$= 64$, `n_layer`$= 4$, `hidden_size`$= 256$.

It is worth noting that we tested the model's inference latency on the CPU since many robots lack GPUs, and energy consumption considerations make lightweight deployment crucial.

Our test environment is shown in Table 8:

Table 8. Information on the latency measurement environment.

	Value
Warm Up	500 dummy state inputs
CPU	Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz
Memory Configured Clock Speed	2400MHz
OS	Ubuntu 18.04.5 LTS, Linux 4.15.0-135-generic
Python	3.8.18
GCC	8.4.0
Torch Version	1.12.1

Furthermore, in the Ablation Study detailed in Appendix B, we consistently measure the inference latency of 8 models simultaneously to ensure environmental consistency across each set of experimental data.

D. Real-World Applications

We provide a more detailed image (i.e., video screen capture) in Fig. 10 of real-world robot deployments, as a supplement to Sec. 4.2. It also includes the comparison of FCNet and Transformer.

In this image, it can be seen that the robot corresponding to Transformer appears to fall in scenarios such as going upstairs indoors and going downstairs indoors. We put a more detailed video with more tests in the supplementary material.

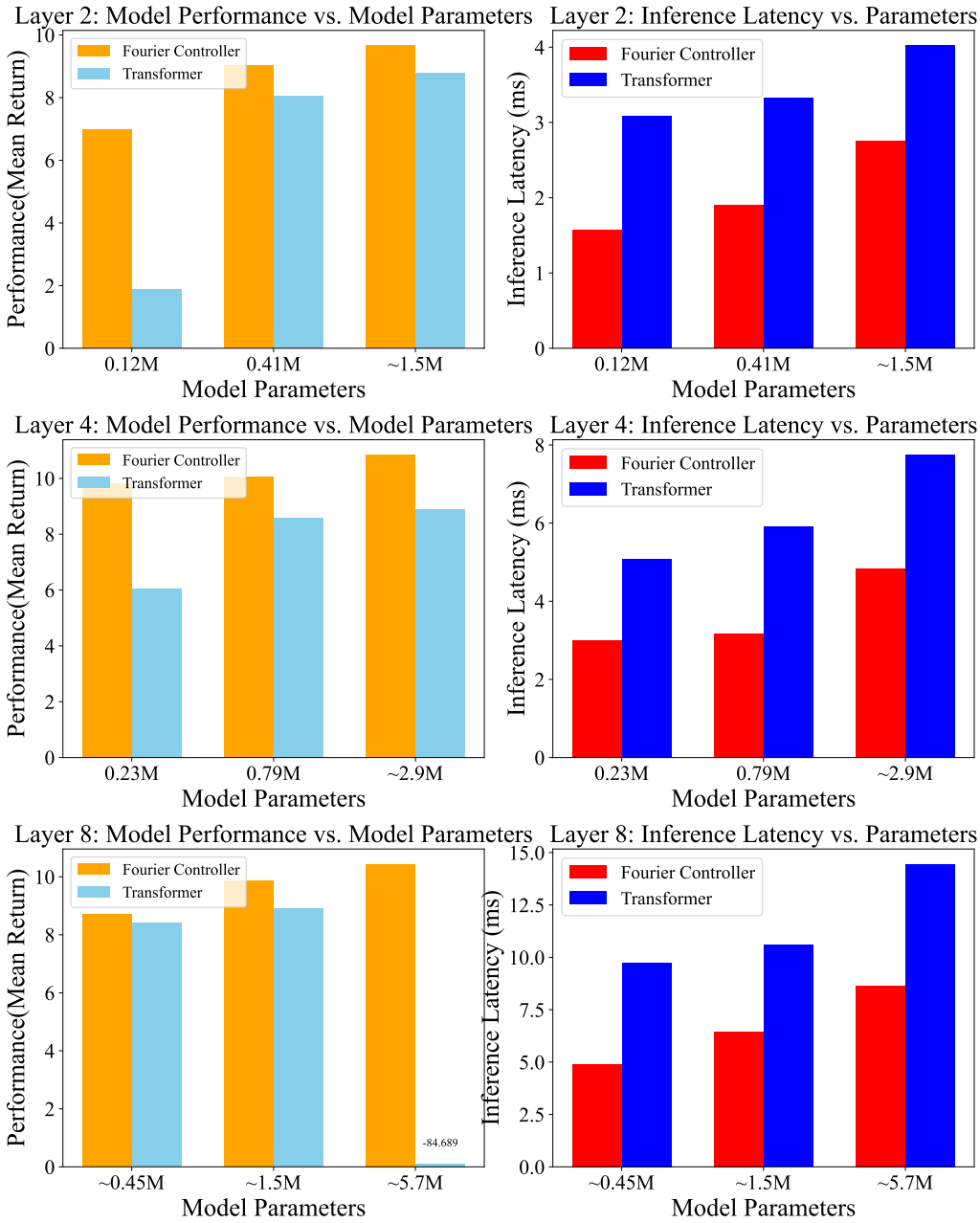

Figure 8. The performance and CPU inference latency of each model on the legged robotics dataset (measured by mean return). The horizontal axis indicates the model's parameter count, adjustable by varying the number of layers and hidden size in both Transformer and Fourier Controller. Throughout these experiments, hyperparameters like the learning rate remained constant. Notably, in the Transformer experiments with n_layer=8 and approximately 5.7M model parameters, a high learning rate induces oscillations in the training process.

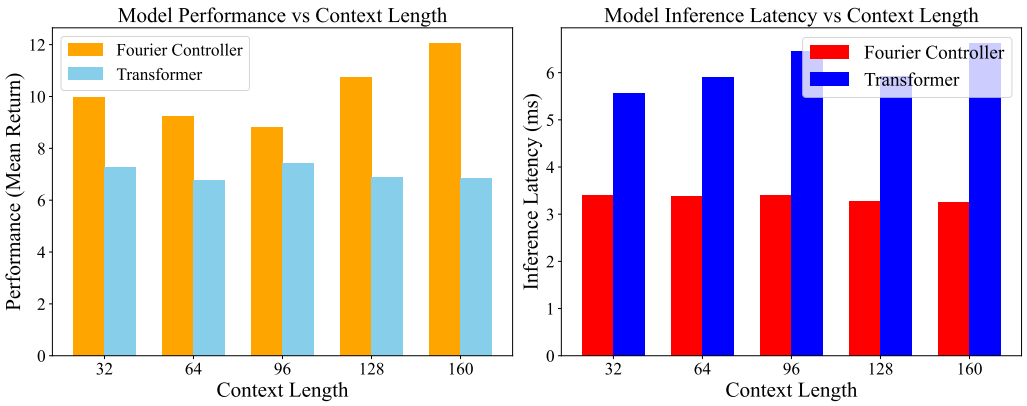

Figure 9. The performance and CPU inference latency of each model on the legged robotics dataset (measured by mean return). The horizontal axis indicates the context length of the model. This study extends the experiment shown in Fig. 4, where we vary the context length and then evaluate over 1500×3 trajectories, averaging both return and inference latency. We refrained from testing longer sequence lengths because doing so, while keeping the total data volume constant, would decrease the number of trajectories, further leading to data insufficiency.

Figure 10. Deploying FCNet to real-world legged robots.