Controlling Deep Generative Models via Physical and Mathematical Priors

by
Nate Gillman
B.A., Wesleyan University, Middletown, CT, 2020
M.S., Brown University, Providence, RI, 2022

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Integrative Studies
(Mathematics & Computer Science) at Brown University

Providence, Rhode Island
May 2026

This dissertation by Nate Gillman is accepted in its present form by
the Department of Integrative Studies
(Mathematics & Computer Science) as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date _____ _____

Dr. Chen Sun, Director

Recommended to the Graduate Council

Date _____ _____

Dr. Carsten Eickhoff, Reader

Date _____ _____

Dr. Charles Herrmann, Reader

Date _____ _____

Dr. Jeffrey Hoffstein, Reader

Approved by the Graduate Council

Date _____ _____

David P. Lindstrom
Dean of the Graduate School

Dedication

My parents always told me that education is the one thing that no one can ever take away from you. This dissertation is dedicated to my parents. Mom and Dad: I love you both so much, thanks for everything.

Acknowledgments

My deepest thanks goes to my PhD advisor, Chen Sun, for adopting me into his lab. When we met, I had a lot of pure math experience but relatively little direction in applied AI. I couldn't have asked for a kinder, more patient, or wiser advisor. Thank you, Chen, for demonstrating what integrity, warmth, and humility look like in research.

I also want to thank the rest of my thesis committee for guiding my rather non-traditional journey. I started at Brown specializing in analytic number theory under Jeff Hoffstein; when my interests shifted, he championed my switch and has supported me ever since. Carsten Eickhoff, my first computer science advisor, welcomed me into his group and made my switch into AI research possible. Finally, Charles Herrmann happily mentored me through *another* pivot into large-scale video modeling, despite my lack of experience in the subfield. Jeff, Carsten, and Charles: thank you for taking a chance on me. Special thanks go to Tom Goodwillie and Stephen Bach for their guidance on my other committees.

I am deeply grateful for all of the administrative and technical support teams that have made my experience at Brown possible and also enjoyable. Thank you to Audrey Aguiar, Lori Nascimento, and Doreen Pappas in the math department, and Lauren Clarke, Eugenia DeGouveia, Jesse Polhemus, Dawn Reed, John Tracey-Ursprung, John Bazik, and Singh Saluja in computer science.

I've had the privilege of collaborating with many wonderful students while at Brown, including William Rudman, Taylor Rayne, Michael Freeman, Daksh Aggarwal, Chia-Hong Hsu, Calvin Luo, Evan Luo, Yinghua Zhou, Zitian Tang, and Arjan Chakravarthy. I was especially fortunate to have worked with Michael Freeman and Daksh Aggarwal, who co-authored every single project that I led over the past three years. That collaboration has been incredibly fun and rewarding. I'm also thankful to my many external collaborators at Google DeepMind and Google Research: Yonglong Tian, Saurabh Singh, Deqing Sun, Ameesh Makadia, and Carlos Esteves.

My background in pure math research laid the foundation for my AI research. Deep thanks are due to Jeff Hoffstein again; I came to Brown to study analytic number theory with him, which I did for a very short time, and it was a blast. Thank you to Ken Ono for teaching me about the role of passion and intensity in research. I am also grateful to my undergraduate mentors at Wesleyan: Ilesanmi Adeboye, David Constantine, Cameron Hill, David Pollack, and Felipe Ramirez. My mentors and advocates during High School seeded my early love for mathematics: Laura Goetz, Jon Goetz, Matt Davis, Brian Schuler, John Chase, Kristina Petulante, Douglas McDonald, Leslie McDonald, Evan Anderson, Matt Heyman, Amin Gholampour, and Kasso Okoudjou.

None of this would have been possible without the environment of Chen's PALM lab. Many thanks to my

labmates for the fun times and fruitful discussions: Apoorv Khandelwal, Calvin Luo, Shijie Wang, Tian Yun, Zilai Zeng, and Zitian Tang.

Lastly, I must thank my family and friends for keeping me tethered to reality. My parents, Lesli and Todd Gillman, modeled hard work, perseverance, and intellectual honesty as they raised me, which really came in handy when my code wouldn't compile. I'm incredibly thankful to my human sisters, Abigail and Rebecca, for their love. I'm also thankful for my dog sisters, Phoebe and FLOTUS, for their love (and for never asking me when I'm going to graduate). I also owe a massive thank you to Liat Schreiber for her love and support through all of the stressful deadlines.

And of course, to all my other friends. I am terrified to list you all by name because PhD sleep deprivation has destroyed my working memory and I will inevitably leave someone out. So, if we have ever shared a meal, played a game, or let out a commiserating sigh over the past few years: thank you, **[Insert Your Name Here]**. I literally could not have finished this dissertation without you, **[Insert First Name Again]**.

Contents

⋆ Each chapter of this thesis has been published elsewhere: Ch. 1, SCSC at ICML 2024; Ch. 2, FH at ICLR 2025; Ch. 3, FP at NeurIPS 2025; and Ch. 4, GF at CVPR 2026.

List of Tables

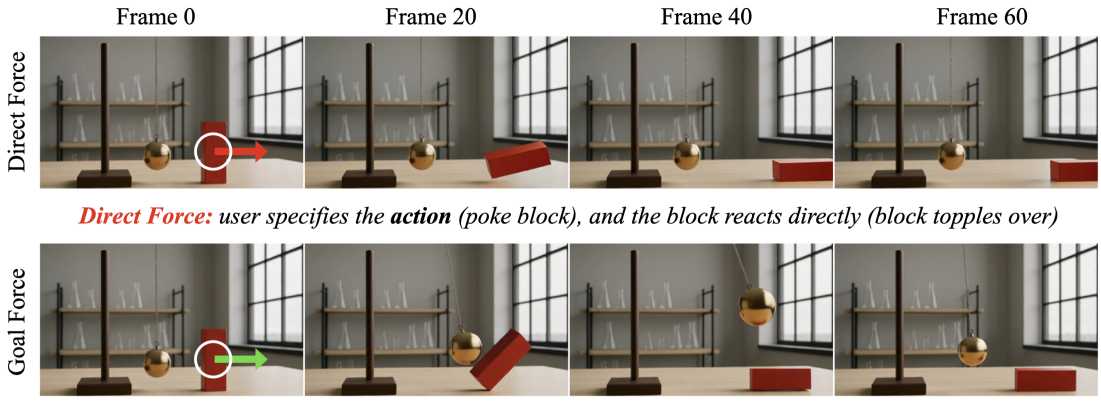
List of Figures

Introduction

Humans develop an intuitive understanding of physical dynamics from infancy. We possess an innate grasp of *forward causality*, which acts as an internal "physics engine" that allows us to mentally simulate how an environment will react to a specific action [40]. Adults use this simulation to accurately predict complex outcomes, such as the stability of a precarious block tower [11]. Equally fundamental is our capacity for *inverse planning* under physical constraints, which allows us to harness our intuitive physics to achieve desired future states [8]. For example, when attempting to sink a billiard ball, we do not merely guess at random actions; rather, we reason backward from the target destination to calculate the exact strike which ensures that outcome.

A central aspiration of artificial intelligence is to build systems equipped with this same intuitive understanding of the physical world, as this capability is essential for agents to safely plan and interact within complex environments. At first glance, recent advancements in video foundation models [121, 16] might suggest we are closing in on this goal. By scaling generative architectures on vast quantities of data, these models implicitly learn to simulate realistic dynamics, allowing them to generate high-fidelity video sequences that obey our expectations of the physical world. For example, if we give a video model an image of toppling dominos, it can generate a visually realistic continuation by generalizing from millions of observed trajectories of colliding objects seen during training. However, standard architectures do not natively expose interfaces for explicit physical reasoning; for example, standard generative architectures lack explicit representations or control interfaces for variables like mass or momentum. A fundamental gap remains between passively matching a training data distribution and actively modeling causal dynamics. This thesis attempts to bridge this gap by demonstrating how the intuitive physics already captured by these statistical models can be successfully elicited, structured, and controlled through the strategic injection of mathematical and physical priors.

This distinction becomes critical when attempting to build action-conditioned transition dynamics models, sometimes referred to as "world models" [67, 44]. These systems, which predict the future state of an environment given its current state and an agent's actions, have obvious utility for interactive media, such as video games and content creation. More importantly, they are vital for simulating counterfactuals and long-tail events in safety-critical embodied AI. For example, an autonomous driving system must learn to navigate rare, out-of-distribution occurrences, such as an elephant crossing a street. Such scenarios are exceedingly difficult to capture in natural driving logs. By utilizing generative models to synthesize videos of these highly specific scenarios, practitioners can generate the training data necessary to robustly train and localize decision-making models before deploying them in the real world.



Direct Force: *user specifies the* ***action*** *(poke block), and the block reacts directly (block topples over)*

Goal Force: *user specifies the* ***goal*** *(block forced to right), then the model generates an* ***action*** *to accomplish the goal (pendulum strikes block)*

Figure 1: **Force Prompting via Direct Forces and Goal Forces.** A core objective of this thesis is giving users physically grounded ways to interact with generative models. This figure contrasts two different levels of physical control. Top, **Direct Force**: The user pushes the target object directly, and the model generates the immediate physical reaction *(explored in Chapter 3).* In this case, that involves directly toppling over the block. Bottom, **Goal Force**: The user specifies what they want to happen to the target object. Instead of moving the object directly, the model must reason backward to figure out *how* to achieve that goal *(explored in Chapter 4).* In this case, that involves inventing a chain of events where a pendulum swings to strike the block.

However, if these generative models are to be trusted for embodied planning and safety-critical training data synthesis, their physical reasoning must be robust, reliable, and controllable. Standard probabilistic frameworks, such as maximum likelihood estimation and score matching, are designed primarily to minimize the statistical distance between generated samples and the training data. Because they lack explicit mathematical or physical inductive biases, these objectives excel at capturing visual correlations but fail to provide the interfaces needed to actively guide and reason about physical dynamics. Consequently, their learned physics remains purely statistical and lacks explicit grounding, meaning that relying solely on these data-driven objectives to build interactive world models exposes four challenges.

First, without grounding in physical invariants, models are highly vulnerable to distributional drift when exposed to synthetic or out-of-distribution data. As world models are increasingly tasked with generating synthetic training data, and inevitably ingest AI-generated content from the web during their own training, they risk entering "self-consuming loops" [5] where physical inaccuracies compound. Without a physics correction mechanism to anchor the data distribution, this iterative training leads to severe mode collapse. Second, a representational mismatch exists between the inherently continuous nature of physical variables, and the discrete tokenization relied upon by modern sequence models [6]. Standard language modeling architectures treat discrete buckets as independent, unstructured categories, stripping away the mathematical relationships between neighboring values. This leaves the model without the continuity inductive bias necessary to accurately learn and represent continuous probability distributions. Third, standard text-to-video interfaces are too ambiguous to specify precise physical interventions. While foundation models can synthesize plausible forward dynamics, they lack the explicit physical conditioning interfaces required to act as true interactive simulators. Without an explicit force conditioning interface, the video model is unable

to map interventional inputs, such as specific physical force vectors, to accurate dynamical outcomes. And fourth, generative objectives optimize for forward prediction and do not naturally support goal-conditioned planning. Reasoning backward from a desired physical outcome to infer the necessary antecedent actions is computationally misaligned with standard "next-frame" prediction, meaning models cannot intrinsically solve causal, goal-conditioned tasks without new training priors.

To address these challenges, this thesis proposes a framework that embeds mathematical and physical priors directly into the architectures and training procedures of deep generative models. By grounding these models in deterministic physics simulators, continuous mathematical functions, and explicit physical controls, we offer concrete methods to adapt sequential "next-frame" prediction models into world models that give users precise control over the physics in the scene. This framework is developed across four chapters:

- **Chapter 1** addresses the *training data contamination issue* in the context of iterative learning. We propose the Self-Correcting Self-Consuming framework, which uses a deterministic physics simulator as a mechanism to correct any corrupted training data before feeding them to the model. We provide theoretical and empirical evidence that incorporating this physical prior prevents mode collapse when models are trained on contaminated data including their own synthetic outputs.

- **Chapter 2** addresses the *mathematical representation of continuity*. To overcome the discretization artifacts of token-based language modeling architectures, we introduce the Fourier Head. By integrating the mathematical prior of Fourier series directly into the neural network, we adapt discrete language models to more accurately model complex, continuous probability distributions such as time series.

- **Chapter 3** addresses *intuitive physics interaction* in the context of video generative models. We propose Force Prompting, which is a framework that allows video diffusion models to learn intuitive physics control from synthetic data. This allows users to precisely guide open-world video generation using explicit physical force vectors, removing the need for 3D assets or external physics engines at inference time.

- **Chapter 4** addresses *causal planning* by proposing the Goal Force framework. We examine how models can learn to execute a causal sequence of actions toward a specific physical goal. By training models to propagate physical forces through intermediate causal chains, we demonstrate that video generators can act as probabilistic visual planners, reasoning backward from a goal state to infer the necessary physical cause.

The findings presented in this thesis deepen our understanding of how to embed mathematical and physical priors into deep generative models. Ultimately, these techniques establish foundational steps toward building interactive models of the world that are reliable, controllable, and capable of explicit causal reasoning within physical constraints.

Chapter 1

Self-Correcting Self-Consuming Loops For Generative Model Training

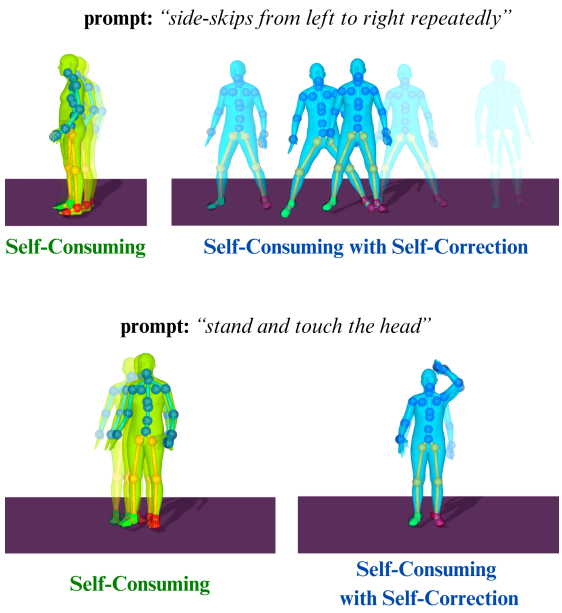


Figure 1.1: What happens after iteratively training a text-conditioned generative model for human motion synthesis for 50 generations? We simulate a self-consuming loop by creating synthetic data with the latest generative model, and mixing them with the original data to continue training the next generative model. We observe that by self-correcting the synthetic data with a physics simulator, the model can successfully avoid **collapse** and generate **high-quality human motion**. Faded poses represent poses from further back in time. This work provides theoretical and empirical justification for the self-correcting self-consuming loop.

Generative models have been used to synthesize training data for various learning tasks, to varying degrees of success. For example, for the tasks of image classification and contrastive representation learning, recent work [7, 116] finds that using data synthesized from generative models rivals using real data. Unfortunately, there is a gloomier outlook when attempting to generalize this framework to generative model training.

On one hand, there is evidence to suggest that training a generative model with its own outputs in a self-consuming manner will lead to collapse [5]. For example, after 50 iterations of self-consuming training, a human motion diffusion model [115] collapses and fails to follow the text prompts or the laws of physics (see the two examples on the left of Figure 1.1).

On the other hand, evidence suggests that such a framework could avoid collapse, but only when a "moderate" amount of synthetic data is used [13]. Worse still, this self-consuming scenario might happen without us knowing, and without us being able to quantify how much synthetic data is being used during training, due to the wide spread of AI generated content on the internet.

Intuitively, model collapse might be delayed or avoided by incorporating higher quality human generated data [5], or by manually fixing the "mistakes" in machine created data. Considering the size of datasets used in practice [103], neither of these options is a scalable solution.

In this chapter, we aim to provide a theoretical analysis of how certain operations would avoid collapse in self-consuming loops, without any assumptions on the "moderateness" of synthetic data corruption. We introduce the mathematical abstraction of a *self-correction operation*. This operation maps synthesized data that are sampled from the generative model to data that are better representatives from the target probability distribution that the model is attempting to approximate. Instead of training on a combination of real data and synthesized data, we propose training on a combination of real data and synthesized *and then self-corrected* data. Note that injecting fresh human generated data can be viewed as a special case of this operation.

Our **main theoretical findings** (Theorem 1.3.3):

(1) The self-consuming model with self-correction is *exponentially* more stable than the self-consuming model without any self-correction.

(2) The self-correction procedure guarantees less unwanted variance during self-consuming model training.

In our theoretical study, we assume that correction is *ideal* in order to obtain rigorous performance guarantees. In our empirical study, we evaluate whether the same conclusions hold for *noisy* self-correction functions. We propose to automate this "self-correction" process by relying on programmed expert knowledge rather than a human-in-the-loop, such that the function can be applied at scale. We focus on the human motion synthesis task [42], and implement the self-correction function with a physics simulator-based imitation model [82]. Our **empirical results** confirm that our theoretical findings hold in practice:

(1) As illustrated in Figure 1.1, the self-correcting self-consuming model generates higher-quality human motion than the one without any self-correction.

(2) The self-correction function allows self-consuming loops to avoid collapse even at a high synthetic data to real data ratio (e.g. 100%).

Our theory and experiments suggest that self-correction should stabilize self-consuming model training for any generative modeling task for which there exists a high quality "self-correction" function. We have released all the code associated with this chapter.[1]

1.1 Related Work

1.1.1 Learning Representations with Synthetic Data

Real curated datasets are costly to obtain, so there has been much interest in generating synthetic data as training data for various vision tasks. Azizi et al. [7] demonstrates that text-to-image diffusion models such as Imagen [101] can generate synthetic examples that augment the ImageNet dataset for better image classification. He et al. [47] studies how synthetic data from text-to-image models, when used exclusively, can be used as training data for image recognition tasks. Similarly, Tian et al. [116] finds that using synthetic outputs from a text-to-image model results in contrastive models whose downstream performance rivals that of CLIP [95] on visual recognition tasks, including dense prediction. And the work in Jahanian et al. [55] explored methods for multi-view representation learning by using the latent space of the generative models to generate multiple "views" of the synthetic data. The above works collectively provide evidence that *some* representation learning tasks, when trained on synthetic data from some *given* generative models, yield excellent results.

1.1.2 Training Generative Models on Synthetic Data

Another line of reseach investigates the use of synthetic data for training *generative* models. Shumailov et al. [105] and Martínez et al. [84] show that the use of model generated content in generative model training results in model degradation, likely because self-consuming loops remove low-density areas from the estimated probability manifold. Alemohammad et al. [5] formalize three different kinds of self-consuming generative models: the fully synthetic loop, the synthetic augmentation loop, and the fresh data loop. In all of these loops, they iteratively re-train the model from scratch for every new generation. They empirically find that only the fresh data loop avoids model degradation.

Another recent work [13] considers the problem of iteratively fine-tuning in the context of synthetic augmentation loops. They find that self-consuming augmentation loops do not necessarily collapse, so long as the synthetic augmentation percentage is sufficiently low. The authors use techniques from the field of performative stability [93] to prove the existence of a convergence phenomenon in the space of model parameters. Our work differs from prior work as we conduct analysis on self-consuming generative model training when the synthetic data can be optionally *corrected*. The correction can be performed with a human-in-the-loop, or by incorporating learned or programmed expert knowledge, as explored for natural language [102, 127, 129] and human motion [138, 132]. We validate our theory with a practical self-correcting operations designed for image generation and human motion synthesis tasks.

[1] Project page: https://nategillman.com/sc-sc.html

Algorithm 1 Iterative Fine-tuning of a Generative Model With Correction

Input: $\mathcal{D}_{\text{real}} := \{x_i\}_{i=1}^n$, $\mathcal{A}$, $\mathcal{A}_{\text{ft}}$, π_γ // ground truth data, learning procedure, fine-tuning procedure, correction function
Parameters: T, λ, γ // number of retraining iterations, proportion of generated data, correction strength
$p_{\theta_0} \leftarrow \mathcal{A}(\mathcal{D}_{\text{real}})$ // learn generative model from scratch on true data
for $t = 1$ **to** T **do**
  $\mathcal{D}_{\text{synth}} \leftarrow \{\pi_\gamma(\tilde{x}_i)\}_{i=1}^{\lfloor \lambda \cdot n \rfloor}$, with $\tilde{x}_i \sim p_{\theta_{t-1}}$ // sample $\lfloor \lambda \cdot n \rfloor$ synthetic data points, pass through correction function
  $p_{\theta_t} \leftarrow \mathcal{A}_{\text{ft}}(\mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{synth}}; p_{\theta_{t-1}})$ // fine-tune previous generation using augmented dataset
end for
Return $[p_{\theta_0}, p_{\theta_1}, p_{\theta_2}, \ldots, p_{\theta_T}]$

1.2 Overall Training Procedure

We describe our proposed procedure in concise language in Algorithm 1, and we explain it in more detail here. We train the zero'th generation from scratch on the ground truth dataset $\mathcal{D}_{\text{real}} := \{x_i\}_{i=1}^n$, and we stop training when the model is close to convergence. For all the following generations, we fine-tune the previous generation's latest checkpoint on a combination of the ground truth dataset $\mathcal{D}_{\text{real}}$, as well as $\lfloor \lambda \cdot n \rfloor$ synthetic data points which are generated from the previous generation's latest checkpoint, and then passed through the *correction function* π_γ.

The correction function π_γ is parameterized by the *correction strength* $\gamma \in \mathbb{R}_{\geq 0}$, which controls how much influence the correction function has on the input data points towards increasing a given point's likelihood with respect to the target distribution. The other main hyperparameter $\lambda \in \mathbb{R}_{\geq 0}$ is the *synthetic augmentation percent*, and it controls the ratio of synthetic data to real data in each iteration of fine-tuning. When $\gamma = 0$, we recover iterative re-training with synthetic augmentation considered in [13]. And if we choose the synthetic augmentation percent to be $\lambda = 0$, then each generation simply corresponds to fine-tuning the model on the same dataset that it was trained on initially.

We now use *iterative fine-tuning* interchangeably with the more general term self-consuming loop. We also consider the idealized correction function for our theoretical analysis, and a broader family of practical *correction functions* for different data types.

1.3 Theoretical Analysis

1.3.1 Preliminaries

We mostly follow the notation from [13], except for introducing the correction function π_γ. Let us denote by p_{data} the ground truth probability distribution that we want to train a generative model to estimate. Suppose we have some dataset $\mathcal{D}_{\text{real}} = \{x_i\}_{i=1}^n$ sampled from p_{data}. We write $\hat{p}_{\text{data}} = (1/n) \sum_{i=1}^n \delta_{x_i}$. More generally, we use a hat to denote the empirical distribution over finitely many samples from the corresponding distribution.

Suppose that we have a class of generative models parameterized by $\Theta \subset \mathbb{R}^d$. We denote by p_θ a probability distribution in this class with model parameters $\theta \in \Theta$. We define the optimal model parameters

within this class to be

$$\theta^\star = \arg\max_{\theta' \in \Theta} \mathbb{E}_{x \sim p_{\text{data}}}[\log p_{\theta'}(x)], \tag{1.3.1}$$

where we break ties by minimizing $\|\theta^\star\|$. Typically, such optimal parameters yield a model $p_{\theta^\star}$ which closely approximates the oracle ground truth distribution p_{data}, but doesn't equal it exactly; accordingly, we define the Wasserstein-2 distance between the distributions to be

$$\varepsilon := d_W(p_{\theta^\star}, p_{\text{data}}). \tag{1.3.2}$$

The model weights for the first generation are naturally defined according to the optimization

$$\theta_0^n := \arg\max_{\theta' \in \Theta} [\mathbb{E}_{x \sim \hat{p}_{\text{data}}}[\log p_{\theta'}(x)]]. \tag{1.3.3}$$

This corresponds to training on the finite subset $\mathcal{D}_{\text{real}}$. Next, let us suppose that the model weights from generation t are denoted θ_t^n. We will formalize a procedure for updating these weights for the next generation to obtain θ_{t+1}^n. For this, we need to define our correction function, and then we will use it to define the weight update.

Definition 1.3.1. *For any probability distribution, and for any $\gamma \in \mathbb{R}_{\geq 0}$, we define the* correction of strength γ *of distribution p_θ to be the distribution*

$$\pi_\gamma p_\theta(x) := \frac{p_\theta(x) + \gamma p_{\theta^\star}(x)}{1+\gamma}, \tag{1.3.4}$$

where $p_{\theta^\star}$ is defined in equation 1.3.1. For any augmentation percentage $\lambda \geq 0$, we define the weight update *mapping to be*

$$\begin{aligned} \pi_\gamma \mathcal{G}_\lambda^n(\theta) &:= \operatorname*{local\,argmax}_{\theta' \in \Theta} \hat{\mathcal{H}}(\theta, \theta') \\ &:= \operatorname*{local\,argmax}_{\theta' \in \Theta} \Big[\mathbb{E}_{x \sim \hat{p}_{\text{data}}}[\log p_{\theta'}(x)]] \\ &\qquad + \lambda \mathbb{E}_{x \sim \widehat{\pi_\gamma p_\theta}}[\log p_{\theta'}(x)] \Big], \end{aligned} \tag{1.3.5}$$

where $\hat{p}_{\text{data}}$ and $\widehat{\pi_\gamma p_\theta}$ are empirical distributions of size n and $\lfloor \lambda \cdot n \rfloor$ respectively.

To continue our discussion from before, our iterative weight update is defined as $\theta_{t+1}^n := \pi_\gamma \mathcal{G}_\lambda^n(\theta_t^n)$.

Note that we use an global maximization in equation 1.3.3 when defining the initial parameters θ_0^n, but we use a local maximization when computing our parameter update in equation 1.3.5. This difference is analogous to the differences between how model weights update during initial training, where parameter updates are more global, and during fine-tuning, where parameter updates are more local.

Understanding the correction $\pi_\gamma p_\theta(x)$

For $\gamma = 0$, the correction mapping in equation 1.3.4 simplifies to $\pi_0 p_\theta = p_\theta$, which is just the original distribution; this corresponds to no correction at all. For $\gamma = 1$, it is $\pi_1 p_\theta = (p_\theta + p_{\theta^\star})/2$. And for $\gamma = \infty$, it

is $\pi_\infty p_\theta = p_{\theta^\star}$, which corresponds to the optimal distribution. So as γ increases from 0 to ∞, the distribution $\pi_\gamma p_\theta$ has a likelihood profile that matches p_θ less, and $p_{\theta^\star}$ more. As $p_{\theta^\star}$ is the optimal model in our generative model class, this means that as γ increases from 0 to ∞, we have that $\pi_\gamma p_\theta(x)$ is a PDF which better represents the target likelihood that we want to estimate through training the generative model.

In our theoretical formulation, we consider correction functions that correct the probability distribution p_θ, rather than the more intuitive (and practical) case of a correction function that corrects individual points that the distribution is defined over. In Appendix A.3, we specify sufficient conditions under which a pointwise correction function is guaranteed to correspond to a distribution-wise correction function of the same form as those which we consider in our theoretical study and therefore can enjoy the theoretical stability guarantees we prove. We also provide a concrete example of a projection function, in the Gaussian case, which provably satisfies those conditions. We conduct a series of experiments on this toy example in Section 1.4.

Understanding the weight update $\pi_\gamma \mathcal{G}^n_\lambda(\theta)$

The weight update $\pi_\gamma G^n_\lambda(\theta)$ in equation 1.3.5 is a formalization of the intended output of fine-tuning p_θ on $\mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{synth}}$, where $\mathcal{D}_{\text{real}} = \{x_i\}_{i=1}^n$ is the ground truth dataset of size n, and $\mathcal{D}_{\text{synth}} = \{\tilde{x}_i : \tilde{x}_i \sim \widehat{\pi_\gamma p_\theta}\}_{i=1}^{\lfloor \lambda \cdot n \rfloor}$ is the synthesized-and-corrected dataset of size $\lfloor \lambda \cdot n \rfloor$. In other words, in an ideal run of stochastic gradient descent fine-tuning, the model weights θ should update to $\pi_\gamma \mathcal{G}^n_\lambda(\theta)$, as defined in equation 1.3.5, when trained on $\mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{synth}}$.

Intuitively, the weight update $\theta \mapsto \pi_\gamma \mathcal{G}^n_\lambda(\theta)$ avoids the loss of variance in the generated data by ensuring that at each step, the model is trained on synthetic data which is likelier to have been sampled from the diverse target distribution. This positive phenomenon is more pronounced when the correction strength γ is larger.

1.3.2 Assumptions

In order to prove our main result, we need some regularity assumptions about the learning procedure. Informally speaking, we will assume that the class of generative models that we consider is smoothly parameterized by its model weights; the loss landscape is concave near the ideal model weights; and the class of generative models does an increasingly good job approximating the target data distribution as the dataset size increases. We formally quantify and state these hypotheses in Assumption 1.3.2.

Assumption 1.3.2. The following are true.

1. There exists some $L > 0$ such that, for all θ sufficiently close to $\theta^\star$, the mapping $x \mapsto \nabla^2_\theta \log p_\theta(x)$ is L-Lipschitz.
2. The mapping $\theta \mapsto \mathbb{E}_{x \sim p_{\text{data}}}[\log p_\theta(x)]$ is continuously twice differentiable locally around $\theta^\star$, and there exists some $\alpha > 0$ such that $\mathbb{E}_{x \sim p_{\text{data}}}\left[\nabla^2_\theta \log p_\theta(x)\right]|_{\theta^\star} \preceq -\alpha I_d \prec 0$.
3. There exist $a, b, \varepsilon_{\text{OPT}} \geq 0$ and a neighborhood U of $\theta^\star$ such that, for any $\delta \in (0, 1)$, with probability

$1-\delta$ over the samplings, we have[2]

$$\|\pi_\gamma \mathcal{G}_\lambda^n(\theta) - \pi_\gamma \mathcal{G}_\lambda^\infty(\theta)\| \leq \varepsilon_{\text{OPT}} + \frac{a}{\sqrt{n}}\sqrt{\log \frac{b}{\delta}}. \tag{1.3.6}$$

for all $\theta \in U$ and $n \in \mathbb{N}$. Denote this bound by $\tau_n(\delta)$.

In Assumption 1.3.2 (2), the notation "$\preceq$" corresponds to the Loewner order on symmetric matrices: we write that $A \preceq B$ if $B - A$ is positive semi-definite, and $A \prec B$ if $B - A$ is positive definite. In particular, Assumption 1.3.2 (2) implies that the matrix $\mathbb{E}_{x\sim p_{\text{data}}}\left[\nabla_\theta^2 \log p_\theta(x)\right]|_{\theta^\star}$ is negative definite, and its largest eigenvalue is at most $-\alpha$. And Assumption 1.3.2 (3) mirrors the main assumption in [13]; it is motivated by generalization bounds in deep learning, see e.g. [56, 57]. The interested reader can consult Appendix A.2 for more details on this assumption.

1.3.3 Iterative Fine-Tuning with Correction

We now have the language to state our main result, which essentially says that if the initial parameters θ_0 are sufficiently close to the optimal model parameters $\theta^\star$, and if the augmentation percentage λ is sufficiently small, then under iterative fine-tuning with correction, we can expect our subsequent model parameters to stay close to $\theta^\star$.

Theorem 1.3.3 (Stability of Iterative Fine-Tuning with Correction)**.** *Fix an augmentation percentage $\lambda \in \mathbb{R}_{>0}$ and a correction strength $\gamma \in \mathbb{R}_{\geq 0}$. Suppose we have an iterative fine-tuning procedure defined by the rule $\theta_{t+1}^n = \pi_\gamma \mathcal{G}_\lambda^n(\theta_t^n)$, and suppose that Assumption 1.3.2 holds. Define the constant*

$$\rho(\lambda) := \rho(\lambda; \alpha, \varepsilon, L) := \frac{\lambda(\alpha + \varepsilon L)}{\alpha - \lambda(\alpha + \varepsilon L)}$$

and fix any $\delta \in (0,1)$. If θ_0 is sufficiently close to $\theta^\star$, and if $\lambda\left(1 + \frac{\varepsilon L}{\alpha}\right) < \frac{1+\gamma}{2+\gamma}$, then $\rho(\lambda)/(1+\gamma) < 1$, and it follows that the stability estimate holds with probability $1-\delta$:

$$\begin{aligned} &\|\theta_t^n - \theta^\star\| \\ &\quad \leq \tau_n(\delta/t) \sum_{i=0}^{t} \left(\frac{\rho(\lambda)}{1+\gamma}\right)^i + \left(\frac{\rho(\lambda)}{1+\gamma}\right)^t \|\theta_0^n - \theta^\star\| \end{aligned} \tag{1.3.7}$$

for all $t > 0$.

We prove Theorem 1.3.3 in Appendix A.1.

Remark 1.3.4. *If we apply Theorem 1.3.3 with correction strength $\gamma = 0$, then the iterative fine-tuning procedure trains successively on a combination of raw synthetic data that has not been corrected using a correction function and ground truth data. This is exactly the case considered in [13]. Accordingly, the bound in equation 1.3.7, applied with $\gamma = 0$, exactly recovers their result.*

[2] The map $\pi_\gamma G_\lambda^\infty$ is defined similarly to $\pi_\gamma G_\lambda^n$ in equation 1.3.5, but with $\hat{p}_{\text{data}}$ replaced with p_{data}, and with $\widehat{\pi_\gamma p_\theta}$ replaced with $\pi_\gamma p_\theta$. See Appendix A.1 for more details. This estimate is identical to the analogous Assumption 3 used in [13], with the only difference being it is applied to our iterative fine-tuning update function. See Appendix A.2 for further discussion.

Corollary 1.3.5. *Under the assumptions from Theorem 1.3.3, iterative fine-tuning with any amount of correction outperforms iterative fine-tuning without correction–in the sense that it is exponentially more stable, and it results in better model weights.*

Proof of Corollary 1.3.5. We apply Theorem 1.3.3 with $\gamma = 0$, which corresponds to no correction, as well as with $\gamma > 0$, which corresponds to any amount of correction. For any $\gamma > 0$, we notice that the RHS of equation 1.3.7 is strictly smaller than when $\gamma = 0$. This guarantees better stability as $t \to \infty$, as well as model weights θ_t^n closer to $\theta^\star$. □

Example 1.3.6. If we apply Theorem 1.3.3 with correction strength $\gamma \to \infty$, then the bound equation 1.3.7 in Theorem 1.3.3 limits to $\tau_n(\delta/t)$. This implies that the practical iterate θ_t^n approaches the ideal model paramaters, and is at worst some constant away, that depends on error from the optimization procedure, as well as statistical error from using finitely many ground truth data samples n.

Note that Theorem 1.3.3 relies on the assumption that the initial model parameters θ_0 are sufficiently close to the ideal model parameters $\theta^\star$, and also that the augmentation percentage λ is sufficiently small. We hypothesize that these assumptions can be relaxed in the case where a correction function participates in the iterative fine-tuning procedure–intuitively, the correction function should compensate for errors that arise from θ_0^n being worse, as well as errors that arise from incorporating more synthetic data. We frame this in the following conjecture.

Conjecture 1.3.7. *In the case of iterative fine-tuning with correction, we may relax how close the initial model parameters θ_0^n need to be to the optimal model parameters $\theta^\star$, as well as choose a larger synthetic augmentation percentage λ, while still retaining the improved stability estimate equation 1.3.7.*

We provide empirical evidence for Conjecture 1.3.7 in Section 1.6 on the human motion synthesis task. In fact, Theorem 1.3.3 represents partial progress towards this conjecture. Namely, according to Theorem 1.3.3, for large correction strength γ, we can effectively choose a synthetic augmentation percentage that is twice as large as we would be able to without any correction, and still be able to meet the assumptions of the theorem. This is because $\lim_{\gamma \to \infty} \frac{1+\gamma}{2+\gamma} = 1$, which is twice as large as the bound when $\gamma = 0$.

1.4 Toy Example: Gaussian

We first assume oracle knowledge of the ground truth distribution, and use a toy example to directly demonstrate the impact of the correction strength γ on model performance and stability as stated in Theorem 1.3.3 and Corollary 1.3.5. Our ground truth distribution is a 2-dimensional isotropic Gaussian centered at the origin, i.e., $\theta^\star = ((0,0), I_2)$, and our correction is "distribution-wise" in this idealized scenario. We consider the more practical setting, where we don't have oracle knowledge of the target distribution a priori, and where the data correction is "point-wise", in the empirical studies in the following two sections. Further, in Appendix A.3, we show that, in theory, sufficiently well-behaved pointwise correction functions indeed correspond to distribution-wise correction functions.

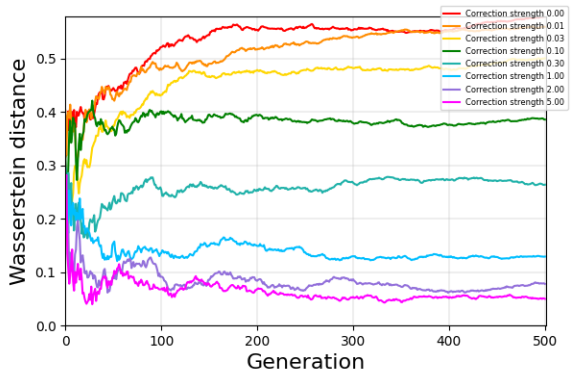


Figure 1.2: Empirical results from our Gaussian toy example. The graph demonstrates that increasing the correction strength γ, with a fixed augmentation ratio of $\lambda = 0.5$, improves performance and stability after self-consuming iterations.

Concretely, our ground truth dataset contains 50 points sampled from the target distribution, which are used to estimate $\theta_0^{50} = (\mu_0, \Sigma_0) \in \mathbb{R}^6$. We fix our synthetic augmentation percentage to be $\lambda = 0.5$, and inductively synthesize a new dataset $\mathcal{D}_{\text{synth}} = \{y_i \sim \mathcal{N}(\mu_t, \Sigma_t)\}_{i=1}^{25}$. We implement a correction function to map $\mathcal{D}_{\text{synth}}$, which was sampled from $p_{\theta_t^{50}}$, to a dataset $\mathcal{D}_{\text{corrected}}$, which is likelier to have been sampled from the target density $p_{\theta^\star}$. We do this by sampling $\mathcal{D}_{\text{corrected}}$ from the *middle density* corresponding to a given correction strength γ:

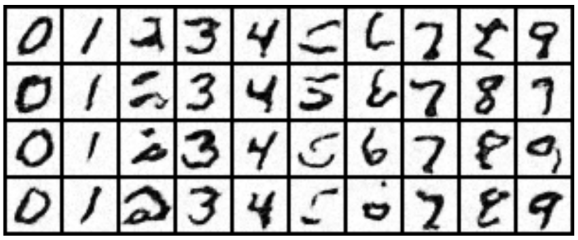
$$\pi_\gamma \hat{p}_{\theta_t^{50}}(x) := \frac{\hat{p}_{\theta_t^{50}}(x) + \gamma p_{\theta^\star}(x)}{1+\gamma}, \tag{1.4.1}$$

where $\hat{p}_{\theta_t^{50}}$ is the empirical PDF obtained from $\mathcal{D}_{\text{synth}}$.

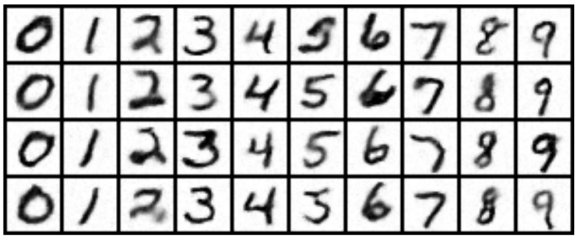
We logarithmically accrue synthetic data points to simulate the case of fine-tuning. We obtain the updated model parameters θ_{t+1}^{50} by computing the sample mean and covariance on this augmented dataset. In Figure 1.2, we present the Wasserstein distance between the origin-centered isotropic Gaussian target distribution and the distribution defined by the parameters θ_t^{50} at each iteration t. Our results illustrate how increasing the correction strength γ adds stability and results in convergence near better Wasserstein scores in later generations, in accordance with Theorem 1.3.3. The experiments also demonstrate how even a very small increase in γ can improve performance over the baseline, in accordance with our claim of exponential improvement in Corollary 1.3.5.

1.5 Toy Example: MNIST

Our proof uses the optimal target PDF $p_{\theta^\star}$ to define the correction function π_γ. This is empirically validated by the Gaussian toy experiment, which assumes knowing the true target distribution. In practice, the correction function only depends on the ability to map synthesized data to data which is *likelier* to have been sampled from the ground truth distribution. Crucially, *this can be achieved without having a complete description of the target distribution.* For example, with our human motion experiments, we will demonstrate that point-wise



Iterative Fine-tuning



Iterative Fine-tuning with Self-Correction

Figure 1.3: Empirical results from our MNIST toy example. These synthesized images demonstrate that after 50 self-consuming iterations at 150% augmentation percentage, the model which is trained using **iterative fine-tuning with self-correction** is able to generate higher quality samples than the model trained using **iterative fine-tuning** without any self-correction.

correction based on the laws of physics is one proxy approach to make a sample more likely, without knowing the true target distribution.

One has the freedom to explore alternative approaches to data correction for more general data types, such as images. For example, one simple heuristic is to identify the "anchor" or "exemplar" images, which are intuitively representative and likely. The correction function can then be implemented as mapping or morphing synthesized data towards its nearest anchor, to make the synthesized data more representative and likely. In this section, we implement this approach on MNIST and study its performance.

For our MNIST [66] experiments, we train a diffusion model [49] for class-conditional image generation, using a train split of size $n = 12000$. For our iterative fine-tuning experiments, we train the model for 20 epochs, then synthesize $\lambda \cdot 12000/10$ images for each digit, and then augment the ground truth dataset with these to train on for the next generation; every following generation follows the same procedure, but only trains for a single epoch. We vary our experiments over augmentation percentages $\lambda \in \{0.2, 0.5, 1.0, 1.5\}$. To define our self-correction operation, we first compute K-means clusters over the training split for each digit. Our iterative fine-tuning with self-correction experiments use the same setup described above, except instead of training on the synthesized images, we train on the synthesized and then corrected images, where "correcting" an image means finding the nearest centroid in the K centroids for that digit that we computed at the start of training. We swept the values $K \in \{1, 2, 4, \ldots, 1024\}$, and we found that any reasonably large K results in the same general trend where self-correction improves the metrics and stability. We report our results for $K = 16$, which performs the best.

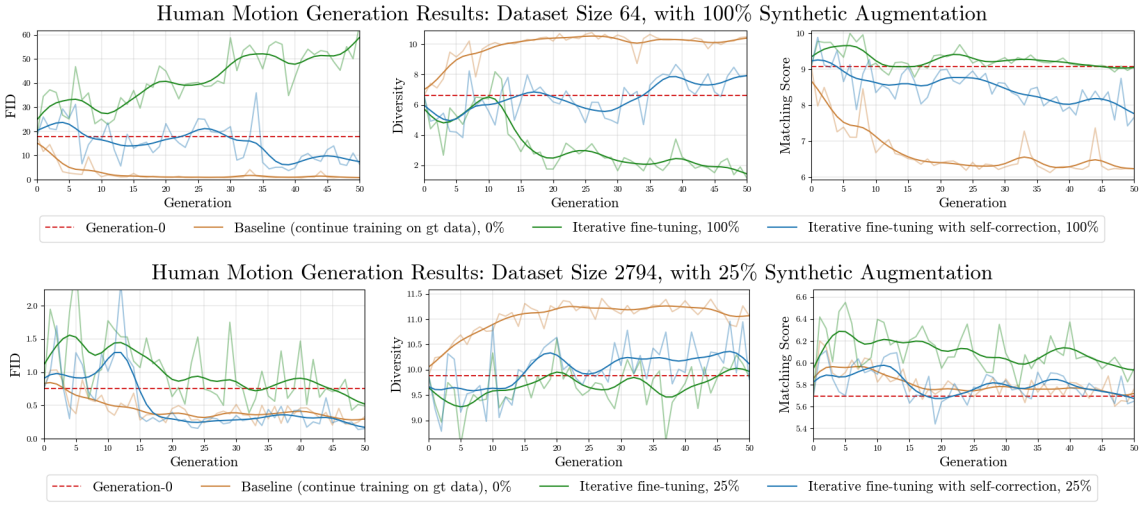


Figure 1.4: Results from our human motion experiments on iterative fine-tuning with self-correction. These graphs show evaluation metrics for the last checkpoint for every generation. This is the checkpoint used for sampling in the iterative fine-tuning experiments, and it is also the checkpoint where training is resumed with this new partially synthesized dataset. We can see that **with self-correction**, the iterative fine-tuning procedure more stably converges to a better FID score, and more quickly. When the dataset size is smaller ($n = 64$, above) we can see that iterative fine-tuning **with no self-correction** has a flat Matching score, as well as diverging FID and Diversity scores, indicating model collapse. And when the dataset size is larger ($n = 2794$, below), there is less collapse for iterative fine-tuning with no self-correction, although the variance of the FID score is worse, as is the average FID across generations. In both cases, we see that iterative fine-tuning **with self-correction** outperforms iterative fine-tuning **with no self-correction**, and is competitive with the **baseline** after many generations.

We present images synthesized using our trained models in Figure 1.3. These synthesized images demonstrate that iterative fine-tuning eventually generates many low quality and illegible digits, and this problem is solved by applying our self-correction operation. Further experiment details, including graphs of the FID metrics for each generation that provide rigorous evidence for this trend across augmentation percentages, can be found in Appendix A.4. Our empirical results demonstrate that applying self-correction improves performance during iterative fine-tuning for our MNIST image generation task across self-consuming generations, and this relative performance is amplified when the augmentation percentage is larger. The behavior that we observe is consistent with our theoretical results in Section 1.3, as well as our human motion experiments in Section 1.6.

1.6 Human Motion Synthesis

Theorem 1.3.3 states that, in theory, iterative fine-tuning with correction should be more stable than iterative fine-tuning without correction. Crucially, the stability estimates that we prove rely on the dataset size, the synthetic augmentation percentage, how expressible the generative model class is, and having an idealized

correction function. To validate how our theory works beyond toy examples, we conduct a case study on human motion synthesis with diffusion models [115]. We believe this is a natural setting to test our iterative fine-tuning with correction framework, because synthesizing natural motions is a challenging problem, but there is a natural and intuitive way to automatically correct them at scale–namely, using a physics simulator.

1.6.1 Generative Model

For our generative model, we use the Human Motion Diffusion Model (MDM) [115]. This is a classifier-free diffusion-based generative model for the text-to-motion generation task, where the model receives as input a description of a motion sequence (e.g. "get down on all fours and crawl across the floor"), and outputs a sequence of skeleton poses which attempt to embody that prompt. Synthesizing human motion is challenging not only for the diverse and compositional text prompts, but also due to failure of physics obeying-ness (e.g. feet skating, floating, penetrating a surface), which is not explicitly enforced by deep generative models.

1.6.2 Physics Simulator as Self-Correction Function

For our self-correction function, we use Universal Humanoid Control (UHC) [82], which is an imitation policy that operates inside the MuJoCo physics simulator [117]. Given an input sequence of humanoid skeleton poses, UHC attempts to imitate the motion sequence, constrained by the laws of physics imposed by the physics simulator, and it outputs a new motion sequence that is the closest possible approximation it can replace it with. For example, if an input motion sequence violates the laws of physics by having a foot penetrate through the floor, then the motion sequence output by UHC will attempt to remove that physically impossible artifact while maintaining the semantic integrity of the original input motion. We use VPoser [92] and SMPL [78] to translate joint representations between the human motion generator and the physics simulator.

The physics simulator allows us to self-correct a synthesized motion automatically. Our underlying assumption is that by enforcing the physics obeying-ness (via the simulator) and closeness to the synthesized motion (via the imitation objective), the self-correction function would act as similar as an idealized corrector as possible.

1.6.3 Experimental setup

We preprocess the MoVi [36] subset of HumanML3D [42] using the official code implementation of HumanML3D. We filter out movements involving interactions with chairs, as UHC by default does not handle human-object interactions. We take as our train split the train split from HumanML3D, intersected with our filtered subset of MoVi, and likewise for the test split. This procedure yields a train set of size $n = 2794$ and a test set of size 546. We further randomly select a smaller training set of $n \in \{64, 128, 256\}$ examples, to simulate the more challenging scenario when the initial generative model is sub-optimal (due to data scarcity). The smaller data also enables us to explore larger synthetic augmentation percentage due to compute constraints. From here, the iterative re-training procedure follows Algorithm 1. We spell it out in this concrete experimental setup.

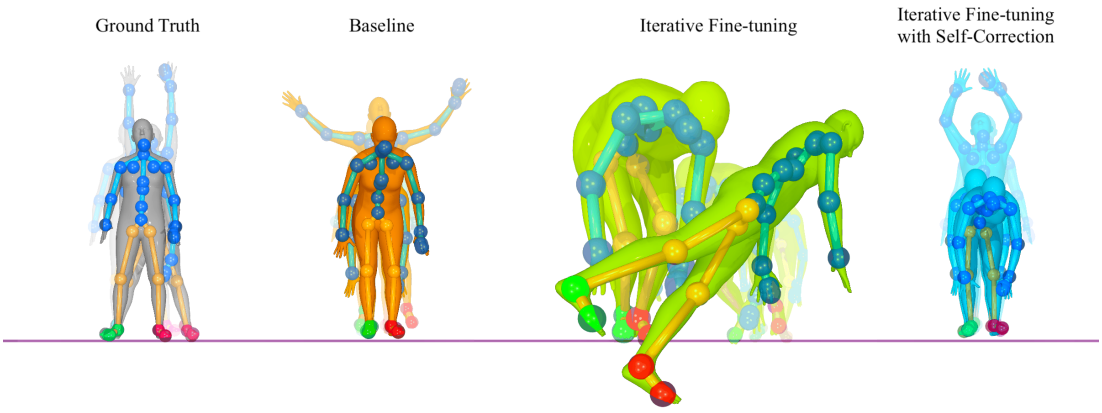


Figure 1.5: How does the self-correction operation affect iterative fine-tuning, qualitatively? Here we present some visualizations. The prompt which describes the ground truth motion, and which we use to generate the three other motions, is: *"a person stands with feet wide, stretches both hands up over his head and then swings down by the waist and hangs arms down before standing up"*. We can see that the **iterative fine-tuning model** produces a motion where the human moves closer to the camera than the others; this is evidence of model collapse, as moving feet is irrelevant to the prompt. Additionally, this motion produces single frames that suddenly snap to a physically impossible position–note the leg penetration through the ground plane. These negative artifacts do not exist in the motions synthesized from the **ground truth**, **baseline model**, or **iterative fine-tuning with self-correction model**. Lastly, we note that the iterative fine-tuning motion depicted here is semantically similar to crawling. We observe in our experiments with smaller dataset sizes that the iterative fine-tuning model generates less diverse outputs than the baseline model and the iterative fine-tuning with self-correction model, and that this crawling pattern appears more often in the latter. Each snapshot is taken at exactly frame 105 of their respective videos. The two motions on the right come from models that were iteratively fine-tuned for 50 generations, with a train set of size $n = 64$, and a synthetic augmentation percentage of 25%. For all pictures of the human, the camera is fixed at the same position, and for consistency the image is not resized.

We first train on the ground truth train split until the model is nearly converged, using all the default hyperparameters from MDM. We evaluate and save this last checkpoint from generation 0. From here, for each generation $t \in \{1, 2, \ldots, 50\}$, we run three sets of experiments.

A. *Baseline*: fine-tune the latest checkpoint from generation $t-1$ for m batches on ground truth dataset $\mathcal{D}_{\text{real}}$.

B. *Iterative fine-tuning:* fine-tune the latest checkpoint from generation $t-1$ on $\mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{synth},t-1}$ for m batches. Here, $\mathcal{D}_{\text{synth},t-1}$ is a synthetic dataset of size $\lfloor \lambda \cdot n \rfloor$ generated from the checkpoint for generation $t-1$, using randomly chosen prompts from the train split.

C. *Iterative fine-tuning with self-correction:* fine-tune the latest checkpoint from generation $t-1$ on $D_{\text{real}} \cup \text{UHC}(\mathcal{D}_{\text{synth},t-1})$ for m batches. Here, $\text{UHC}(\mathcal{D}_{\text{synth},t-1})$ denotes a synthetic dataset of size $\lfloor \lambda \cdot n \rfloor$ generated from the latest checkpoint for generation $t-1$, using randomly chosen prompts from the train split, which is then corrected by UHC.

We experiment with synthetic augmentation percentages $\lambda \in \{0.05, 0.10, 0.15, 0.20, 0.25\}$ on the larger

dataset; we set the number of batches seen during generation 0 to be 3125, and the number of batches seen for each later generation to be $m = 625$. Separately, we experiment with synthetic augmentation percentages $\lambda \in \{0.25, 0.50, 0.75, 1.00\}$ on the smaller datasets; we set the number of batches seen during generation 0 to be $78 * k$ for dataset size $64 * k$, and the number of batches seen for each later generation $t > 0$ to be $m = 16$. We choose to control how many data points the model sees across each generation, rather than controlling some other quantity like the number of epochs, as this allows each experiment to compare against its baseline in a controlled way, which in turn allows them to compare against each other in a controlled way.

We compute every evaluation one time for each checkpoint using the evaluation script provided in the original MDM codebase. Regardless of the train split size, we perform sampling for evaluation using all 546 motion sequences from the test split, since the FID score is sensitive to generated dataset size. We use the same hyperparameters as those used for MDM, including batch size 64, AdamW [80] with learning rate $1e-4$, and classifier-free guidance parameter 2.5. And for UHC we used the `uhc_explicit` model for imitation.

1.6.4 Quantitative Analysis of Results

For each of these experiments we report the metrics from MDM, as used by [42]: FID measures how similar the distribution of generated motions is to the ground truth distribution; Diversity measures the variance of the generated motions; and Matching Score measure how well the generated motions embody the given text prompt. In Figure 1.4 we present results from experiments on our 64-size dataset with 100% synthetic augmentation, as well as our 2794-size dataset with 25% synthetic augmentation.

Our experimental results confirm our theoretical results, that iterative fine-tuning with self-correction outperforms iterative fine-tuning without self-correction, in the sense that the graphs are generally more stable across generations, and approach better evaluation metric values. In particular, Theorem 1.3.3 and Corollary 1.3.5 claim that any amount of idealized self-correction will improve the stability bound during iterative fine-tuning. Our results in Figure 1.4 demonstrate that the FID score is lower and more stable across generations when applying self-correction, and generally higher and less stable than the baseline, where there is no self-consuming training at all. We conduct experiments across multiple seeds, and we find empirically that this general phenomenon holds consistently, where the self-correction technique consistently yields improved training dynamics over iterative fine-tuning with no correction. Graphs from these runs can be found in Appendix A.7.

Our experimental results also provide empirical evidence for Conjecture 1.3.7. Observe that in the baseline experiments in Figure 1.4, the FID score decreases across generations, which indicates that the initial model parameters θ_0^n are not that close to the optimal model parameters $\theta^\star$; additionally, the augmentation percentages considered in the graph are 25% and 100%. Conjecture 1.3.7 claims that performing self-correction during iterative fine-tuning improves performance, even when the initial model weights are sub-optimal and simultaneously the synthetic augmentation percentage is large. This claim is confirmed by Figure 1.4. We direct the curious reader to Appendix A.6, where we present graphs for all of the above listed training set sizes and augmentation percentages, providing additional empirical evidence for Theorem 1.3.3, Corollary 1.3.5, and Conjecture 1.3.7.

1.6.5 Qualitative Analysis of Results

We visually inspect the generated human motion sequences in order to analyze what concrete effect the self-correction has on iterative fine-tuning. We find that the correctness and diversity of synthesized motions are improved by the self-correction procedure, in agreement with our quantitative analysis in Subsection 1.6.4. We present snapshots of our synthesized motions in Figure 1.5, and we analyze the motions in the caption. In short, we find that physics-disobeying artifacts such as floor penetration or floating become more pronounced without the self-correction. We also find that in the model without self-correction, the humanoid sometimes performs movements completely unrelated to the prompt; our model with self-correction fixes these negative phenomena. We direct the curious reader to Appendix A.5, where we present more examples from our qualitative analysis, as well as our project webpage, where we provide side-by-side video comparisons.

1.7 Conclusion

This chapter investigates the learning of generative models when the training data includes machine-generated contents. We investigate how self-correction functions, which automatically correct synthesized data points to be more likely under the true data distribution, can stabilize self-consuming generative model training. Our theoretical results show that self-correction leads to exponentially more stable model training and smaller variance, which we illustrate with a Gaussian toy example. We then demonstrate how physics simulators can serve as a self-correction function for the challenging human motion synthesis task, where models trained with our self-correcting self-consuming loops generate higher quality motions, and manage to avoid collapse even at a high synthetic data to real data ratio. Future work includes exploring self-correcting functions for more diverse applications, such as language modeling and text-to-image generation, and investigating when self-consuming training may lead to overall better generative models.

Chapter 2

Fourier Head: Helping Large Language Models Learn Complex Probability Distributions

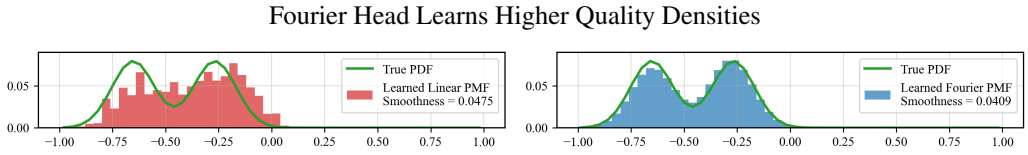


Figure 2.1: We task an MLP with learning to approximate a continuous bimodal density using a categorical distribution and a cross-entropy objective. We observe that a standard linear head fails to distinguish between the two modes, and overfits to high-frequency noise in the training set. In contrast, our proposed Fourier head learns a smoother, more accurate categorical distribution.

Human language can be viewed as a discretization for a *continuous*, often *probabilistic* representation of the world that is construed in our mind [109]. 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 [77], decision making with continuous or discrete actions [21], future anticipation and time-series forecasting [6], or simply drawing random numbers given a probability distribution [51].

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 [6]. 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 the Fourier head is forced to approximate the categorical distributions using a finite number of frequencies. 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 builds upon the Fourier Basis Density Model [27].

Our main contributions are as follows.

Contribution #1: We reveal the underlying principle on the trade-off between the Fourier head's expressive power and the "smoothness" of the predicted distributions. We prove a theorem which demonstrates a scaling law for the Fourier head. Namely, as we increase the quantity of Fourier coefficients learned by the Fourier head, the layer can model increasingly complicated distributions; however, the Fourier head will necessarily fit to more high-frequency noise, thereby outputting categorical distributions which are less smooth.

Contribution #2: We propose a practical implementation of the Fourier head capable of sequential prediction tasks by modeling complex multi-modal distributions. Additionally, 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.

Contribution #3: 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 agent, we improve returns across four benchmark games by as much as 377%. 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.

2.1 Fourier Head

2.1.1 Fourier Head: Motivation

When practitioners apply LLMs to model complex probability distributions over non-linguistic tokens, 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:

Example 1: The Decision Transformer [21] 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 decoder-only transformer architecture.

Example 2: The Chronos time series foundation model [6] 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.** We consider the very flexible class of truncated Fourier series with N frequencies. These are functions of the form

$$f(x) = a_0 + \sum_{k=1}^{N} \big(a_k \cos(k\pi x) + b_k \sin(k\pi x)\big). \tag{2.1.1}$$

Fourier series are a classical tool for solving quantitative problems [110] because functions like Equation 2.1.1 are universal function approximators, with the approximation improving as N increases.

2.1.2 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 2. The Fourier head is constructed using the Fourier Basis Density Model from [27]. 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 [27]. We direct the curious reader to Appendix A.10.1 for a low-dimensional demonstration of the Fourier head in action.

Algorithm 2 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

INPUT $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$

$(\alpha_0, \beta_0, \ldots, \alpha_N, \beta_N) \leftarrow Ax$

$a_k \leftarrow \alpha_k + i\beta_k \in \mathbb{C}$, for every $k = 0, \ldots, N$ // compute autocorrelation coefficients

$c_k \leftarrow \sum_{\ell=0}^{N-k} a_\ell a_{\ell+k}^* \in \mathbb{C}$, for every $k = 0, \ldots, N$ // compute Fourier coefficients

$p(z) = \frac{1}{2} + \Re\left(\sum_{k=1}^{N} \frac{c_k}{\Re(c_0)} \exp(ik\pi z)\right)$ // define Fourier PDF over $[-1, 1]$

$b_k \leftarrow -1 + \frac{1+2k}{m}$, for every $k = 0, \ldots, m-1$ // define m bin centerpoints

$y_k \leftarrow \frac{p(b_k)}{\sum_{j=0}^{m-1} p(b_j)}$, for every $k = 0, \ldots, m-1$ // evaluate PDF at m bin centerpoints

OUTPUT $(y_1, \ldots y_m) \in \mathbb{R}^m$ // by design, $\sum_{k=1}^{m} y_k = 1$ and each $y_k \geq 0$

2.1.3 Fourier Head: Considerations for Training

We highlight the main design choices of the Fourier head so that users may apply it most effectively.

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 $p_x(z)$ defined over $z \in [-1, 1]$. 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 2.2.3 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 [27] 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 2, 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 data binning can impact performance. As discussed, the Fourier head should only be applied when nearby bins are 'similar' in some sense, requiring a semantically meaningful bin ordering. When bins represent quantized numerical values over a continuous latent space, a 'mixed-precision' binning strategy can improve performance. For example, to model values in $[-15, 15]$ with most data in $[-1, 10]$, allocating more bins to the dense interval improves performance. Given m total bins, a hyperparameter $d \in [0, 1)$ controls allocation, with $\lfloor d \cdot m \rfloor$ bins for the sparse interval and the rest for the dense range (estimated from training data). Fourier theory supports this approach, as increasing precision in dense regions de-localizes the quantized data distribution, localizing the Fourier spectrum. This accelerates higher frequency decay, enabling effective learning with lower-frequency Fourier heads. Separately, we note that [27] suggests re-parameterizing the periodic domain to the real line, though we do not use this in our work.

Weight initialization: The learned parameters for the Fourier head consist of the learned linear layer which extracts autocorrelation parameters a_k. In PyTorch, linear layers use He initialization [46] by default, which ensures that the linear layer outputs values close to zero in expectation. Similarly, initializing the Fourier densities to be uniform $p(z) \approx 1/2$ improves learning dynamics. 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.

2.2 Theoretical Analysis of Fourier Head

2.2.1 "Smoothness": A Metric for High Frequency Content

In this subsection we propose a smoothness metric which inputs a categorical distribution $y = (y_1, \ldots, y_m) \in \mathbb{R}^m$, 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 2.2.1. *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 2.2.1.

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 this chapter. 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

$$s(f) := \int_0^\infty \int_a^b \alpha_\sigma D[f(x), g_\sigma(x) * f(x)] dx d\sigma. \tag{2.2.1}$$

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 2.2.1 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 tractably quantify their smoothness. Accordingly, we define a specific case of this as follows.

Definition 2.2.2 (Smoothness metric for categorical distributions)**.** *Suppose $y = (y_1, \ldots, y_m) \in \mathbb{R}^m$ is a categorical distribution, so every $y_k \geq 0$ and $\sum_{k=1}^m y_k = 1$. Denote by $g_\sigma \in \mathbb{R}^{2m-1}$ the discrete Gaussian kernel of standard deviation σ and radius $m-1$. Define the weights $\alpha_\sigma = 6/\pi^2\sigma^2$. Then we define the **smoothness** of y to be the constant*

$$s(y) := \sum_{\sigma=1}^{\infty} \alpha_\sigma \|y - g_\sigma * y\|_2. \tag{2.2.2}$$

We direct the curious reader to Appendix A.9, where we conduct additional experiments to justify this choice of smoothness metric for our experiments.

2.2.2 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 2.2.3. *(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. Suppose that there exists some $t \geq 2$ such that the Fourier coefficients of f_x decay on the order of $1/k^t$. Denote by $f_{x,N}$ the truncation of f_x to its first N frequencies, denote by $\vec{b} \in \mathbb{R}^m$ the m bin centerpoints in $[-1, 1]$, and denote by $y^{(N)} = f_{x,N}(\vec{b})/(f_{x,N}(b_0) + \cdots + f_{x,N}(b_{m-1})) \in \mathbb{R}^m$ the discretization of $f_{x,N}$ into m bins. Then, there exist constants $C_1, C_2 > 0$ such that*

$$s(y^{(N)}) = C_1 - \frac{C_2}{N^{2t-1}} + O(1/N^{2t}). \tag{2.2.3}$$

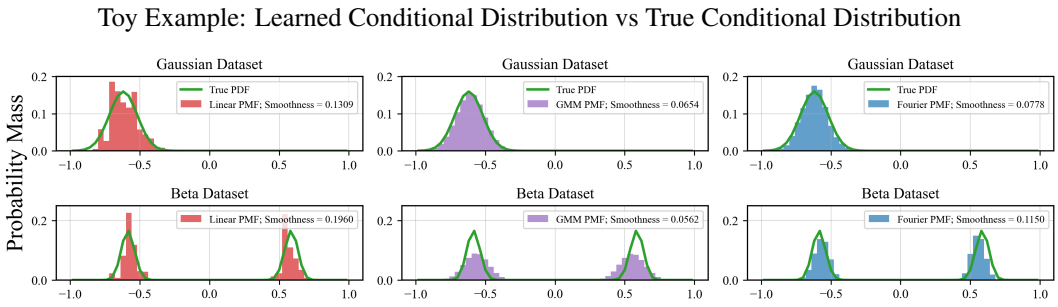


Figure 2.2: Comparison between the PMFs learned by the linear head, GMM head, and the Fourier head, for two of the datasets in the toy example–Gaussian and Beta. (The GMM dataset is in Figure 2.1.) We observe that the Fourier head learns a smoother categorical distribution than the linear head over its predicted values. Furthermore, the Fourier head better fits the true conditional PDF; this is reflected in the KL divergence and smoothness metrics.

Note that the smoothness scaling law asymptotic in Equation 2.2.3 shows that as N increases, so does $s(y^{(N)})$. Further, note that if the Fourier spectrum of the underlying distribution decays quicker (controlled by t) then the rate at which smoothness degrades is slower; this is because if what we are learning has little high frequency content, then increasing the frequencies shouldn't affect the smoothness of the learned distribution very much. In part (2), our assumption that the Fourier coefficients decay at least quadratically is reasonable since if f_x is at least twice continuously differentiable, we already know its Fourier coefficients corresponding to the k-th frequency are in $O(1/k^2)$ [110, Ch.2, Cor. 2.4]. Our Fourier weight decay regularization helps toward ensuring that this condition is met in practice as well. We include a full proof in Appendix A.8.

2.3 Toy Examples

2.3.1 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 A.10.2.

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 the interval $[-1, 1]$ 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

	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 2.1: 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.

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.[1] 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 2.1. Figure 2.2 presents sample visualizations of the learned conditional distributions alongside the true densities. And in Appendix A.10.2, 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 2.2.3, as it demonstrates that for all datasets, as frequency increases, 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 A.10.2. Additionally, in Figure A21 (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.

2.3.2 Are LLMs Random Number Generators?

Suppose that we query an LLM with the following prompt, repeatedly: *"The following is a list of normally distributed random numbers in the interval [-1, 1] with mean 0.55 and std 0.10: 0.57, 0.36, "*. Would the

[1] Note that we also demonstrate the possibility of training a *continuous* version of the Fourier head, using a maximum-likelihood based objective. Accordingly, we carry out experiments in the continuous domain analogous to those we did in the quantized domain; for more details, see Appendix A.10.3

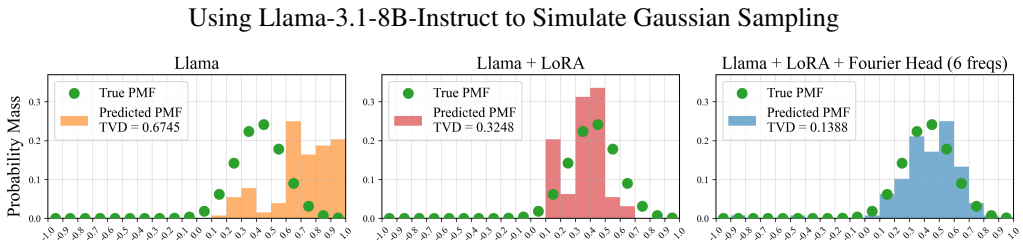


Figure 2.3: We demonstrate that the baseline Llama model does a poor job simulating Gaussian sampling, as measured by the Total Variation Distance between the ground truth quantized Gaussian histogram, and the empirical histogram of samples. We find that LoRA fine-tuning improves the results by a factor of ≈ 2.07, and that using the Fourier head improves the output distribution by a factor of ≈ 4.86.

model outputs be approximately Gaussian? In this empirical study, we simulate Gaussian sampling using the Llama-3.1-8B-Instruct model [31]. We demonstrate that this language model struggles to generate high quality numerical Gaussian samples. We consider two possible interventions: LoRA fine-tuning the base Llama model, and LoRA fine-tuning the base model while also replacing the linear classification head with a Fourier head. We find that LoRA fine-tuning improves the learned distribution significantly, and replacing the linear head with the Fourier head improves the distributions even further. We present an illustrative example of this phenomenon in Figure 2.3. See Appendix A.10.4 for experiment details and some related works.

2.4 Large-Scale Study: Offline Reinforcement Learning

The Decision Transformer [21] reframes reinforcement learning as sequentially modeling rewards, states, and actions. We evaluate its performance on the Seaquest game from the Atari [12] 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. The original architecture uses a decoder-only language model [94] to encode context and map it through a linear layer, producing a categorical distribution over actions. At test time, the agent samples from this distribution to select its next action. We replace the linear classification head with a Fourier head, introducing a prior that semantically similar actions (e.g., 'move left' and 'move up left') should have similar

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 2.2: 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.

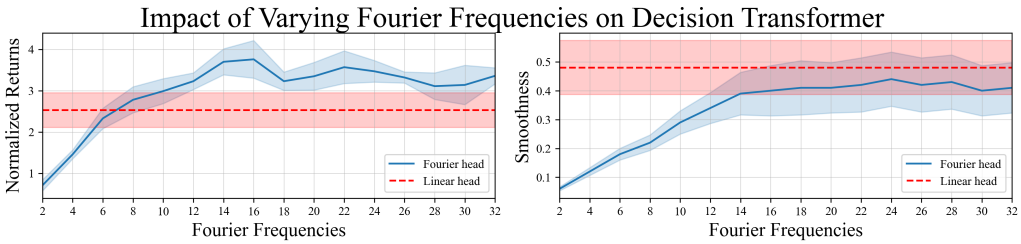


Figure 2.4: We present empirical results for how the quantity of Fourier frequencies impacts returns and smoothness for the imitation learning task. For normalized returns, higher is better; for smoothness, lower is better. We can see that the Fourier agent achieves higher normalized returns than the linear baseline agent when sufficiently many Fourier frequencies are used, while still learning smoother next-action distributions.

likelihoods. Our results show the Fourier head improves returns by as much as 46% in the reward-conditioned setting, using identical training hyperparameters.

Task: In the Seaquest game, the agent moves a submarine to avoid enemies, shoot at enemies, and rescue divers. 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 [21]. This dataset consists of 500k transitions experienced by an online deep Q-network agent [87] during training on the Seaquest game.

Model architecture: [21] used the GPT-1 model [94] 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 with N frequencies and Fourier regularization $\gamma = 10^{-6}$. In our experiments we consider frequencies $N \in \{2, 4, 6, 8, \ldots, 30, 32\}$.

Model Evaluation: We present results for the linear baseline and Fourier-N head ($N \in 2, 4, 6, \ldots, 30, 32$) across four Atari games, showing mean reward totals for rollouts at the best epoch across four seeds. Table 2.2 demonstrates significant return gains with the Fourier head. For example, Seaquest returns increase by up to 46.2%, while Gravitar sees as much as a 377% boost. Figure 2.4 shows improved Seaquest performance as the number of frequencies grows, with learned PMFs becoming less smooth, aligning with Theorem 2.2.3. Qualitative results in Figure A24 (Appendix) highlight the smoother PMFs produced by the Fourier head. Additional results for BankHeist, DoubleDunk, and Gravitar in Figure A27 (Appendix) confirm that the Fourier agent consistently outperforms the linear baseline while maintaining smoother next-action distributions.

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 A25 (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

Chronos Time Series Model	MASE (↓)	WQL (↓)	Smoothness (↓)
Linear	0.883	0.750	0.1689 ± 0.1087
Fourier-550	**0.852**	**0.749**	$\mathbf{0.0283 \pm 0.0224}$

Table 2.3: We present large-scale experiments on Chronos time series forecasting. The best-performing Fourier model outperforms the linear baseline both terms of the continuity of the learned probability mass functions (smoothness) for the quality of the forecasts (MASE, WQL).

performance of the Linear head and the Fourier head, and obtain a similar result. In Figure A26 (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.

2.5 Large-Scale Study: Probabilistic Time Series Forecasting

The Chronos time series foundation models [6] "learn the language of time series". They do this by approaching time series forecasting as language modeling, by 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 [96]. 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 A28 (Appendix), we can see that the linear next-token prediction PMFs fit to the noise, and appear very jagged.

The motivation for replacing the linear head with the Fourier head is to "smooth" out the distribution in the left side of Figure A28, to help the forecasting model better learn the signal, and ignore the noise. In this figure, we can see that the Fourier head accomplishes this successfully.

In this section, we study how the 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 outperforms the Chronos linear baseline on the MASE metric, while learning next token multinomials which are at least 8x smoother, with fewer parameters than the baseline.

Dataset: We use the same training dataset for large-scale pretraining that Ansari et al. [6] 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 [6]. The reader can check Appendix A.11.2 for details on the training and evaluation datasets we used.

Model architecture: We use the Chronos model, which is built using the T5 architecture [96]. 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 2.1.3. We also use Fourier weight decay regularization with $\gamma = 10^{-6}$. 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 [6] (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 2.3 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 A5 (Appendix) show that mixed precision binning and regularization improve the MASE and smoothness for the Fourier head, and that using more Fourier frequencies improves MASE and WQL. Additionally, we show that the Fourier head yields more accurate forecasts than the linear head across dataset sizes and model sizes (Figures A29 and A30, Appendix).

2.6 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 [83], time series forecasting [6, 24, 59, 89, 100, 58, 146, 123], music generation [28, 2, 23, 137], and as well as in decision making [69, 21].

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 [86] but has been challenging for modeling samplers for probability distributions [51]. In a related direction, Razeghi et al. [97] 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 [108] or RL contexts [33]. While even frozen LLMs with no further training show interesting empirical results as regressors [119], 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 pre-processing 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. [27] 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. [107] proposed sinusoidal activation functions, which can be seen as learning the *frequencies* of a Fourier series; in contrast, we 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.

2.7 Conclusion

We propose the Fourier head and demonstrate its positive impact on performance on several tasks. We prove a scaling law that characterizes 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.

Chapter 3

Force Prompting: Video Generation Models Can Learn and Generalize Physics-based Control Signals

Humans develop an intuitive understanding of how objects respond to forces since infancy [128, 118]: a gentle poke causes a plant to sway, while a breeze creates rippling patterns across fabric. Do video generation models, which encode powerful visual and motion priors through internet-scale pretraining, possess a similar level of intuitive physics understanding? And if so, how to elicit their capabilities to interact with force inputs? A positive answer to these questions would provide a more flexible and expressive interface for video content creation, enable interactive video generation with user input (e.g., generating a video game), and eventually lead to an intuitive world model for intelligent agents to plan and make decisions with.

We introduce **Force Prompting**, a step towards incorporating force-based control (direction and magnitude) into video generation models. We explore two distinct categories of force prompts: *local* force prompts, such as instantaneous pokes or pulls applied to specific regions, and *global* force prompts, such as sustained directional wind that affects the entire scene uniformly. Crucially, as manually collecting force annotations from natural videos is both costly and difficult, we instead leverage physics simulators (e.g., Blender) to hand-craft perfectly annotated training data. With our data creation pipeline, we specify a collection of objects along with the force conditions, and simulate the resulting dynamics to obtain the paired training videos. We hypothesize that such *sim2real* generalization is feasible because state-of-the-art video generation models already encode strong priors about visual dynamics, and our paired force-video data serves the role of eliciting their understanding of the physics-based control signals.

We implement Force Prompting by introducing additional force control as local or global vector fields on a video generation model [135] conditioned on initial frame and text. We also curate an evaluation benchmark of diverse objects and motion types to evaluate global and local force prompts. As illustrated in Figure 3.1, our main finding is that despite the synthetic visual appearance and few objects (flying flags and rolling balls) in our training data, video generation models can indeed learn to execute fine-grained force prompts, and exhibit

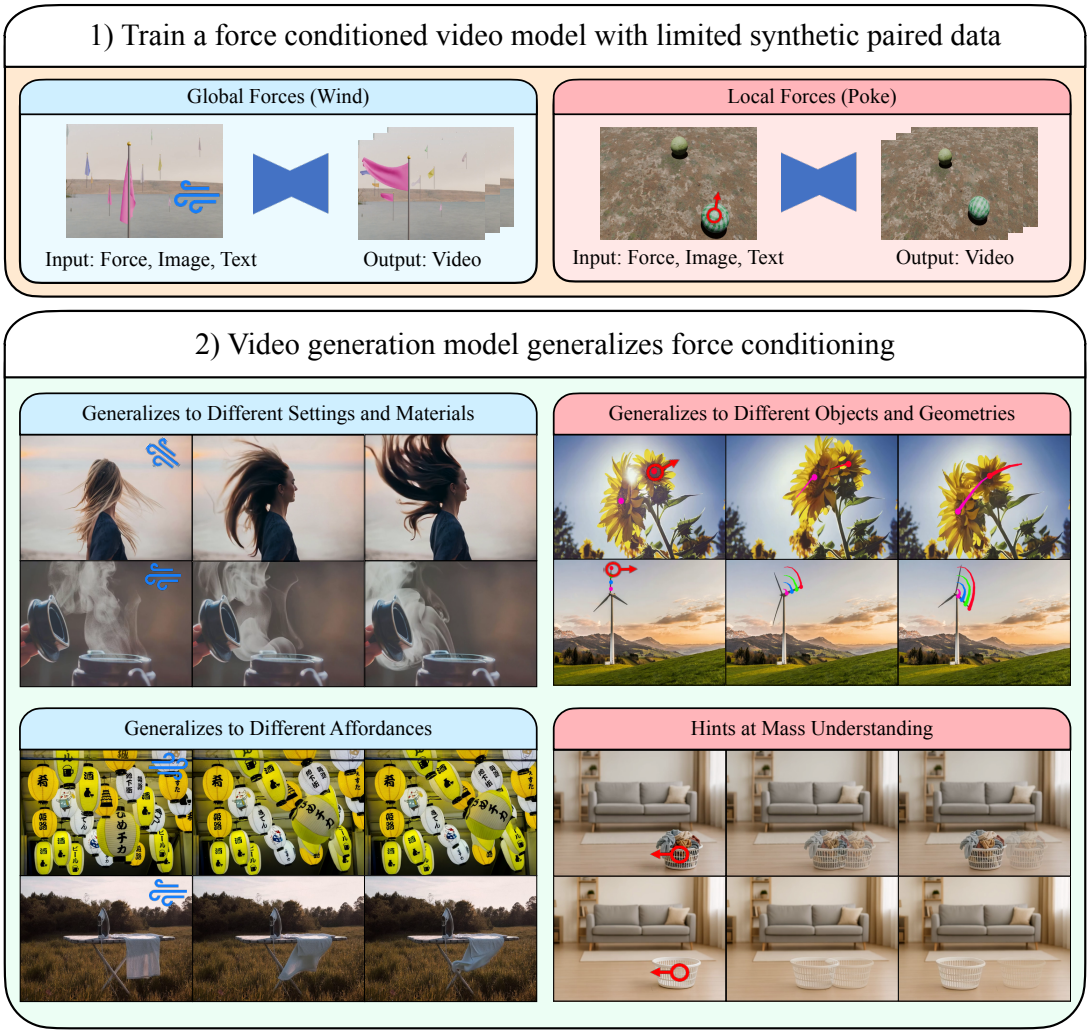


Figure 3.1: **Force prompting** allows users to apply either global or local forces to objects in an image and then generate the resultant video. Despite being trained on a limited set of synthetic videos (15k for global force and 23k for local force), we observe significant generalization to different settings, materials, objects, geometries, affordances, and some initial hints at mass understanding. Trajectory visualization or alpha overlay are incorporated to better illustrate movement for some examples.

surprisingly strong *generalization* behavior across diverse settings, object shapes and materials, geometry, and affordances. Through extensive human evaluations, we demonstrate that Force Prompting exhibits superior adherence to physical instruction while maintaining realistic motion and visual quality, when compared to text-conditioned baselines. This validates our hypothesis that synthetic data can teach video generation models intuitive physics and control without damaging their video priors. We further show that simply extrapolating the future by treating forces as local trajectories is insufficient, and our approach significantly outperforms the state-of-the-art in trajectory-controlled video generation [35]. Notably, Force Prompting can be trained in approximately a single day on four NVIDIA A100 GPUs. We also try to understand the cause of this strong generalization and perform a careful ablation on the training data. We find two elements that appear important

to generalization: visual diversity in the training data with respect to the control signal and the usage of certain text keywords at training time, which appear to help elicit the understanding of force control signals.

In summary, our main contributions are as follows:

1. We introduce physical forces as conditioning signals for video generation through two models: one for localized point forces and another for global wind forces.
2. We find that video models can execute precise force prompts with broad generalization to different settings, objects, geometries, and affordances despite minimal training data (15K videos) and modest computational resources (one day on four A100 GPUs). We also attempt to understand the source of this generalization and perform careful ablations on the training data, finding two key elements: visual diversity with respect to the control signal, as well as the usage of text keywords at training time, which appear to help elicit understanding of force control signals.
3. We show that our force-conditioned model has some degree of mass understanding, where the same force can cause a lighter object to move farther than a heavier one.

We release all datasets, code, and models on our project page, https://force-prompting.github.io/.

3.1 Related Works

Video generation: In the last several years, video generation models have made rapid progress in visual quality and realistic dynamics [106, 50, 14, 38, 10, 16]. In particular, Sora [16] was one of the first video generation models to demonstrate truly compelling diverse real-world physical phenomena and directly advocated for the future use of using video generators as simulators for the physical world. In the last half year, significant progress has been made by open source models such as CogVideoX [135] and Wan 2.1 [121], even approaching the quality of closed-source models. While these models act as strong video priors, they primarily use text and images as input and lack precise control over general actions or other physical inputs.

Controllable video generation: As video models have rapidly progressed, so too has the accompanying field of controllability for these models with the majority of work in this domain focusing on either camera control [45, 143, 112] or various paradigms of motion control [136, 22, 124, 104, 90, 130, 35, 70, 88, 141], such as drag-based, trajectory-based, and optical flow-based techniques. Many of the existing motion control models [136, 22, 141] require the complete pre-specified trajectory, specifying the location of the pixel on every generated frame. This reliance on full temporal information makes it difficult to use these models for simulation or prediction tasks.

Motion Prompting [35], a concurrent work, uses spatio-temporally sparse trajectories as a conditioning signal, enabling users to specify motion over a few frames for video extrapolation. While this might superficially resemble force control, crucial distinctions exist. First, global phenomena like wind or fluid dynamics are naturally expressed as forces but are difficult or impossible to represent with trajectories. Second, applied forces fundamentally depend on an object's mass or material properties - a dependency absent when specifying motion or location (e.g., identical forces induce greater displacement in lighter objects). Third, specifying an

object's location across a few frames is not equivalent to an applied force; the same observed motion could result from numerous alternative causes, such as camera movement or internal object changes. We compare to Motion Prompting and demonstrate significantly better adherence to the conditioning force.

Interactive world models: Paralleling the interest in video generation models, interactive world models [44] have gained significant attention. Despite extensive research in this area, investigations have predominantly concentrated on video game environments [120, 18, 17]. While a few contemporary studies have begun exploring real-world applications (e.g., [9, 1]), none explores interactions besides camera control or text. In contrast, our work focuses on interaction through physical forces.

Physical simulators and hybrid approaches: Early work [25, 26] on generating video based on intuitive forces extracts modal bases of vibrating objects in 2D image space; these works, as well as their modern adaptations [71], represented motion as a series of vibrations with different frequencies and intensities, which works well for vibration-like motions but struggles to represent many types of motion, such as linear motion. This led to an alternative research direction explicitly incorporating physics solvers [20, 144, 65, 131, 140, 53, 75, 74, 3]. However, almost all of these techniques require the 3D geometry of the scenes. Recent work has focused on combining both physics simulators and generative models, trying to get the best of both worlds: accurate dynamics from the simulator and better appearance from generative models. For example, PhysGen [76] uses a rigid-body physics solver to model object collisions and then renders these scenes through a video generator, and PhysMotion [113] uses a combination of a 3D physics solver and a video generation model. However, due to their usage of physics simulators, they are limited in the types of dynamics they can model. In contrast, we explore using the video generation as a simulator and do not use a physics simulator at inference time. We mention some concurrent works as well: [68] also explores the use of simulated videos to finetune generative models, but their focus is on modeling object freefall as opposed to learning physics-based control; [72] explores action-conditioned video generation, but their model requires the use of a physics simulator at inference time; and [122] explores force-conditioned video generation, but their framework requires learning a 3D point cloud trajectory model from synthetic data, and then passing that 4D temporal volume into a point cloud-conditioned video generation model.

3.2 Method: Force Prompting

The goal of Force Prompting is to enable users to interact with images through physical forces. To this end, we explore two distinct force prompts paradigms: a global model that allows users to animate an entire scene with directional wind forces, and a local model that enables precise interaction through localized point forces applied to specific objects within the image. Our video generation method takes as input a triple (τ, ϕ, π), where τ is the text prompt, $\phi \in \mathbb{R}^{c \times h \times w}$ is the initial frame with height h width w and c channels, and π is the physics control signal which represents the force being applied: for the wind force model, this is simply a force vector (magnitude, angle) $\in \mathbb{R}^2$, and for the point force model, this is a force vector (magnitude, angle) $\in \mathbb{R}^2$ along with pixel coordinates $(x, y) \in \mathbb{R}^2$ specifying where to apply the force. The goal is to generate a video $v \in \mathbb{R}^{f \times c \times h \times w}$. While we train the global force and local force models with different synthetic datasets and

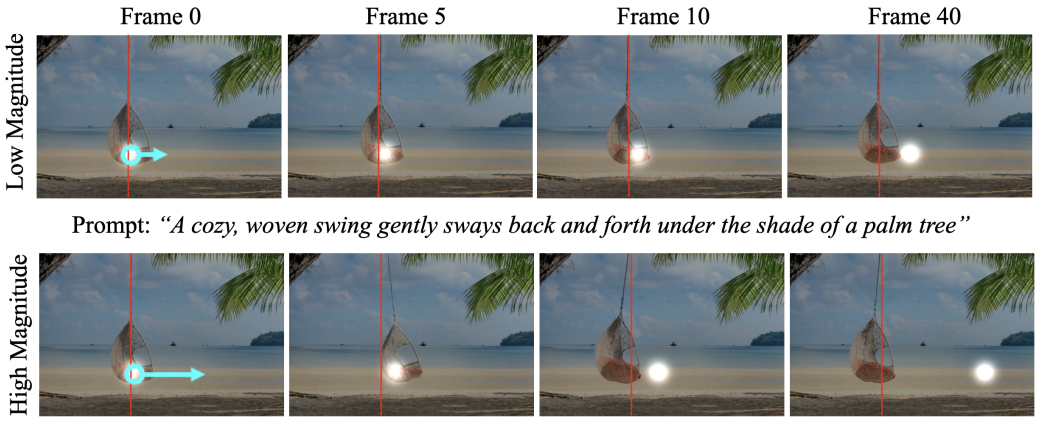


Figure 3.2: **Visualizing the point force control signal.** The magnitude of applied force is proportional to the gaussian blob's velocity in the control signal, producing proportionally stronger impulses. Stronger forces (bottom) generate faster-moving blobs and correspondingly larger physical responses than gentler forces (top). Note, red line added at the same location in each image for visualization. In our method, we enable the force prompt to dictate the object's trajectory, deliberately excluding such specifics from the text prompt.

encode the force inputs differently for each, both models share identical architectures and training procedures.

3.2.1 Synthetic training data

To construct our global wind force dataset, we use a physics simulator to generate videos of flags waving in the wind. And we construct our local point force dataset in two parts: for the first part, we use a physics simulator to generate videos of a ball rolling across the ground; and for the second part, we use a model [140] which integrates 3D Gaussians and a physics simulator to generate videos of a plant being poked. We provide more detail below.

Global force dataset: We use Blender to construct a dataset of flags waving in response to varying wind conditions. In order to generate a diverse dataset, we randomize multiple parameters for each video: flag quantity ($\text{Unif}\{1, \ldots, 64\}$), flag color (from a set of 100), flag positions, camera placement, HDRIs (High Dynamic Range Images), which are 360-degree panoramic images used for lighting and background purposes (selected from 50 options on Polyhaven), wind direction in $[0, 360)$, and wind speed in $[0, 1]$, where 0 corresponds to no wind, and 1 corresponds to very strong wind. Each video captures the flags' transitions from stationary to wind-affected state. Our training dataset has 15k videos.

Local force dataset: The first scenario in our dataset comprises 12k videos of balls, with one of them rolling in response to being pushed by an unseen point-wise force (the force actor is *not rendered*), and the other balls remaining stationary. We generate these videos using Blender with randomized parameters: ball quantity ($\text{Unif}\{2, 3, 4\}$), ball textures (soccer balls using a Polyhaven mesh [$p = 2/3$] , or bowling balls modeled as smooth spheres [$p = 1/3$]), ball colors (from a set of 108), ball positions, camera position, ground textures (from 42 Polyhaven options), target ball selection, force angle in $[0, 360)$, and force magnitude in [0,1]. We assign the bowling ball to be four times the mass of the soccer ball with the goal of teaching the model

mass-based dynamics. The second scenario (11k videos) utilizes PhysDreamer [140], a *generative-simulator hybrid*, and features videos of a carnation swaying back and forth in response to being poked by an unseen force. We generate these videos with randomized camera position, contact points, force angles, and magnitudes. We use a mixed dataset with the goal of teaching the model that a point force can result in both simple linear motion, and complex oscillatory dynamics, depending on what type of object the force is applied to. In both scenarios, the force magnitude 0 corresponds to a very gentle poke, and the force magnitude 1 corresponds to a much stronger poke.

For both datasets, we project forces from 3D space onto the 2D pixel plane using the camera's parameters. This transformation maps force vectors and object positions from the physical world coordinate system to screen coordinates, allowing us to model forces within the image frame. We generate detailed text prompts using the GPT-4o API, creating a unique descriptions for each HRDI background and ground texture, plus a single shared prompt for all PhysDreamer carnation videos.

3.2.2 Local and Global Force Prompts

As the wind force is applied globally, and the point force is applied locally, we propose two different force encoding strategies.

Encoding strategy, global force: The wind force control signal is parameterized by a force $F \in [0, 1]$ and an angle $\theta \in [0, 360)$. The goal is to develop a tensor representation for the physics prompt π, which we denote by $\tilde{\pi} \in \mathbb{R}^{f \times c \times h \times w}$. Here, $f = 49$ is the number of frames, $c = 3$ is the number of color channels, and $h = 480$ and $w = 720$ are the height and width of the generated video. We define the first channel of $\tilde{\pi}$ to be $-1 + 2 \cdot F \in [-1, 1]$, the second channel to be $\cos\theta$, and the third angle to be $\sin\theta$. This defines a smooth map $[0, 1] \times [0, 360) \to \mathbb{R}^{f \times c \times h \times w}$ which encodes the angle and magnitude of the wind force field.

Encoding strategy, local force: The point force control signal π specifies a localized force, so it is parameterized by the pixel coordinates $(x, y) \in \{0, \ldots, w-1\} \times \{0, \ldots, h-1\}$ in addition to the force magnitude $F \in [0, 1]$ and angle $\theta \in [0, 360)$. At a high level, we define the tensor representation $\tilde{\pi} \in \mathbb{R}^{f \times c \times h \times w}$ for the control signal π to be a sequence of frames where a Gaussian blob starts at the pixel location (x, y), and then moves in the direction θ at a constant velocity, for a total distance affinely proportional to the force F. Full mathematical details in Appendix A.12.3. This defines a continuous map $\{0, \ldots, w-1\} \times \{0, \ldots, h-1\}] \times [0, 1] \times [0, 360) \to \mathbb{R}^{f \times c \times h \times w}$ which encodes the coordinates where the force is applied, as well as the point force magnitude and angle, into a tensor representation $\tilde{\pi}$. We present a visual example of this in Figure 3.2. In the case of the local force, we note that the displacement of the Gaussian blob is nonzero when the force is $F = 0$, as our training dataset convention is that $F = 0$ indicates a small force.

We note that force values across the ball rolling and plant poking training videos are not calibrated to any absolute physical scale. Instead, they follow intuitive relative physics where smaller force values (approaching $F = 0$) correspond to gentle pokes resulting in minimal initial displacement, while larger force values produce stronger pokes with correspondingly greater initial displacement. We also wish to highlight that our force prompting models are fundamentally different from video generative models with trajectory-based control such as [141, 35]. This is because the gaussian blob which serves as the force indicator for the point force

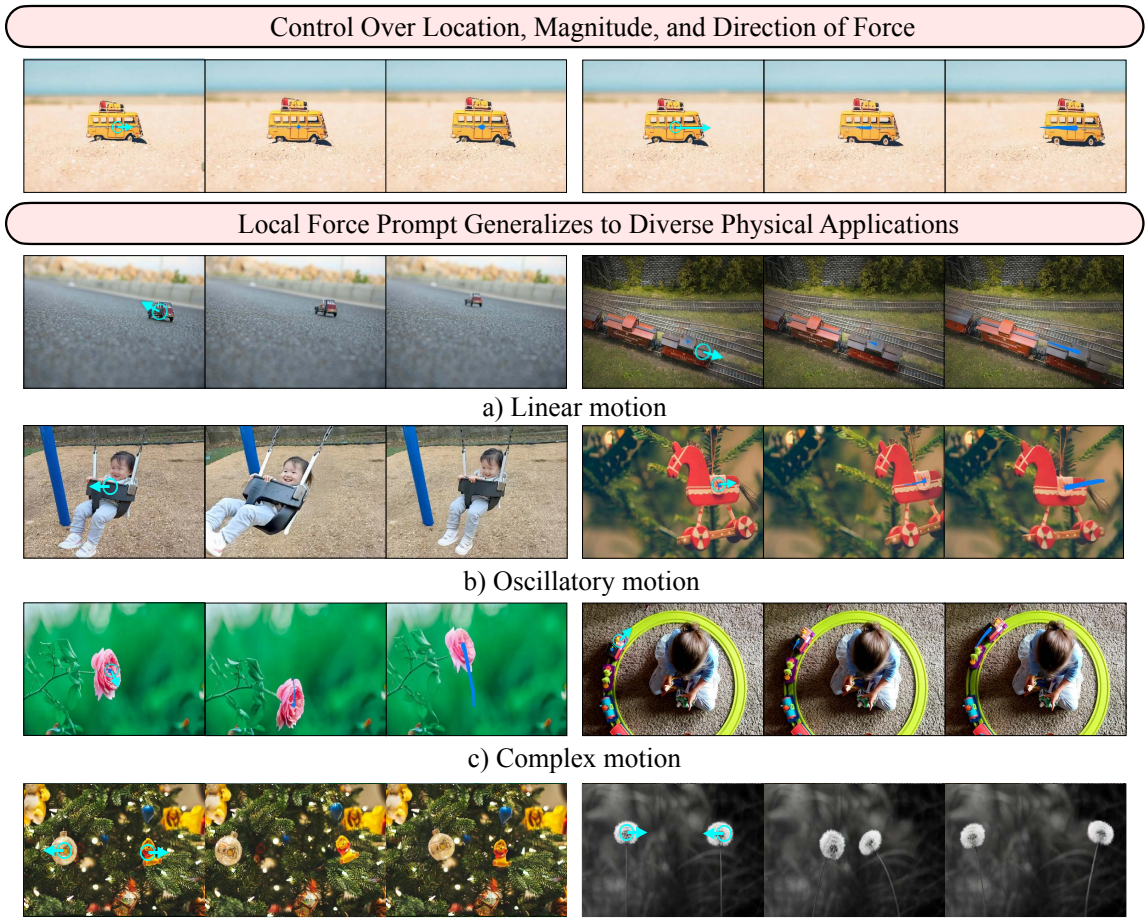


Figure 3.3: **Qualitative results for the Local Force (Poke) model**. *Top section:* For local forces, the control signal can specify both the location, magnitude, and direction of the force. *Bottom section:* despite the limited training data, the model generalizes to different types of motion. We add blue lines to visualize a time-lapse of some objects' movements.

model is generally far away from the pixels that it affects, as demonstrated in the complex oscillatory motion of the swaying flower in Figure 3.2. Similarly, the wind force control signal under-specifies which points must move to which locations, as that control signal is global and causal.

3.2.3 Architecture and Training

We build the force prompting models on top of CogVideoX-5B-I2V [135], a video generative model which accepts text and initial frame as conditional inputs. This model generates 49-frame videos at 8-fps. In order to integrate force prompt conditioning, we add a ControlNet [139] which inputs a physics control prompt π, processing it through downscaling, encoding, and temporal compression before combining with hidden

Point Force Model	Linear Motion			Oscillatory Motion			Complex Motion		
	Force Adh.	*Real. Physics*	*Visual Qual.*	*Force Adh.*	*Real. Physics*	*Visual Qual.*	*Force Adh.*	*Real. Physics*	*Visual Qual.*
Text-only, zero-shot	72%	50%	48%	67%	48%	52%	73%	48%	49%
Text-only, fine-tuned	79%	53%	52%	62%	52%	58%	74%	55%	54%
Motion Prompting	91%	93%	100%	89%	76%	99%	86%	76%	98%

Global Force Model	Tethered Motion			Aerodynamic Motion			Fluid Dynamics		
	Force Adh.	*Real. Physics*	*Visual Qual.*	*Force Adh.*	*Real. Physics*	*Visual Qual.*	*Force Adh.*	*Real. Physics*	*Visual Qual.*
Text-only, zero-shot	91%	50%	54%	97%	48%	47%	84%	53%	47%
Text-only, fine-tuned	62%	48%	47%	57%	70%	50%	71%	58%	49%
Motion Prompting	93%	82%	100%	90%	75%	100%	90%	80%	95%

Table 3.1: **Comparison to baselines.** *Top:* Local point force model. *Bottom:* Global wind force model. We present % win rates of our method against baselines in 2AFC human study results (i.e. values above 50% indicate a preference for Force Prompting) for force adherence, realistic physics, and visual quality. We find that none of the other methods provide consistent adherence to the input force.

states via a zero convolution. The ControlNet clones the first six transformer layers and fine-tunes them while keeping the base model's transformer layers frozen. We base our implementation on [62] with modifications to adhere more closely to the original ControlNet design. We train the models on a four 80 GB A100 GPU cluster for 5000 training steps, which takes approximately one day. Training uses an instantaneous batch size per device of 1, with two gradient accumulation steps, for an effective batch size of 8. Full hyperparameter details are listed in Appendix A.12.1.

3.3 Quantitative and Qualitative Results

We propose a benchmark dataset for both force prompting models using images that we curate from Pexels. We conduct a 2AFC human study ($N = 10$) using Prolific comparing our force prompting model against three baselines on these benchmark datasets.

Baseline models: The first baseline is *text-only, zero-shot*, which uses the original CogVideoX model and describes the intended force with a string and appending it to the end of the original text prompt. Two example prompt string suffixes are "the apple is moved very forcefully, upwards and to the left", and "the wind is medium strength, blowing right". The second baseline is *text-only, fine-tuned*, which has the same ControlNet architecture as our force prompting model, but with zero-tensor control signals, as well as force suffixes added to the end of the text prompts during training. Our third baseline is *Motion Prompting* [35], built on Lumiere [10] (run by the authors). It is the only track-conditioned model that accepts temporally sparse tracks as conditioning signal. We simulate force prompt's impulse by tracing push paths from target objects for the first 3 frames. While the model is meant to accept temporally sparse trajectories, 3 frames of trajectory is out of domain for the intended use case of Motion Prompting.

Human study for local force benchmark: We create a benchmark by curating 63 images from Pexels

demonstrating three categories of physical interactions: 1) *linear* movement patterns (toy car, toy train on straight track, hot air balloon); 2) *oscillatory* movement patterns (windmill, pendulums, ornament, and swing); and 3) *complex* movement patterns (toy train on circular track, various plants including ivy, apple tree, and flowers). Table 3.1 presents human evaluation results for point forces, showing that despite training only on ball rolling (linear) and plant poking (complex) scenarios, our force prompting model demonstrates strong generalization across all motion categories. We note that this model successfully handles multiple forces "zero-shot" during inference, despite only being trained to handle a single force, as seen in Figure 3.3 and detailed in Appendix A.13.1.

Human study for global force benchmark: We create a benchmark by curating 41 images from Pexels which demonstrate three different types of physical properties. The first is *tethered motion* (hair, cloth, clothing on person, paper lantern attached to hook). The second is *aerodynamic motion* (bubbles, falling leaves, inflatable tube in pool, floating litter, confetti). And the third is *fluid dynamics* (fog, smoke, snow, steam). In Table 3.1 we present human evaluation results for the global wind force model. Note that the base CogVideoX model is good at generating videos for all three motion categories (tethered, aerodynamic, fluid dynamic). However, our training data only has tethered motion (flags waving on a flagpole). We observe that the global wind control model trained only on labeled videos with tethered motion results in a model with generalized control over aerodynamic motion and fluid motion as well. We visualize some of these generalization patterns in Figure 3.4.

Human study comparing to PhysDreamer: The point force model, trained on data from a single carnation, demonstrates remarkable generalization to other plants, as we illustrate in Figures 3.1 and 3.3. To evaluate this generalization quantitatively, we compare our approach against PhysDreamer [140], which employs 3D assets and an integrated physics simulator. Using their benchmark dataset of six plant species, our results in Table 3.2 show that the point force model successfully generalizes to various roses, tulips, and alocasia without specific training on these plants. While we do not claim to replace physics-based simulation approaches, our purely neural method offers exceptional generalizability and produces responses that align with "intuitive physics," effectively conveying plausible physical interactions to human evaluators. We also also conduct an extended qualitative comparison with 8 other physics simulation models; see Appendix A.14.2 for more details.

3.4 Ablation Studies

3.4.1 Ablation Study #1: Composition of Synthetic Dataset

How do synthetic dataset design choices affect model generalization? In this section we analyze the impact of dataset diversity on force modeling tasks. Sample results are illustrated in Figure 3.5, more in depth results are in Figure A31 (Appendix), and additional videos are on the project webpage.

Point force training dataset ablation: For the localized point force task, we conduct an ablation study by removing "distractor balls" from scenes, leaving only a single ball affected by the point force. Our results show that the presence distractor balls significantly improves force localization. Without them, the model exhibits undesirable behaviors: when poking one hot air balloon, all balloons move slightly; when poking a

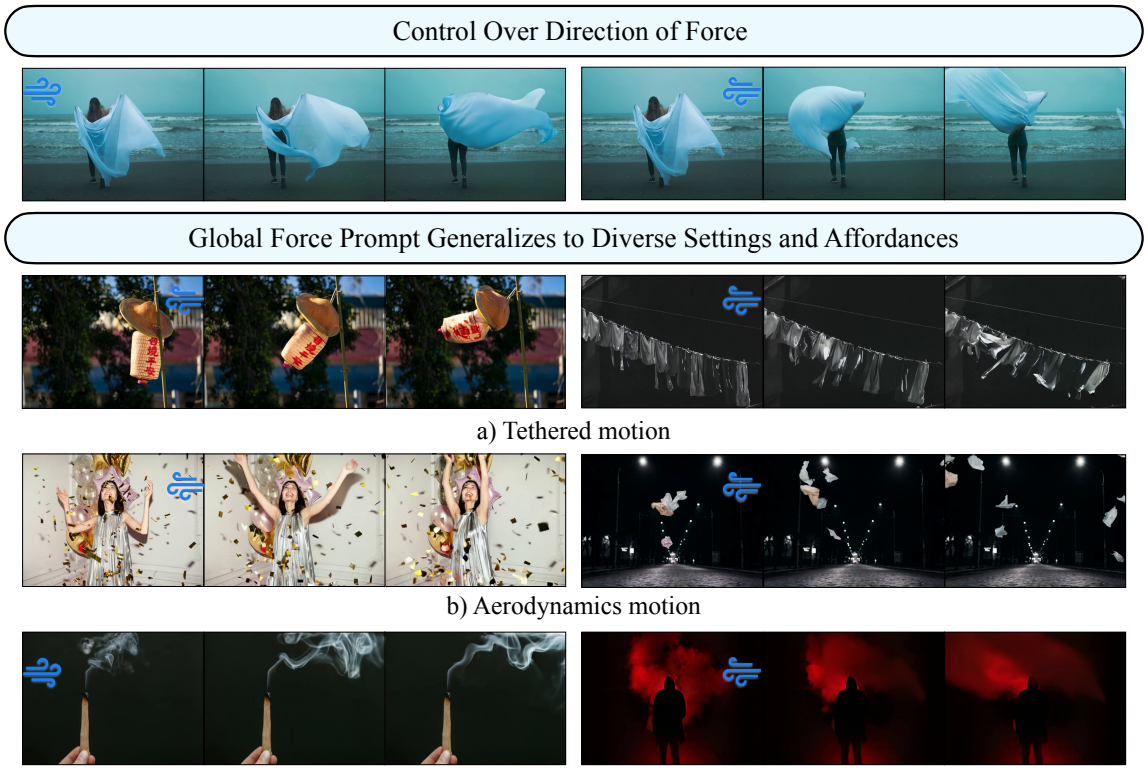


Figure 3.4: **Qualitative results for the Global Force (Wind) model**. *Top:* from the same starting image, different directions for the force result in different videos. *Bottom:* while the model was only trained on flags, it can generalize to many different settings producing different types of motion.

rose in a glass vase, both the rose and vase move together, failing to isolate the force application. Visuals are in Figure A31 as well as the project webpage.

Global force training dataset ablation: For the global wind force task, we evaluate two diversity factors: flag quantity and background variety. We find that training with a single background leads to models that follow force physics but frequently fail to differentiate between foreground and background, reducing visual quality. Similarly, when restricting scenes to contain only one flag instead of a variable number ($\text{Unif}\{1, \ldots, 64\}$), the model successfully models cloth mechanics but fails to generalize to other materials. In these cases, smoke from campfires remains unaffected by wind, and confetti either doesn't respond or stays unnaturally suspended. We also observe that bubbles don't respond to wind, while human limbs incorrectly billow like cloth. These failures indicate that insufficient scene diversity causes the model to overfit to stationary backgrounds and limited material interactions. These findings are illustrated in Figure A31 as well as the project webpage. Additionally, we trained a unified model to learn both point force prompts and wind force prompts. We found that this results in more dynamic backgrounds, but has slightly less robust point force control. Additional details are in Appendix A.14.3.

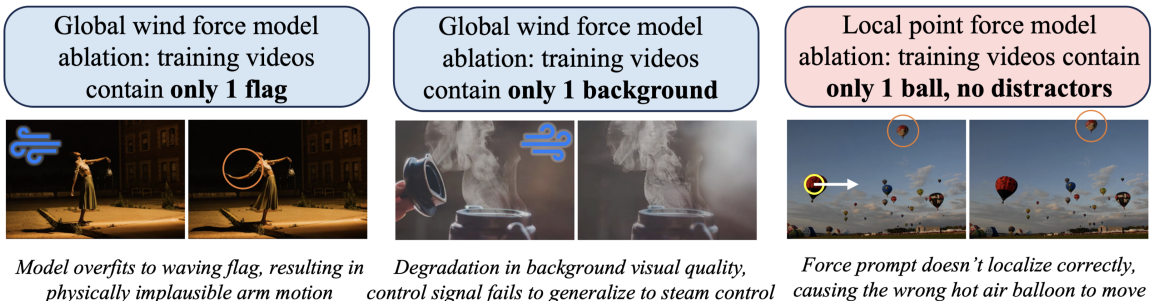


Figure 3.5: **Results from our ablation studies on synthetic dataset design choices.** *Left:* when the global wind force model is trained on a dataset with only one flag, it overfits, causing the woman's arm to wave unnaturally like fabric. *Middle*: when trained with a single background, the global force model has significantly degraded overall visual quality. *Right:* when trained without distractor objects, the point force model cannot properly localize motion, applying forces indiscriminately rather than to the intended target.

	Alocacia	Carnation	Rose (Orange)	Rose (Red)	Rose (White)	Tulip	**Mean**
Motion Realism	40%	50%	50%	60%	40%	50%	48.33%
Visual Quality	20%	40%	50%	40%	20%	50%	36.67%
Force Adherence	60%	70%	50%	50%	50%	70%	58.33%

Table 3.2: **Comparison to PhysDreamer.** Values represent the percentage of evaluators preferring the Force Prompting model over PhysDreamer, an approach that uses physics simulation during generation. Values above 50% indicate preference for our force prompting model. The results show that Force Prompting outperforms PhysDreamer on force adherence and achieves comparable performance on motion realism, while PhysDreamer maintains an advantage in visual quality.

3.4.2 Ablation Study #2: Text Prompt Specificity

How does specificity of the text prompt affect model outputs? In this ablation study, we investigate how material descriptions in text prompts affect model generalization through a 2×2 grid search ablation study. We train and test our wind model with and without wind-related keywords (wind/breeze/blow). Our results in Figure A32 and the project webpage show that omitting these keywords during training significantly increases failure cases in our benchmark dataset—fog remains static, lanterns collapse unexpectedly, and steam appears without cause. In contrast, models trained with wind-specific terminology demonstrate superior generalization to diverse wind scenarios. Interestingly, the presence of these keywords during inference has less impact than during training, though using wind terminology generally produces more robust results.

3.5 Mass Understanding

This section examines the model's capacity for *mass understanding*, which we define to be the ability to recognize that objects with different apparent masses should respond distinctively to the same applied force. A model with robust mass understanding would demonstrate physically intuitive behaviors: a book sliding

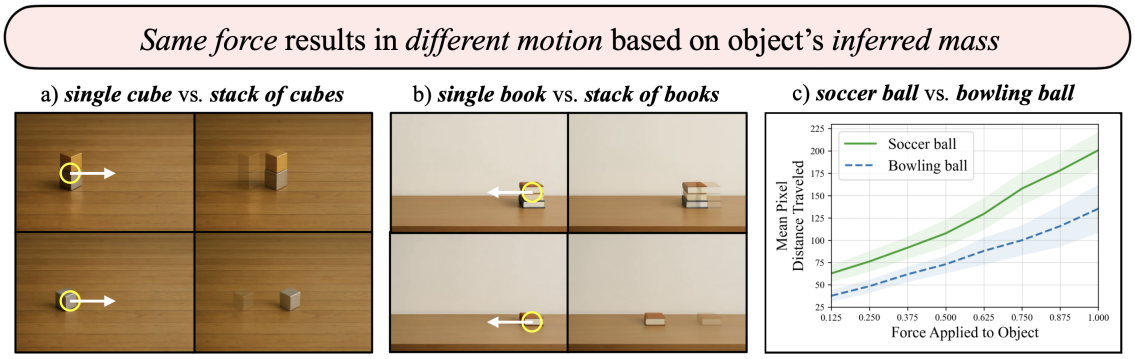


Figure 3.6: **Mass understanding:** We find that the model has some degree of understanding of mass, in that the same force applied to two objects with different masses will result in different amounts of motion. We demonstrate this qualitatively in (a) and (b) and quantitatively in (c), showing that this result is consistent across a range of force magnitudes. See additional examples in the project webpage.

further than a stack of books when pushed with equal force, or a wooden ornament swinging more freely than its identical metal counterpart under the same impulse. We focus our quantitative analysis on the ball-rolling scenario, as it allows for objective measurement using automatic object detection. Then, we focus our qualitative analysis on other scenarios which present greater challenges for obtaining reliable metrics at scale, such as the swinging ornament.

Force-Mass Relationship Quantitative Study: To quantitatively assess the model's mass understanding, we design an experiment to measure whether soccer balls roll farther than bowling balls when subjected to identical forces. We generate initial condition images across four ground surfaces (dirt, grass, stone, and wood), with three color variations each for both bowling balls and soccer balls. Additional experiment details are in Appendix A.12.2. Results presented in Figure 3.6 confirm two key physical principles: the distance traveled increases linearly with applied force for both ball types, and soccer balls consistently travel farther than bowling balls across all force magnitudes, demonstrating the model's intuitive understanding of mass-dependent physics in this scenario.

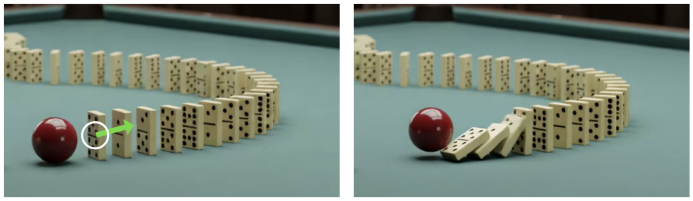
Force-Mass Relationship Qualitative Study: We evaluate mass understanding across four benchmark tasks featuring geometrically identical objects with different implied masses. Our test scenarios includes ornaments (wooden versus cast iron), laundry baskets (empty versus filled with clothes), book stacks (one, two, or three books), and cube stacks (single versus double cube). To ensure experimental control, we utilize the GPT-Image-1 API to generate initial frames with variations where only the implied mass differs between conditions. Figure 3.6 presents some of these results, with demonstrating that lighter objects consistently travel farther when subjected to identical forces. This pattern remains robust across four random seeds. This behavior suggests an emergent understanding of mass-dependent physics in our force-prompted model. Other results are in the project webpage. Additionally, we find that the mass understanding behavior persists in the zero-shot multiple objects setting. We include these experimental details in Appendix A.13.2.

3.6 Conclusion

We introduce Force Prompting, enabling users to interact with generative video models through physically meaningful controls including localized point forces and global wind effects. Our approach demonstrates that video generation models can successfully learn to respond to force-based conditioning from limited synthetic training data, generalizing remarkably well to diverse objects, materials, and scenarios without requiring physics simulators at inference time. These results suggest a promising direction for developing intuitive world models that respond to natural physical interactions, with potential applications in both creative content generation and embodied AI planning.

Chapter 4

Goal Force: Teaching Video Models To Accomplish Physics-Conditioned Goals



*Goal Force: user specifies **goal** (green arrow to right), then video model generates **action** to accomplish goal (ball collides into domino)*

*Goal Force: user specifies **goal** (green arrow upwards), then video model generates **action** to accomplish goal (robot arm picks up flower)*

Figure 4.1: Given a force-conditioned task, **goal force** enables video models to generate the antecedent action to accomplish the task.

The past two years have witnessed a paradigm shift in video generation, evolving from coarse, rudimentary clips to near-photorealistic sequences [16, 10, 1]. This progress has sparked considerable interest in leveraging these models as "world models" for robotics and planning. One of the most exciting possibilities for using "world models" in planning involves generating a video that transitions from a current state (an initial frame) towards a specified goal state [43, 63]. Consider a soccer player at the start of a game: the initial frame shows the ball at midfield, and the objective is to score. Existing approaches predominantly rely on text or static images to define these goals. However, for complex physical tasks involving multi-step dynamics, these modalities often prove insufficient. Text is frequently too abstract; a soccer player's intent is rarely just to

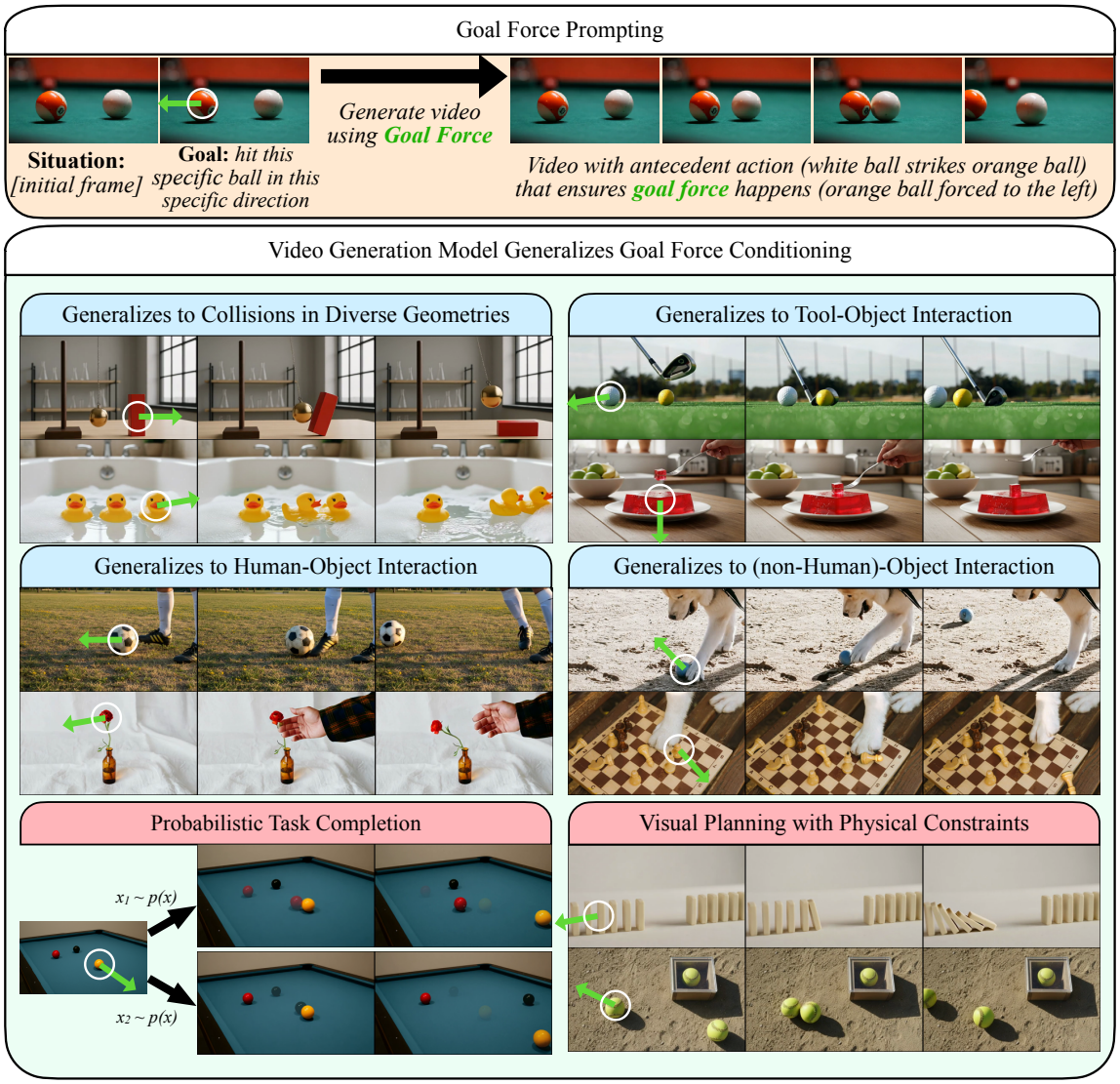


Figure 4.2: **Goal Force**: A user provides an input image and a **goal force**, and the model generates a video containing a force that locally causes the goal force. Our model generalizes to diverse objects and interactions and enables visual planning, respecting the physical properties of the objects and their environments.

"shoot at the goal," but rather to strike the ball with specific force and precision. Conversely, specifying a goal via a target image is often overly burdensome or infeasible, potentially requiring a user to render the exact lighting of a ball entering the net.

In contrast, humans approach tasks differently than through abstract text or pixel-perfect images alone. We often decompose long, abstract tasks into concrete sub-goals that, particularly in sports, possess distinct physical properties like spatial location, dynamics, and motion. When taking a penalty kick, a soccer player

does not focus merely on the static end state of the ball in the net, nor do they simply rely on the abstract concept of scoring. Instead, they aim to impart a specific trajectory and velocity—a "goal force"—onto the ball. This chapter proposes a method that aligns with this intuition: defining goals through desired forces and intermediate dynamics. By specifying these goal forces, rather than limiting users to static endpoints or requiring direct, low-level scene manipulation, we offer a mechanism that is both precise enough for physics-based planning and intuitive for human users.

To accomplish this, we introduce a framework that conditions video generation on explicit goal force vectors. We curate a dataset of paired videos and "goal forces," adapting a state-of-the-art open-source video model to accept these forces as a control signal. Our training strategy relies on the hypothesis that learning fundamental physical interactions can bootstrap complex reasoning. We train the model on simple, synthetic examples of causal primitives, such as elastic collisions and falling dominos. Crucially, we find that this grounded training enables *non-trivial generalization* to highly diverse scenarios (Figure 4.2).

Our empirical results demonstrate that the model learns to propagate forces through time and space, handling chains of events where one object exerts force on another, which in turn influences a third. Remarkably, this capability extends to zero-shot tool usage; for instance, the model can infer how to use a golf club to impart the desired force onto a ball, and to pick up a rose via its stem as opposed to its petals (Figure 4.1), despite only being trained on simpler collision data. This suggests the model is not merely memorizing patterns but acting as an implicit neural physics simulator.

Our main contributions are as follows:

1. We propose *Goal Force*, a new task and model which teaches video models to plan a causal chain of physical interactions to achieve a specified goal force. This moves beyond prior direct-force methods and changes how goals can be specified in world models.

2. We propose a training paradigm with a novel multi-channel control signal (for goal forces, direct forces, and mass) that teaches the model to act as an implicit neural physics simulator, requiring no simulator at inference.

3. We demonstrate powerful out-of-domain generalization: despite training only on simple synthetic data (*e.g.*, balls, dominos), our model leverages the base video model's rich prior to generate complex, physically-plausible scenarios involving tool use, human-object interaction, and intricate multi-object collisions.

We release training and evaluation code, model weights, synthetic training data, and benchmark datasets at our project page, https://goal-force.github.io/.

4.1 Related Works

Our approach builds upon recent advancements in video generation and physics-based modeling, which were broadly reviewed in Chapter 3. For some background on **video generative models** [106, 16, 121, 134], **interactive world models** [44, 120, 1, 60, 67], the integration of **physics simulators and hybrid approaches**

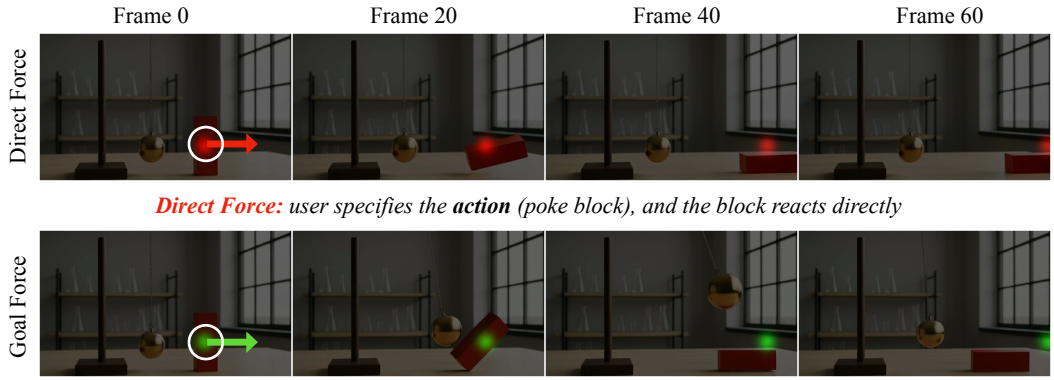


Figure 4.3: **Force prompt and goal force prompt result in different behaviors.** With a **direct force** applied to the red block (top), the effect is directly materialized (i.e. the block falls over). The force in this case is encoded in the red channel of the control signal as a moving Gaussian blob. In contrast, with a **goal force** applied to the red block (bottom), the model must find the antecedent motion to achieve the goal force (i.e. the pendulum swings to knock over the block). The force in this case is encoded in the green channel of the control signal as a moving Gaussian blob. We visualize the control signal overlaid on top of the video via alpha blending.

[140, 76, 72, 122], and general **controllable video generation** techniques [45, 141, 35], we refer the reader to Section 3.1.

Chapter 3 introduced Force Prompting [37], which allows for direct physical control via user-specified force vectors. Our *Goal Force* framework moves beyond this by enabling the model to reason about and plan a causal chain of forces—for example, generating the antecedent action of hitting ball A in order to achieve a desired goal force on ball B. Here, we focus on the specific area of related work most relevant to this goal-directed, causal planning.

Planning with videos: Video models have been applied to solve decision-making problems in robotic applications [85, 73]. A video generative model can serve as reward functions [32, 52], dynamics models [133, 120], and pixel-based planners [64, 4, 145]. For example, UniPi [30] and Adapt2Act [81] employ text-conditioned video generative models to predict visual plans that depict future outcomes, which are then converted into robotic actions with inverse dynamics models. With our introduced framework, such visual planners can take goal forces, in addition to text, to specify the desired goals.

4.2 Method: Prompting with Goal Force

Our method reframes force-conditioned video generation from specifying a *direct force* (*e.g.*, [37, 140, 76]) to declaring a desired *goal force*. Given a starting frame ϕ and a text prompt τ, the user specifies a "goal

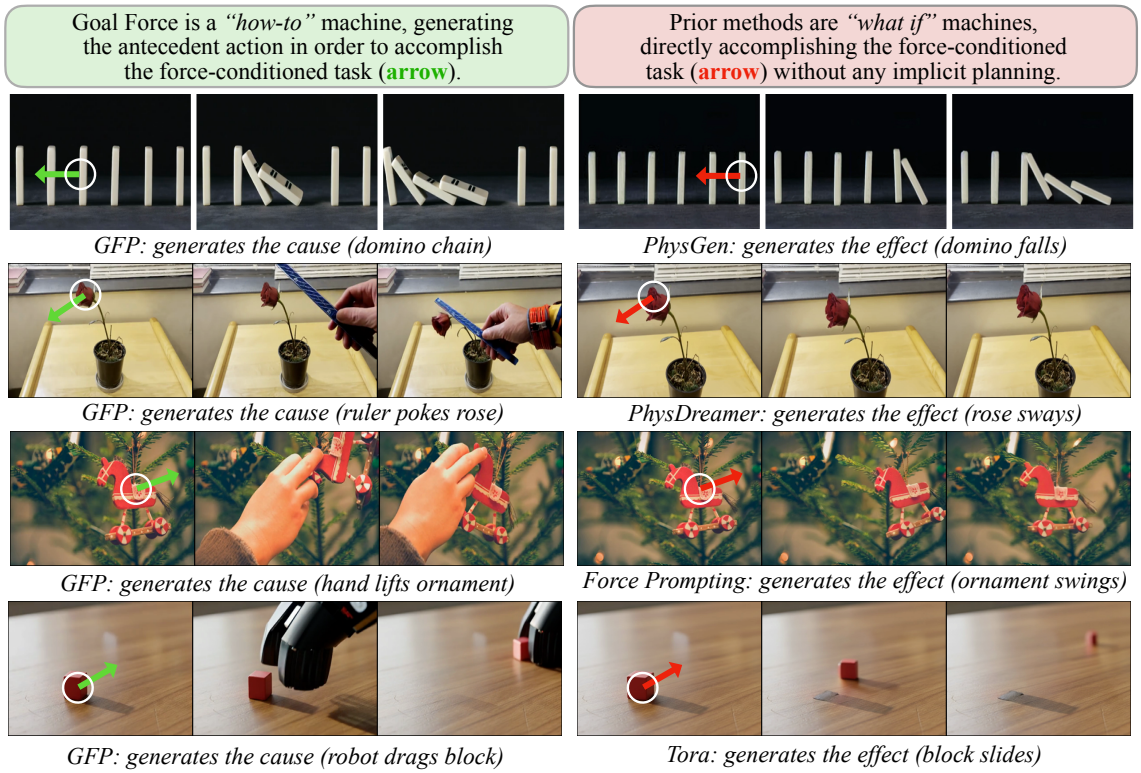


Figure 4.4: In prior methods (right), the user provides a force, and the model directly applies the force to the target object. In our method (left), the user provides a goal force, and the model generates the causes that achieve the desired effect on the target object. The top three methods (PhysGen [76], PhysDreamer [140], and Force Prompting [37]) all accept forces as conditioning; the fourth method, Tora [141], accepts trajectories rather than forces, so we condition on an acceptable trajectory.

force" on a target object (make ball B move right). The model's task is to generate a video v that synthesizes a physically-plausible *antecedent causal chain* (ball A striking ball B) to achieve that goal.

We achieve this by training a video generative model to act as an implicit neural physics planner. The core of our approach is a novel training paradigm built on a multi-channel physics control signal and a curriculum of synthetic data.

4.2.1 Multi-Channel Physics Control Signal

We introduce a 3-channel physics control tensor $\tilde{\pi} \in \mathbb{R}^{f\times 3\times h\times w}$, where f is the number of frames, h and w are the spatial dimensions, and each of the 3 channels encodes a specific physical property. This tensor $\tilde{\pi}$ is the spatial-temporal encoding of the abstract user prompt.

Channel 0: Direct Force. Encodes an immediate, direct force (the "cause"). Following [37], we represent this as a "moving Gaussian blob" video, where the blob's trajectory and duration are affinely proportional to the force vector (location, angle, and magnitude).

Channel 1: Goal Force. Encodes the desired *outcome* (the "effect") on a target object. This channel uses the *same* moving Gaussian blob representation to specify the desired force (and resulting motion) on the *target* object. We visualize the practical difference between a Goal force and a Direct force in Figure 4.3.

Channel 2: Mass. Encodes privileged physical information, such as relative object mass. We represent this as a *static* Gaussian blob in this channel, centered on the object, with a radius affinely proportional to its mass. The mass signal is optional, and offers an interface for users to provide more fine-grained, object-level physical properties, when they are available. When not provided, Goal Force can instead resort to the physical priors encoded in video generative models themselves, a behavior referred to as "mass understanding" in [37].

Force and Mass Normalization. We note that force and mass values are not calibrated to an absolute physical scale. Instead, they follow an intuitive, relative scale normalized *within* each synthetic dataset (dominos, balls, plants). Our model learns this relative concept, as the Gaussian blob encoding is also defined proportionally to the value range of a given domain. This allows the model to generalize the *idea* of force (*e.g.*, "small poke" vs. "large poke") without requiring a unified, absolute scale.

4.2.2 Goal Reaching via Implicit Planning

We train the model on a synthetic dataset of simple causal chains (colliding balls, falling dominos) and complex dynamics (swaying flowers), generated using Blender and PhysDreamer [140]. This dataset contains three scenarios:

- **Dominos (3k videos):** Generated in Blender, these videos show a line of dominos where a direct force on one initiates a chain reaction, linking the "cause" to a "goal force" on a downstream domino.
- **Rolling Balls (6k videos):** Blender scenes of multiple balls. A direct force is applied to a "projectile" ball, which is aimed to either collide with a "target" ball (4.5k videos) or miss it (1.5k videos).
- **PhysDreamer Carnation (3k videos):** Videos of a flower swaying after being poked, generated with PhysDreamer [140], a method that integrates 3D Gaussians and a physics simulator. This component teaches the model complex, non-rigid dynamics from a direct force.

Full data generation details are in Appendix A.17.2.

This synthetic data for the ball collisions and domino collisions provides ground-truth pairs of (direct force, resulting goal force). Our key training strategy is to **randomly mask the causal information**. For each training video, we provide *either* the direct force (in Ch 0) *or* the goal force (in Ch 1), zeroing out the other.

Benchmark	Two Object Collision			Multi Object Collision			Human Object Interaction			Tool Object Interaction		
	Force Adh.	*Real. Motion*	*Visual Qual.*	*Force Adh.*	*Real. Motion*	*Visual Qual.*	*Force Adh.*	*Real. Motion*	*Visual Qual.*	*Force Adh.*	*Real. Motion*	*Visual Qual.*
Text-only, zero-shot	73.4%	67.2%	66.0%	72.0%	69.0%	66.8%	70.5%	47.5%	48.9%	74.5%	61.6%	58.7%
Text-only, fine-tuned	66.9%	60.3%	60.7%	67.0%	57.5%	61.0%	56.8%	48.4%	50.4%	60.3%	60.3%	55.8%

Table 4.1: **Human study comparing Goal Force method to text-only baselines.** Numbers indicate the percentage of human pairwise preferences for Goal Force Prompting over each text-only baseline on each benchmark dataset. The proposed model consistently yields superior goal force adherence against both baselines, with minimal degradation of motion realism and visual quality.

(And in scenes without collisions, namely 1/4 of the ball scenes and all the plant scenes, we only provide the direct force in Ch 0.) This forces the model to learn the physical reasoning:

- **Goal $\rightarrow$ Plan:** Given a goal force, the model must infer and generate the antecedent direct-force event.
- **Action $\rightarrow$ Outcome:** Given a direct force, the model must simulate the resulting collision and secondary force.

The mass channel (Ch 2) is also randomly masked during training. This teaches the model to leverage privileged physics information when available but also to rely on its internal, learned physics prior to estimate properties (like mass) from appearance when it is not. The text prompt's role is to set the semantic context (*e.g.*, "a pool table") and guide the model toward a plausible distribution of videos. It does not, however, specify the low-level causal plan, such as which ball should strike another. This ambiguity is intentional: it forces the model to leverage its internal prior to plan a valid antecedent action, constrained only by the specific objective of the goal force prompt.

4.2.3 Architecture and Training Details

We build our model on Wan2.2 [121], a Mixture-of-Experts diffusion model. We use a ControlNet [139] module to condition on our physics signal $\tilde{\pi}$. We fine-tune this ControlNet only for the high-noise expert, as this expert is primarily responsible for global structure and low-frequency dynamics [29], which aligns with our physics-planning task. The ControlNet module clones the first 10 DiT layers from the pretrained Wan2.2, fine-tuning them and feeding their outputs to the frozen base model via zero-convolutions. We encode the goal force prompt π using the frozen Wan2.2 encoder and pass the result through a randomly initialized patch embedding layer before feeding it to the ControlNet DiT layers. We fine-tune the model for 3,000 steps with an effective batch size of 4 (1 per device on 4 NVIDIA 80GB A100s), which completes in under 48 hours. We use videos of 81 frames at 16 FPS during training and inference.

4.3 Experimental Comparisons

4.3.1 Comparison to Text-Only Baselines

To evaluate Goal Force, we first compare to baselines that use text-only conditioning. We create a new benchmark of 75 challenging scenes curated from permissively licensed web sources {Pexels, Pixabay, Unsplash} as well as generative models {Nano-banana, GPT-Image-1}. We then conduct a 2AFC human study ($N = 40$) on Prolific, comparing our full Goal Force model against those baselines.

Baselines. We compare against two models:

1. *Text-only (Zero-shot):* Wan2.2 base model, prompted with a text suffix, *e.g.*, *"...a golf ball rolls across the grass, colliding with another ball. The secondary object is moved with very strong force to the left."*.

2. *Text-only (Fine-tuned):* Our ControlNet architecture finetuned on our synthetic data, but with the physics control signal zeroed out, relying only on the text suffixes provided during training.

Human Study for Generalization. Our benchmark spans four categories of increasing generalization from our training data: (1) two-object collisions (cantaloupes, pendulum striking object, pool ball, rubber duck toys in water, bars of soap, soccer balls, softballs), (2) multi-object collisions (ball colliding with domino, golf balls, tennis balls) (3) human-object interaction (hand interacting with ornaments, toy car; we also include in this category a dog interacting with a ball, and a cat knocking over a chess piece), and (4) tool-object interaction (golf club hitting golf ball, and a fork touching a dome of jello). Participants evaluated videos on three axes: *Goal Force Adherence* (Does the video accomplish the specified goal?), *Realistic Motion*, and *Visual Quality*.

Table 4.1 compares the performance of Goal Force against the text-only baselines. These results demonstrate that our model outperforms both baselines on goal force adherence, demonstrating that the text prompt is not sufficient, confirming that the explicit physics control signal is critical for solving the task. The results also demonstrate that this goal force adherence is achieved with minimal degradation of visual quality and motion realism. Despite training only on synthetic balls, dominos, and a single flower, our model generalizes effectively, enabling complex, out-of-domain interactions like tool use and human-object planning, as visualized in Figure 4.2.

4.3.2 Comparison to Prior Methods

The Goal Force prompting task is new, and prior force-conditioned methods (*e.g.*, PhysGen [76], PhysDreamer [140], and Force Prompting [37]) are not designed to solve it. These models can only simulate a *direct* force (the effect), not plan the *antecedent* action required to achieve a *goal* force (the cause). As shown qualitatively in Figure 4.4, when given a goal force prompt, those prior methods misinterpret it as a direct, non-causal poke on the target object. Similarly, motion-conditioned models like ToRA [142] can follow a specified trajectory but fail to adhere to causality, often moving the target object before an antecedent event (like a hand) arrives. A qualitative comparison against these prior works is provided in Figure 4.4. Also, while prior methods cannot perform Goal Force prompting, our model is still capable of performing direct Force Prompting (FP).

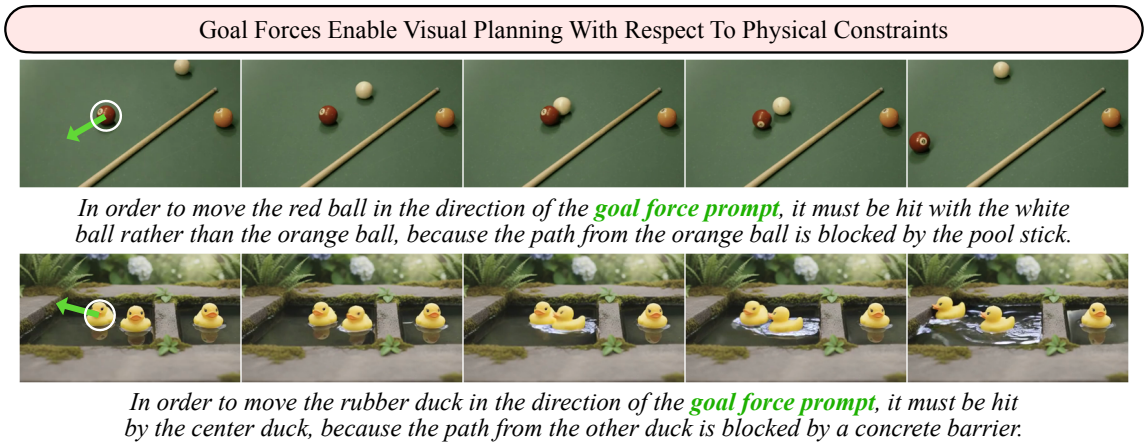


Figure 4.5: **Given a goal force prompt, the model chooses the physically correct way to execute it.** Top: even though there exist multiple plausible initiators, the model correctly selects the white ball as the initiator to achieve the desired force on the target. Bottom: With multiple plausible rubber ducks that could initiate the force, the model selects the initiator that is not blocked by a physical barrier.

4.4 Goal Force Enables Visual Planning

We now evaluate a core claim of our work: that Goal Force enables a form of visual planning. We test this by analyzing three key properties of the generated plans: their physical accuracy, their diversity, and their awareness of privileged physics information such as mass.

4.4.1 Visual Plans are Accurate

We first test if the model's visual planning adheres to physical constraints. We create a benchmark of 22 scenes containing "natural blockers" (Figure 4.5), where each scene features target objects in a contextually appropriate environment, and distractor objects are physically constrained from initiating the goal force. A successful plan requires the model to identify and select a valid, unconstrained object to execute the causal chain.

For each scene, we generate 50 videos. To isolate the planning logic from the base video diffusion model's artifacts, we filter out trials exhibiting stochastic visual degradation (*e.g.*, object hallucination) prior to

Table 4.2: **Visual planning accuracy across scenes.** Our model achieves a high success rate in selecting a physically valid force initiator across diverse, complex scenarios. List of all scenes is in Table A8.

Scene	# Valid	# Success	% Accuracy
Pool	49	48	97.96
Paper Balls	50	49	98.00
Kitchen Lemon	50	50	100.00
Coffee Cups	44	41	93.18
Duckie	40	34	85.00
Accessories	50	47	94.00
Curling Stones	49	37	75.51
Rubik's Cube	49	46	93.88

analysis. We define **accuracy** as the percentage of valid trials where the goal force is initiated by the correct, unconstrained object, rather than by a distractor or through spontaneous, non-causal motion.

Results. We report accuracy for selected scenes in Table 4.2. Results for all test scenes are included in Table A8, and Figure A37 presents visualizations of each scene. A random baseline achieves at most 33.3% accuracy given our distractor design. The model demonstrates strong physical reasoning across most of the scenes. In the pool example (Figure 4.5, top), a stick blocks the orange ball. Our model correctly selects the white ball as the initiator in 98% of valid trials. On the rubber duckie benchmark (Figure 4.5, bottom), it selects the correct initiator. We observe that most failure cases involve the target object moving spontaneously, rather than the model choosing an incorrect, constrained initiator. We also observe this trend of physically grounded visual planning generalizes to other natural scenarios, including the ones shown in Figure 4.2.

4.4.2 Visual Plans are Diverse

Beyond accuracy, we test if our model produces a *diverse* set of valid plans rather than suffering from mode collapse. We design a multi-modal task: a line of six dominos where the goal is to topple the rightmost (sixth) domino block. This goal can be achieved by initiating a chain reaction from any of the five preceding dominos. A deterministic model would repeatedly target the same domino, whereas we hypothesize Goal Force will sample from the full distribution of valid plans.

To quantify this, we propose a diversity metric $\delta(p)$ based on the Jensen-Shannon Divergence (JSD). Let $\hat{p}(x)$ be the empirical probability mass function (PMF) over the set of the $N = 5$ targetable dominos, $S = \{0, 1, 2, 3, 4\}$. We define our **diversity metric** as:

$$\delta(p) = 1 - \text{JSD}(\hat{p} \parallel \text{Unif}(S)). \tag{4.4.1}$$

This metric is normalized to provide an interpretable score. A perfectly diverse model sampling uniformly from all 5 dominos ($\hat{p} = \text{Unif}(S)$) achieves the maximum score of $\delta(p) = 1.0$. Conversely, a fully deterministic model exhibiting complete mode collapse (i.e., $\hat{p}$ is a Dirac delta function on a single domino) yields the baseline score of $\delta(p) \approx 0.39$.

Results. We present our findings in Table 4.3. Across 26 random seeds, our model achieves a diversity score of 0.6577, significantly higher than the deterministic baseline (0.3900) and distinct from distributions with

Table 4.3: **Diversity metric ($\delta(p)$) scores for the 5-domino task.** Higher is better (Max: 1.0). Our model (0.6577) shows significant diversity compared to the deterministic baseline (0.3900).

Distribution (p)	**Score ($\delta(p)$)**
Our Model (Goal Force)	**0.6577**
Reference: Unif{0..4} (Max diversity)	1.0000
Reference: Unif{0..3}	0.8920
Reference: Unif{0..2}	0.7635
Reference: Unif{0..1}	0.6042
Reference: Unif{0} (Deterministic)	0.3900

collapsed support (*e.g.*, $\mathrm{Unif}\{0, 1\}$). This demonstrates that our model successfully explores a multi-modal distribution of valid plans rather than collapsing to a single solution.

4.4.3 Visual Plans Leverage Privileged Physics

Next, we test if the model's visual plans can use privileged mass information provided in the control signal to help guide their plans. Our experiments focus on ball collision. In this setting, a physically-grounded plan must account for mass; for example, achieving a specific goal force on a heavier target requires a stronger impact.

We design a ball collision task with a fixed goal force magnitude, varying the projectile and target masses. In Figure 4.6, we test our model on two such scenarios. One is in-distribution with a scene and viewpoint similar to our training data. The other features an out-of-distribution background, viewpoint, lighting, and ball size. We expect the model to learn two principles: (1) if projectile mass is constant, a heavier target requires a faster projectile; (2) if target mass is constant, a heavier projectile can move slower.

To quantitatively measure the ball collision, we use Faster R-CNN [99] to detect the positions of the two balls. Then we determine the collision time and compute the projectile's moving speed accordingly. We generate 15 videos for each combination of masses and average the speed over the samples. We observe that in the in-distribution scenario, the projectile's speeds satisfy all four desired speed magnitude relationships. In the out-of-distribution scenario, our results satisfy three of them, while the fourth is very close. This demonstrates the model's capability in leveraging privileged physics information for visual planning.

4.5 Conclusion

We introduce Goal Force, a paradigm that shifts generative video control from specifying a direct force (the cause) to declaring a desired goal force (the effect). We demonstrate that by training on simple, synthetic causal primitives, a video model can learn to function as an implicit neural physics planner. This enables the model to reason backward from a user-defined goal and generate a physically plausible, antecedent causal chain to achieve it. Our key finding is that this planning capability generalizes to complex, out-of-domain scenarios involving tool use and human-object interactions. This work represents a step toward interactive

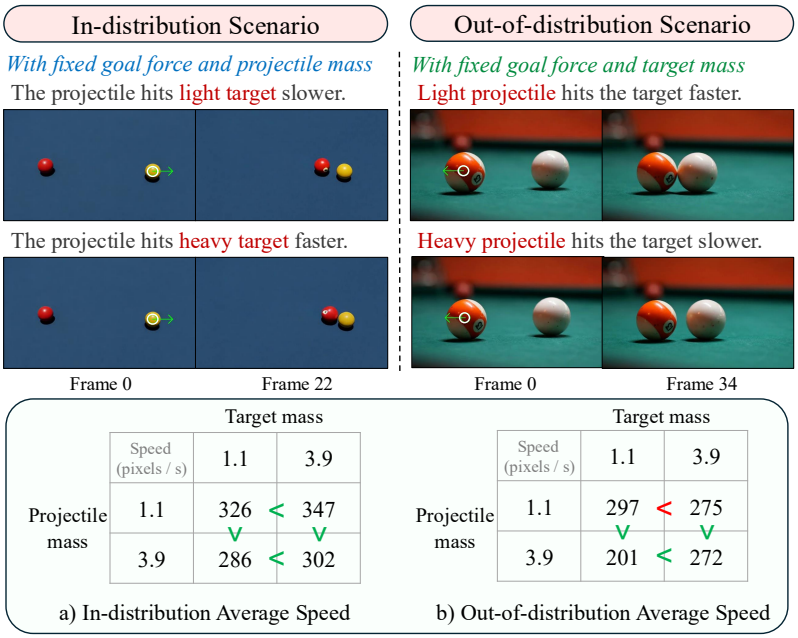


Figure 4.6: **Visual plans take advantage of mass information.** We test goal force prompting on in-distribution (left) and out-of-distribution (right) scenarios. In both scenarios, our model can adjust the moving speed of the projectile accordingly when the object masses are changed to cause the desired force magnitude. The direction of the "<" sign indicates the desired numerical relationship; green indicates satisfaction, red indicates violation.

world models that can not only simulate a physical reaction but also reason about and plan the actions required to achieve a desired physical outcome.

Bibliography

[1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

[2] 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.

[3] Luca Savant Aira, Antonio Montanaro, Emanuele Aiello, Diego Valsesia, and Enrico Magli. Motioncraft: Physics-based zero-shot video generation. *arXiv preprint arXiv:2405.13557*, 2024.

[4] Anurag Ajay, Seungwook Han, Yilun Du, Shuang Li, Abhi Gupta, Tommi Jaakkola, Josh Tenenbaum, Leslie Kaelbling, Akash Srivastava, and Pulkit Agrawal. Compositional foundation models for hierarchical planning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[5] Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, and Richard Baraniuk. Self-consuming generative models go MAD. In *The Twelfth International Conference on Learning Representations*, 2024.

[6] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Syndar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=gerNCVqqtR.

[7] Shekoofeh Azizi, Simon Kornblith, Chitwan Saharia, Mohammad Norouzi, and David J. Fleet. Synthetic data from diffusion models improves imagenet classification. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

[8] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[9] Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. *arXiv preprint arXiv:2412.03572*, 2024.

[10] Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, et al. Lumiere: A space-time diffusion model for video generation. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024.

[11] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the national academy of sciences*, 110(45):18327–18332, 2013.

[12] 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.

[13] Quentin Bertrand, Avishek Joey Bose, Alexandre Duplessis, Marco Jiralerspong, and Gauthier Gidel. On the stability of iterative retraining of generative models on their own data. In *The Twelfth International Conference on Learning Representations*, 2024.

[14] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[15] Peter J Brockwell and Richard A Davis. *Time series: theory and methods*. Springer science & business media, 1991.

[16] Tim Brooks, Bill Peebles, Connor Homes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators, 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

[17] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[18] Haoxuan Che, Xuanhua He, Quande Liu, Cheng Jin, and Hao Chen. Gamegen-x: Interactive open-world game video generation. *arXiv preprint arXiv:2411.00769*, 2024.

[19] Boyuan Chen, Hanxiao Jiang, Shaowei Liu, Saurabh Gupta, Yunzhu Li, Hao Zhao, and Shenlong Wang. Physgen3d: Crafting a miniature interactive world from a single image. *CVPR*, 2025.

[20] Hsiao-yu Chen, Edith Tretschk, Tuur Stuyck, Petr Kadlecek, Ladislav Kavan, Etienne Vouga, and Christoph Lassner. Virtual elastic objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[21] 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.

[22] Tsai-Shien Chen, Chieh Hubert Lin, Hung-Yu Tseng, Tsung-Yi Lin, and Ming-Hsuan Yang. Motion-conditioned diffusion model for controllable video synthesis. *arXiv preprint arXiv:2304.14404*, 2023.

[23] 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.

[24] 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.

[25] Abe Davis, Katherine L Bouman, Justin G Chen, Michael Rubinstein, Fredo Durand, and William T Freeman. Visual vibrometry: Estimating material properties from small motion in video. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 2015.

[26] Abe Davis, Justin G Chen, and Frédo Durand. Image-space modal bases for plausible manipulation of objects in video. *ACM Transactions on Graphics (TOG)*, 2015.

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

[28] 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.

[29] Sander Dieleman. Diffusion is spectral autoregression. Blog post, September 2024. URL https://sander.ai/2024/09/02/spectral-autoregression.html. Accessed: YYYY-MM-DD.

[30] Yilun Du, Sherry Yang, Pete Florence, Fei Xia, Ayzaan Wahid, brian ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B. Tenenbaum, Leslie Pack Kaelbling, Andy Zeng, and Jonathan Tompson. Video language planning. In *International Conference on Learning Representations (ICLR)*, 2024.

[31] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[32] Alejandro Escontrela, Ademi Adeniji, Wilson Yan, Ajay Jain, Xue Bin Peng, Ken Goldberg, Youngwoon Lee, Danijar Hafner, and Pieter Abbeel. Video prediction models as rewards for reinforcement learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[33] Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024.

[34] 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.

[35] Daniel Geng, Charles Herrmann, Junhwa Hur, Forrester Cole, Serena Zhang, Tobias Pfaff, Tatiana Lopez-Guevara, Carl Doersch, Yusuf Aytar, Michael Rubinstein, Chen Sun, Oliver Wang, Andrew Owens, and Deqing Sun. Motion prompting: Controlling video generation with motion trajectories. *arXiv preprint arXiv:2412.02700*, 2024.

[36] Saeed Ghorbani, Kimia Mahdaviani, Anne Thaler, Konrad Kording, Douglas James Cook, Gunnar Blohm, and Nikolaus F Troje. Movi: A large multi-purpose human motion and video dataset. *Plos one*, 16(6):e0253157, 2021.

[37] Nate Gillman, Charles Herrmann, Michael Freeman, Daksh Aggarwal, Evan Luo, Deqing Sun, and Chen Sun. Force prompting: Video generation models can learn and generalize physics-based control signals, 2025. URL https://arxiv.org/abs/2505.19386.

[38] Rohit Girdhar, Mannat Singh, Andrew Brown, Quentin Duval, Samaneh Azadi, Sai Saketh Rambhatla, Akbar Shah, Xi Yin, Devi Parikh, and Ishan Misra. Emu video: Factorizing text-to-video generation by explicit image conditioning. *arXiv preprint arXiv:2311.10709*, 2023.

[39] 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.

[40] Alison Gopnik and David M Sobel. Detecting blickets: How young children use information about novel causal powers in categorization and induction. *Child development*, 71(5):1205–1222, 2000.

[41] Jia Gu, Liang Pang, Huawei Shen, and Xueqi Cheng. Do llms play dice? exploring probability distribution sampling in large language models for behavioral simulation. *arXiv preprint arXiv:2404.09043*, 2024.

[42] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5152–5161, 6 2022.

[43] Ziyu Guo, Xinyan Chen, Renrui Zhang, Ruichuan An, Yu Qi, Dongzhi Jiang, Xiangtai Li, Manyuan Zhang, Hongsheng Li, and Pheng-Ann Heng. Are video models ready as zero-shot reasoners? an empirical study with the MME-CoF benchmark. *arxiv*, October 2025.

[44] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[45] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for text-to-video generation, 2024.

[46] 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*, pages 1026–1034, 2015.

[47] Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition? In *ICLR*, 2023.

[48] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[49] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[50] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[51] 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.

[52] Tao Huang, Guangqi Jiang, Yanjie Ze, and Huazhe Xu. Diffusion reward: Learning rewards via conditional video diffusion. *arXiv preprint arXiv:2312.14134*, 2023.

[53] Tianyu Huang, Haoze Zhang, Yihan Zeng, Zhilu Zhang, Hui Li, Wangmeng Zuo, and Rynson WH Lau. Dreamphysics: Learning physical properties of dynamic 3d gaussians with video diffusion priors. *arXiv preprint arXiv:2406.01476*, 2024.

[54] 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.

[55] Ali Jahanian, Xavier Puig, Yonglong Tian, and Phillip Isola. Generative models as a data source for multiview representation learning. In *International Conference on Learning Representations*, 2022.

[56] Daniel Jakubovitz, Raja Giryes, and Miguel RD Rodrigues. Generalization error in deep learning. In *Compressed Sensing and Its Applications: Third International MATHEON Conference 2017*, pages 153–193. Springer, 2019.

[57] Kaiyi Ji, Yi Zhou, and Yingbin Liang. Understanding estimation and generalization error of generative adversarial networks. *IEEE Transactions on Information Theory*, 67(5):3114–3129, 2021.

[58] 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.

[59] 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.

[60] Bingyi Kang, Yang Yue, Rui Lu, Zhijie Lin, Yang Zhao, Kaixin Wang, Gao Huang, and Jiashi Feng. How far is video generation from world model: A physical law perspective. *arxiv*, June 2025.

[61] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[62] Denis Karachev and Yuancheng Xu. Cogvideox controlnet extention. https://github.com/TheDenk/cogvideox-controlnet, 2025.

[63] Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B Tenenbaum. Learning to act from actionless videos through dense correspondences. *arXiv preprint arXiv:2310.08576*, 2023.

[64] Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B. Tenenbaum. Learning to act from actionless videos through dense correspondences. In *International Conference on Learning Representations (ICLR)*, 2024.

[65] Simon Le Cleac'h, Hong-Xing Yu, Michelle Guo, Taylor Howell, Ruohan Gao, Jiajun Wu, Zachary Manchester, and Mac Schwager. Differentiable physics simulation of dynamics-augmented neural objects. *IEEE Robotics and Automation Letters*, 2023.

[66] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[67] Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

[68] Chenyu Li, Oscar Michel, Xichen Pan, Sainan Liu, Mike Roberts, and Saining Xie. Pisa experiments: Exploring physics post-training for video diffusion models by watching stuff drop. *arXiv preprint arXiv:2503.09595*, 2025.

[69] 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.

[70] Yaowei Li, Xintao Wang, Zhaoyang Zhang, Zhouxia Wang, Ziyang Yuan, Liangbin Xie, Yuexian Zou, and Ying Shan. Image conductor: Precision control for interactive video synthesis. *arXiv preprint arXiv:2406.15339*, 2024.

[71] Zhengqi Li, Richard Tucker, Noah Snavely, and Aleksander Holynski. Generative image dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[72] Zizhang Li, Hong-Xing Yu, Wei Liu, Yin Yang, Charles Herrmann, Gordon Wetzstein, and Jiajun Wu. Wonderplay: Dynamic 3d scene generation from a single image and actions. *arXiv preprint arXiv:2505.18151*, 2025.

[73] Junbang Liang, Ruoshi Liu, Ege Ozguroglu, Sruthi Sudhakar, Achal Dave, Pavel Tokmakov, Shuran Song, and Carl Vondrick. Dreamitate: Real-world visuomotor policy learning via video generation. *arXiv preprint arXiv:2406.16862*, 2024.

[74] Jiajing Lin, Zhenzhong Wang, Shu Jiang, Yongjie Hou, and Min Jiang. Phys4dgen: A physics-driven framework for controllable and efficient 4d content generation from a single image. *arXiv preprint arXiv:2411.16800*, 2024.

[75] Fangfu Liu, Hanyang Wang, Shunyu Yao, Shengjun Zhang, Jie Zhou, and Yueqi Duan. Physics3d: Learning physical properties of 3d gaussians via video diffusion. *arXiv preprint arXiv:2406.04338*, 2024.

[76] Shaowei Liu, Zhongzheng Ren, Saurabh Gupta, and Shenlong Wang. Physgen: Rigid-body physics-grounded image-to-video generation. In *ECCV*, 2024.

[77] 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.

[78] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[79] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[80] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[81] Calvin Luo, Zilai Zeng, Yilun Du, and Chen Sun. Solving new tasks by adapting internet video knowledge. *arXiv preprint arXiv:2504.15369*, 2025.

[82] Zhengyi Luo, Ryo Hachiuma, Ye Yuan, and Kris Kitani. Dynamics-regulated kinematic policy for egocentric pose estimation. In *Advances in Neural Information Processing Systems*, 2021.

[83] 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.

[84] Gonzalo Martínez, Lauren Watson, Pedro Reviriego, José Alberto Hernández, Marc Juarez, and Rik Sarkar. Towards understanding the interplay of generative artificial intelligence and the internet. In Fabio Cuzzolin and Maryam Sultana, editors, *Epistemic Uncertainty in Artificial Intelligence*, pages 59–73, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-57963-9.

[85] Robert McCarthy, Daniel C.H. Tan, Dominik Schmidt, Fernando Acero, Nathan Herr, Yilun Du, Thomas George Thuruthel, and Zhibin Li. Towards generalist robot learning from internet video: A survey. *arXiv preprint arXiv:2404.19664*, 2024.

[86] 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.

[87] 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.

[88] Koichi Namekata, Sherwin Bahmani, Ziyi Wu, Yash Kant, Igor Gilitschenski, and David B Lindell. Sg-i2v: Self-guided trajectory control in image-to-video generation. *arXiv preprint arXiv:2411.04989*, 2024.

[89] 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.

[90] Muyao Niu, Xiaodong Cun, Xintao Wang, Yong Zhang, Ying Shan, and Yinqiang Zheng. Mofa-video: Controllable image animation via generative motion field adaptions in frozen image-to-video diffusion model. *arXiv preprint arXiv:2405.20222*, 2024.

[91] Akshay Paruchuri, Jake Garrison, Shun Liao, John Hernandez, Jacob Sunshine, Tim Althoff, Xin Liu, and Daniel McDuff. What are the odds? language models are capable of probabilistic reasoning. *arXiv preprint arXiv:2406.12830*, 2024.

[92] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[93] Juan Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, and Moritz Hardt. Performative prediction. In *International Conference on Machine Learning*, pages 7599–7609. PMLR, 2020.

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

[95] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[96] 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.

[97] 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, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 840–854, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.59.

[98] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[99] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

[100] 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.

[101] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[102] William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

[103] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

[104] Xiaoyu Shi, Zhaoyang Huang, Fu-Yun Wang, Weikang Bian, Dasong Li, Yi Zhang, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, et al. Motion-i2v: Consistent and controllable image-to-video generation with explicit motion modeling. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

[105] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arxiv:2305.17493*, 2023.

[106] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

[107] 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.

[108] 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.

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

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

[111] Elias M Stein and Rami Shakarchi. *Real Analysis: Measure Theory, Integration, and Hilbert Spaces*, volume 3. Princeton University Press, 2005.

[112] Wenqiang Sun, Shuo Chen, Fangfu Liu, Zilong Chen, Yueqi Duan, Jun Zhang, and Yikai Wang. Dimensionx: Create any 3d and 4d scenes from a single image with controllable video diffusion. *arXiv preprint arXiv:2411.04928*, 2024.

[113] Xiyang Tan, Ying Jiang, Xuan Li, Zeshun Zong, Tianyi Xie, Yin Yang, and Chenfanfu Jiang. Physmotion: Physics-grounded dynamics from a single image. *arXiv preprint arXiv:2411.17189*, 2024.

[114] Terry Tao, Dec 2014. URL https://terrytao.wordpress.com/2014/12/09/254a-notes-2-complex-analytic-multiplicative-number-theory/#nxx.

[115] Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023.

[116] Yonglong Tian, Lijie Fan, Kaifeng Chen, Dina Katabi, Dilip Krishnan, and Phillip Isola. Learning vision from models rivals learning vision from data. *arXiv preprint arXiv:2312.17742*, 2023.

[117] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

[118] Tomer D Ullman, Elizabeth Spelke, Peter Battaglia, and Joshua B Tenenbaum. Mind games: Game engines as an architecture for intuitive physics. *Trends in cognitive sciences*, 21(9):649–665, 2017.

[119] 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.

[120] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. *arXiv preprint arXiv:2408.14837*, 2024.

[121] Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui,

Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[122] Chen Wang*, Chuhao Chen*, Yiming Huang, Zhiyang Dou, Yuan Liu, Jiatao Gu, and Lingjie Liu. Physctrl: Generative physics for controllable and physics-grounded video generation. In *NeurIPS*, 2025.

[123] 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.

[124] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video generation. In *ACM SIGGRAPH 2024 Conference Papers*, 2024.

[125] Megan Wei, Michael Freeman, Chris Donahue, and Chen Sun. Do music generation models encode music theory? In *International Society for Music Information Retrieval*, 2024.

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

[127] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations*, 2023.

[128] Friedrich Wilkening and Trix Cacchione. Children's intuitive physics. *The Wiley-Blackwell handbook of childhood cognitive development*, pages 473–496, 2010.

[129] Tsung-Han Wu, Long Lian, Joseph E Gonzalez, Boyi Li, and Trevor Darrell. Self-correcting llm-controlled diffusion models. *arXiv preprint arXiv:2311.16090*, 2023.

[130] Weijia Wu, Zhuang Li, Yuchao Gu, Rui Zhao, Yefei He, David Junhao Zhang, Mike Zheng Shou, Yan Li, Tingting Gao, and Di Zhang. Draganything: Motion control for anything using entity representation. In *European Conference on Computer Vision*, 2024.

[131] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *CVPR*, 2024.

[132] Sirui Xu, Zhengyuan Li, Yu-Xiong Wang, and Liang-Yan Gui. Interdiff: Generating 3d human-object interactions with physics-informed diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14928–14940, 2023.

[133] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 1(2):6, 2023.

[134] Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, Song Han, and Yukang Chen. LongLive: Real-time interactive long video generation. *arxiv*, October 2025.

[135] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

[136] Shengming Yin, Chenfei Wu, Jian Liang, Jie Shi, Houqiang Li, Gong Ming, and Nan Duan. Dragnuwa: Fine-grained control in video generation by integrating text, image, and trajectory. *arXiv preprint arXiv:2308.08089*, 2023.

[137] 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.

[138] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[139] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[140] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T Freeman. Physdreamer: Physics-based interaction with 3d objects via video generation. In *ECCV*, 2024.

[141] Zhenghao Zhang, Junchao Liao, Menghao Li, Zuozhuo Dai, Bingxue Qiu, Siyu Zhu, Long Qin, and Weizhi Wang. Tora: Trajectory-oriented diffusion transformer for video generation. *arXiv preprint arXiv:2407.21705*, 2024.

[142] Zhenghao Zhang, Junchao Liao, Menghao Li, Zuozhuo Dai, Bingxue Qiu, Siyu Zhu, Long Qin, and Weizhi Wang. Tora: Trajectory-oriented diffusion transformer for video generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2063–2073, 2025.

[143] Guangcong Zheng, Teng Li, Rui Jiang, Yehao Lu, Tao Wu, and Xi Li. Cami2v: Camera-controlled image-to-video diffusion model. *arXiv preprint arXiv:2410.15957*, 2024.

[144] Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and simulation of elastic objects with spring-mass 3d gaussians. In *European Conference on Computer Vision*, 2024.

[145] Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, D. Y. Yeung, and Chuang Gan. Robodreamer: Learning compositional world models for robot imagination. *arXiv preprint arXiv:2404.12377*, 2024.

[146] 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.

Appendix

A.1 SCSC: Mathematical Theory: The Proof of Theorem 1.3.3

In this appendix, we provide a full account of the mathematical details of the theorems and their proofs appearing in the main body of Chapter 1. Our proof technique has the same framework as [13] because our theoretical analysis generalizes theirs to the case where you have a self-correction function in the self-consuming loop.

A.1.1 Mathematical Setup and Notation

Definition A.1.1. *Define the optimal model parameters to be*

$$\theta^\star \in \arg\max_{\theta' \in \Theta} \mathbb{E}_{x \sim p_{\text{data}}}[\log p_{\theta'}(x)], \tag{A.1.1}$$

chosen so that $\|\theta^\star\|$ has minimal norm within this set. Let θ be any model parameters. Then the correction of strength γ *of distribution p_θ towards $p_{\theta^\star}$ is a new distribution, denoted $\pi_\gamma p_\theta$, defined according to the rule*

$$\pi_\gamma p_\theta(x) := \frac{p_\theta(x) + \gamma p_{\theta^\star}(x)}{1+\gamma}.$$

This is illustrated in Figure A1. Let θ_t be the parameters of the model trained after t generations. We define the iterative fine-tuning with correction update mapping *to be*

$$\pi_\gamma \mathcal{G}_\lambda^\infty(\theta) := \operatorname*{local\,argmax}_{\theta' \in \Theta} \mathcal{H}(\theta, \theta') := \operatorname*{local\,argmax}_{\theta' \in \Theta} [\mathbb{E}_{x \sim p_{\text{data}}}[\log p_{\theta'}(x)]] + \lambda \mathbb{E}_{x \sim \pi_\gamma p_\theta}[\log p_{\theta'}(x)]] \tag{A.1.2}$$

$$\pi_\gamma \mathcal{G}_\lambda^n(\theta) := \operatorname*{local\,argmax}_{\theta' \in \Theta} \hat{\mathcal{H}}(\theta, \theta') := \operatorname*{local\,argmax}_{\theta' \in \Theta} [\mathbb{E}_{x \sim \hat{p}_{\text{data}}}[\log p_{\theta'}(x)]] + \lambda \mathbb{E}_{x \sim \widehat{\pi_\gamma p_\theta}}[\log p_{\theta'}(x)]]. \tag{A.1.3}$$

Notice that in the finite case, we're optimizing by taking samples from an empirical distribution. In contrast, in the infinite case, there is zero statistical error, since the parameter update is done with access to an infinite sampling budget at each generation t. The finite case is the more practical case, when we have some statistical error (so we only have access to finite sampling at each generation). Since the parameter space of the generative model class might be limited, there might be a small difference between the distribution

corresponding to the optimal parameters and the target distribution p_{data}; we capture this difference via the Wasserstein-2 distance and denote

$$\varepsilon := d_W(p_{\theta^\star}, p_{\text{data}}). \tag{A.1.4}$$

Let

$$\mathcal{H}_1(\theta') := \mathbb{E}_{x\sim p_{\text{data}}}[\log p_{\theta'}(x)], \qquad \mathcal{H}_2(\theta,\theta') := \mathbb{E}_{x\sim \pi_\gamma p_\theta}[\log p_{\theta'}(x)]. \tag{A.1.5}$$

and note that $\mathcal{H}(\theta,\theta') = \mathcal{H}_1(\theta') + \lambda\mathcal{H}_2(\theta,\theta')$.

We first establish that the correction map is truly a mapping of probability distributions as well as some of its elementary properties.

Lemma A.1.2. *The correction map has the following properties.*

1. *$\pi_\gamma p_\theta$ is a probability distribution.*
2. *Strengths $0, 1, \infty$ correspond to p_θ, the average of p_θ and $p_{\theta^\star}$, and $p_{\theta^\star}$, respectively.*
3. *For any $x \in \mathbb{R}^n$, if $\gamma > 1$, then*

$$\|\pi_\gamma p_\theta(x) - p_{\theta^\star}(x)\| \le \|\pi_\gamma p_\theta(x) - p_\theta(x)\|,$$

and if $\gamma < 1$, then the inequality is flipped. In other words, $\pi_\gamma p_\theta$ is a better estimate of the ideal distribution $p_{\theta^\star}$ than p_θ is, precisely when the projection strength is more than 1.

Proof. For the first point, $\pi_\gamma p_\theta$ is a probability distribution because it is a convex combination of probability distributions. For example, we can compute that

$$\int_{\mathbb{R}^d} \pi_\gamma p_\theta dx = \frac{1}{1+\gamma}\int_{\mathbb{R}^d} p_\theta(x)dx + \frac{\gamma}{1+\gamma}\int_{\mathbb{R}^d} p_{\theta^\star}(x)dx = \frac{1}{1+\gamma}\cdot 1 + \frac{\gamma}{1+\gamma}\cdot 1 = 1.$$

The second point follows immediately from the definition of $\pi_\gamma p_\theta$. For the third point, we can estimate that

$$\begin{aligned}
\|\pi_\gamma p_\theta(x) - p_{\theta^\star}(x)\| &= \left\| \frac{p_\theta(x) + \gamma p_{\theta^\star}(x)}{1+\gamma} - \frac{p_{\theta^\star}(x)(1+\gamma)}{1+\gamma} \right\| \\
&= \frac{1}{1+\gamma}\cdot \|p_\theta(x) - p_{\theta^\star}(x)\| \\
&\le \frac{\gamma}{1+\gamma}\cdot \|p_{\theta^\star}(x) - p_\theta(x)\| \\
&= \left\| \frac{p_\theta(x) + \gamma p_{\theta^\star}(x)}{1+\gamma} - \frac{p_\theta(x)(1+\gamma)}{1+\gamma} \right\| \\
&= \|\pi_\gamma p_\theta(x) - p_\theta(x)\|
\end{aligned}$$

when $\gamma > 1$. The inequality flips when $\gamma < 1$. □

Intuitively, it is clear that we cannot hope to prove general results about generative models without assuming something about the mapping $\theta \mapsto p_\theta$. We now state the two assumptions we require in order to make our theoretical arguments; note that they are precisely the same assumptions made in [13]. The first assumption is a local Lipschitzness property that we will exploit via the Kantorovich-Rubenstein duality:

Assumption A.1.3. For θ close enough to $\theta^\star$, the mapping $x \mapsto \nabla_\theta \nabla_\theta \log p_\theta(x)$ is L-Lipschitz.

The second assumption is a local regularity and concavity condition:

Assumption A.1.4. The mapping $\theta \mapsto \mathbb{E}_{x\sim p_{\text{data}}}[\log p_\theta(x)]$ is continuously twice differentiable locally around $\theta^\star$ and $\mathbb{E}_{x\sim p_{\text{data}}}[\nabla_\theta \nabla_\theta \log p_\theta(x)]_{\theta^\star} \preceq -\alpha I_d \prec 0$.

We next show the existence and uniqueness of $\pi_\gamma \mathcal{G}_\lambda^\infty(\infty)$ locally around $\theta^\star$.

Proposition A.1.5 (The Local Maximum Likelihood Solution is Unique)**.** *The following are true:*

A. There exists an open neighborhood $U \subset \mathbb{R}^d$ containing $\theta^\star$ and a continuous function $g : U \to \mathbb{R}^d$ such that $g(\theta^\star) = \theta^\star$, and

$$\nabla_{\theta'} \mathcal{H}(\theta, \theta')|_{\theta, g(\theta)} = 0 \tag{A.1.6}$$

for every $\theta \in U$.

B. Given optimal model parameters $\theta^\star$ as in equation A.1.1 that follow Assumptions A.1.3 and A.1.4, we have that, if $\varepsilon L < \alpha$, then for all $\lambda > 0$ and θ in a small enough neighborhood U around $\theta^\star$, there exists a unique local maximizer $\pi_\gamma \mathcal{G}_\lambda^\infty(\theta)$ in U.

Proof. **We first prove part A.** It suffices to apply the Implicit Function Theorem to the map

$$\mathbb{R}^{2d} \to \mathbb{R}^d : (\theta, \theta') \mapsto \nabla_{\theta'} \mathcal{H}(\theta, \theta')|_{\theta, \theta'} \tag{A.1.7}$$

in an open neighborhood of $(\theta^\star, \theta^\star)$. To do this, we need to show the following:

i) The map vanishes at $(\theta^\star, \theta^\star)$, i.e.

$$\nabla_{\theta'} \mathcal{H}(\theta, \theta')|_{\theta^\star, \theta^\star} = 0. \tag{A.1.8}$$

ii) The Jacobian matrix at $(\theta^\star, \theta^\star)$ is invertible, i.e.,

$$\nabla_{\theta'} \nabla_{\theta'} \mathcal{H}(\theta, \theta')|_{\theta^\star, \theta^\star} \qquad \text{is invertible.} \tag{A.1.9}$$

We first prove i). Recall from the definition equation A.1.2 that $\pi_\gamma \mathcal{G}_\lambda^\infty(\theta) = \arg\max_{\theta' \in \Theta} \mathcal{H}(\theta, \theta')$. This means that for any θ, $\pi_\gamma \mathcal{G}_\lambda^\infty(\theta)$ is the choice of θ' which maximizes $\mathcal{H}(\theta, \theta')$. In particular, for $\theta = \theta^\star$, we have that $\theta' = \pi_\gamma \mathcal{G}_\lambda^\infty(\theta^\star)$ is the choice which maximizes $\mathcal{H}(\theta^\star, \theta')$. But $\pi_\gamma \mathcal{G}_\lambda^\infty(\theta^\star) = \theta^\star$ by Proposition A.1.6. This implies that its derivative is zero at $\theta' = \theta^\star$, meaning $\nabla_{\theta'} \mathcal{H}(\theta, \theta')|_{\theta^\star, \theta^\star} = 0$, as needed.

Now we prove ii). In order to show that the matrix equation A.1.9 is invertible, it suffices to show it is close to another matrix which is invertible. A natural choice is the matrix

$$M = (1 + \lambda) \nabla_{\theta'} \nabla_{\theta'} \mathbb{E}_{x\sim p_{\text{data}}}[\log p_{\theta'}(x)]|_{\theta^\star}. \tag{A.1.10}$$

First, note that this matrix indeed exists; by Assumption 2 A.1.4, we know the map $\theta' \mapsto \mathbb{E}_{x\sim p_{\text{data}}}[\log p_{\theta'}(x)]$ is continuously twice differentiable locally near $\theta^\star$. We can estimate that the matrices equation A.1.9 and

equation A.1.10 are indeed close as follows:

$$\begin{aligned}
&\|\nabla_{\theta'}\nabla_{\theta'}\mathcal{H}(\theta,\theta')|_{\theta^\star,\theta^\star} - (1+\lambda)\nabla_{\theta'}\nabla_{\theta'}\mathbb{E}_{x\sim p_{\text{data}}}[\log p_{\theta'}(x)]_{\theta^\star}\| \\
&\quad = \|\nabla_{\theta'}\nabla_{\theta'}[\mathbb{E}_{x\sim p_{\text{data}}}\log p_{\theta'}(x) + \lambda\mathbb{E}_{x\sim\pi_\gamma p_\theta}p_{\theta'}(x)]|_{\theta^\star,\theta^\star} - (1+\lambda)\nabla_{\theta'}\nabla_{\theta'}\mathbb{E}_{x\sim p_{\text{data}}}[\log p_{\theta'}(x)]_{\theta^\star}\| \\
&\quad = \lambda\|[\nabla_{\theta'}\nabla_{\theta'}\mathbb{E}_{x\sim\pi_\gamma p_\theta}\log p_{\theta'}(x)]|_{\theta^\star,\theta^\star} - \nabla_{\theta'}\nabla_{\theta'}\mathbb{E}_{x\sim p_{\text{data}}}[\log p_{\theta'}(x)]_{\theta^\star}\| \\
&\quad = \lambda\|[\nabla_{\theta'}\nabla_{\theta'}\mathbb{E}_{x\sim p_{\theta^\star}}\log p_{\theta'}(x)]_{\theta^\star} - [\nabla_{\theta'}\nabla_{\theta'}\mathbb{E}_{x\sim p_{\text{data}}}\log p_{\theta'}(x)]_{\theta^\star}\| \\
&\quad = \lambda\|[\mathbb{E}_{x\sim p_{\theta^\star}}\nabla_{\theta'}\nabla_{\theta'}\log p_{\theta'}(x)]_{\theta^\star} - [\mathbb{E}_{x\sim p_{\text{data}}}\nabla_{\theta'}\nabla_{\theta'}\log p_{\theta'}(x)]_{\theta^\star}\| \\
&\quad \le L\lambda\mathbb{E}_{(x,x')\sim p_{\theta^\star}\times p_{\text{data}}}\|[\nabla_{\theta'}\nabla_{\theta'}\log p_{\theta'}(x)]_{\theta^\star} - \nabla_{\theta'}\nabla_{\theta'}\log p_{\theta'}(x)]_{\theta^\star}\| \\
&\quad \le \lambda\varepsilon L
\end{aligned}$$

where the first equality follows from the definition of $\mathcal{H}$ in equation A.1.5; the second equality follows from some cancellation; the third equality follows the fact that the derivatives are constant with respect to θ, and $\pi_\gamma p_{\theta^\star} = p_{\theta^\star}$ by Lemma A.1.2; we exchange the derivative and the expectation in equation 4 using the Dominated Convergence Theorem, since Assumption 1 A.1.3 says that $x \mapsto \nabla_\theta\nabla_\theta \log p_\theta(x)$ is L-Lipschitz; the fifth estimate follows from Kantorovich-Rubinstein Duality; and the final estimate is the definition of Wasserstein distance equation A.1.4.

Finally, we verify M is indeed invertible. Assumption 2 A.1.4 implies that the largest eigenvalue of M is at most $-(1+\lambda)\alpha$. Therefore, since all eigenvalues of M are nonzero, M is invertible. We can now apply the implicit function theorem to equation A.1.7, and part A follows immediately.

Next, we prove part B. Let $d_U = \sup_{\theta\in U} d_W(p_{\theta^\star}, p_\theta)$. To verify that $g(\theta)$ is a local maximizer of equation A.1.7, it suffices to show that $\nabla_{\theta'}\nabla_{\theta'}\mathcal{H}(\theta, g(\theta)) \prec 0$. By Assumption 2 A.1.4, we know $\nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_1(\theta^\star) \prec -\alpha I_d$ and since $\theta' \mapsto \nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_1(\theta')$ is *continuously* twice differentiable locally near $\theta^\star$, we also have $\nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_1(g(\theta)) \prec -\alpha I_d$. Thus, we have

$$\begin{aligned}
\nabla_{\theta'}\nabla_{\theta'}\mathcal{H}(\theta, g(\theta)) &= \nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_1(g(\theta')) + \lambda\nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_2(\theta, g(\theta)) \\
&= (1+\lambda)\nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_1(g(\theta)) + \lambda(\nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_2(\theta, g(\theta)) - \nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_1(g(\theta))) \\
&\preceq -\alpha(1+\lambda)I_d + \lambda L\left(\frac{1}{1+\gamma}d_W(p_\theta, p_{\theta^\star}) + \varepsilon\right)I_d,
\end{aligned}$$

where the last step follows from Kantorovich-Rubsenstein duality:

$$
\begin{aligned}
&\|\nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_2(\theta,\theta') - \nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_1(\theta')\| \\
&\quad\leq \|\nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_2(\theta,\theta') - \nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_2(\theta^\star,\theta')\| + \|\nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_2(\theta^\star,\theta') - \nabla_{\theta'}\nabla_{\theta'}\mathcal{H}_1(\theta')\| \\
&\quad= \|\int_{\mathbb{R}^d} \nabla_{\theta'}\nabla_{\theta'}\log p_{\theta'}(x)\frac{p_\theta(x)+\gamma p_{\theta^\star}(x)}{1+\gamma}\,dx - \int_{\mathbb{R}^d}\nabla_{\theta'}\nabla_{\theta'}\log p_{\theta'}(x)p_{\theta^\star}(x)\,dx\| \\
&\qquad+ \|\mathbb{E}_{x\sim p_{\text{data}}}[\log p_{\theta'}(x)] - \mathbb{E}_{x\sim p_{\theta^\star}}[\log p_{\theta'}(x)]\| \\
&\quad\leq \frac{1}{1+\gamma}\|\int_{\mathbb{R}^d}\nabla_{\theta'}\nabla_{\theta'}\log p_{\theta'}(x)\,(p_\theta(x)-p_{\theta^\star}(x))\;dx\| + L\varepsilon \\
&\quad= \frac{1}{1+\gamma}\|\mathbb{E}_{x\sim p_\theta}[\log p_{\theta'}(x)] - \mathbb{E}_{x\sim p_{\theta^\star}}[\log p_{\theta'}(x)]\| + L\varepsilon \\
&\quad\leq \frac{L}{1+\gamma}d_W(p_\theta,p_{\theta^\star}) + L\varepsilon \\
&\quad\leq \frac{L}{1+\gamma}d_U + L\varepsilon
\end{aligned}
$$

Thus, to have $\nabla_{\theta'}\nabla_{\theta'}\mathcal{H}(\theta,g(\theta)) \prec 0$, it is sufficient that

$$-\alpha(1+\lambda) + \lambda L\left(\frac{1}{1+\gamma}d_U + \varepsilon\right) < 0,$$

which is guaranteed for all $\lambda > 0$ by $\alpha > L\varepsilon$ and $d_U \leq \frac{\alpha(1+\gamma)}{\lambda}$. This concludes the proof. □

Further, as we would expect, $\theta^\star$ is a fixed point of $\pi_\gamma\mathcal{G}_\lambda^\infty$:

Proposition A.1.6 (The optimal parametric generative model is a fixed point)**.** *For any given data distribution p_{data}, any $\theta^\star$ as defined by equation A.1.1, and for all $\lambda > 0$, we have $\pi_\gamma\mathcal{G}_\lambda^\infty(\theta^\star) = \theta^\star$.*

Proof. Unpacking definition equation A.1.2 shows that $\pi_\gamma\mathcal{G}_\lambda^\infty(\theta^\star) = \mathcal{G}_\lambda^\infty(\theta^\star)$, and we know by Proposition 4 from [13] that $\mathcal{G}_\lambda^\infty(\theta^\star) = \theta^\star$. □

A.1.2 Convergence of Iterative Fine-tuning with Correction for Infinite Sampling

We now have the required setup to state and prove a convergence result for iterative fine-tuning assuming infinite access to underlying probablity distributions. We need the following result, which is a technical lemma that provides a computation of the Jacobian of $\pi_\gamma G_\lambda^\infty$ at $\theta^\star$ as well as a spectral bound, both essential for the proof of Theorem A.1.8.

Lemma A.1.7. *We define the matrices*

$$A := (\nabla^2_{\theta',\theta'}\mathcal{H}_1(\theta'))|_{\theta^\star} \tag{A.1.11}$$

$$B := \nabla^2_{\theta,\theta'}\mathbb{E}_{x\sim p_\theta}[\log p_{\theta'}(x)]\big|_{\theta^\star,\theta^\star} \tag{A.1.12}$$

$$C := \nabla^2_{\theta',\theta'}\mathbb{E}_{x\sim p_\theta}[\log p_{\theta'}(x)]\big|_{\theta^\star,\theta^\star} \tag{A.1.13}$$

Recall the definition of $\pi_\gamma\mathcal{G}_\lambda^\infty(\theta)$ from equation A.1.2. Since γ and λ are fixed, denote $\pi\mathcal{G}(\theta) = \pi_\gamma\mathcal{G}_\lambda^\infty(\theta)$. Finally, let $\mathcal{J}(\pi\mathcal{G}(\theta)) := \nabla_\theta\pi_\gamma\mathcal{G}_\lambda^\infty(\theta)|_\theta$ denote the Jacobian of $\pi_\gamma\mathcal{G}_\lambda^\infty(\theta)$.

I. There exists an open neighborhood $U \subseteq \Theta$ containing $\theta^\star$ such that for all $\theta \in U$, we have

$$\mathcal{J}(\pi\mathcal{G}(\theta)) = -\left(\nabla^2_{\theta',\theta'}\mathcal{H}(\theta, \pi\mathcal{G}(\theta))\right)^{-1} \cdot \lambda\nabla^2_{\theta,\theta'}\mathcal{H}_2(\theta, \pi\mathcal{G}(\theta)). \tag{A.1.14}$$

II. We have that $\nabla^2_{\theta,\theta'}\mathcal{H}_2(\theta^\star, \theta^\star) = \frac{B}{1+\gamma}$, and $B = -C$, so the Jacobian of $\pi\mathcal{G}$ at $\theta^\star$ is

$$\mathcal{J}(\pi\mathcal{G}(\theta^\star)) = (I + \lambda A^{-1}C)^{-1} \cdot \frac{\lambda}{1+\gamma} A^{-1}C \tag{A.1.15}$$

III. The spectral norm of $A^{-1}C$ can be bounded as

$$\|A^{-1}C\| \leq 1 + \frac{L\varepsilon}{\alpha}. \tag{A.1.16}$$

Proof. **We first prove I.** We apply Proposition A.1.5. Part A of that proposition gives us a function $g : U \to \mathbb{R}^d$ such that $\nabla_{\theta'}\mathcal{H}(\theta, \theta')_{\theta, g(\theta)} = 0$. But part B of that proposition says that there exists a unique local maximizer inside U, and this local maximizer is $\pi_\gamma \mathcal{G}^\infty_\lambda$. This implies that $\nabla_{\theta'}\mathcal{H}(\theta, \theta')_{\theta, \pi_\gamma\mathcal{G}^\infty_\lambda(\theta)} = 0$. Next, we implicitly differentiate this equation with respect to θ. Recall that when you have an equation of the form $f(x, y) = 0$, and implicitly differentiate it in the form $f(x, g(x)) = 0$ with respect to x, you obtain $\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{\partial g}{\partial x} = 0$, and solving for $\frac{\partial g}{\partial x}$ yields $\frac{\partial g}{\partial x} = -\left(\frac{\partial f}{\partial y}\right)^{-1}\frac{\partial f}{\partial x}$. We apply this formula with

$$(x, f, g) = (\theta, \theta \mapsto \nabla_{\theta'}\mathcal{H}(\theta, \theta')_{\theta, \pi_\gamma\mathcal{G}^\infty_\lambda(\theta)}, \theta \mapsto \pi_\gamma\mathcal{G}^\infty_\lambda(\theta))$$

and obtain equation A.1.14, as desired.

Now we prove II. We can compute that

$$\begin{aligned}
\nabla^2_{\theta',\theta}\mathcal{H}_2(\theta, \theta') &= \nabla_{\theta'}\nabla_\theta \mathbb{E}_{x\sim\pi_\gamma p_\theta}[\log p_{\theta'}(x)] && \text{(A.1.17)}\\
&= \nabla_{\theta'}\nabla_\theta \int_{x\in\mathbb{R}^d} \log p_{\theta'}(x)\left(\frac{p_\theta(x) + \gamma p_{\theta^\star}(x)}{1+\gamma}\right)dx && \text{(A.1.18)}\\
&= \frac{1}{1+\gamma}\nabla_{\theta'}\nabla_\theta \int_{x\in\mathbb{R}^d} \log p_{\theta'}(x) p_\theta(x) dx && \text{(A.1.19)}\\
&= \frac{1}{1+\gamma}\nabla^2_{\theta',\theta}\mathbb{E}_{x\sim p_\theta}[\log p_{\theta'}(x)] && \text{(A.1.20)}\\
&= \frac{1}{1+\gamma}B && \text{(A.1.21)}
\end{aligned}$$

where the third equality holds because the integral containing $p_{\theta^\star}$ is constant with respect to θ. Next, we can compute that

$$\begin{aligned}
B &= \int_X \nabla_{\theta'}\log p_{\theta'}(x)\nabla_\theta p_\theta(x)dx\Big|_{\theta^*,\theta^*} && \text{(A.1.22)}\\
&= \int_X [\nabla_\theta \log p_\theta(x)][\nabla_\theta p_\theta(x)]dx\Big|_{\theta^*,\theta^*} && \text{(A.1.23)}\\
&= \int_X \nabla_\theta[p_\theta(x)\nabla_\theta \log p_\theta(x)]dx\Big|_{\theta^*,\theta^*} - \int_X p_\theta(x)(\nabla_\theta\nabla_\theta \log p_\theta(x))dx\Big|_{\theta^*,\theta^*} && \text{(A.1.24)}\\
&= \int_X \nabla_\theta\left[p_\theta(x)\frac{\nabla_\theta p_\theta(x)}{p_\theta(x)}\right]dx\Big|_{\theta^*,\theta^*} - \nabla^2_{\theta',\theta'}\mathbb{E}_{x\sim p_\theta}[\log p_{\theta'}(x)]\Big|_{\theta^*,\theta^*} && \text{(A.1.25)}\\
&= -C, && \text{(A.1.26)}
\end{aligned}$$

where the third equality follows from the product rule for gradients,

$$\nabla_\theta[p_\theta(x)\nabla_\theta \log p_\theta(x)] = p_\theta(x)(\nabla_\theta\nabla_\theta \log p_\theta(x)) + [\nabla_\theta p_\theta(x)][\nabla_\theta \log p_\theta(x)]. \tag{A.1.27}$$

Finally, we will prove the formula equation A.1.15 by manipulating equation A.1.14. We begin with the rightmost factor in equation A.1.14. If we apply these equalities that we just obtained, then we get

$$\begin{aligned}
\mathcal{J}(\pi\mathcal{G}(\theta^\star)) &= -\left(\nabla^2_{\theta',\theta'}\mathcal{H}(\theta^\star,\theta^\star)\right)^{-1} \cdot \lambda\nabla^2_{\theta',\theta}\mathcal{H}_2(\theta^\star,\theta^\star) \\
&= -(A+\lambda C)^{-1}\cdot\frac{\lambda}{1+\gamma}B \\
&= -(I+\lambda A^{-1}C)^{-1}\cdot\frac{\lambda}{1+\gamma}A^{-1}B \\
&= (I+\lambda A^{-1}C)^{-1}\cdot\frac{\lambda}{1+\gamma}A^{-1}C
\end{aligned}$$

where the first equality follows from equation A.1.15 along with the fixed point Proposition A.1.6, and we are using that A is invertible by Assumption 2 A.1.4, which implies all eigenvalues of A are nonzero; in the fourth step we used that $B = -C$. This proves part II.

Now we prove III. We can bound the operator norm $\|A^{-1}C\|$ as follows:

$$\|A^{-1}C\| = \|I + A^{-1}(C-A)\| \le \|I\| + \|A^{-1}\|\cdot\|C-A\| \le 1 + \alpha^{-1}\|C-A\|, \tag{A.1.28}$$

where the first estimate comes from subadditivity and submultiplicativity, and the second comes from the fact that, since A is symmetric, $\|A\| = \max_{\lambda\in\sigma(A)}|\lambda|$, where $\sigma(A)$ is the spectrum of A. Formally, we know by Assumption A.1.4 that A has eigenvalues $e_1 < e_2 < \cdots < e_n \le -\alpha < 0$ and so $|e_n| > \alpha$. Therefore, A^{-1} has eigenvalues $1/e_n < 1/e_{n-1} < \cdots < 1/e_1 < 0$ and thus $1/|e_n| > 1/|e_{n-1}| > \cdots > 1/|e_1|$, which gives us the bound $\|A^{-1}\| = 1/|e_n| < 1/\alpha$ on the matrix norm. Next, we can estimate that

$$\begin{aligned}
||C-A|| &= \|\nabla^2_{\theta',\theta'}\mathbb{E}_{x\sim p_{\theta^\star}}[\log p_{\theta'}(x)]|_{\theta^\star} - \nabla^2_{\theta',\theta'}\mathbb{E}_{x\sim p_{\text{data}}}[\log p_{\theta'}(x)]|_{\theta^\star}\| \\
&= \|\mathbb{E}_{x\sim p_{\theta^\star}}[\nabla^2_{\theta',\theta'}\log p_{\theta^\star}(x)] - \mathbb{E}_{x\sim p_{\text{data}}}[\nabla^2_{\theta',\theta'}\log p_{\theta^\star}(x)]\| \\
&\le L d_W(p_{\theta^\star}, p_{\text{data}}) \\
&= L\varepsilon,
\end{aligned}$$

where in the second equality we exchange the derivative and the expectation in equation 4 using the Dominated Convergence Theorem, since Assumption 1 A.1.3 says that $x \mapsto \nabla_\theta\nabla_\theta \log p_\theta(x)$ is L-Lipschitz; and in the last estimate, we used Kantorovich-Rubenstein duality. This, combined with the estimate equation A.1.28, yields the bound in equation A.1.16. □

We are finally ready to prove our theorem that guarantees convergence to the optimal parameters in the infinite sampling case under certain assumptions, one being the that the initial model parameters θ_0 are sufficiently close to $\theta^\star$:

Theorem A.1.8 (Convergence of Iterative Fine-tuning, Infinite Sampling Case)**.** *Suppose we have an iterative fine-tuning procedure defined by the rule $\theta^\infty_{t+1} = \pi_\gamma\mathcal{G}^\infty_\lambda(\theta^\infty_t)$. Let $\theta^\star$ be the parameter vector for the optimal*

generative model, as in equation A.1.1. We assume that $\theta^\star$ follows Assumptions A.1.3 and A.1.4 from [13]. Suppose also that $\lambda\left(1+\frac{\varepsilon L}{\alpha}\right) < \frac{1+\gamma}{2+\gamma}$. Then, the Jacobian of $\pi_\gamma G_\lambda^\infty$ satisfies the following bound:

$$\|\nabla_\theta \pi_\gamma \mathcal{G}_\lambda^\infty(\theta^\star)\|_2 \leq \frac{1}{1+\gamma} \cdot \frac{\lambda(\alpha+\varepsilon L)}{\alpha - \lambda(\alpha+\varepsilon L)} < 1. \tag{A.1.29}$$

Consequently, there exists a $\delta > 0$ such if $\theta_0 \in \Theta$ satisfies $\|\theta_0 - \theta^\star\| \leq \delta$, then starting training at θ_0 and having $\theta_{t+1} = \pi_\gamma \mathcal{G}_\lambda^\infty(\theta_t)$, we have that $\lim_{t\to\infty} \theta_t \to \theta^\star$. Furthermore, if we define

$$\rho(\lambda) = \frac{\lambda(\alpha+\varepsilon L)}{\alpha - \lambda(\alpha+\varepsilon L)}, \tag{A.1.30}$$

then we obtain the asymptotic stability estimate[1]

$$\|\theta_t - \theta^\star\| \leq \left(\frac{\rho(\lambda)}{1+\gamma}\right)^t \|\theta_0 - \theta^\star\|. \tag{A.1.31}$$

Proof. **We first prove the Jacobian bound equation A.1.29.** By hypothesis, we know $\lambda(1+\frac{L\varepsilon}{\alpha}) < 1$, so by Lemma A.1.7(III), we have $\lambda||A^{-1}C|| < 1$. Thus, we can write

$$(I+\lambda A^{-1}C)^{-1} = \sum_{k=0}^{\infty} (-\lambda A^{-1}C)^k$$

and so

$$\|(I+\lambda A^{-1}C)^{-1}\| \leq \sum_{k=0}^{\infty} \lambda^k ||A^{-1}C||^k = \frac{1}{1-\lambda||A^{-1}C||}.$$

Applying Lemma A.1.7(2), we get

$$||\mathcal{J}(G(\theta^\star))|| \leq ||(I+\lambda A^{-1}C)^{-1}|| \cdot \frac{\lambda}{1+\gamma}||A^{-1}C|| \leq \frac{\lambda}{1+\gamma} \cdot \frac{||A^{-1}C||}{1-\lambda||A^{-1}C||}.$$

Now, it is straightforward to see the RHS above is at most the bound in equation A.1.29 if and only if $\alpha\|A^{-1}C\| < \alpha + \varepsilon L$. But this bound holds because of Lemma A.1.7(III). This proves the Jacobian bound equation A.1.29, but does not prove that the bound is less than 1. For this, we must show that

$$\frac{1}{1+\gamma} \cdot \frac{\lambda(\alpha+\varepsilon L)}{\alpha - \lambda(\alpha+\varepsilon L)} < 1. \tag{A.1.32}$$

By clearing denominators and grouping like terms, we can see that this is equivalent to

$$\lambda\left(1+\frac{\varepsilon L}{\alpha}\right) < \frac{1+\gamma}{2+\gamma}, \tag{A.1.33}$$

which is precisely guaranteed by our hypothesis.

We now apply the the Jacobian bound equation A.1.29 to prove the asymptotic stability estimate equation A.1.31. Assume λ is sufficiently small so that $\rho(\lambda)/(1+\gamma) < 1$. Then for every $\rho' \in (\rho(\lambda)/(1+\gamma), 1)$, there exists $\delta > 0$ sufficiently small so that every $\theta_0 \in \Theta$ which satisfies $\|\theta_0 - \theta^\star\| < \delta$ has the

[1] [13] could have presented their results in this stronger form, without the big O notation, with very little extra work.

property that $\|\nabla_\theta \pi_\gamma G_\lambda^\infty(\theta_0)\|_2 < \rho'$. Because the map $\pi_\gamma G_\lambda^\infty$ has Jacobian matrix norm less than 1 in the δ-ball around $\theta^\star$, it is a contraction mapping in this neighborhood. Concretely, this means that

$$\|\pi_\gamma \mathcal{G}_\lambda^\infty(\theta) - \pi_\gamma \mathcal{G}_\lambda^\infty(\theta')\| \leq \frac{\rho(\lambda)}{1+\gamma}\|\theta - \theta'\|, \tag{A.1.34}$$

for every θ, θ' in the δ-ball around $\theta^\star$. In particular, for $(\theta, \theta') = (\theta_t, \theta^\star)$ we obtain

$$\|\theta_{t+1} - \theta^\star\| = \|\pi_\gamma \theta_t - \theta^\star\| = \|\pi_\gamma \mathcal{G}_\lambda^\infty(\theta_t) - \pi_\gamma \mathcal{G}_\lambda^\infty(\theta^\star)\| \leq \frac{\rho(\lambda)}{1+\gamma} \cdot \|\theta_t - \theta^\star\|.$$

By induction, the above estimate implies that if θ_0 is in a δ-ball around $\theta^\star$, then so is every successive θ_t. Therefore the desired estimate equation A.1.31 now follows by induction on t. □

Remark A.1.9. *Taking $\gamma = 0$ recovers exactly the result in [13]. Importantly, the correction function π_γ provides leverage in determining how large the augmentation percentage λ can be: choosing a larger correction strength γ allows us to choose a larger augmentation percentage λ while still retaining theoretical guarantees for convergence. Additionally, for the same choice of augmentation percentage λ, a larger correction strength γ provides a guarantee for an improved rate of convergence. See Conjecture 1.3.7.*

A.1.3 Stability of Iterative Fine-tuning with Correction for Finite Sampling

Finally, we prove a stability result for iterative fine-tuning with correction in the presence of statistical error. To do this, we require an assumption that essentially provides probabilistic guarantee that the chosen generative model learns the underlying distribution increasingly better if it has access to more samples:

Assumption A.1.10. There exist $a, b, \varepsilon_{\text{OPT}} \geq 0$ and a neighborhood U of $\theta^\star$ such that, for any $\delta \in (0,1)$, with probability $1-\delta$ over the samplings, we have

$$(\forall \theta \in U)(\forall n \in \mathbb{N}) \qquad \|\pi_\gamma \mathcal{G}_\lambda^n(\theta) - \pi_\gamma \mathcal{G}_\lambda^\infty(\theta)\| \leq \varepsilon_{\text{OPT}} + \frac{a}{\sqrt{n}}\sqrt{\log \frac{b}{\delta}}. \tag{A.1.35}$$

See Appendix A.2 for a discussion about this assumption; we investigated whether to assume a similar bound to the one they assumed in [13], or prove our bound from theirs. In fact, we prove in Appendix A.2 that you can in fact deduce something nearly as strong as Assumption A.1.10 from Assumption 3 in their paper, so we made Assumption A.1.10 for the sake of a cleaner, more parallel exposition.

Theorem A.1.11 (Iterative Fine-Tuning Stability Under Correction)**.** *Suppose we have an iterative fine-tuning procedure defined by the rule $\theta_{t+1}^n = \pi_\gamma \mathcal{G}_\lambda^n(\theta_t^n)$. In words, this means that the augmentation percentage is $\lambda \in (0, \infty)$ and the correction strength is $\gamma \in [0, \infty)$. Under the same assumptions of Theorem A.1.8 and Assumption A.1.10, there exist $0 < \rho < 1$ and $\delta_1 > 0$ such that if $\|\theta_0^n - \theta^\star\| \leq \delta_1$, then for any $\delta_2 \in (0,1)$, with probability $1 - \delta_2$, we have*

$$\|\theta_t^n - \theta^\star\| \leq \left(\varepsilon_{OPT} + \frac{a}{\sqrt{n}}\sqrt{\log \frac{bt}{\delta}}\right) \sum_{i=0}^{t} \left(\frac{\rho(\lambda)}{1+\gamma}\right)^i + \left(\frac{\rho(\lambda)}{1+\gamma}\right)^t \|\theta_0^n - \theta^\star\|. \tag{A.1.36}$$

Proof. By the triangle inequality, we can estimate that

$$\begin{aligned}\|\theta_t^n - \theta^\star\| &\leq \|\theta_t^n - \pi_\gamma \mathcal{G}_\lambda^\infty(\theta_{t-1}^n))\| + \|\pi_\gamma \mathcal{G}_\lambda^\infty(\theta_{t-1}^n) - \theta^\star\| \\ &= \|\pi_\gamma \mathcal{G}_\lambda^n(\theta_{t-1}^n) - \pi_\gamma \mathcal{G}_\lambda^\infty(\theta_{t-1}^n)\| + \|\pi_\gamma \mathcal{G}_\lambda^\infty(\theta_{t-1}^n) - \pi_\gamma \mathcal{G}_\lambda^\infty(\theta^\star)\|, \end{aligned} \tag{A.1.37}$$

where we applied the fixed point Proposition A.1.6. By Assumption A.1.10, the left summand in equation A.1.37 is at most $\varepsilon_{\text{OPT}} + \frac{a}{\sqrt{n}}\sqrt{\log \frac{b}{\delta}}$, with probability $1-\delta$. Next, recall that in equation A.1.34 in the proof of Theorem A.1.8, we proved that that $\pi_\gamma G_\lambda^\infty$ is a contraction mapping of factor $\rho(\lambda)/(1+\gamma)$ sufficiently close to U; this implies that the right summand in equation A.1.37 is at most $\frac{\rho(\lambda)}{1+\gamma}\|\theta_{t-1}^n - \theta^\star\|$. Together, these yield the recurrence estimate

$$\mathbb{P}\left(\|\theta_t^n - \theta^\star\| \leq \varepsilon_{\text{OPT}} + \frac{a}{\sqrt{n}}\sqrt{\log \frac{b}{\delta}} + \frac{\rho(\lambda)}{1+\gamma}\|\theta_{t-1}^n - \theta^\star\|\right) \geq 1-\delta. \tag{A.1.38}$$

Iterating this recurrence for successive time steps yields

$$\mathbb{P}\left(\|\theta_t^n - \theta^\star\| \leq \left(\varepsilon_{\text{OPT}} + \frac{a}{\sqrt{n}}\sqrt{\log \frac{b}{\delta}}\right)\sum_{i=0}^{t}\left(\frac{\rho(\lambda)}{1+\gamma}\right)^i + \left(\frac{\rho(\lambda)}{1+\gamma}\right)^t \|\theta_0^n - \theta^\star\|\right) \geq (1-\delta)^t. \tag{A.1.39}$$

Note that equation A.1.39 holds for any $\delta \in (0,1)$. In particular, we can apply equation A.1.39 with $\delta := \delta/t$. In this case, the Bernoulli inequality lets us estimate that $(1-\delta/t)^t \geq 1-\delta$. This completes the proof, with $\delta_2 = \delta$. □

Remark A.1.12. *Theorem A.1.11 recovers the result from [13] in the case where the correction strength is $\gamma = 0$. But for a fixed augmentation percentage λ, for any correction strength $\gamma > 0$, this gives stronger stability guarantees than in [13].*

Remark A.1.13. *In a previous version of this manuscript, we claimed that there was an error in the statement of the corresponding theorem in [13]. In this version, we retract that claim; we have corresponded with those authors, and they updated their manuscript with additional details to justify their statement.*

A.1.4 Discussion: The Main Limitation

Our empirical results are for generative modeling tasks where we have access to some "self-correction" operation that is easy to compute, as well as automatic; see Sections 1.5 and 1.6 for more details about these correction functions. Therefore, the main limitation of our work is that one can only hope to use this self-correction procedure to stabilize training in scenarios where there is some "self-correction" function. For our MNIST experiments, we built a self-correction function from scratch using clustering statistics. And for our human motion experiments, we used an off-the-shelf human motion imitation model that other researchers built.

A.2 SCSC: Discussion about Assumption 1.3.2

In this section, we show how with a mild boundedness assumption on our generative model parameter update function, we can deduce our Assumption A.1.10 (which is the same as Assumption 1.3.2, part 3) from the

following assumption used in [13].

Assumption A.2.1. There exist $a, b, \varepsilon_{\text{OPT}} \geq 0$ and a neighborhood U of $\theta^\star$ such that, for any $\delta \in (0,1)$, with probability $1-\delta$ over the samplings, we have

$$(\forall \theta \in U)(\forall n \in \mathbb{N}) \qquad \|\mathcal{G}_\lambda^n(\theta) - \mathcal{G}_\lambda^\infty(\theta)\| \leq \varepsilon_{\text{OPT}} + \frac{a}{\sqrt{n}}\sqrt{\log \frac{b}{\delta}}. \tag{A.2.1}$$

Now, if we make the additional assumption that our generative model parameter update function is locally bounded near $\theta^\star$ then we obtain the following.

Proposition A.2.2. *Suppose Assumption A.2.1 holds. Suppose also that there exists $B < \infty$ such that for all $n > 0$ and θ sufficiently close to $\theta^\star$,*

$$\|\mathcal{G}_\lambda^n(\theta) - \mathcal{G}_\lambda^n(\theta^\star)\| < B\|\theta - \theta^\star\|.$$

Then there exist $a, b, c, \varepsilon_{OPT} \geq 0$ and a neighborhood U of $\theta^\star$ such that, for any $\delta \in (0,1)$, with probability $1-\delta$ over the samplings, we have

$$(\forall \theta \in U)(\forall n \in \mathbb{N}) \qquad \|\pi_\gamma \mathcal{G}_\lambda^n(\theta) - \pi_\gamma \mathcal{G}_\lambda^\infty(\theta)\| \leq c \cdot d_U + \varepsilon_{OPT} + \frac{a}{\sqrt{n}}\sqrt{\log \frac{b}{\delta}}, \tag{A.2.2}$$

where $d_U = \sup_{\theta \in U} \|\theta - \theta^\star\|$.

Proof. By the triangle inequality, we have

$$\|\pi_\gamma \mathcal{G}_\lambda^n(\theta) - \pi_\gamma \mathcal{G}_\lambda^\infty(\theta)\| \leq \|\pi_\gamma \mathcal{G}_\lambda^n(\theta) - \mathcal{G}_\lambda^n(\theta)\| + \|\mathcal{G}_\lambda^n(\theta) - \mathcal{G}_\lambda^\infty(\theta)\| + \|\mathcal{G}_\lambda^\infty(\theta) - \pi_\gamma \mathcal{G}_\lambda^\infty(\theta)\|. \tag{A.2.3}$$

We bound each term in the RHS: firstly, note the middle term is bounded by Assumption A.2.1. The first term is bounded as follows:

$$\begin{aligned}
\|\mathcal{G}_\lambda^n(\theta) - \pi_\gamma \mathcal{G}_\lambda^n(\theta)\| &\leq \|\mathcal{G}_\lambda^n(\theta) - \mathcal{G}_\lambda^n(\theta^\star)\| + \|\pi_\gamma \mathcal{G}_\lambda^n(\theta^\star) - \pi_\gamma \mathcal{G}_\lambda^n(\theta)\| \\
&\leq B\|\theta - \theta^\star\| + B\|\theta - \theta^\star\| \\
&\leq 2Bd_U,
\end{aligned}$$

where in the first step we used that $\mathcal{G}_\lambda^\infty(\theta^\star) = \pi_\gamma \mathcal{G}_\lambda^\infty(\theta^\star)$. Similarly, the last term is bounded as follows:

$$\begin{aligned}
\|\mathcal{G}_\lambda^\infty(\theta) - \pi_\gamma \mathcal{G}_\lambda^\infty(\theta)\| &\leq \|\mathcal{G}_\lambda^\infty(\theta) - \mathcal{G}_\lambda^\infty(\theta^\star)\| + \|\pi_\gamma \mathcal{G}_\lambda^\infty(\theta^\star) - \pi_\gamma \mathcal{G}_\lambda^\infty(\theta)\| \\
&\leq \rho(\lambda)\|\theta - \theta^\star\| + \frac{\rho(\lambda)}{1+\gamma}\|\theta - \theta^\star\| \\
&= \rho(\lambda)\frac{2+\gamma}{1+\gamma}\|\theta - \theta^\star\| \\
&\leq \rho(\lambda)\frac{2+\gamma}{1+\gamma}d_U,
\end{aligned}$$

where in the second step we applied equation A.1.34. Using these bounds in equation A.2.3 and taking $c = 2B + \rho(\lambda)\frac{2+\gamma}{1+\gamma}$ completes the proof. □

Note that the constant $c \cdot d_U < c$ (for U sufficiently small) can really be viewed as a part of the optimization constant ε_{OPT} since it is controlled by the choice of generative model class.

A.3 SCSC: Point-wise correction corresponds to distribution-wise correction

In this section we provide a sufficient condition under which you can associate a distribution-wise correction mapping (like the one we consider in the work, π_γ) to a point-wise correction mapping (which is the one you are more likely to find in the wild).

Definition A.3.1. *Let $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ and define the empirical cumulative distribution function Φ_X by*

$$\Phi_X(v) := \Phi_X(v; \{x_1, \ldots, x_n\}) := \frac{1}{n} \sum_{i=1}^{n} \chi_v(x_i),$$

where for $v \in \mathbb{R}^m$, $\chi_v : \mathbb{R}^m \to \{0, 1\}$ is the indicator function for the set $\prod_{i=1}^{n}(-\infty, v_i]$. For a continuous distribution, the cumulative distribution function is defined in the usual way.

Definition A.3.2. *Suppose that we have a model p_θ and an arbitrary function $\Pi : \mathbb{R}^m \to \mathbb{R}^m$. Then we say that Π is a* valid point-wise correction function *for p_θ if there exists a $\gamma \in [0, \infty]$ such that*

$$\lim_{n \to \infty} \left(\mathbb{E}_{X^n \sim p_\theta^n} \sup_{v \in \mathbb{R}^m} \|\Phi_{\Pi(X^n)}(v) - \Phi_{\pi_\gamma p_\theta}(v)\| \right) \to 0, \tag{A.3.1}$$

almost surely, where the expectation is over all samplings $X^n = \{x_1, \ldots, x_n\}$ of size n from p_θ.

Intuition A.3.3. This is saying that the CDFs for $\pi_\gamma p_\theta$ and $\Pi(X \sim p_\theta^n)$ are equal in expectation, for large enough n. This is one way of saying that $\pi_\gamma p_\theta$ and $\Pi(X \sim p_\theta^n)$, for large enough n, are nearly identical probability distributions.

Definition A.3.4. *If the limit in equation A.3.1 exists, then we define the* distribution-wise projection function *corresponding to Π to be*

$$\pi_\gamma p_\theta = \frac{1}{1+\gamma} p_\theta + \frac{\gamma}{1+\gamma} p_{\theta^\star}, \tag{A.3.2}$$

and we define the projection strength of the point-wise correction function Π *to be γ. Recall that $\pi_\gamma p_\theta = \frac{1}{1+\gamma} p_\theta + \frac{\gamma}{1+\gamma} p_{\theta^\star}$. So intuitively, equation A.3.1 implies that the projection function Π maps samples from p_θ to a different space such that they look like they come from a combination of the original distribution p_θ and $p_{\theta^\star}$, at least at the level of CDFs.*

Remark A.3.5. *Such a γ, if it exists, is unique. Furthermore, if $p_\theta = p_{\theta^\star}$, then $\gamma = \infty$.*

The limit condition in Definition A.3.2 is abstract, and can be hard to swallow. We present an example of a simple point-wise correction for the Gaussian toy example that we consider in Section 1.4, whose corresponding distribution-wise correction is exactly one would expect it to be–the weighted average of the corresponding Gaussians. Recall that we demonstrated empirically in Figure 1.2 that Theorem 1.3.3 holds for that example. The projection function is depicted in Figure A1.

Example A.3.6. Let $G_1(x)$ be the pdf of $\mathcal{N}(0, \sigma_1^2 I_d)$ (initial distribution, corresponds to θ) and $G_2(x)$ the pdf of $\mathcal{N}(0, \sigma_2^2 I_d)$ (target distribution, corresponds to $\theta^\star$). Given $x_1, \ldots, x_n \sim G_1$, we define Π^γ as follows: Fix any $\gamma \in \mathbb{R}_{\geq 0}$, and let $y_1, \ldots, y_n \sim (\hat{G}_1^{(n)}(x) + \gamma G_2(x))/(1+\gamma)$, where $\hat{G}_1^{(n)}$ is the PDF of the empirical distribution defined by $\{x_1, \ldots, x_n\}$; in practice we implement $\hat{G}_1^{(n)}$ as a histogram. Then choose a random $\sigma \in S_n$ (S_n = group of permutations on n symbols). Finally, we define $\Pi^\gamma(x_i) := y_{\sigma(i)}$ for $1 \leq i \leq n$.

Next, we define the projection set $\Pi X^{(n)} := \{\Pi^\gamma(x_i)\}_{1 \leq i \leq n}$, and define the PDF $\pi_\gamma \hat{G}_1^{(n)}(x) := \frac{1}{1+\gamma}\hat{G}_1^{(n)}(x) + \frac{\gamma}{1+\gamma}G_2(x)$, and let $\Phi_{\pi_\gamma \hat{G}_1^{(n)}}$ represent the cumulative distribution function of the Gaussian $\pi_\gamma \hat{G}_1^{(n)}$. Then, since $\Pi^\gamma(x_i) \sim \pi_\gamma \hat{G}_1^{(n)}$, we have by the uniform law of large numbers that

$$\lim_{n\to\infty} \left(\mathbb{E}_{\{x_i \sim G_1\}_{i=1}^n} \sup_{v \in \mathbb{R}^m} \left\|\Phi_{\Pi X^{(n)}}(v) - \Phi_{\pi_\gamma G_1}(v)\right\|\right) \to 0 \tag{A.3.3}$$

almost surely. Therefore Π^γ is a valid point-wise correction function, and its corresponding distribution-wise projection function is π_γ.

Remark A.3.7. *In the example we considered in Section 1.4, we could have included a total distance traveled minimization condition, but here for this proof we don't even need to use that hypothesis. (In the proof, this would have corresponded to the additional assumption that we've chosen a $\sigma \in S_n$ such that $\sum_{i=1}^n \|x_i - y_{\sigma(i)}\|$ is minimized.) This implies that different point-wise correction functions can correspond to the same distribution-wise correction function.*

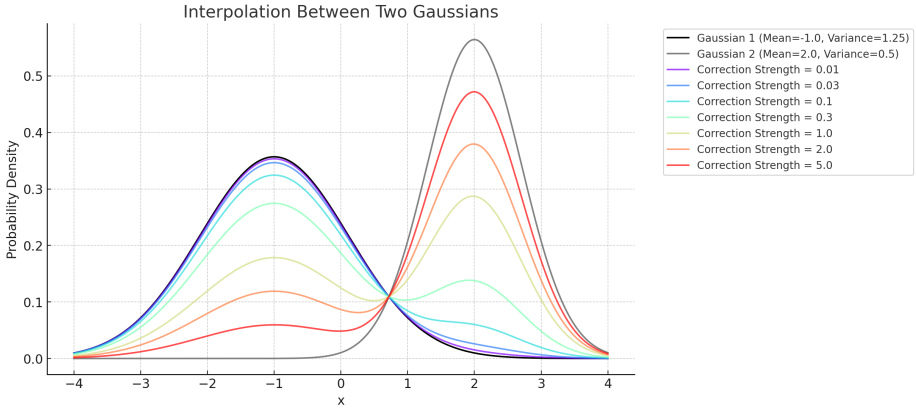


Figure A1: Illustration of the distribution-wise projection function, like in our Gaussian toy example. Correcting one Gaussian in the direction of another, like we consider in Section 1.4, corresponds to finding the "(weighted) average Gaussian" that lives between the two.

A.4 SCSC: More MNIST Experiment Details

We train a Denoising Diffusion Probabilistic Model (DDPM) [49] on the 20% of the MNIST dataset [66]. We use classifier-free guidance [48] with guidance parameter 0.5, and 400 diffusion steps. We used a batch size of

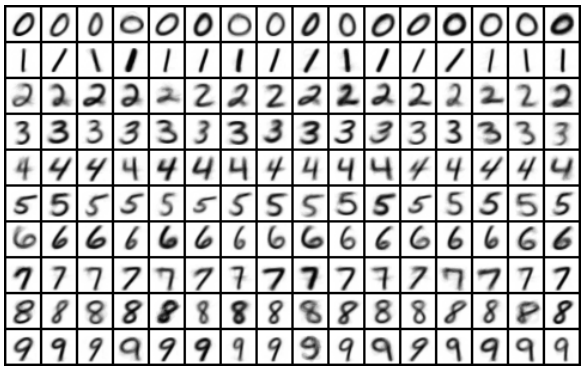


Figure A2: For every digit, we perform K-means clustering with K=16. We show here the cluster centroids, which intuitively are anchor images within the manifold of all possible images.

256. We train generation 0 for 20 epochs, with a linear decay learning rate schedule starting at $1e-4$ and ending at $(1e-4)/20$. We train each following generations for a single epoch, with a fixed learning rate of $(1e-4)/20^2$.

To compute our metrics, we first train a LeNet model [66] on MNIST, and then we sample an equal number of digits from each class using the checkpoint that we're trying to evaluate. To compute the FID score, we extract embeddings from the last fully connected LeNet layer for the synthesized examples, as well as for the held out test examples, and compute FID score as normal, by computing the Wasserstein distance between the Gaussians. Note that we use embeddings for LeNet trained on MNIST, rather than the Inception network trained on ImageNet, because MNIST isn't comprised of natural images. This is consistent with the convention in [5].

For the self-correction operation, we compute the K-means clusters, with $K = 16$, once at the start of training. And we "correct" a synthesized motion by mapping it to the nearest cluster mean corresponding to its digit. In Figure A2 we present the clusters, and we present graphs of our FID scores across augmentation percentages in Figure A3.

A.5 SCSC: Additional Human Motion Generation Qualitative Results

In Figures A4, A5, and A6, we present additional qualitative observations and analysis of our synthesized motions. We present more evidence that iterative fine-tuning with self-correction yields physically plausible motions comparable to the baseline, whereas iterative fine-tuning without self-correction yields motions that are incorrect for various reasons. See the captions of the referenced figures for analysis of some characteristic failure modes of the iterative fine-tuning loop *without* self-correction.

A technical note: for all figures, we render the motions from the same environment and camera position. We consolidate each render into the same image *without* resizing it. This means that if a figure appears larger relative to the others, the human moved closer to the camera. Some motions will have transparent frames of past positions; the more transparent the image, the farther back in the past it was in the motion sequence.

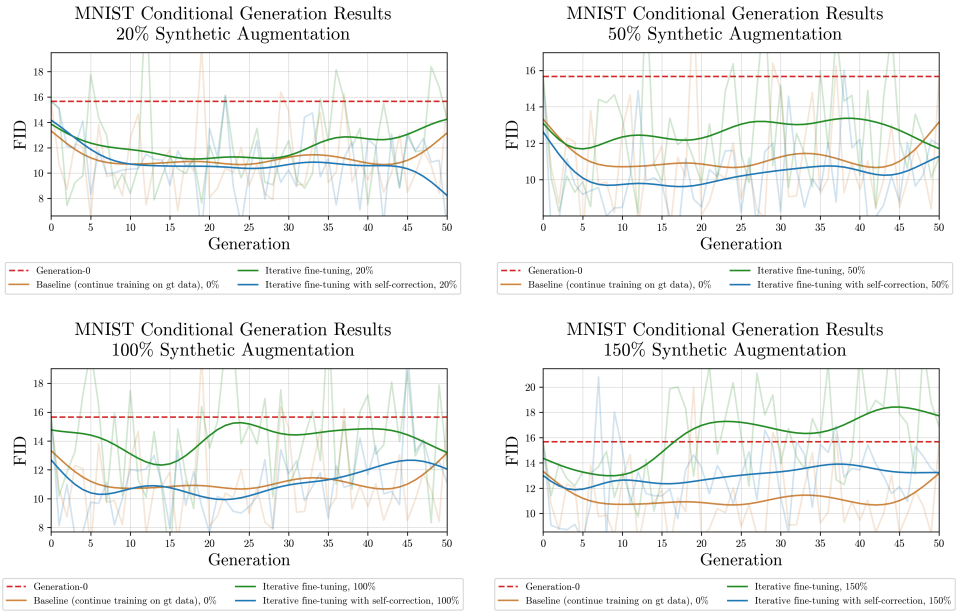


Figure A3: Results from MNIST experiments with iterative fine-tuning with and without self-correction. These graphs show the FID score on the last checkpoint for every generation; this is the checkpoint used for sampling in the self-consuming loop experiments, and it is also the checkpoint where training is resumed with this new partially synthesized dataset. These results demonstrate that **iterative fine-tuning with self-correction** generally outperforms **iterative fine-tuning.**

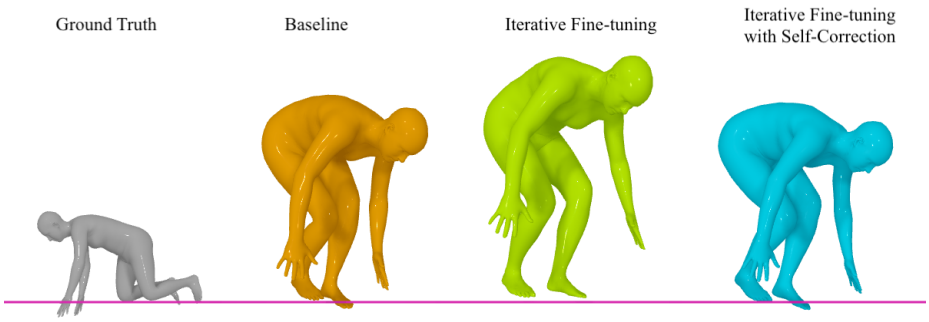


Figure A4: Here we see the negative *floating* phenomenon exacerbated by **iterative fine-tuning**, whereas **iterative fine-tuning with self-correction** generates a motion with floor contact integrity comparable to the **ground truth** and **baseline**. The floatic metric is formally defined in [138] as the distance between the lowest vertex on the human mesh and the floor plane. All three sequences were generated using the same prompt: *person got down and is crawling across the floor.* Each snapshot was taken at exactly frame 87. The green figure appears larger than the other two only because it is closer to the camera. The two motions on the right were synthesized after 50 generations training with 25% synthetic augmentation, trained on $n = 64$ data points.

Finally, in each figure, the text prompt for all generated motions was the same –the prompt being the one associated with the ground truth motion in the HumanML3D [42] training data, which we also visualize. Note that the coloring in the humanoid figures corresponds to the coloring in the graphs.

A.6 SCSC: Additional Human Motion Generation Quantitative Results

See Figures A7, A8, A9 for results when the dataset size is $n \in \{64, 128, 256\}$ and the synthetic augmentation percentage is $\lambda \in \{0.25, 0.50, 0.75, 1.00\}$. And see Figures A10 and A11 for additional results on our iterative fine-tuning experiments when the dataset size is $n = 2794$ and the synthetic augmentation percentage is $\lambda \in \{0.05, 0.10, 0.15, 0.20, 0.25\}$. The graphs provide evidence across 17 experiment settings that our iterative fine-tuning procedure with self-correction yields better training performance than iterative fine-tuning with no self-correction for the motion synthesis task, in accordance with Theorem 1.3.3.

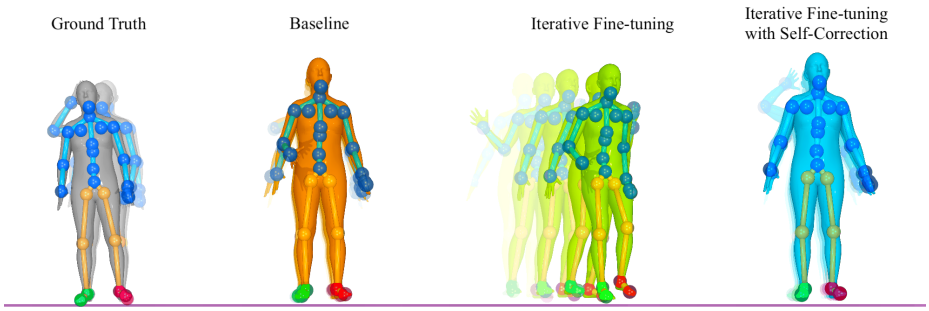


Figure A5: All four of the above motions correspond to the prompt: *a person raises right hand to face looks around and puts hand down back to side.*. The model which is trained with **iterative fine-tuning** outputs spurious motion that slides the figure to the right. And in the video for this example, the human rotates their forearm unnaturally and forcefully. In contrast, the **baseline** and **iterative fine-tuning with self-correction** models' motions both accurately embody the prompt. Each generated snapshot is taken at exactly frame 142 while the ground truth's image is frame 70 in its sequence. The two motions on the right were synthesized after 42 generations with 10% synthetic augmentation, where the ground truth dataset has size $n = 2794$.

A.7 SCSC: Consistency Across Seeds: Additional Human Motion Generation Quantitative Results

In Figures A12, A13, A14, and A15, we present experimental results from runs across three more seeds for our human motion experiments when the dataset size is $n = 64$. We find that the self-correction technique consistently yields improved training dynamics over iterative fine-tuning without correction.

A.8 FH: Proof of Fourier Head Scaling Law, Theorem 2.2.3

In this section we prove Theorem 2.2.3, 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. This justifies why we only theoretically study the case $1 \ll N < m/2$ in the scaling law.

A.8.1 Definitions

Consider an input $x \in \mathbb{R}^n$ to the Fourier head, 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 will assume that f_x is

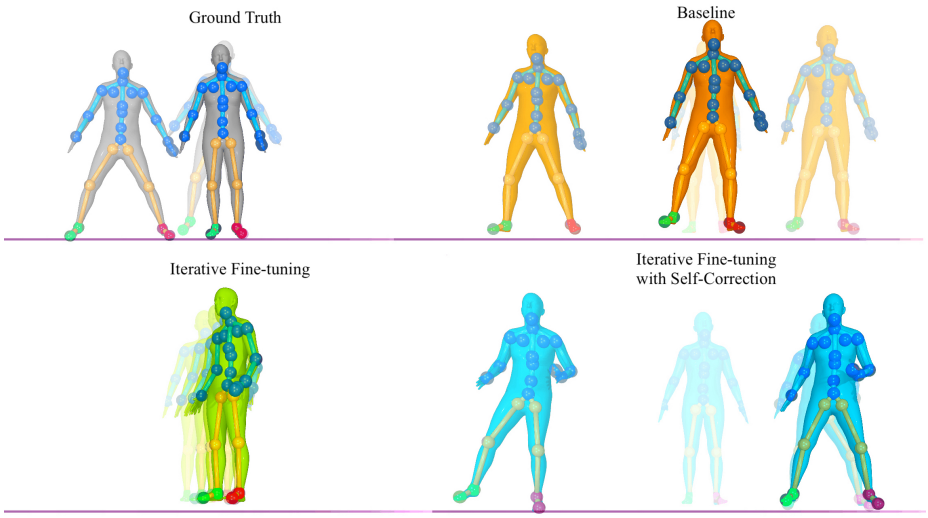


Figure A6: Here we observe that **iterative fine-tuning** fails to produce any meaningful motion sequence, but the **iterative fine-tuning with self-correction** and **baseline** models generate results consistent with their prompt: *walks side ways but back and forth*. Each snapshot for the generated motions was taken at exactly frame 120 while the ground truth image is a snapshot from frame 69. These images were synthesized after 50 generation of the model that was trained on $n = 64$ data points at 25% synthetic augmentation.

periodic, since the Fourier head learns a 2-periodic Fourier density. We denote by $f_{x,N}$ the truncation of the Fourier series of f_x to its first N frequencies. Note that $f_{x,N}$ also integrates to 1 over $[-1, 1]$ since its first Fourier coefficient is the same as that of f_x. Further, $f_{x,N}$ is non-negative on $[-1, 1]$ since its Fourier coefficients, being a subsequence of the coefficients of f_x, are non-negative definite; a periodic function with non-negative definite Fourier coefficients is non-negative by Herglotz's Theorem [15, Corollary 4.3.2]. For completeness, we will recall the convolution formulas, specialized to the cases we consider in our argument.

Definition A.8.1 (Discrete convolution)**.** *Let $b_j := -1 + \frac{2j+1}{m}, 0 \le j < m$ be the center points of the m bins in $(-1, 1)$, and let us denote $\vec{b} := (b_0, \ldots, b_{m-1})$. Denote by $G_\sigma(z) := \frac{e^{-z^2/2\sigma^2}}{\sqrt{2\pi}\sigma}$ the Gaussian PDF with standard deviation σ. Then the discrete Gaussian convolution filter of radius $m - 1$ is*

$$g_\sigma := \frac{G_\sigma([1-m, 2-m, 3-m, \ldots, m-1])}{S(m, \sigma)} \in \mathbb{R}^{2m-1}, \tag{A.8.1}$$

where the normalization constant is

$$S(m, \sigma) := \sum_{k=1-m}^{m-1} G_\sigma(k). \tag{A.8.2}$$

The discrete convolution of $g_\sigma \in \mathbb{R}^{2m-1}$ and $f_{x,N}(\vec{b}) \in \mathbb{R}^m$ is the vector $(g_\sigma * f_{x,N})(\vec{b}) \in \mathbb{R}^m$ whose j'th

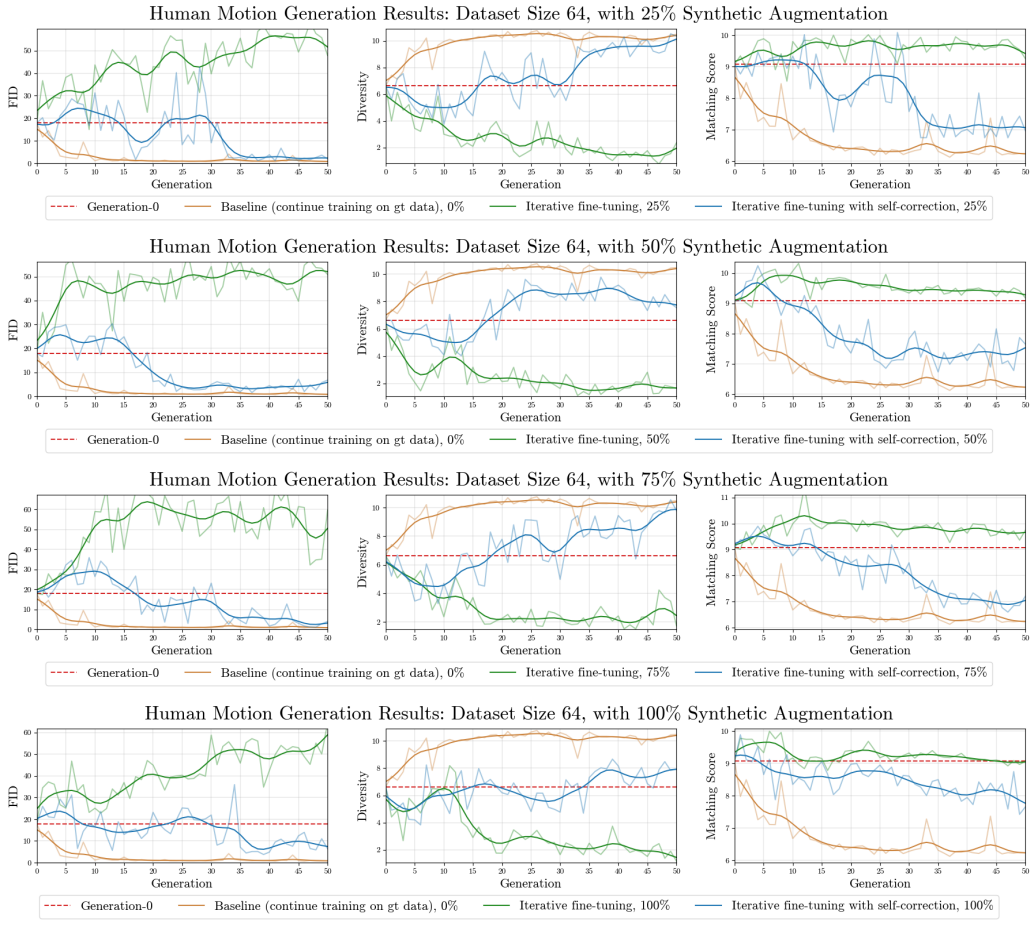


Figure A7: Results from our human motion experiments with iterative fine-tuning with and without self-correction, where the training set has size 64. These are graphs for evaluation metrics on the last checkpoint for every generation; this is the checkpoint used for sampling in the self-consuming loop experiments, and it is also the checkpoint where training is resumed with this new partially synthesized dataset. These results demonstrate that **iterative fine-tuning with self-correction** generally outperforms **iterative fine-tuning**, and is sometimes even competitive with **baseline** performance.

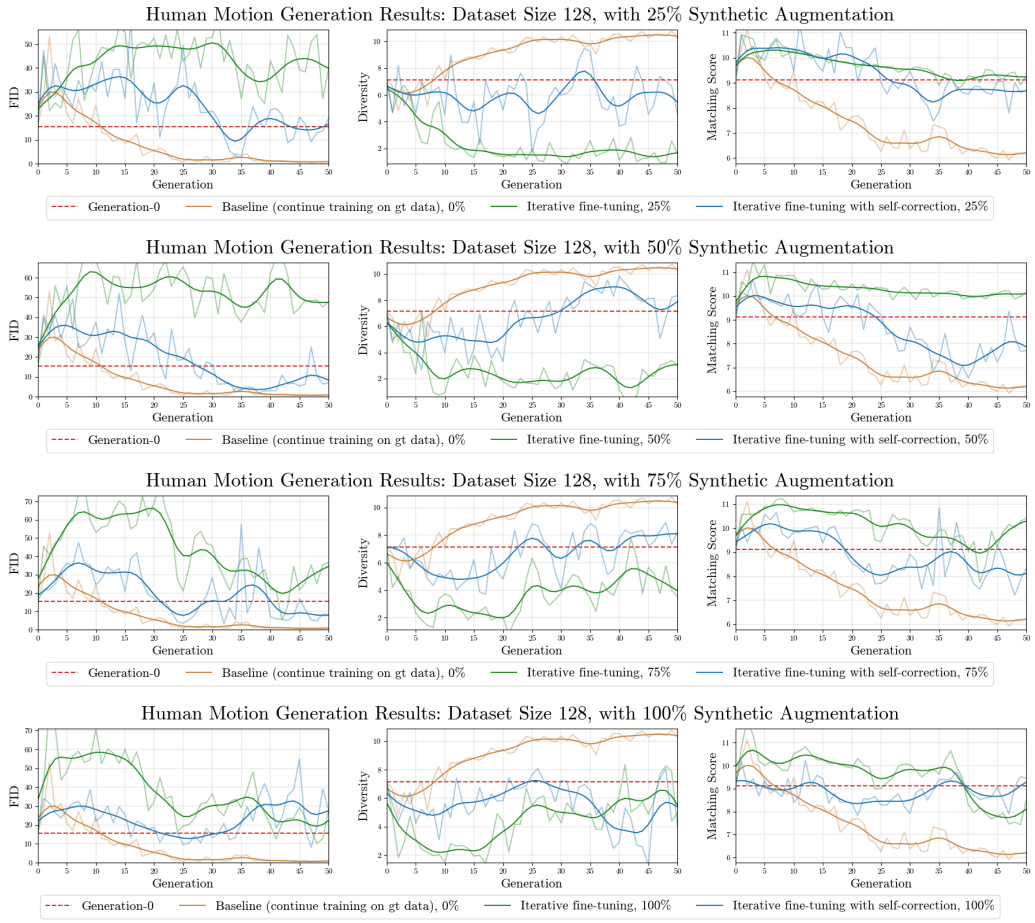


Figure A8: Results from our human motion experiments with iterative fine-tuning with and without self-correction, where the training set has size 128. These are graphs for evaluation metrics on the last checkpoint for every generation; this is the checkpoint used for sampling in the self-consuming loop experiments, and it is also the checkpoint where training is resumed with this new partially synthesized dataset. These results demonstrate that **iterative fine-tuning with self-correction** generally outperforms **iterative fine-tuning**, and is sometimes even competitive with **baseline** performance. Notably, the performance gain of **iterative fine-tuning with self-correction** over **iterative fine-tuning** is less pronounced than when the dataset size is $n = 64$.

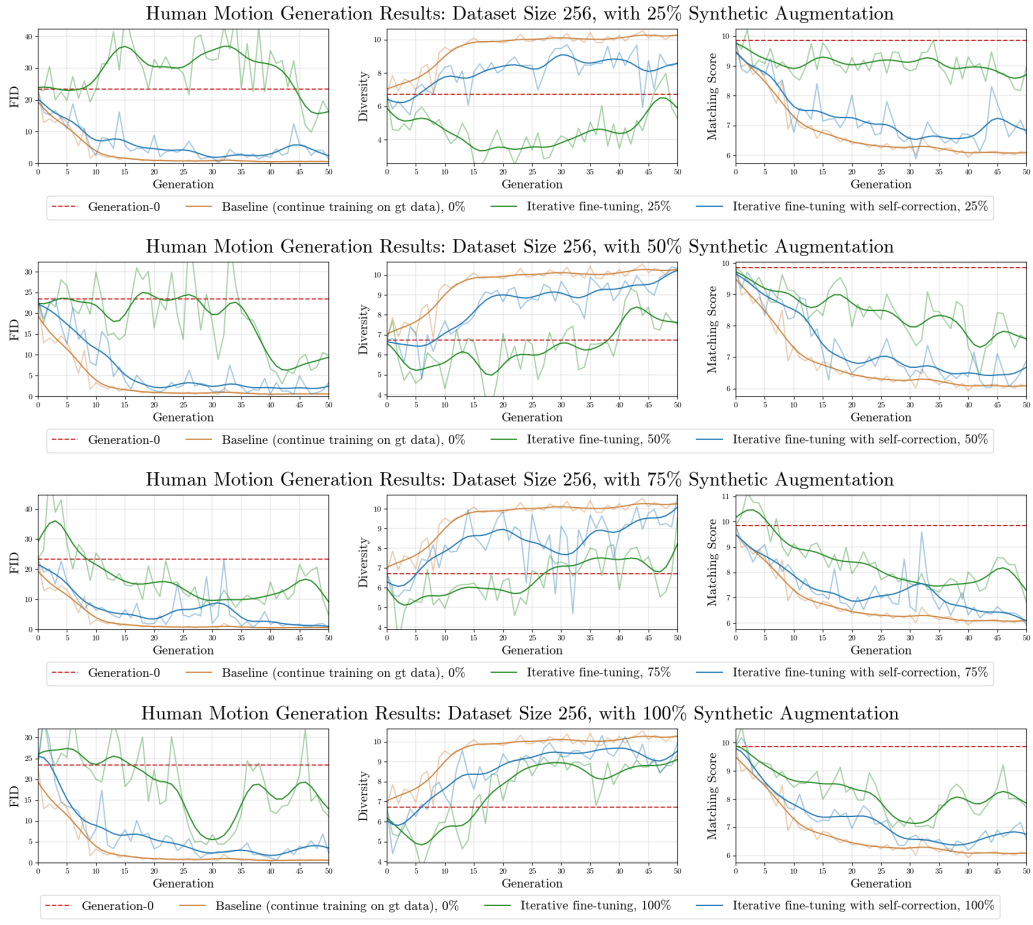


Figure A9: Results from our human motion experiments with iterative fine-tuning with and without self-correction, where the training set has size 256. These are graphs for evaluation metrics on the last checkpoint for every generation; this is the checkpoint used for sampling in the self-consuming loop experiments, and it is also the checkpoint where training is resumed with this new partially synthesized dataset. These results demonstrate that **iterative fine-tuning with self-correction** generally outperforms **iterative fine-tuning**, and is sometimes even competitive with **baseline** performance.

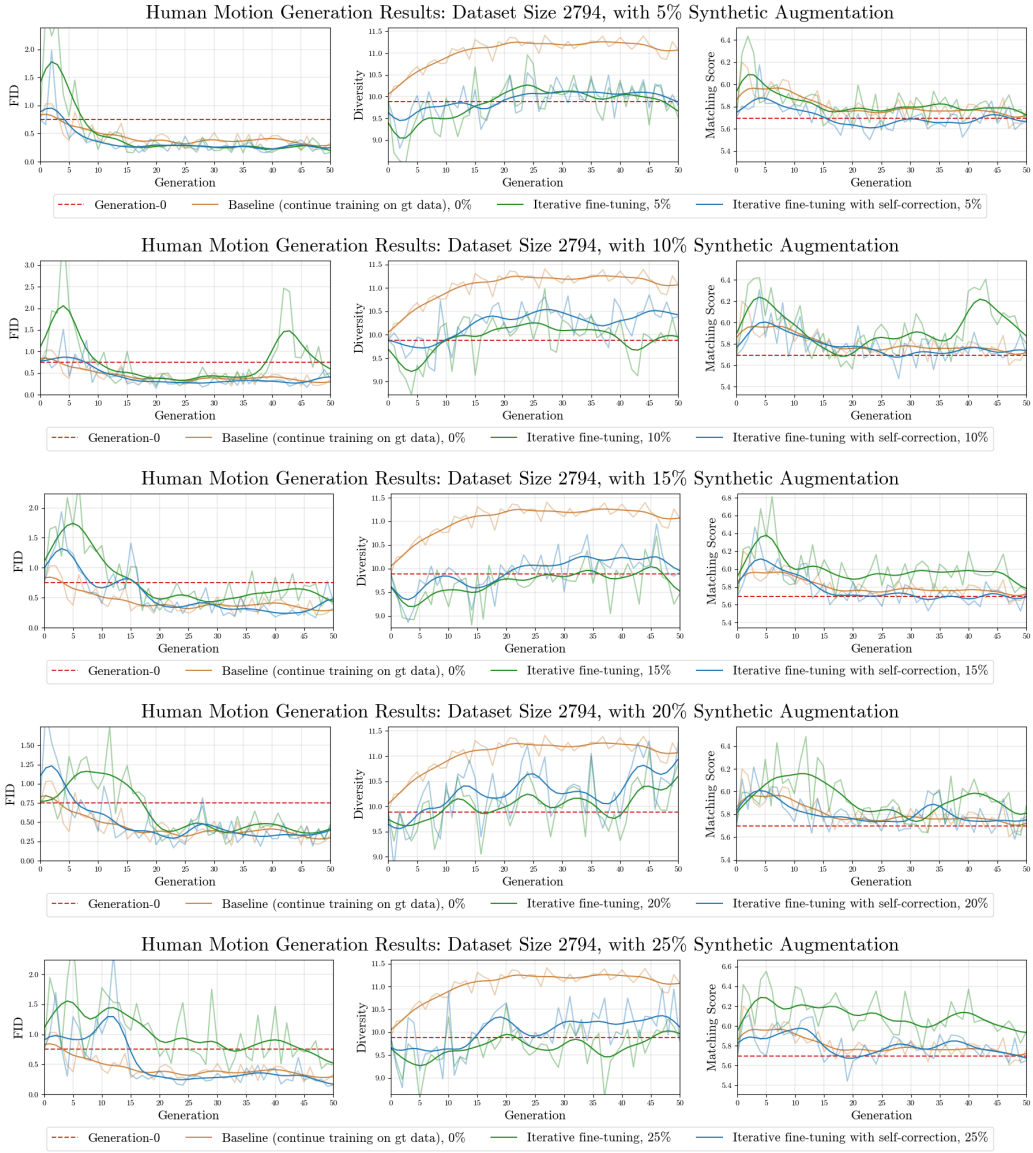


Figure A10: Results from our human motion experiments on iterative fine-tuning with dataset size $n = 2794$. These are graphs for evaluation metrics on the last checkpoint for every generation; this is the checkpoint used for sampling in the augmentation loop experiments, and it is also the checkpoint where training is resumed with this new synthesized dataset. In these results, it appears as though iterative fine-tuning **with self-correction** has less variance during training than iterative fine-tuning with **with no self-correction**, and generally has better FID scores later in training. Notably, the these two curves are closer together than they were in the cases $n \in \{64, 128, 256\}$.

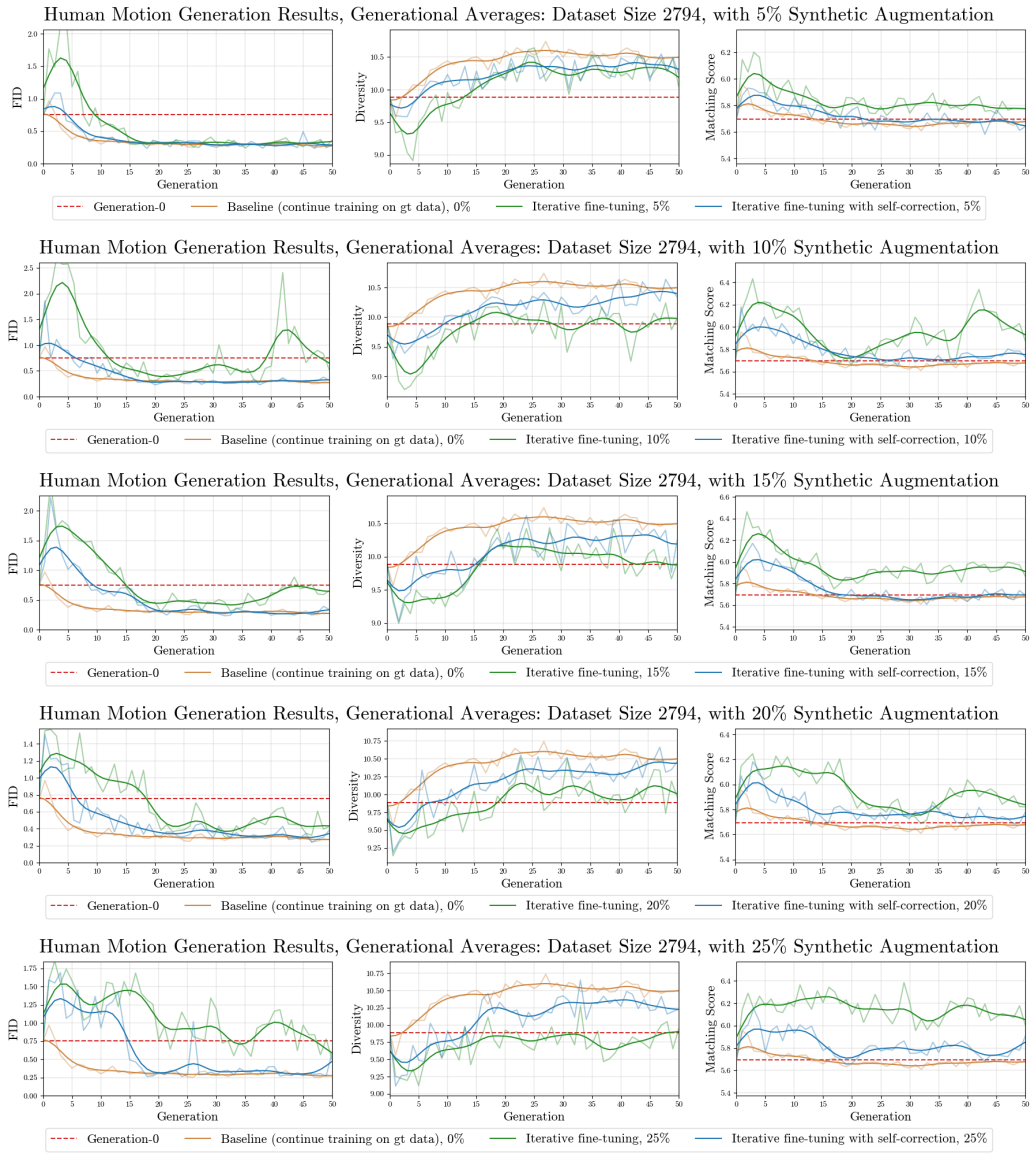


Figure A11: Results from our human motion experiments on iterative fine-tuning with dataset size $n = 2794$. These are graphs of the average evaluation metrics for every generation. Graphing the average evaluation metrics makes the training dynamics trend over time more clear. With this additional smoothing, it is more clear that iterative fine-tuning **with self-correction** outperforms iterative fine-tuning **with no self-correction**, and is competitive with the **baseline** after many generations; in fact, it appears to converge to the baseline (on average) for every synthetic augmentation percentage that we considered.

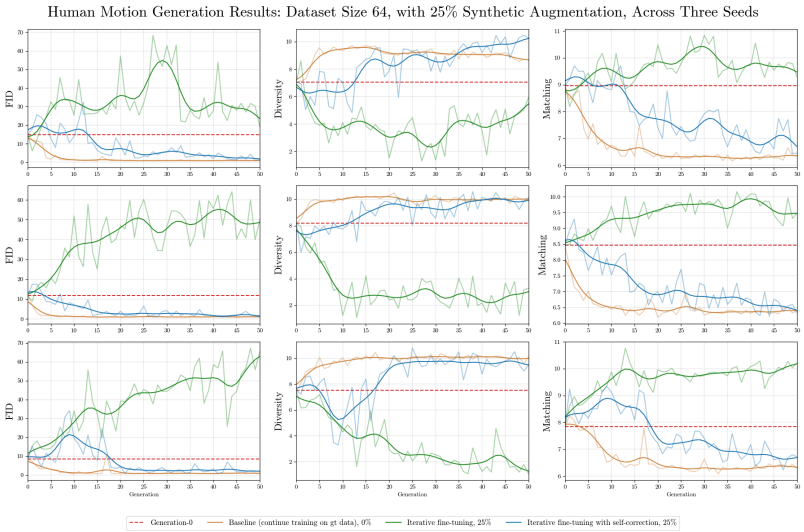


Figure A12: Results from our human motion experiments on iterative fine-tuning, with dataset size $n = 64$ and 25% augmentation percentage. Each row corresponds to a different random seed. We can see that iterative fine-tuning **with self-correction** consistently outperforms iterative fine-tuning **with no self-correction**, and the FID score appears to converge to the **baseline** after many generations.

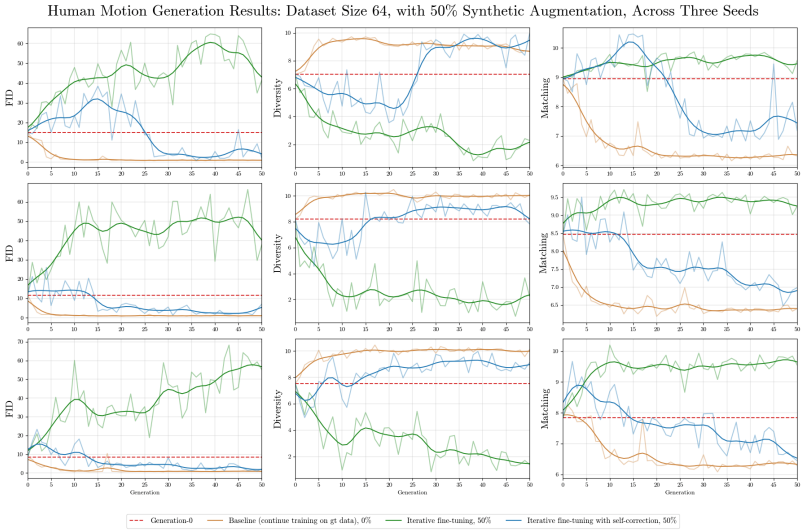


Figure A13: Results from our human motion experiments on iterative fine-tuning, with dataset size $n = 64$ and 50% augmentation percentage. Each row corresponds to a different random seed. We can see that iterative fine-tuning **with self-correction** consistently outperforms iterative fine-tuning **with no self-correction**, and the FID score appears to converge to the **baseline** after many generations.

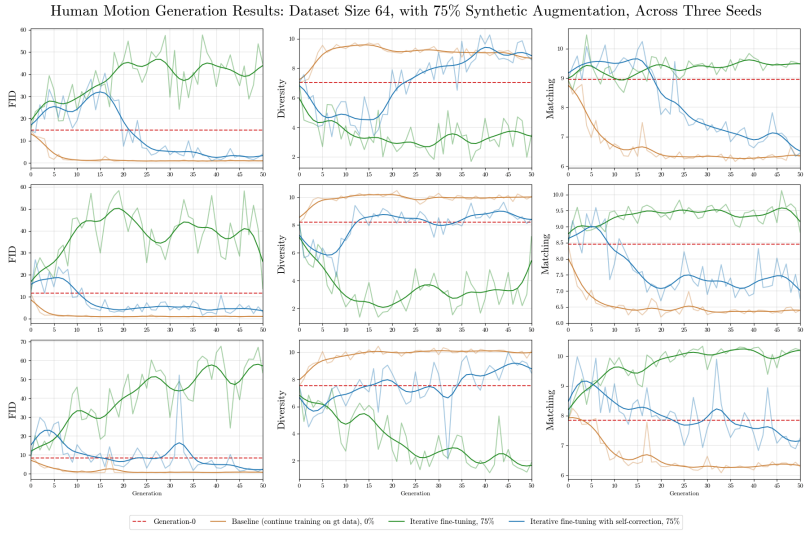


Figure A14: Results from our human motion experiments on iterative fine-tuning, with dataset size $n = 64$ and 75% augmentation percentage. Each row corresponds to a different random seed. We can see that iterative fine-tuning **with self-correction** consistently outperforms iterative fine-tuning **with no self-correction**, and the FID score appears to converge near the **baseline** after many generations.

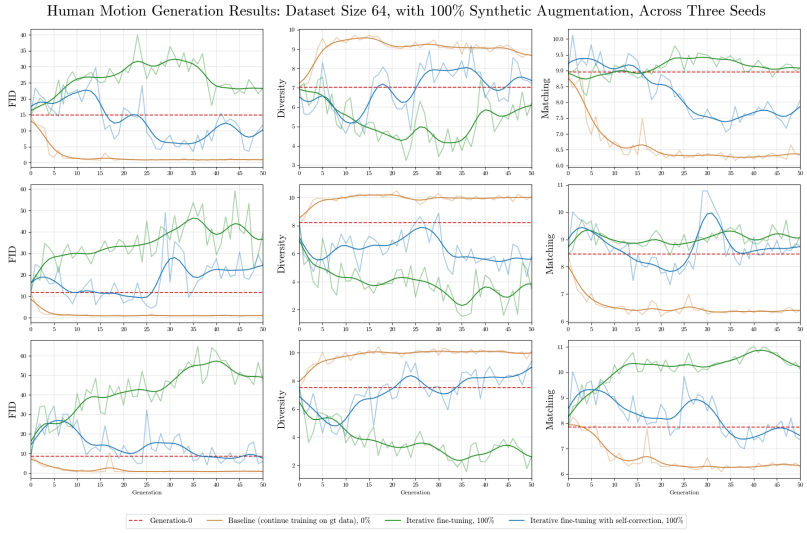


Figure A15: Results from our human motion experiments on iterative fine-tuning, with dataset size $n = 64$ and 100% augmentation percentage. Each row corresponds to a different random seed. We can see that iterative fine-tuning **with self-correction** consistently outperforms iterative fine-tuning **with no self-correction**. However, we see less stability than in the runs with a lower augmentation percentage. This is in accordance with Theorem 1.3.3.

coordinate is given by

$$(g_\sigma * f_{x,N})(b_j) = \frac{1}{S(m,\sigma)} \sum_{k=1-m}^{m-1} G_\sigma(k) \cdot f_{x,N}(b_{j-k}). \tag{A.8.3}$$

Definition A.8.2 (Continuous convolution)**.** *The continuous Gaussian convolution filter $\tilde{g}_\sigma : [-2,2] \to \mathbb{R}_{>0}$ is*

$$\tilde{g}_\sigma(z) = \frac{G_{\frac{2\sigma}{m}}(z)}{S(m,\sigma)} = \frac{m}{2S(m,\sigma)} G_\sigma\left(\frac{mz}{2}\right). \tag{A.8.4}$$

This function $\tilde{g}_\sigma(z)$ is a normalized truncation of a Gaussian PDF with mean 0 and standard deviation $2\sigma/m$. The continuous convolution of $\tilde{g}_\sigma : [-2,2]$ and the periodic function $f_{x,N} : [-1,1] \to \mathbb{R}$ is

$$\tilde{g}_\sigma * f_{x,N}(z) := \int_{-2}^{2} \tilde{g}_\sigma(u) f_{x,N}(z-u)\, du. \tag{A.8.5}$$

A.8.2 Overview of Proof

In this subsection, we provide an overview of the proof of Theorem 2.2.3 by presenting the statements of the lemmata that we will need, and connecting each one to the overall argument. In the next subsection, we rigorously prove the scaling law by careful applications of these lemmata. And in the following subsection, we will rigorously prove each of the lemmata.

This first lemma allows us to replace the discrete Gaussian convolution in the definition with a continuous Gaussian convolution.

Lemma A.8.3. (Discrete convolution is close to continuous convolution) *If we define the constant $B_1(m,\sigma) := 1 + \frac{G_\sigma(m)}{S(m,\sigma)}$, then we have that*

$$\|f_{x,N}(\vec{b}) - g_\sigma * f_{x,N}(\vec{b})\|_2 = \|B_1(m,\sigma) f_{x,N}(\vec{b}) - \tilde{g}_\sigma * f_{x,N}(\vec{b})\|_2 + \sqrt{m} O(1/N^{2t+1}). \tag{A.8.6}$$

Furthermore, $B_1(m,\sigma)$ satisfies the following bound, uniformly in σ,

$$B_1(m,\sigma) \le 1 + \frac{1}{2m-1}. \tag{A.8.7}$$

This next lemma, a standard result from analytic number theory allows us to upper bound the sums of the norms of the Fourier series coefficients. This is proved in various places, see e.g. [114, Equation 21].

Lemma A.8.4 (Asymptotic expansion of Riemann zeta function)**.** *Consider the Riemann zeta function $\zeta(t) := \sum_{k=1}^{\infty} \frac{1}{k^t}$. If $t \ge 2$, then*

$$\sum_{k=1}^{N} \frac{1}{k^t} = \zeta(t) - \frac{1}{t-1} \frac{1}{N^{t-1}} + O(1/N^t). \tag{A.8.8}$$

This next lemma allows us to extract the main asymptotic behavior in the scaling law.

Lemma A.8.5. (Main term asymptotic) *Denote by $a_0(x)$ the constant coefficient of $f_{x,N}$. Let us suppose that the Fourier coefficients of $f_{x,N}$ decay like $B_3(x)/k^t$, and define the constant*

$$B_2(\sigma,m,x) := \sqrt{a_0(x)^2 B_1(m,\sigma)^2 + 2B_1(m,\sigma)^2 B_3(x)^2 \zeta(2t)}. \tag{A.8.9}$$

Then we know that

$$\|B_1(m,\sigma)f_{x,N}(\vec{b}) - \tilde{g}_\sigma * f_{x,N}(\vec{b})\|_2 \tag{A.8.10}$$

$$= \sqrt{m}\left(B_2(\sigma,m,x) - \frac{B_1(m,\sigma)^2 B_3(x)^2}{2t-1}\cdot\frac{1}{N^{2t-1}} + O(1/N^{2t})\right). \tag{A.8.11}$$

Furthermore, $B_2(\sigma,m,x)$ is bounded from above and below as a function of σ and m.

This final lemma allows us to relate the continuous case, where our analysis works out easier, to the discrete case, where our smoothness metric is actually defined.

Lemma A.8.6. (The average value of the truncated Fourier PDF is $1/2$) *If $N < m/2$, then*

$$\sum_{j=0}^{m-1} f_{x,N}(b_j) = \frac{m}{2}. \tag{A.8.12}$$

A.8.3 Proving Theorem 2.2.3 Using the Lemmata

We now prove the 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 must assume that f_x, the conditional distribution being learned by the Fourier head, is sufficiently smooth. For example, if f_x is twice continuously differentiable, then the Fourier coefficients corresponding to the k-th frequency of f_x are in $O(1/k^2)$ [110, Ch.2, Cor. 2.4]. Thus, our assumption that the Fourier coefficients decay quadratically is reasonable, and our Fourier weight decay regularization helps ensure that this condition is met in practice as well. In our theorem, we generalize this hypothesis to the cases where the Fourier coefficients corresponding to the k-th frequency of f_x are in $O(1/k^t)$.

Theorem 2.2.3. *(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. Suppose that there exists some $t \geq 2$ such that the Fourier coefficients of f_x decay on the order of $1/k^t$. Denote by $f_{x,N}$ the truncation of f_x to its first N frequencies, denote by $\vec{b} \in \mathbb{R}^m$ the m bin centerpoints in $[-1,1]$, and denote by $y^{(N)} = f_{x,N}(\vec{b})/(f_{x,N}(b_0) + \cdots + f_{x,N}(b_{m-1})) \in \mathbb{R}^m$ the discretization of $f_{x,N}$ into m bins. Then, there exist constants $C_1, C_2 > 0$ such that*

$$s(y^{(N)}) = C_1 - \frac{C_2}{N^{2t-1}} + O(1/N^{2t}). \tag{2.2.3}$$

Proof of Claim 2 of Theorem 2.2.3. We can estimate that

$$\begin{aligned}
s(y^{(N)}) &= \frac{1}{\sum_{j=0}^{m-1} f_{x,N}(b_j)} \sum_{\sigma=1}^{\infty} \alpha_\sigma \|f_{x,N}(\vec{b}) - g_\sigma * f_{x,N}(\vec{b})\|_2 && \text{(Definition 2.2.2)} \\
&= \frac{1}{\sum_{j=0}^{m-1} f_{x,N}(b_j)} \sum_{\sigma=1}^{\infty} \alpha_\sigma \Big(\|B_1(m,\sigma) f_{x,N}(\vec{b}) - (\tilde{g}_\sigma * f_{x,N})(\vec{b})\|_2 && \text{(Lemma A.8.3)} \\
&\qquad + \sqrt{m} O(1/N^{2t+1}) \Big) \\
&= \frac{\sqrt{m}}{\sum_{j=0}^{m-1} f_{x,N}(b_j)} \sum_{\sigma=1}^{\infty} \alpha_\sigma \Big(B_2(\sigma,m,x) - \frac{B_1(m,\sigma)^2 B_3(x)^2}{(2t-1)N^{2t-1}} && \text{(Lemma A.8.5)} \\
&\qquad + O(1/N^{2t}) + O(1/N^{2t+1}) \Big) \\
&= \frac{2}{\sqrt{m}} \cdot \left(C_3 - \frac{C_4}{N^{2t-1}} + O(1/N^{2t}) \right). && \text{(Lemmata A.8.5, A.8.6)}
\end{aligned}$$

In the last step we used the convergence of the respective series (which follows from boundedness of $B_2(\sigma,m,x)$ and $B_1(m,\sigma)$ in σ) and we assigned C_3 and C_4 to be those sums. This completes the proof. □

Proof of Claim 1 of Theorem 2.2.3. The proof of this claim 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 [110, Ch. 2, Cor. 2.4]. In other words, the function f_N being learnt by the Fourier head converges uniformly and absolutely to f. □

A.8.4 Proving the Lemmata

In this subsection, we will restate and prove Lemmata A.8.3, A.8.5, and A.8.6.

Lemma A.8.3. (Discrete convolution is close to continuous convolution) *If we define the constant* $B_1(m,\sigma) := 1 + \frac{G_\sigma(m)}{S(m,\sigma)}$*, then we have that*

$$\|f_{x,N}(\vec{b}) - g_\sigma * f_{x,N}(\vec{b})\|_2 = \|B_1(m,\sigma) f_{x,N}(\vec{b}) - \tilde{g}_\sigma * f_{x,N}(\vec{b})\|_2 + \sqrt{m} O(1/N^{2t+1}). \tag{A.8.6}$$

Furthermore, $B_1(m,\sigma)$ *satisfies the following bound, uniformly in* σ*,*

$$B_1(m,\sigma) \leq 1 + \frac{1}{2m-1}. \tag{A.8.7}$$

Proof of Lemma A.8.3. Extending $f_{x,N}$ periodically to $[-2,2]$, we can compute that the continuous convolution $\tilde{g}_\sigma * f_{x,N}(z)$ is

$$\begin{aligned}
(\tilde{g}_\sigma * f_{x,N})(z) &= \int_{-2}^{2} \tilde{g}_\sigma(u) f_{x,N}(z-u)\, du && \text{(A.8.13)} \\
&= \frac{m}{2S(m,\sigma)} \int_{-2}^{2} G_\sigma\left(\frac{mu}{2}\right) f_{x,N}(z-u)\, du && \text{(A.8.14)} \\
&= \frac{1}{S(m,\sigma)} \int_{-m}^{m} G_\sigma(s) f_{x,N}\left(z - \frac{2s}{m}\right) ds, && \text{(A.8.15)}
\end{aligned}$$

where in the third step we applied the change of variables $s = \frac{mu}{2}$. We claim that this is precisely a continuous approximation of the discrete convolution in Definition 2.2.2. To see this, we will apply the Euler-Maclaurin formula. This formula says that the integral in Equation A.8.15 is a Riemann sum over rectangles of width 1 evaluated at the right endpoints of each interval, minus an error term $E(m, \sigma)$, as follows:

$$(\tilde{g}_\sigma * f_{x,N})(b_j) + E(m,\sigma) = \frac{1}{S(m,\sigma)} \sum_{k=1-m}^{m} G_\sigma(k) \cdot f_{x,N}\left(b_j - \frac{2k}{m}\right) \tag{A.8.16}$$

$$= \frac{1}{S(m,\sigma)} \sum_{k=1-m}^{m} G_\sigma(k) \cdot f_{x,N}\left(-1 + \frac{2j+1}{m} - \frac{2k}{m}\right) \tag{A.8.17}$$

$$= \frac{1}{S(m,\sigma)} \sum_{k=1-m}^{m} G_\sigma(k) \cdot f_{x,N}\left(-1 + \frac{2(j-k)+1}{m}\right) \tag{A.8.18}$$

$$= \frac{1}{S(m,\sigma)} \left(\sum_{k=1-m}^{m-1} G_\sigma(k) \cdot f_{x,N}\left(b_{j-k}\right) + G_\sigma(m) f_{x,N}(b_{j-m}) \right) \tag{A.8.19}$$

$$= \frac{1}{S(m,\sigma)} \left(S(m,\sigma) \cdot (g_\sigma * f_{x,N})(b_j) + G_\sigma(m) f_{x,N}(b_j) \right) \tag{A.8.20}$$

$$= (g_\sigma * f_{x,N})(b_j) + \frac{1}{S(m,\sigma)} G_\sigma(m) f_{x,N}(b_j), \tag{A.8.21}$$

where the error term is defined as

$$E(m,\sigma) := \frac{1}{S(m,\sigma)} \int_{-m}^{m} \frac{d\left(G_\sigma(s) f_{x,N}\left(z - \frac{2s}{m}\right)\right)}{ds} P_1(s)\, ds \tag{A.8.22}$$

$$+ \frac{1}{2S(m,\sigma)} \left(G_\sigma(m) f_{x,N}(z-2) - G_\sigma(-m) f_{x,N}(z+2) \right), \tag{A.8.23}$$

where $P_1(s) := s - \lfloor s \rfloor - 1/2$ is the periodized Bernoulli polynomial. We will now estimate this error term. Note that since G_σ is an even function and $f_{x,N}$ is periodic with period 2, the difference in A.8.23 is 0. Therefore, we can compute that

$$E(m,\sigma) = \frac{1}{S(m,\sigma)} \int_{-m}^{m} G'_\sigma(s) P_1(s) f_{x,N}\left(z - \frac{2s}{m}\right) ds \tag{A.8.24}$$

$$- \frac{2}{mS(m,\sigma)} \int_{-m}^{m} G_\sigma(s) P_1(s) f'_{x,N}\left(z - \frac{2s}{m}\right) ds. \tag{A.8.25}$$

Using the triangle inequality, we can bound $E(m,\sigma)$ in terms of convolutions with $\tilde{g}_\sigma$:

$$|E(m,\sigma)| \le \frac{1}{S(m,\sigma)} \left| \int_{-m}^{m} G'_\sigma(s) P_1(s) f_{x,N}\left(z - \frac{2s}{m}\right) ds \right| \tag{A.8.26}$$

$$+ \frac{1}{S(m,\sigma)} \left| \int_{-m}^{m} G_\sigma(s) P_1(s) f'_{x,N}\left(z - \frac{2s}{m}\right) \frac{-2}{m}\, ds \right| \tag{A.8.27}$$

$$\le \frac{1}{S(m,\sigma)} \left(\frac{m}{2\sigma^2} \int_{-m}^{m} G_\sigma(s) f_{x,N}\left(z - \frac{2s}{m}\right) ds + \right. \tag{A.8.28}$$

$$\left. + \frac{1}{m} \int_{-m}^{m} G_\sigma(s) \left| f'_{x,N}\left(z - \frac{2s}{m}\right) \right| ds \right) \tag{A.8.29}$$

$$= \frac{m}{2\sigma^2} (\tilde{g}_\sigma * f_{x,N})(z) + \frac{1}{m} (\tilde{g}_\sigma * |f'_{x,N}|)(z), \tag{A.8.30}$$

where in Equation A.8.29 we used that $|P_1(s)| \leq 1/2$ and that $|G'_\sigma(s)| = |s|\, G_\sigma(s)/\sigma^2 \leq mG_\sigma(s)/\sigma^2$ for $s \in [-m, m]$.

Note that since $\tilde{g}_\sigma$ is a truncated Gaussian on $[-2, 2]$, it is infinitely differentiable on the open set $(-2, 2)$, however, it is not differentiable at the endpoints -2 and 2 when treated as a 4-periodic function. This technical difficulty can be resolved using mollifiers: we can replace $\tilde{g}_\sigma$ with $\tilde{g}_\sigma * \varphi_\epsilon$, where $\{\varphi_\epsilon\}$ is a family of mollifiers indexed by $\epsilon > 0$. The key properties of a mollifier are that $\tilde{g}_\sigma * \varphi_\epsilon$ is infinitely differentiable as a 4-periodic function for all $\epsilon > 0$ and $\lim_{\epsilon \to 0} \tilde{g}_\sigma * \varphi_\epsilon = \tilde{g}_\sigma$ [111, Ch. 3]. We are ultimately interested in only bounds on absolute values of $\tilde{g}_\sigma$ convolved with various functions, and since absolute values are continuous and inequalities are preserved under taking limits, all our bounds are still true. In particular, this shows that the k'th Fourier coefficients of $\tilde{g}_\sigma$ decay faster than any polynomial. And on the other hand, by assumption we know that the Fourier coefficients of $f_{x,N}$ decay on the order of $1/k^t$; and we know that $|f'_{x,N}|$ is continuous and 2π periodic, so its Fourier coefficients converge. So by the convolution theorem, we can deduce that the Fourier coefficients of $\tilde{g} * f_{x,N}$ and $\tilde{g}_\sigma * |f'_{x,N}|$ decay faster than any polynomial. Summed over the N frequencies, this shows that $|\tilde{g}_\sigma * f_{x,N}(x)|$ and $|\tilde{g}_\sigma * \left|f'_{x,N}\right|(z)|$ decay faster than any polynomial as well. Since m is fixed and $\sigma \geq 1$, this implies that

$$|E(m, \sigma)| = O(1/N^{2t+1}). \tag{A.8.31}$$

Using Definition A.8.1 and Equation A.8.16, we have that

$$g_\sigma * f_{x,N}(b_j) = \frac{1}{S(m,\sigma)} \sum_{k=1-m}^{m} G_\sigma(k) \cdot f_{x,N}\left(b_j - \frac{2k}{m}\right) - \frac{1}{S(m,\sigma)} G_\sigma(m) f_{x,N}(b_j) \tag{A.8.32}$$

$$= (\tilde{g}_\sigma * f_{x,N})(b_j) + E(m, \sigma) - \frac{1}{S(m,\sigma)} G_\sigma(m) f_{x,N}(b_j). \tag{A.8.33}$$

If we define $C_1(m, \sigma) := \frac{G_\sigma(m)}{S(m,\sigma)}$, then Equation A.8.33 combined with A.8.31 together imply that

$$\left| g_\sigma * f_{x,N}(b_j) - \tilde{g}_\sigma * f_{x,N}(b_j) + C_1(\sigma, m) f_{x,N}(b_j) \right| = O(1/N^{2t+1}). \tag{A.8.34}$$

Finally, we can estimate that

$$\|f_{x,N}(\vec{b}) - g_\sigma * f_{x,N}(\vec{b})\|_2 \tag{A.8.35}$$

$$= \|f_{x,N}(\vec{b}) - \tilde{g}_\sigma * f_{x,N}(\vec{b}) + C_1(\sigma, m) f_{x,N}(\vec{b})\|_2 \tag{A.8.36}$$

$$+ \|g_\sigma * f_{x,N}(\vec{b}) - \tilde{g}_\sigma * f_{x,N}(\vec{b}) + C_1(m, \sigma) f_{x,N}(\vec{b})\| \tag{A.8.37}$$

$$= \|f_{x,N}(\vec{b}) - \tilde{g}_\sigma * f_{x,N}(\vec{b}) + C_1(m, \sigma) f_{x,N}(\vec{b})\|_2 + \sqrt{m}\|O(1/N^{2t+1})\|_2 \tag{A.8.38}$$

$$= \|(1 + C_1(m, \sigma)) f_{x,N}(\vec{b}) - \tilde{g}_\sigma * f_{x,N}(\vec{b})\|_2 + \sqrt{m} O(1/N^{2t+1}). \tag{A.8.39}$$

This completes the first part of the proof. For the second part of the proof, since $S(m, \sigma) \geq (2m-1)G_\sigma(m-1)$, we can estimate that

$$B_1(m, \sigma) \leq 1 + \frac{G_\sigma(m)}{(2m-1)G_\sigma(m-1)} \leq 1 + \frac{1}{2m-1} e^{-(2m-1)/2\sigma^2} \leq 1 + \frac{1}{2m-1}. \tag{A.8.40}$$

This completes the proof. □

Lemma A.8.5. (Main term asymptotic) *Denote by $a_0(x)$ the constant coefficient of $f_{x,N}$. Let us suppose that the Fourier coefficients of $f_{x,N}$ decay like $B_3(x)/k^t$, and define the constant*

$$B_2(\sigma, m, x) := \sqrt{a_0(x)^2 B_1(m,\sigma)^2 + 2B_1(m,\sigma)^2 B_3(x)^2 \zeta(2t)}. \tag{A.8.9}$$

Then we know that

$$\|B_1(m,\sigma) f_{x,N}(\vec{b}) - \tilde{g}_\sigma * f_{x,N}(\vec{b})\|_2 \tag{A.8.10}$$

$$= \sqrt{m}\left(B_2(\sigma, m, x) - \frac{B_1(m,\sigma)^2 B_3(x)^2}{2t-1} \cdot \frac{1}{N^{2t-1}} + O(1/N^{2t})\right). \tag{A.8.11}$$

Furthermore, $B_2(\sigma, m, x)$ is bounded from above and below as a function of σ and m.

Proof of Lemma A.8.5. We will first argue that $B_2(\sigma, m, x)$, as a function of σ and m, is bounded from above and below. Indeed, from its definition, $B_2(\sigma, m, x) \geq \sqrt{a_0(x)^2 B_1(m,\sigma)^2} \geq |a_0(x)|$. But the Fourier PDF has integral 1 over $[-1, 1]$, so its constant term is $a_0(x) = 1/2$. This implies that $B_2(\sigma, m, x) \geq 1/2$. To see $B_2(\sigma, m, x)$ is bounded from above, we simply recall that in Lemma A.8.3 we showed that $|B_1(m,\sigma)| \leq 2$, which implies that $|B_2(\sigma, m, x)| \leq \sqrt{4a_0(x)^2 + 8B_3(x)^2\zeta(2t)}$. This shows that $B_2(\sigma, m, x)$ is bounded above and below as a function of m and σ, as claimed.

Now, let $(d_0(x), \ldots, d_{m-1}(x)) \in \mathbb{R}^m$ be the discrete Fourier transform of $(B_1(m,\sigma) f_{x,N} - \tilde{g}_\sigma * f_{x,N})(\vec{b}) \in \mathbb{R}^m$. For notational simplicity, we will write $B_1 = B_1(m,\sigma)$ as long as σ is fixed. By Plancherel's Theorem, we have

$$\sum_{j=0}^{m-1} |(B_1 f_{x,N} - \tilde{g}_\sigma * f_{x,N})(b_j)|^2 = \frac{1}{m} \sum_{k=0}^{m-1} |d_k(x)|^2. \tag{A.8.41}$$

Let $h_{\sigma,j}$ be the Fourier coefficients of $\tilde{g}_\sigma$, treated as a periodic function with period 4, and defined over $[-2, 2]$:

$$\tilde{g}_\sigma(z) = \sum_{j=-\infty}^{\infty} h_{\sigma,j} e^{\pi i j z/2}. \tag{A.8.42}$$

Since $f_{x,N}$ is defined over $[-1, 1]$ and is periodic with period 2, we can likewise treat it as a function over $[-2, 2]$ with period 4, in which case we can rewrite its Fourier series as

$$f_{x,N}(z) = \sum_{j=-2N}^{2N} \tilde{a}_j(x) e^{\pi i j z/2}, \tag{A.8.43}$$

where

$$\tilde{a}_j(x) = \begin{cases} a_{j/2}(x) & \text{if } j \equiv 0 \pmod 2 \\ 0 & \text{else.} \end{cases} \tag{A.8.44}$$

Then, by the Convolution Theorem, we have

$$(B_1 f_{x,N} - \tilde{g}_\sigma * f_{x,N})(b_l) = \sum_{j=-2N}^{2N} \tilde{a}_j(x)(B_1 - h_{\sigma,j}) \cdot e^{\pi i j b_l/2} \tag{A.8.45}$$

$$= \sum_{k=-N}^{N} \tilde{a}_{2k}(x)(B_1 - h_{\sigma,2k}) \cdot e^{\pi i k b_l} \tag{A.8.46}$$

$$= \sum_{k=-N}^{N} a_k(x)(B_1 - h_{\sigma,2k}) \cdot e^{\pi i k b_l}, \tag{A.8.47}$$

where in the second equality we used the fact that $\tilde{a}_j$ is 0 for odd j and therefore re-indexed using $k = j/2$. Thus, using the definition of DFT along with Equation A.8.47, we get

$$d_k(x) = \sum_{l=0}^{m-1} (B_1 f_{x,N} - \tilde{g}_\sigma * f_{x,N})(b_l) \cdot e^{-2\pi i k l/m} \tag{A.8.48}$$

$$= \sum_{l=0}^{m-1} \sum_{j=-N}^{N} a_j(x)(B_1 - h_{\sigma,2j}) e^{\pi i j b_l} \cdot e^{-2\pi i k l/m} \tag{A.8.49}$$

$$= \sum_{j=-N}^{N} a_j(x)(B_1 - h_{\sigma,2j}) \sum_{l=0}^{m-1} e^{\pi i j(-1+\frac{2l+1}{m})} \cdot e^{-2\pi i k l/m} \tag{A.8.50}$$

$$= \sum_{j=-N}^{N} a_j(x)(B_1 - h_{\sigma,2j}) \cdot e^{\pi i j(-1+1/m)} \sum_{l=0}^{m-1} e^{2\pi i (j-k) l/m}. \tag{A.8.51}$$

First, we claim that at most a single summand (in j) is represented. Towards this, we note that

$$\sum_{l=0}^{m-1} e^{2\pi i (j-k) l/m} = \begin{cases} 0 & j \not\equiv k \pmod{m} \\ m & j \equiv k \pmod{m}. \end{cases} \tag{A.8.52}$$

Then, we note that since $0 < N < m/2$, for each $0 \le k < m$, there is at most one $j \in \{-N, -N+1, \ldots, N\}$, such that $j \equiv k \pmod{m}$. This shows that there is at most a single summand. We will now find the exact formula for each summand. We consider three cases.

- Case 1: $0 \le k \le N$. In this case, $j = k$ satisfies $j \equiv k \pmod{m}$, so this index gives the exponential sum of m.
- Case 2: $N < k < m - N$. In this case, k is too large to be an index in the sum, so we can't choose $j = k$; the next smallest equivalent value is $j = k - m$, which satisfies $j \equiv k \pmod{m}$. But $N - m < j < -N$ in this case, so k is too small to be an index in the sum; therefore, every exponential sum is zero in this range.
- Case 3: $m - N \le k \le m - 1$. In this case, $j = k - m$ satisfies $j \equiv k \pmod{m}$. We have $-N \le j \le 1$, so this is a valid index in the sum.

This gives the following closed formula:

$$d_k(x) = \begin{cases} m \cdot a_k(x)(B_1 - h_{\sigma,2k})e^{\pi ik(-1+1/m)} & 0 \le k \le N \\ 0 & N < k < m - N \\ m \cdot a_{k-m}(B_1 - h_{\sigma,2(k-m)})e^{\pi i(k-m)(-1+1/m)} & m - N \le k \le m - 1. \end{cases} \tag{A.8.53}$$

Using this closed formula in A.8.41, we obtain

$$\sum_{j=0}^{m-1} |(B_1 f_{x,N} - \tilde{g}_\sigma * f_{x,N})(b_j)|^2 \tag{A.8.54}$$

$$= \frac{1}{m} \sum_{k=0}^{N} \left| m \cdot a_k(x)(B_1 - h_{\sigma,2k})e^{\pi ik(-1+1/m)} \right|^2 \tag{A.8.55}$$

$$+ \frac{1}{m} \sum_{k=m-N}^{m-1} \left| m \cdot a_{k-m}(B_1 - h_{\sigma,2(k-m)})e^{\pi i(k-m)(-1+1/m)} \right|^2 \tag{A.8.56}$$

$$= m \sum_{k=0}^{N} |a_k(x)(B_1 - h_{\sigma,2k})|^2 + m \sum_{k=m-N}^{m-1} \left| a_{k-m}(B_1 - h_{\sigma,2(k-m)}) \right|^2, \tag{A.8.57}$$

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 $\tilde{g}_\sigma$ is a real and even function, we know that its Fourier coefficients $h_{\sigma,k}$ are real. Further, since $\tilde{g}_\sigma$ is infinitely differentiable, we also know that $h_{\sigma,k} = O(1/k^t)$ (in fact they decay faster than $1/k^p$ for any $p \ge 1$). Thus, using that $|a_k(x)|$ decays like $B_3(x)/k^t$, we see

$$|a_k(x)(B_1 - h_{\sigma,2k})|^2 = |a_k(x)|^2 (B_1^2 - 2h_{\sigma,2k}(x) + h_{\sigma,2k}^2) \tag{A.8.58}$$

$$= \frac{B_3^2 B_1^2}{k^{2t}} + O\left(\frac{1}{k^{2t}(2k)^t}\right) + O\left(\frac{1}{k^{2t}(2k)^{2t}}\right). \tag{A.8.59}$$

From A.8.59, 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, we can estimate that

$$\sum_{j=0}^{m-1} |(B_1 f_{x,N} - \tilde{g}_\sigma * f_{x,N})(b_j)|^2 \approx m B_1^2 a_0(x)^2 + m \sum_{k=1}^{N} \frac{B_3^2 B_1^2}{k^{2t}} + m \sum_{k=m-N}^{m-1} \frac{B_3^2 B_1^2}{(k-m)^{2t}} \tag{A.8.60}$$

$$= m a_0(x)^2 B_1^2 + m \sum_{k=1}^{N} \frac{B_3^2 B_1^2}{k^{2t}} + m \sum_{k=-N}^{-1} \frac{B_3^2 B_1^2}{k^{2t}} \tag{A.8.61}$$

$$= m a_0(x)^2 B_1^2 + m \sum_{k=1}^{N} \frac{B_3^2 B_1^2}{k^{2t}} + m \sum_{k=1}^{N} \frac{B_3^2 B_1^2}{k^{2t}} \tag{A.8.62}$$

$$= m a_0(x)^2 B_1^2 + 2m \sum_{k=1}^{N} \frac{B_3^2 B_1^2}{k^{2t}}. \tag{A.8.63}$$

Next, we note that our asymptotic in Lemma A.8.4, applied at $2t$, yields

$$\sum_{k=1}^{N} \frac{1}{k^{2t}} = \zeta(2t) - \frac{1}{2t-1} \frac{1}{N^{2t-1}} + O(1/N^{2t}). \tag{A.8.64}$$

Substituting this into A.8.63, we obtain

$$\sum_{j=0}^{m-1} |(B_1 f_{x,N} - \tilde{g}_\sigma * f_{x,N})(b_j)|^2 \tag{A.8.65}$$

$$= m a_0(x)^2 B_1^2 + 2m B_1^2 B_3^2 \left(\zeta(2t) - \frac{1}{2t-1} \frac{1}{N^{2t-1}} + O(1/N^{2t}) \right) \tag{A.8.66}$$

$$= m \left(B_2^2 - \frac{2B_1^2 B_3^2}{2t-1} \frac{1}{N^{2t-1}} + B_1^2 B_3^2 O(1/N^{2t}) \right) \tag{A.8.67}$$

$$= m \left(B_2^2 - \frac{2B_1^2 B_3^2}{2t-1} \frac{1}{N^{2t-1}} + O(1/N^{2t}) \right), \tag{A.8.68}$$

where we defined $B_2 = B_2(\sigma, m, x) := \sqrt{a_0(x)^2 B_1^2 + 2B_1^2 B_3^2 \zeta(2t)}$ as in the statement of the Lemma, and in the third line we applied Lemma A.8.3 to estimate that $B_1 \leq 2$, and we used that $B_3 = B_3(x)$ only depends on x. Then, using the Taylor expansion $(1+x)^{1/2} = 1 + \frac{x}{2} + O(x^2)$ about 0, we can estimate that

$$\left(\sum_{j=0}^{m-1} |(B_1 f_{x,N} - \tilde{g}_\sigma * f_{x,N})(b_j)|^2 \right)^{1/2} \tag{A.8.69}$$

$$= \left(m B_2^2 - \frac{m 2B_1^2 B_3^2}{2t-1} \frac{1}{N^{2t-1}} + m O(1/N^{2t}) \right)^{1/2} \tag{A.8.70}$$

$$= \sqrt{m} B_2 \left(1 - \frac{2B_1^2 B_3^2}{(2t-1)B_2^2} \frac{1}{N^{2t-1}} + \frac{1}{B_2^2} O(1/N^{2t}) \right)^{1/2} \tag{A.8.71}$$

$$= \sqrt{m} B_2 \left(1 - \frac{1}{2} \cdot \frac{2B_1^2 B_3^2}{(2t-1)B_2^2} \frac{1}{N^{2t-1}} + O(1/N^{2t}) \right) \tag{A.8.72}$$

$$= \sqrt{m} \left(B_2 - \frac{B_1^2 B_3^2}{2t-1} \cdot \frac{1}{N^{2t-1}} + O(1/N^{2t}) \right). \tag{A.8.73}$$

To justify our application of the Taylor expansion, we note that $N \gg 1$, and $B_2 = B_2(\sigma, m, x)$ is bounded below as a function of σ and m. This completes the proof. □

Lemma A.8.6. (The average value of the truncated Fourier PDF is $1/2$) *If $N < m/2$, then*

$$\sum_{j=0}^{m-1} f_{x,N}(b_j) = \frac{m}{2}. \tag{A.8.12}$$

Proof of Lemma A.8.6. Denote by a_k the Fourier coefficients of $f_{x,N}$. We can compute that

$$\sum_{j=0}^{m-1} f_{x,N}(b_j) = \sum_{j=0}^{m-1} \sum_{k=-N}^{N} a_k e^{i\pi k\left(-1+\frac{2j+1}{m}\right)} \tag{A.8.74}$$

$$= \sum_{k=-N}^{N} a_k e^{-i\pi k} e^{\frac{i\pi k}{m}} \sum_{j=0}^{m-1} e^{\frac{2\pi i j k}{m}}. \tag{A.8.75}$$

Note that by hypothesis, $N < m/2$, which implies that $|k| < m$ for every outer sum index k. We consider two cases; if $k = 0$, then the innermost summand is $e^{\frac{2\pi i j k}{m}} = 1$; and if $k \neq 0$, then the innermost sum is a

truncated geometric series with first term 1, common ratio $e^{\frac{2\pi ik}{m}}$, and m terms. In summary, the innermost summand is

$$\sum_{j=0}^{m-1} e^{\frac{2\pi ijk}{m}} = \begin{cases} m & k = 0 \\ 0 & k \neq 0, \end{cases} \tag{A.8.76}$$

which implies that $\sum_{j=0}^{m-1} f_{x,N}(b_j) = ma_0$. But $a_0 = 1/2$ because $f_{x,N}$ is a PDF implies it has average value $1/2$ over $[-1, 1]$. This completes the proof. □

A.9 FH: Smoothness Metric

We will examine how the proposed smoothness metric Equation 2.2.1 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:

$$f(x) = \frac{4}{\pi} \sum_{n=1,3,5,\ldots}^{\infty} \frac{1}{n} \sin\left(\frac{n\pi x}{L}\right). \tag{A.9.1}$$

Here, the wavelength is $2L$ [126].

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 A16.

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 A17 demonstrates that this is indeed the case.

Choice of L2 Distance over L1 Distance: The proposed smoothness metric Equation 2.2.1 permits a general measure of discrepancy D, and we've chosen D to be L^2 distance as indicated in 2.2.2. 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 A17). 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* [54], which has a similar purpose in quantifying the "smoothness" 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:

$$H(f; N) = \sum_{n \in N} p(n) \log_2\left(\frac{1}{p(n)}\right) = -\sum_{n \in N} \frac{S_n}{S_{total}} \log_2\left(\frac{S_n}{S_{total}}\right) \tag{A.9.2}$$

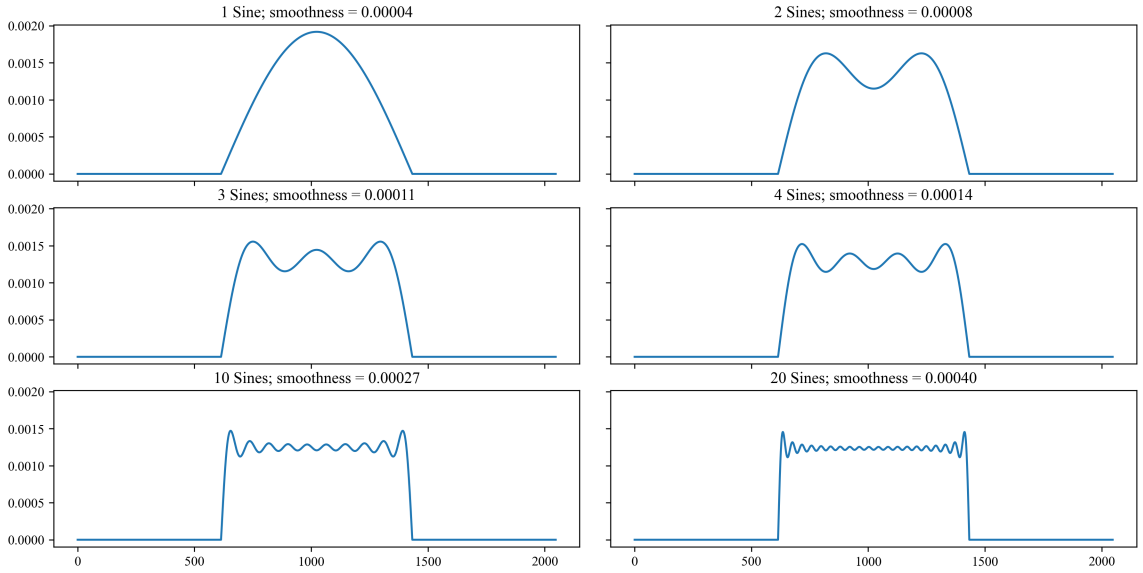


Figure A16: Truncated square waves framed as densities and their smoothness.

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 A1, 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

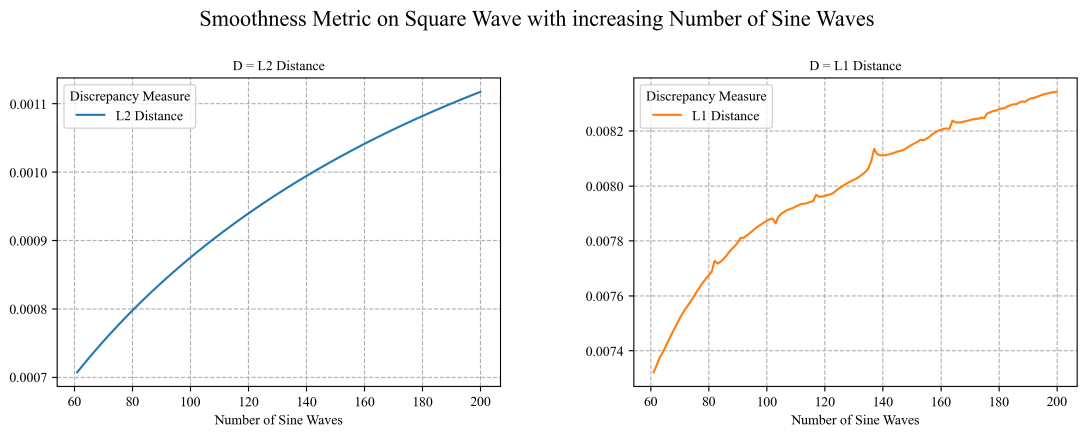


Figure A17: Values of the smoothness metric 2.2.2 on our square-wave-like multinomials as we increase the number of sine waves. We desire the value of this metric to be close to zero when there are few sine waves, and be monotonically increasing with each additional wave, indicating that adding more high frequency content results in a less smooth distribution. 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.

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.

Discrepancy	Noise	Mean $\pm$ 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 A1: 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.

A.10 FH: Additional Experiment Details, Toy Examples

A.10.1 Motivating Example: Audio Spectrogram Transformer

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 [39]. We consider the task of beats per minute (BPM) classification for metronome-like audio samples [125] within the tempo range $\{50, 51, \ldots, 210\}$. While this task is not difficult, we use this audio classification task to illustrate

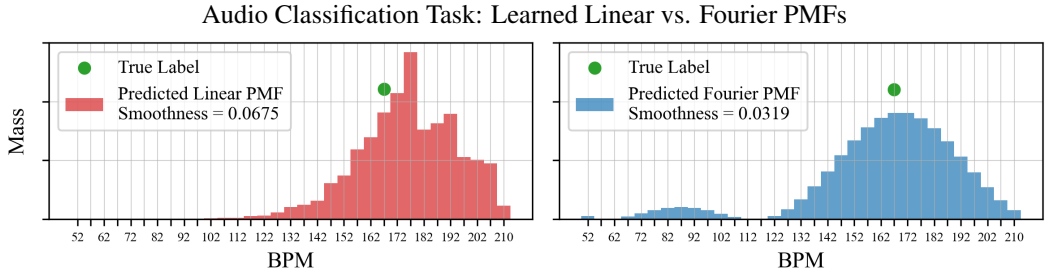


Figure A18: Comparison between the PMF learned by the linear head, and the Fourier head with 2 frequencies, for the toy BPM classification task, on a single audio example. We observe that the Fourier head learns a smoother categorical distribution over its predicted values, and is better centered around 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.

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 [34], 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 the inductive bias of continuity that the Fourier head imparts. In Figure A18 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.

A.10.2 Learning a Continuous Density

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

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.

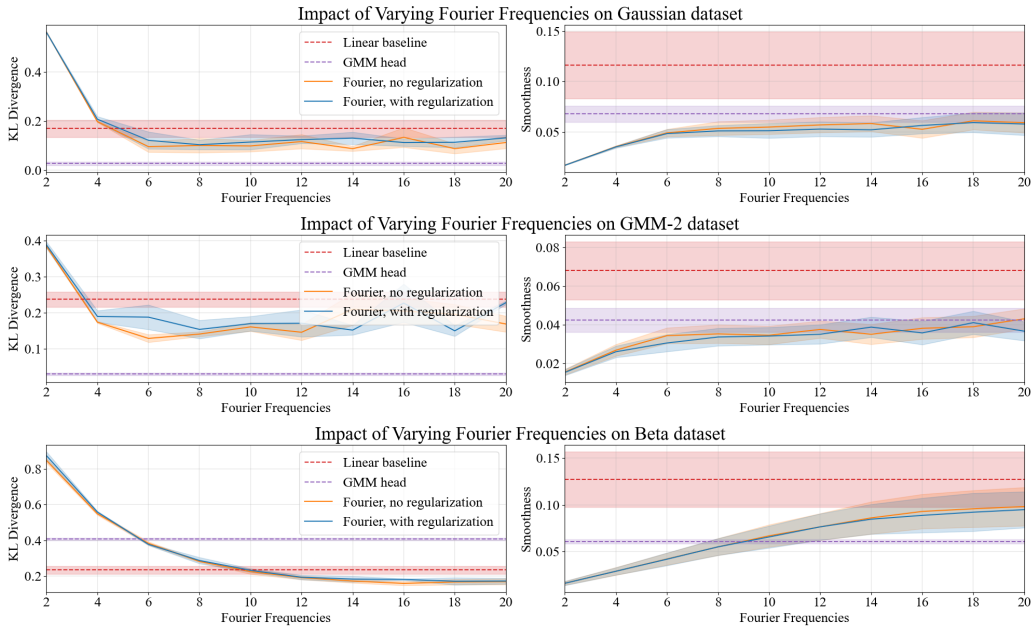


Figure A19: We study how the quantity of Fourier frequencies impacts KL divergence and smoothness for the toy example on each dataset. For both KL divergence and smoothness, lower is better. We observe that the Fourier models with and without regularization performed similarly to each other, and outperformed 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 2.2.3.

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 A19, we present results from training over a range of frequencies, and for each frequency we ran experiments with and without Fourier regularization. In Table A3 we present results on the MSE metric, that show that the Fourier head outperforms the linear classification head.

A.10.3 MLE-based Fourier Head

We carry out experiments in the continuous domain analogous to those we did in the quantized domain from the toy example.

Dataset: We use the same synthetic datasets–Gaussian, GMM-2, and Beta–as in the previous subsection, except we do not quantize the data into bins.

Task: Predict the conditional distribution of z given (x, y).

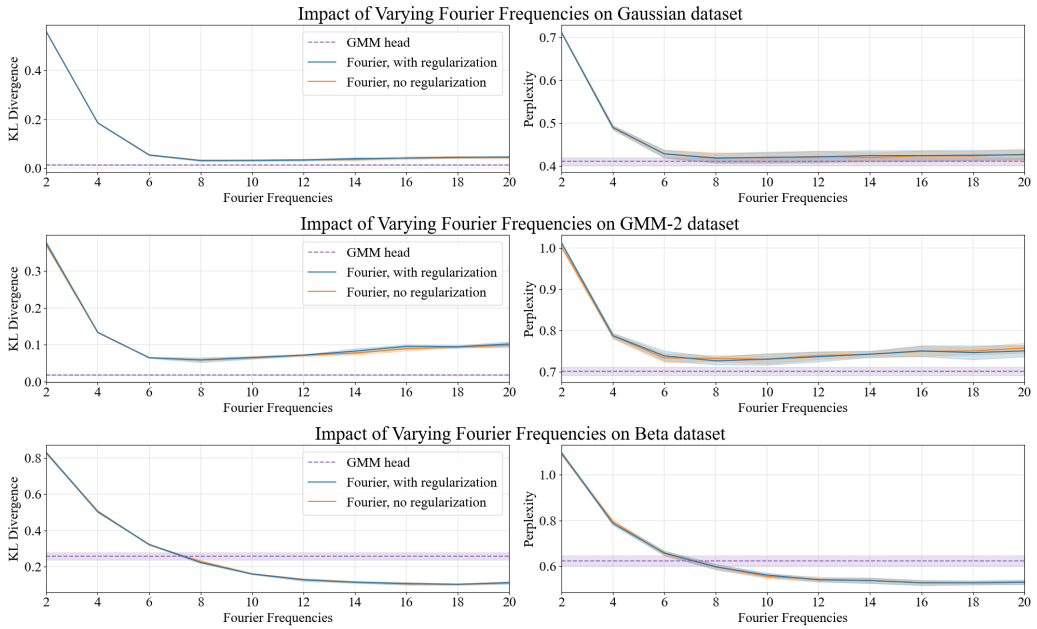


Figure A20: We study how the quantity of Fourier frequencies impacts KL divergence and perplexity for the toy example on each dataset for the MLE experiments. For both KL divergence and perplexity, lower is better. We observe that the Fourier models with and without regularization performed similarly to each other. 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) has the best KL divergence on the Beta dataset for sufficiently large Fourier frequencies, since it is non-Gaussian.

Model architecture: Our model is an MLP with one hidden layer and the final layer is an MLE-based Fourier head which returns the $2N + 1$ learned real coefficients of the Fourier series, mapping $\mathbb{R}^2 \to \mathbb{R}^{64} \to \mathbb{R}^{32} \to \mathbb{R}^{2N+1}$. Alongside the Fourier-MLE model, we consider a baseline where the final layer is a Gaussian model mixture whose means and standard deviations are learned using an MLE objective. For the MLE-Fourier model, we sweep over frequencies $N = 2, 4, \ldots, 20$ and regularization $\gamma \in \{0, 10^{-6}\}$.

Model evaluation: We use two metrics for evaluation. The first metric is the average KL divergence $D_{\text{KL}}(\mathcal{P}(x, y) \| M(x, y))$, where $\mathcal{P}(x, y)$ is the fixed conditional distribution of z given (x, y) and $M(x, y)$ denotes the predicted probability density function of z. Our second metric is perplexity, which is the exponential of the average negative log likelihood of the test set.

Results: The metrics for the best performing model on each dataset are reported in Table A4. Figure A22 presents sample visualizations of the learned conditional distributions alongside the true densities. While, as expected, the GMM-MLE head outperforms the Fourier-MLE head on the Gaussian and GMM-2 datasets due to the Gaussian nature of the datasets, the Fourier-MLE head outperforms the GMM-MLE head on the Beta dataset, highlighting the flexibility of the Fourier-MLE head in learning a large variety of probability distributions. In Appendix A.10.2, we present the results of a study on the impact of number of frequencies and Fourier regularization in the MLE setting.

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

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

Table A2: 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.

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 A3: 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.

A.10.4 Are LLMs Random Number Generators?

Dataset: We create a training dataset using the prompt template: *"The following is a list of normally distributed random numbers in the interval [-1, 1] with mean μ and std σ: $x_1, x_2,$ "* and response template: "x_3", where $(\mu, \sigma) \in \{(-0.55, 0.10), (-0.03, 0.24), (0.42, 0.16), (0.55, 0.10)\}$, and each $x_i \sim \mathcal{N}(\mu, \sigma)$. We write each number using two decimal places. Our training dataset consists of 256 such (prompt, response) pairs, divided evenly among the four distributions.

Model Architecture: We consider three different models: the original Llama-3.1-8B-Instruct model [31], the original model after LoRA fine-tuning, and the original model where we replace the linear classification head with the Fourier head and perform LoRA fine-tuning. For the Fourier head, we use an output dimension of 200 and the original latent space $[-1, 1]$ because of our chosen decimal precision. We conduct LoRA fine-tuning for 16 epochs with a learning rate of $3 * 10^{-4}$ and a linear decay schedule, and a batch size of 64. We release all of our training code on our project page.

Model Evaluation: We compute two metrics: the first is Total Variation Distance; we define this to be one

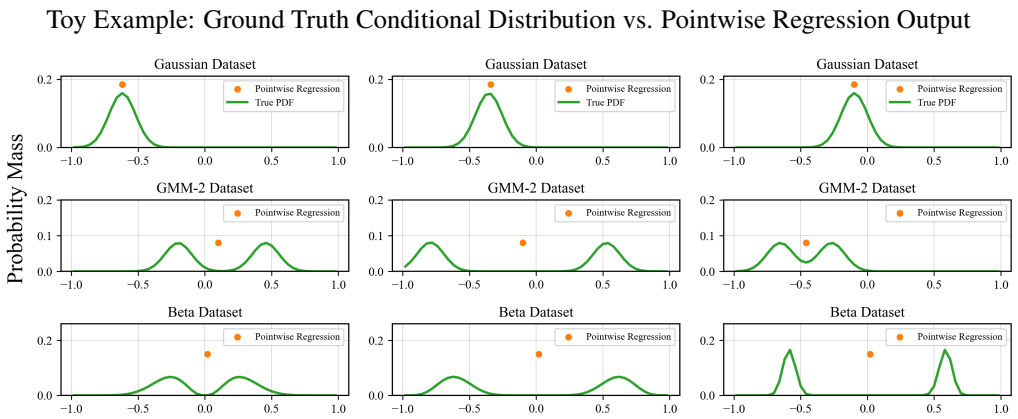


Figure A21: We present some examples of the ground truth conditional distribution versus the point predicted by the Pointwise Regression model. The regression model simply regresses to the mean of the conditional distribution. Accordingly, the regression model performs extremely well for the unimodal Gaussian dataset, and it performs poorly for the bimodal datasets GMM-2 and Beta.

	KL Divergence (↓)		Perplexity (↓)	
Dataset	GMM-MLE	Fourier-MLE	GMM-MLE	Fourier-MLE
Gaussian	**0.012** $\pm$ 0.002	0.034 $\pm$ 0.003	**0.410** $\pm$ 0.014	0.422 $\pm$ 0.019
GMM-2	**0.018** $\pm$ 0.001	0.072 $\pm$ 0.005	**0.702** $\pm$ 0.015	0.740 $\pm$ 0.014
Beta	0.257 $\pm$ 0.03	**0.130** $\pm$ 0.005	0.623 $\pm$ 0.035	**0.542** $\pm$ 0.017

Table A4: We compare metrics between the GMM-MLE head, and the Fourier-MLE head with 12 frequencies and no regularization, for every dataset in our toy example. We aggregate metrics over 4 different seeds and report the standard deviation.

half of the L^∞ distance between the ground truth quantized Gaussian histogram, and the empirical histogram of samples, quantized into 20 bins. Our second metric is the Quantity of Unique Samples. In Figure 2.3 we present example histograms for each of the three models that we consider for the median TVD in each class. Those results demonstrate that the Fourier head learns a more accurate PMF. And in Figure A23 we demonstrate that the Fourier head model consistently obtains a lower TVD and a greater diversity of samples. We hypothesize that the LoRA-finetuned baseline model has fewer diverse samples because it memorizes training data instead of learning the actual distribution. In contrast, the Fourier head is forced to learn a continuous distribution, and samples directly from that distribution.

Related works which consider LLMs as random number generators: Gu et al. [41] explores probability distribution sampling in LLMs in the context of behavioral simulation, which demonstrates an application of using LLMs as random number generators. And Paruchuri et al. [91] explores the tasks of using LLMs for estimating percentiles, drawing samples, and calculating probabilities. Lastly, Hopkins et al. [51] explores sampling from numerical probability distributions using LLMs, but they only consider a single non-uniform density, namely the normal distribution $N(0.5, 0.2887)$. They find that LLMs don't perform well at this task, but they don't thoroughly investigate model-based interventions. In contrast, we thoroughly investigate the impact of fine-tuning, and replacing the token classification head one with a better inductive bias for modeling

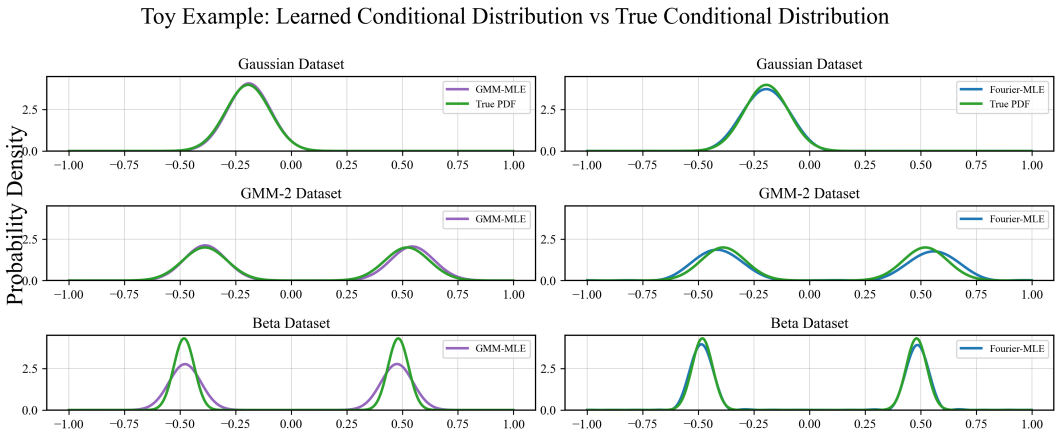


Figure A22: Comparison between the PDFs learned by the GMM-MLE head and the Fourier-MLE head for each of the datasets in the toy example. While GMM-MLE outperforms Fourier-MLE on the Gaussian and GMM-2 datasets, Fourier-MLE performs better on the Beta dataset.

complex probability distributions.

A.11 FH: Additional Experiment Details, Large-Scale Examples

A.11.1 Decision Transformer

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 A24 that the PMFs learned by the Fourier head are smoother. In Figure A27 we present results for more Atari games. In Figure A25, 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 A26, 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.

A.11.2 Chronos

In Figure A28 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 A5 we present results from an ablation study on the quantity of Fourier frequencies, 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 (the T5 model with approximately 20M parameters) and this takes about 48 hours on 8 GPUs. See Table A6 for the datasets we used to train and evaluate Chronos.

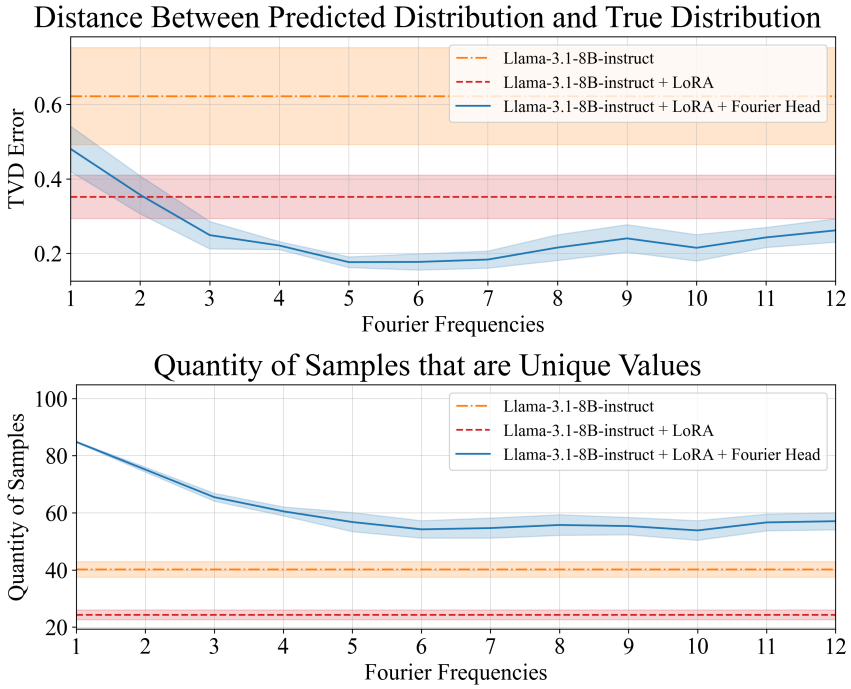


Figure A23: We demonstrate that the Fourier head model consistently obtains a lower total variation distance, as well as a greater diversity of samples. For TVD (top), lower is better, because lower values indicate that the learned distribution is closer to the ground truth distribution. And for quantity of samples (bottom), higher is better, because lower values indicate that the LM has just memorizes specific numbers instead of performing sampling. We present here the mean values across the four distributions, for all ten seeds. We can see that the Fourier head obtains more diverse samples, and learns a distribution closer to the ground truth.

A.12 FP: Implementation details

A.12.1 Training hyperparameters

We train for 5000 steps with an effective batch size of 8 (using gradient accumulation over two steps) and saved model checkpoints every five hundred steps. We used `bf16` mixed precision with `tf32` support and initialized from the `THUDM/CogVideoX-5b-I2V` pretrained weights. ControlNet [139] was initialized from the first six transformer layers, for all three input channels, using a downscaling factor of eight, and a unit weight coefficient. We use the AdamW optimizer [79] (initial learning rate 1×10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, maximum gradient norm 1.0) under a cosine-with-restarts learning-rate schedule (one cycle, 250 warm-up steps). We set our random seed to 42. Training videos contained up to 49 frames at 8 fps.

There are a total of 42 transformer blocks that we may use to initialize our ControlNet. We chose the first six due to memory constraints. With more compute resources, one could potentially achieve higher quality control. We expose the number of transformer blocks to use in the ControlNet as a command line argument for convenience in our training code release.

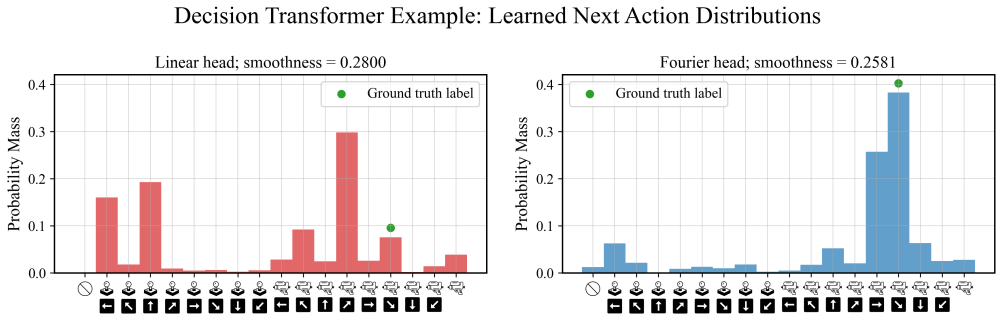


Figure A24: We present example next action distributions for a single step in the Decision Transformer test split. The Fourier agent with 8 frequencies produces a "clump" of actions that is semantically meaningful. Namely, this agent almost certainly wants to shoot in the down right or right direction, presumably because there is a submarine in that direction. In contrast, the linear agent's next-action distribution doesn't clearly depict a strategy, and incorrectly assigns higher likelihoods to incorrect actions. Because the Fourier head outputs a smoother PMF, it learns to concentrate more probability mass near the correct action.

A.12.2 Additional details: mass understanding quantitative study

Each ball is subjected to forces of magnitude $0.125 \cdot n$, where $n \in \{1, \ldots, 8\}$, with 10 videos generated per force value using different random seeds. To ensure experimental control, we utilize the GPT-Image-1 API to generate first frames and their variations, maintaining consistent initial ball positions and shapes across conditions. We automatically detect ball position using a Faster R-CNN model with a ResNet-50-FPN backbone [98], tracking the sports ball class (ID 37). We compute the distance traveled as the Euclidean distance in pixel space between detected bounding box centers in the first and last frames.

A.12.3 Additional details: encoding strategy, point force

For the first frame, we set the pixel values equal to 0 everywhere except for a Gaussian blob of radius 20 centered at (x, y), which gets a value of 1; and at the final frame of the control signal, the blob will have moved a distance of $(\frac{1}{8} + \frac{3}{8}F) \cdot w$ pixels. This ensures that when the force is minimized at $F = 0$, the total displacement is $w/8$, and when the force is maximized at $F = 1$, the total displacement is $w/2$.

A.12.4 Additional details: comparison with PhysDreamer

We took six flower demo videos from the PhysDreamer teaser videos (Alocacia, Carnation, Orange Rose, Red Rose, White Rose, Tulip) and took a screenshot of a still frame from the video that did not have any force annotation on it. We passed these six frames, as well as the six equivalent force prompts (i.e. the same force vector from the original demo) into the Force Prompting model. We presented these videos side by side in the human study, which we served using Qualtrics. Both videos for a given scene were shown to the human annotator simultaneously, with left/right randomization.

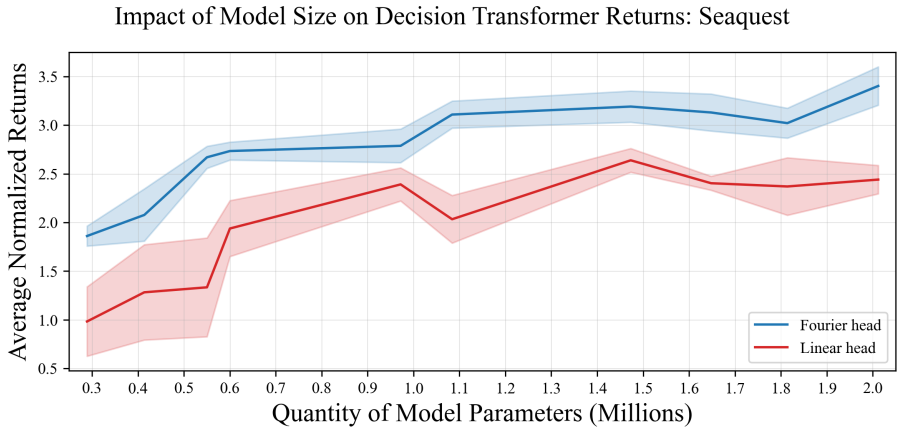


Figure A25: We present an ablation study on the effect of the model size on the relative performance of the Fourier head and the Linear head. The results demonstrate that, across model sizes, the Decision Transformer with a Fourier head is better at learning high-quality next action distributions than the Decision Transformer with a linear head.

A.13 FP: Additional details: zero-shot multiple forces

A.13.1 Multiple forces for multiple objects, benchmark

Our experiments confirm that the model successfully handles multiple simultaneous forces without requiring retraining. This capability emerges zero-shot by simply adding multiple Gaussian blobs to the control signal videos—one for each applied force. We include these videos in the project webpage. The 6 scenes we tested are:

- An image with 2 Christmas tree ornaments
- An image with 2 toy cars
- An image with 2 vases, each with a flower in it
- An image with 2 roses leaning diagonally, growing out of the ground
- An image with 2 dandelions growing in a field
- An image with 2 apples on separate branches of a tree

We generated these images using the GPT-Image-1 API. To test each image, we identified two directions where it would be reasonable to poke each object; for example, the dandelions can each be poked to the left and to the right. Then for each image, we used 4 different multiple-force force prompts, representing the 2×2 different ways of combining the two different directions; for example, we can poke both dandelions to the left, poke both to the right, poke one to the right and the other to the left, then poke one to the left and the other to the right. We computed one video for each, using the same seed for all of them, with no cherry-picking. We

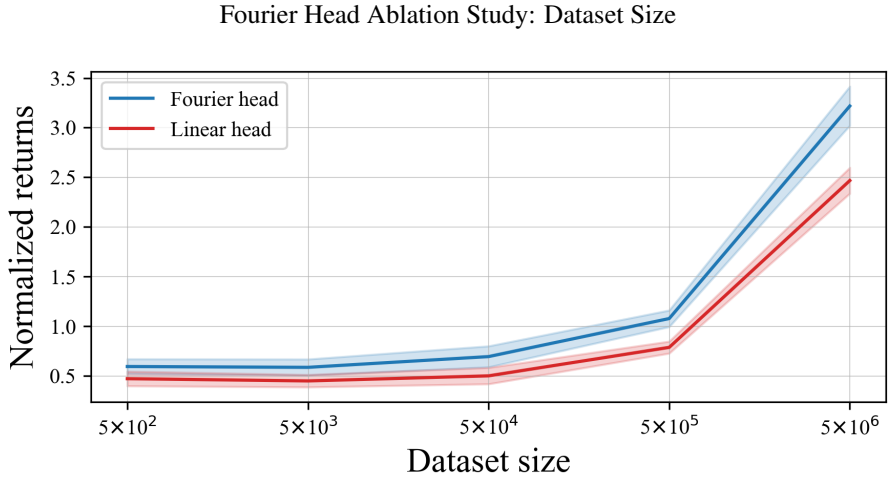


Figure A26: In this ablation study, we analyze whether dataset size has any effect on the relative performance of the Linear head and the Fourier head. Our results show that, across dataset sizes, the Decision Transformer agent with a Fourier head achieves larger returns than the linear head on the Seaquest game.

found that $5/6$ of the videos in this multi-poke benchmark had perfect force adherence, and one of them had nearly perfect force adherence. The only failure case was in the two apples scene; when the left apple is poked to the right and the right apple is poked to the left, they move as if they were both poked to the right.

A.13.2 Multiple forces for multiple objects, mass understanding

We used the GPT-Image-1 API to construct images where two objects of different apparent masses are present. The first image contains an empty laundry basket on the left side of the frame, and a full laundry basket on the right side of the frame. The second image contains a book on the left side of the frame, and a stack of books on the right side of the frame. We passed both sets of images into the multi-poke Force Prompting inference pipeline and instructed the model to poke both sets of objects towards the middle of the frame with the same force magnitude. Across 8 different force magnitudes $\{0.125 * i, i \in \{1, \ldots, 8\}\}$ we found that the lighter object moved much further, with the heavier item barely moving at all. We include videos on our project page.

A.14 FP: Additional qualitative results

A.14.1 Failures and Limitations

Figure A33 illustrates and categorizes failure cases of Force Prompting. We observe model correlation issues—for example, in hair-blowing scenarios, faces sometimes reorient based on wind direction, likely reflecting patterns in training data where hair typically blows backward. Our method is fundamentally constrained by the underlying video prior's physical understanding; we focus on controlling existing physical capabilities rather than improving the model's physics comprehension. We defer to other works that specifically

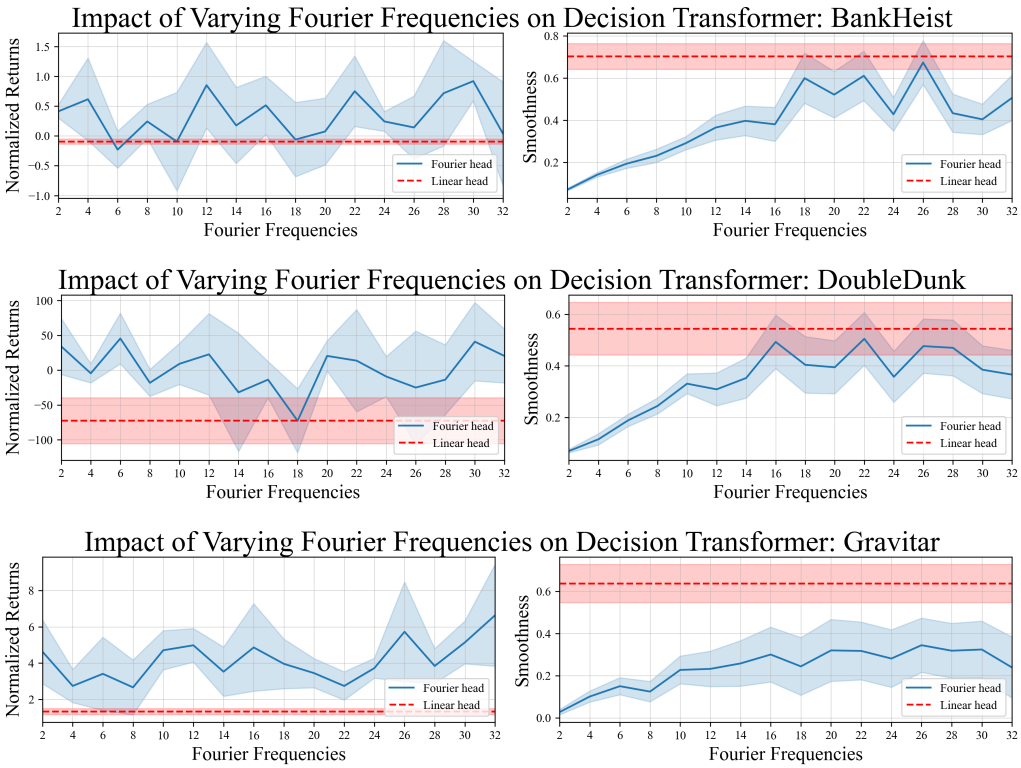


Figure A27: We present empirical results for how the quantity of Fourier frequencies impacts returns and smoothness for additional imitation learning games. For normalized returns, higher is better; for smoothness, lower is better. We can see that for the BankHeist, DoubleDunk, and Gravitar games, the Fourier agent consistently achieves higher normalized returns than the linear baseline agent, while still learning smoother next-action distributions.

aim to enhance physical accuracy in generative models, while noting that our approach benefits from efficiently leveraging the scaling properties of the base model.

A.14.2 Extended comparison with physics simulation models

To demonstrate the point force model's versatility, we curate a benchmark using first-frame images from prominent physics-in-the-loop papers: PhysDreamer [140], DreamPhysics [53], MotionCraft [3], PhysGaussian [131], PhysGen [76], PhysGen3D [19], Physics3D [75], and PhysMotion [113]. We apply our force prompting approach to these diverse scenarios, including poking plants (alocasia, ficus, bouquet of flowers), moving vehicles (boat in water, toy cars), and household objects (rocking horse). Our video results (see project webpage) illustrate that our purely neural method can handle the same visual scenarios almost as effectively as approaches requiring explicit physics simulation at inference time. These qualitative results are in line with our findings in Table 3.2.

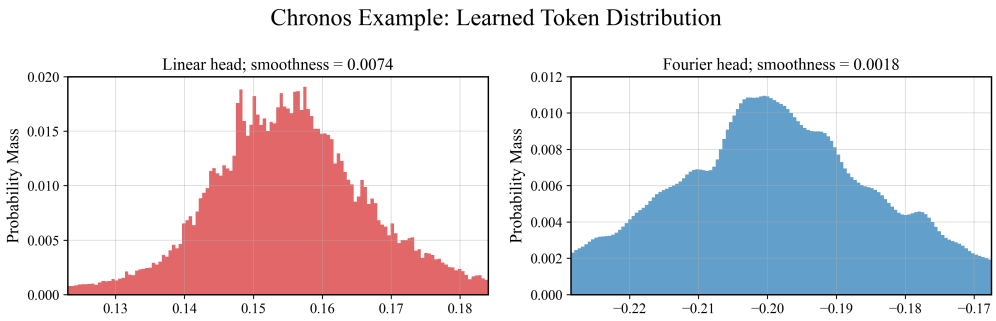


Figure A28: We present the next token value distribution for a single forecasted timestep on the Tourism Monthly dataset. We observe that the Fourier head's learned conditional distribution is smoother, fitting signal more robustly, whereas the linear head overfits to the noise, and is therefore more 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.

A.14.3 Training a unified model

We also train a unified model to learn both point force prompts and wind force prompts. We run inference on this joint model using our point force benchmark, as well as our wind force benchmark. Our findings:

- *More dynamic backgrounds:* in many of the videos, the background moves more dynamically in a natural way. For example, for the apple tree, the surrounding leaves move more naturally after the apple is poked; in the video where a kid is sitting in the middle of a toy train track, the kid is moving more naturally while the train is moving around the track; and in the video with falling leaves in the forest with a woman sitting on a chair in the background, the woman in the chair moves more while the leaves are being blown.
- *Slightly less robust point force control:* on some of the point force videos (e.g. the blueberry bush), the control signal is not respected.

We designed this experiment by sourcing 50% of the training data and control signals in each batch from the synthetic point force dataset, and the other 50% from the synthetic wind force dataset. We trained using the same architecture and number of training steps as the original model.

A.14.4 Scaling the dataset size

We trained from scratch a wind force model which uses half as many synthetic flag waving videos. The only variable that we changed was the dataset size; everything else (including the number of training steps and learning rate scheduler) remained the same. For this model, we found some additional failure cases. One failure case: for the image where a woman holds a sheet on the beach, the model hallucinates a bedsheet in the background, indicating that the model has likely memorized the "waving flag" pattern from the training dataset and is injecting it into the output video. A second failure case: the confetti's response to the wind

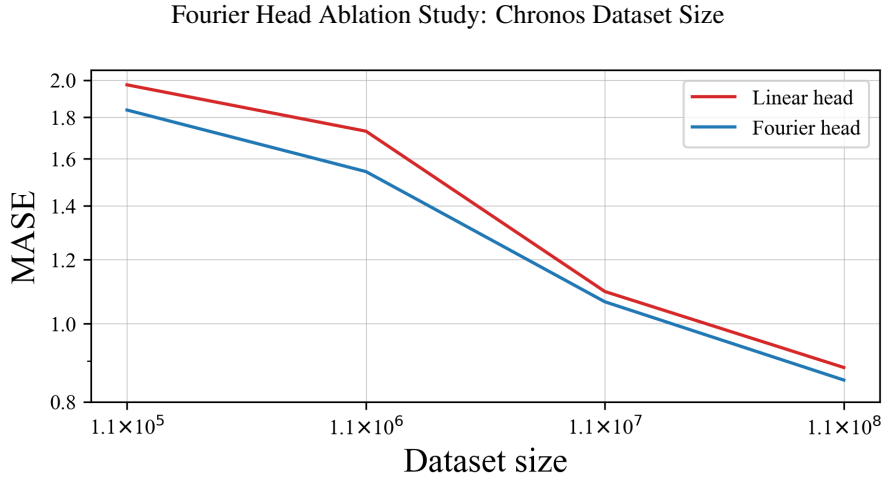


Figure A29: In this ablation study, we analyze whether dataset size has any effect on the relative performance of the linear head and the Fourier head for the probabilistic time series task. Our results show that, across dataset sizes, the Fourier head yields more accurate forecasts than the linear head. For the dataset sizes $1.1 \times 10^5, 1.1 \times 10^6$, and 1.1×10^7, we report the average MASE across four seeds; for the dataset size 1.1×10^8 we report the MASE from Table 2.3. We generate the plot following [61] and observe a similar power-law scaling behavior for both methods, with the Fourier head consistently outperforming the linear head.

isn't as convincing. For example, some of the confetti will blow in the wind's direction but some will stay stationary. This indicates that the model hasn't generalized properly, perhaps because of less training data.

A.15 FP: Additional emergent phenomena

A.15.1 Case study #1: Does the model enforce physical affordances?

The Force Prompting models demonstrate a surprising capability to respect object-specific movement constraints. For example, when a train on a circular track is poked forward, it follows the curved trajectory of the track rather than continuing in a straight line—a behavior similarly observed with windmills respecting their rotational axis. We also note interesting emergent behaviors with multi-part objects: poking the lead car of a toy train forward sometimes pulls the entire train along, while other times only the first car moves; conversely, backward forces consistently push the entire train as a unit.

A.15.2 Case study #2: Does the model understand atomicity of objects?

We evaluate the local force model's sensitivity to the specific pixel chosen as the application point for localized forces. Results demonstrate consistent object movement regardless of which part of the object receives the force. For example, whether poking a train's engine, middle car, or caboose, the entire object responds appropriately to the applied force. This suggests the model has developed a holistic understanding of object

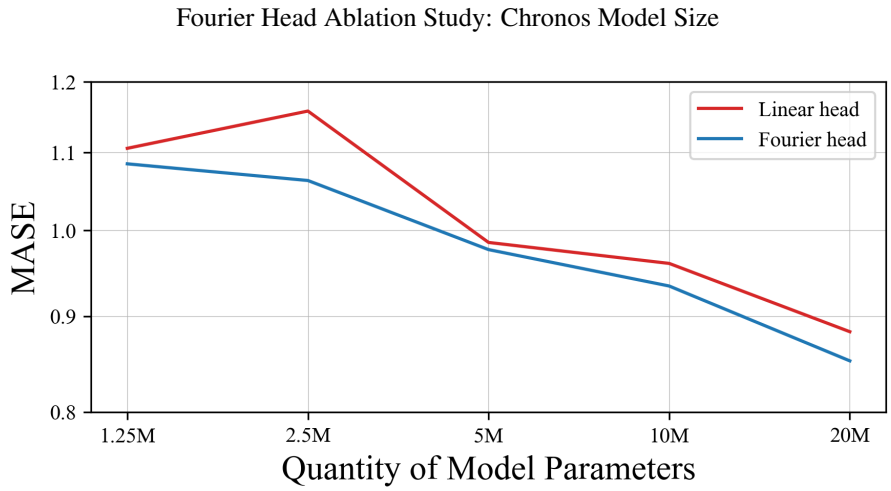


Figure A30: In this ablation study, we analyze whether model size has any effect on the relative performance of the linear head and the Fourier head for the probabilistic time series forecasting task. Our results show that, across model sizes, the Fourier head yields more accurate forecasts than the linear head. For the model sizes 1.25M, 2.5M, 5M, and 10M, we report the average MASE across three seeds; for the model size 20M we report the MASE from Table 2.3. We generate the plot following [61] and observe a similar power-law scaling behavior for both methods, with the Fourier head consistently outperforming the linear head.

wholeness rather than simply responding to pixel-level manipulations.

A.15.3 Case study #3: Does the model preserve cinematic effects of the original image?

The model demonstrates an ability to maintain the original image's stylistic and cinematic properties throughout generated sequences. For example, when animating a toy car from an image with a depth-of-field effect, the model preserves the background blur as the car moves, ensuring visual continuity with the source image's aesthetic. This suggests the model not only understands physical motion but also respects the artistic intent and visual language of the input image. See the project page for videos.

A.16 FP: Survey Details and Instructions

We sourced participants from Prolific, compensating responders $12/hr. Our surveys specify the number of questions and an expected time limit. For example, we present the following to participants at the start of the survey:

> *Thank you for taking this survey!* ***It should take less than 25 minutes.***
>
> *There are 208 questions total. You should aim to spend around 7-8 seconds per question. You will be shown two videos, and you must choose which video more accurately* ***shows the effect of the wind blowing in the direction indicated by the question. Please read each question carefully.***

Chronos Time Series Model	MASE (↓)	WQL (↓)	Smoothness (↓)
Linear	0.883	0.750	0.1689 ± 0.1087
Fourier-64	0.875	0.798	$\mathbf{0.0032 \pm 0.0012}$
Fourier-128	0.872	0.767	0.0068 ± 0.0035
Fourier-256	0.859	0.755	0.0139 ± 0.0087
Fourier-550	**0.852**	**0.749**	0.0283 ± 0.0224
Fourier-550 (no regularization)	0.861	0.753	0.0286 ± 0.0219
Fourier-550 (uniform precision binning)	0.873	0.747	0.0395 ± 0.0252

Table A5: We present large-scale experiments on Chronos time series forecasting. Notably, every Fourier model outperforms the linear baseline on MASE and smoothness metrics. We can see that 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). In the bottom two rows, we present an ablation for our large-scale experiments 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. Note that the choice of binning strategy doesn't affect the performance of the linear baseline.

> ***There are hidden vigilance questions, so please make sure you answer to the best of your ability.*** *We will be rejecting extremely poor quality responses.*
>
> *At the end of the survey, there is a place to put your Prolific ID so we can confirm you've taken the survey.* ***Please respond only once to this survey*** *(you may have done a similar survey in the past day, that is fine) and thank you for your time!*
>
> *Please do not spend more than 25 minutes on this survey! We don't want to waste your time :)*

We then present participants with example questions and what we consider an appropriate response along with our reasoning, a screen shot of an example can be found in Figure A34. We then present participants with questions following the example for them to answer. They may select their preference from two videos by selecting the radio button underneath their selection, which is depicted in Figure A35.

A.17 GF: Additional Experiment Details

A.17.1 Comparison to Prior Works: Direct Force Prompting Quantitative Comparison

We encode the the goal force prompt in the second channel of the control signal, and we encode the direct force prompt (which is a similar task to PhysDreamer [140], Force Prompting [37], and PhysGen [76]) in the first channel of the control signal. In Table A7 we compare the "Direct Force Prompting" capability of our model to those three models via a 2AFC human study ($N = 10$) conducted on Prolific. We gathered two benchmarks: a PhysGen benchmark, consisting of four scenes highlighted on that work's project page; as well as a PhysDreamer benchmark, consisting of three scenes highlighted on that work's project page. We compare our model to PhysGen and Force Prompting on the PhysGen benchmark, and we compare our model

to PhysDreamer and Force Prompting on the PhysDreamer benchmark. Note that PhysGen models rigid body mechanics, whereas PhysDreamer models oscillations.

A.17.2 Synthetic Data Generation

In this section, we provide an in-depth discussion of the methods and specific parameters utilized for generating our synthetic training data. This data is used to train the Goal Force model to act as an implicit neural physics planner.

For all synthetic datasets, a key step in creating the multi-channel control signal $\tilde{\pi}$ is the projection of 3D forces and object properties onto the 2D image plane. We use the camera's parameters to map 3D force vectors and object world coordinates into 2D pixel coordinates, enabling us to accurately model physical interactions within the video frames.

Dominos Dataset

We generated 3k videos of domino chain reactions using Blender. The setup models a causal chain where an initial direct force on one domino results in a predictable goal force on a downstream target domino.

To ensure diversity and robustness, we randomized the following parameters per video:

- **Domino Count:** Uniformly sampled from $\mathrm{Unif}\{3, \ldots, 10\}$.
- **Scene Geometry:** Randomized placement and orientation of the domino line.
- **Causality:** Choice of the initial target domino and the direction of the chain reaction (i.e., hitting the domino in front or behind).
- **Visuals:** Randomized camera position, ground textures (from 42 Polyhaven options), and High Dynamic Range Images (HDRIs) for lighting and background (from 50 Polyhaven options).
- **Force Magnitude:** Continuous values from $[0, 1]$, where 0 represents the minimum force required for the domino to topple and 1 represents a maximal, strong impulse.

Each video is accompanied by a JSON file that records the names of the initial and adjacent contact dominos, along with the complete 2D pixel coordinates for all dominos across every frame.

Rolling Balls Dataset

This dataset comprises 6k videos generated in Blender, split into two primary categories to capture both collision and non-collision causal interactions:

1. Collision Set (4.5k videos): A "projectile" ball, acted upon by an unseen point force, is aimed to collide with one specific "target" ball within a group of initially stationary "distractor" balls.
2. Non-Collision Set (1.5k videos): The projectile ball is aimed such that it misses the target ball.

For the Collision Set, we ensured a diverse range of physical scenarios by randomizing:

- **Ball Count:** $\mathrm{Unif}\{3, \ldots, 9\}$.
- **Physical Properties:** Ball colors, ball masses $\mathrm{Unif}(1.0, 4.0)$ kg, and all ball positions.
- **Visuals:** Randomized camera position and ground textures.
- **Force Calculation:** To guarantee collision, a minimum required force is calculated based on the projectile mass, distance to the target, and a randomized collision time ($\mathrm{Unif}(2.5, 4.5)$ seconds). This minimum force is scaled by $Unif(1.2, 1.6)$ to introduce physical variation.

The collision videos are evenly split between straight-on and indirect collisions. For both types, the script first calculates the precise angular window necessary for the projectile to hit the target.

- For straight-on collisions, the force is aimed directly at the center of this calculated angular window.
- For indirect collisions, the force angle is randomly sampled within this window, resulting in an off-center strike.

This mixed-collision approach helps the model learn diverse post-collision behaviors.

For the Non-Collision Set, we randomized: ball quantity ($Unif\{3, \ldots, 5\}$), ball textures, positions, camera angle, ground textures, target ball selection, force angle ($[0, 360°)$), and force magnitude ($[0, 1]$).

For all ball videos, a JSON file records initial 2D/3D coordinates and physics parameters. For the videos featuring indirect collisions, we also save the complete 2D pixel trajectory of the target ball. For the non-collision videos, we save the final 2D trajectory angle of the projectile ball.

Plants Dataset

This dataset, generated using PhysDreamer [140] (which integrates 3D Gaussians and a physics simulator), focuses on non-rigid body dynamics. The videos show a plant (carnation) swaying after being subjected to a direct force. We randomized the following parameters:

- **Initial Conditions:** Camera position and initial object configuration.
- **Force Application:** Contact points, force angles, and force magnitudes in $[0, 1]$, where 0 is a gentle poke and 1 is a strong impulse.

A.17.3 Ablation studies

How important is the mass channel?

We find that masking the mass channel during training and relying on text instead for mass leads to worse performance than reported in Figure 4.6.

How important is the direct force channel?

We find that masking the direct force channel during training causes the model to fail on some of the complex, out-of-domain causal chains, such as human-object interactions.

How specific does the text prompt need to be?

We conducted ablation studies to confirm that the action itself does not need to be specified in the text prompt (*e.g.*, "the dog paw *causes a ball to move*" works just as well as "the dog paw *nudges the ball*"). When the source of action is unspecified, the model resorts to its pre-trained prior to choose a cause, which may sometimes be an "invisible" force. We highlight that, even when the source of action is specified in the text prompt, the low-level physical dynamics are not (*e.g.*, we never specify to "hit the ball at that angle with that force"). Additionally, the ability to specify the interaction source and type is a desirable feature (*e.g.*, for robotics applications), as it allows users to provide the embodiment and the high-level plan information.

Table A6: All datasets that are used for our time series forecasting experiments. We built our time series forecasting experiments on top of Chronos [6], 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

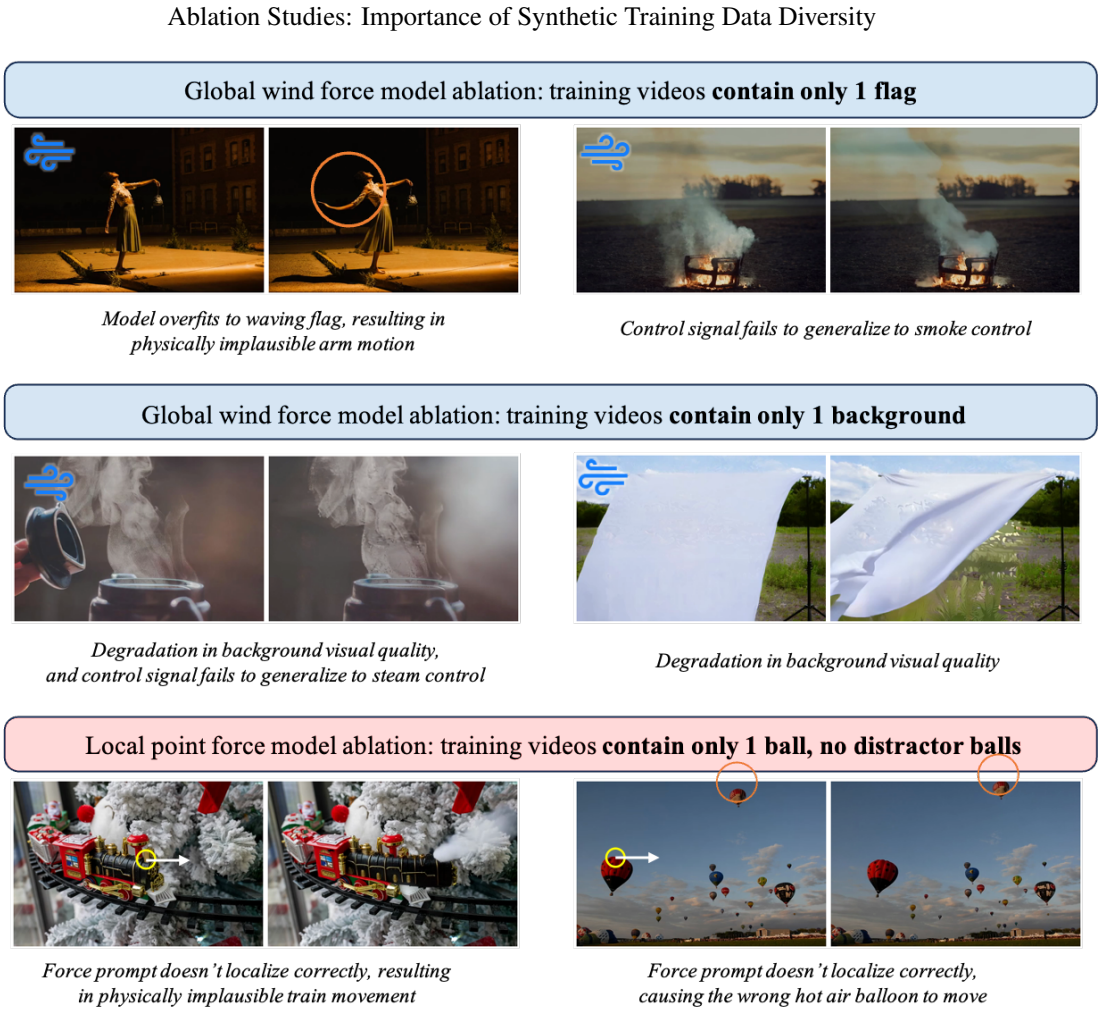


Figure A31: **Results from our ablation studies on synthetic dataset design choices.** *Top:* when the global wind force model is trained on a dataset with only one flag, it overfits, causing the woman's arm to wave unnaturally like fabric and failing to generalize to fluid dynamics scenarios such as smoke. *Middle:* when trained with a single background, the global force model fails to differentiate between foreground and background elements, significantly degrading overall visual quality. *Bottom:* when trained without distractor objects, the point force model cannot properly localize motion, applying forces indiscriminately rather than to the intended target.

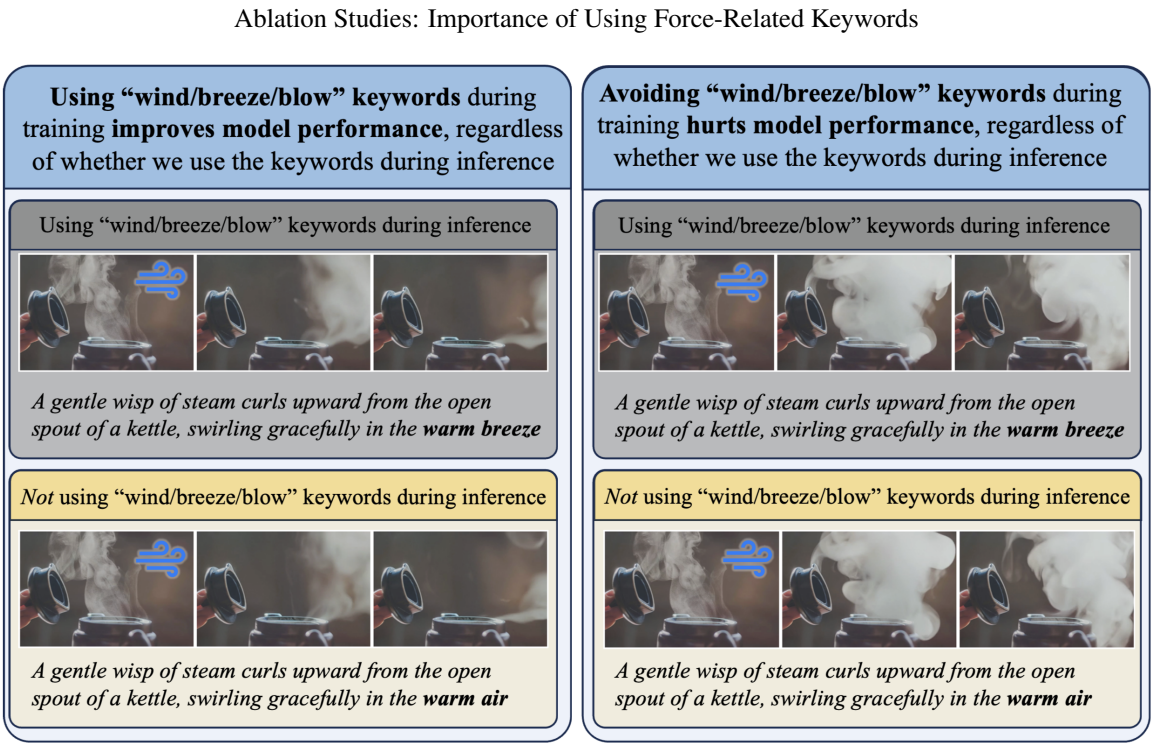


Figure A32: **Results from our ablation studies on text prompt specificity.** In this ablation study, we investigate how material descriptions in text prompts affect model generalization through a 2×2 grid search ablation study. We train and test our wind model with and without wind-related keywords (wind/breeze/blow). Our results demonstrate that omitting these keywords during training significantly increases failure cases in our benchmark dataset. For example, steam is conjured out of thin air instead of being blown correctly. In contrast, models trained with wind-specific terminology demonstrated superior generalization to diverse wind scenarios.

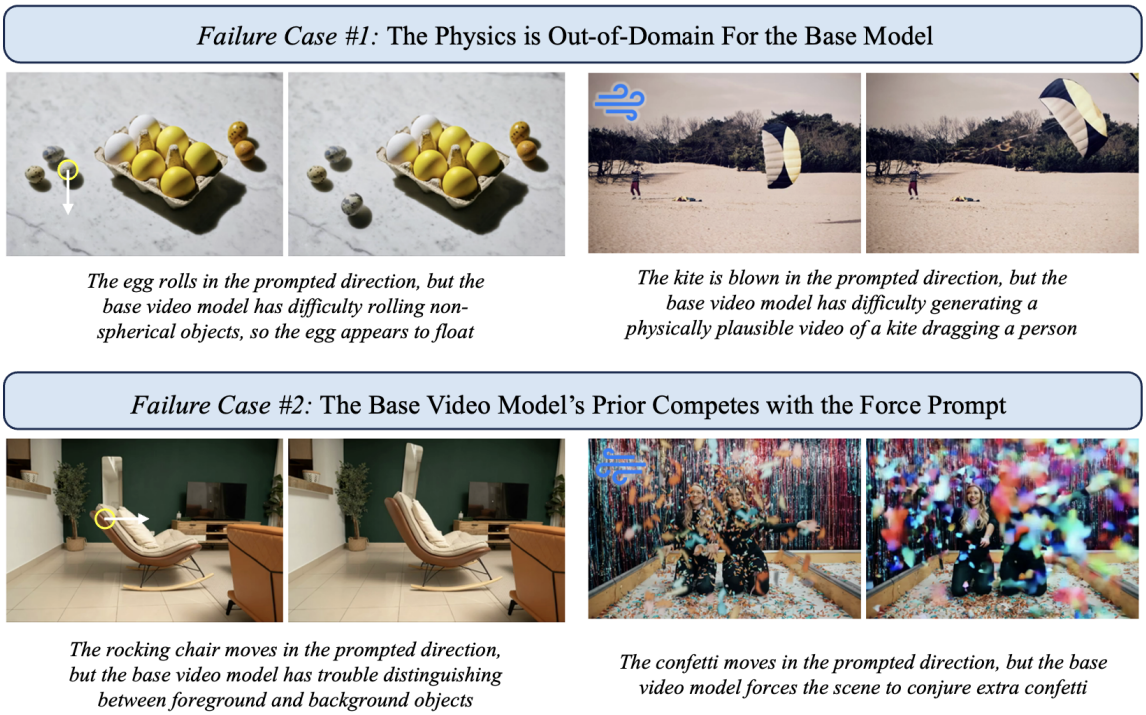


Figure A33: **Analysis of failure cases.** We illustrate and categorize failure cases of Force Prompting. The *top row* shows scenarios where the generated physical motion is out-of-domain for the base CogVideoX model, leading to partial adherence to the force prompt. The *bottom row* depicts failures in visual fidelity or physical realism when the video prior conflicts with the force prompt's intent. More examples are available on our project webpage.

Visual Quality	Dominos	Pool balls	Stone tower	Wall toy	Orange Rose	White Rose	Tulip
Force Prompting	90.0%	80.0%	60.0%	50.0%	100.0%	80.0%	60.0%
PhysGen	60.0%	100.0%	100.0%	80.0%	–	–	–
PhysDreamer	–	–	–	–	50.0%	50.0%	50.0%

Force Adherence	Dominos	Pool balls	Stone tower	Wall toy	Orange Rose	White Rose	Tulip
Force Prompting	90.0%	90.0%	80.0%	90.0%	50.0%	60.0%	50.0%
PhysGen	90.0%	80.0%	80.0%	40.0%	–	–	–
PhysDreamer	–	–	–	–	40.0%	30.0%	60.0%

Table A7: **Human study comparing the Direct Force capability of the Goal Force method to prior works.** Numbers indicate the percentage of human pairwise preferences for Goal Force Prompting's direct force capability (i.e. encoding the force in the first channel) over each baseline on each benchmark dataset. The results demonstrate that Goal Force achieves consistently higher visual quality, as well as superior force adherence against the majority of baselines. Notably, our method achieves these results without relying on physics simulators or 3D assets at inference, unlike PhysDreamer and PhysGen. We note that PhysGen models rigid body mechanics, whereas PhysDreamer models oscillations, so they can't be directly compared to one another.

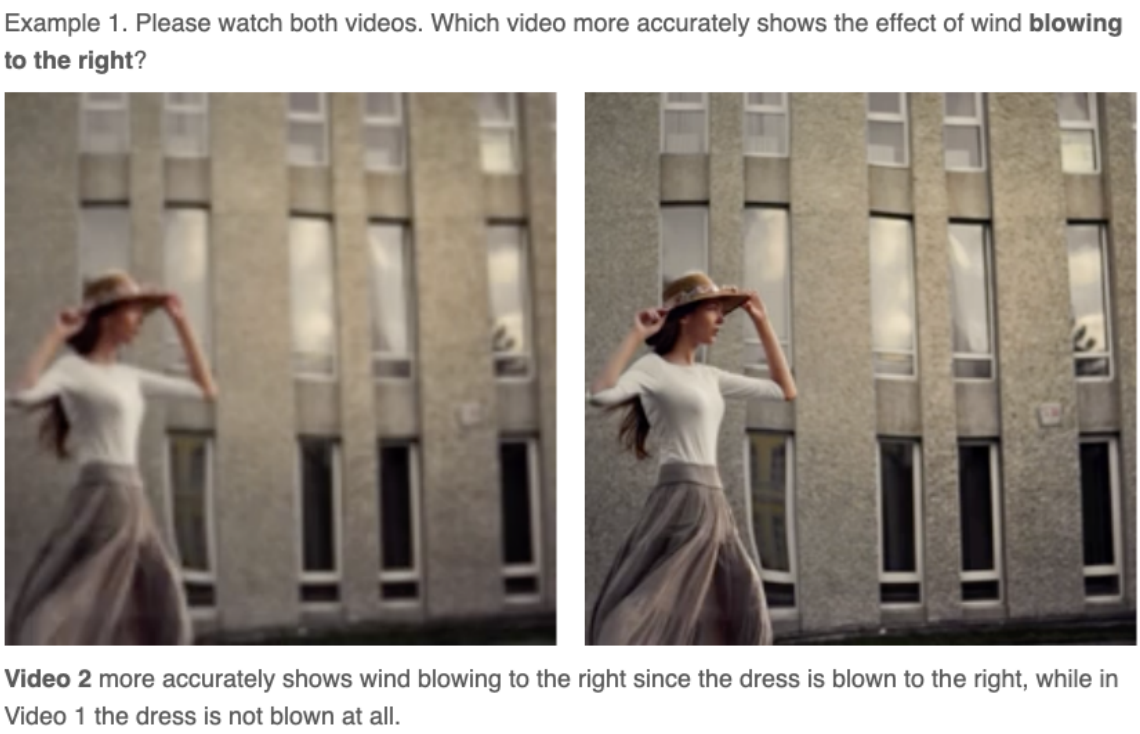


Figure A34: **A demonstration question from one of our surveys.** Participants are shown an example question with a response along with the reasoning for that response.

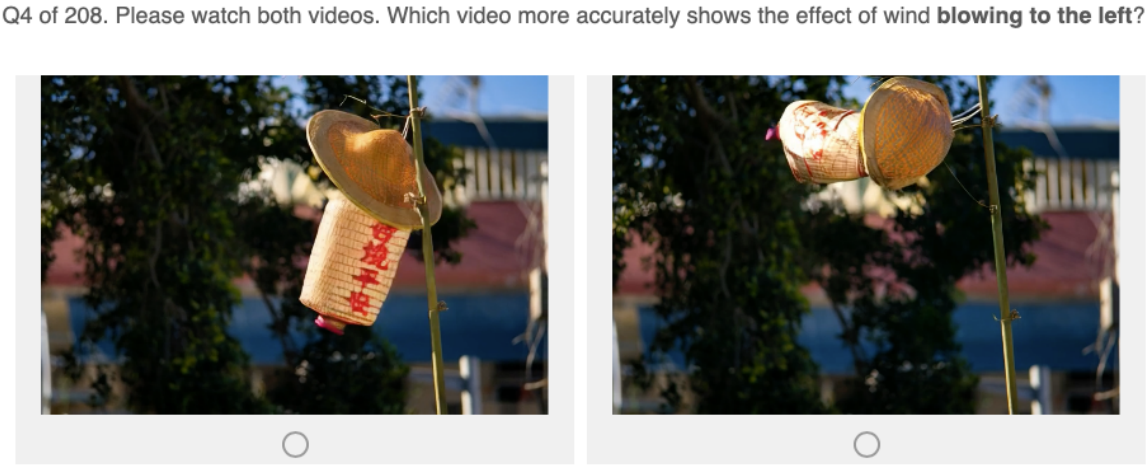


Figure A35: **A question from one of our surveys.** Participants are shown two videos side to side, with radio buttons beneath that they may use to make a selection of which better adheres to the question. The videos play automatically and simultaneously.

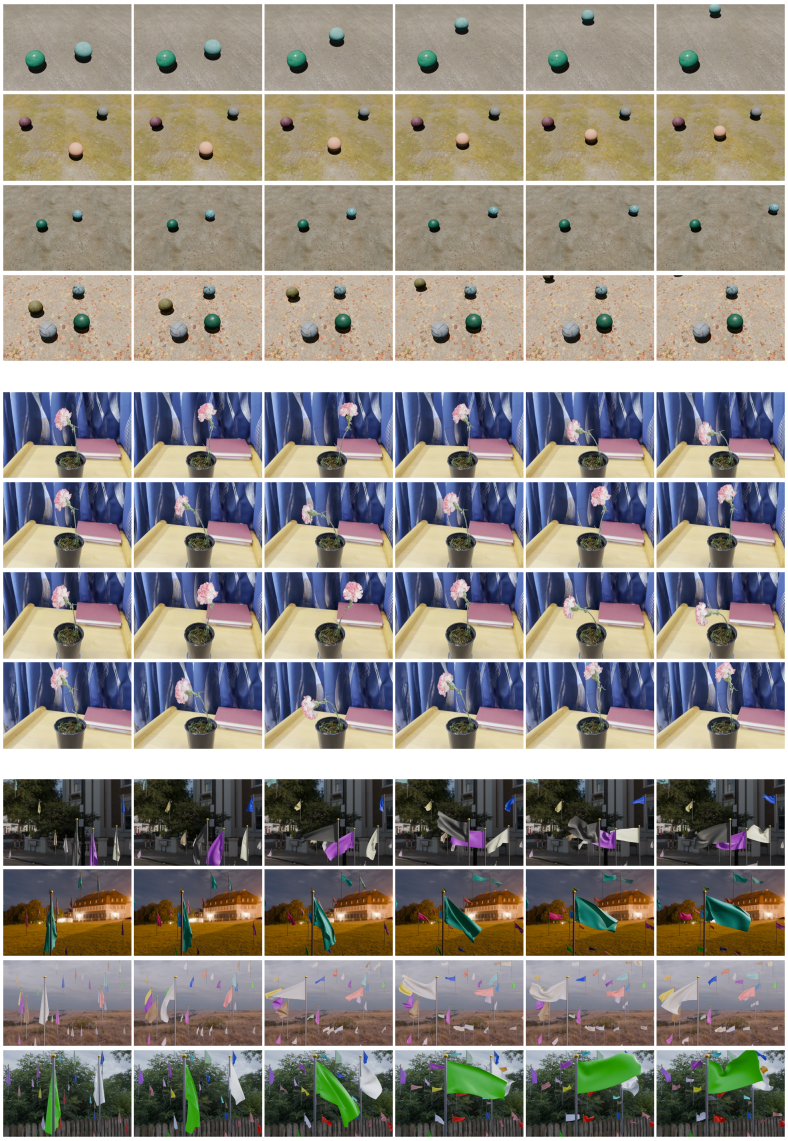


Figure A36: **Samples from our synthetic training datasets.** Top (ball) and middle (flower) are timelapses from our point force training dataset; bottom (flag) are timelapses from the global force training dataset. Our key finding is that video generation models can generalize well when adapted to follow physical force conditioning from videos synthesized by Blender, even with limited demonstrations on few ojects.

Scene	# Valid	# Success	% Accuracy
Dominos	50	50	100.00
Pool Scene 1	49	48	97.96
Pool Scene 2	22	12	54.55
Duckie Scene 1	40	34	85.00
Duckie Scene 2	37	24	64.86
Duckie Scene 3	41	38	92.68
Kitchen Lemon	50	50	100.00
Kitchen Cantaloupes	50	39	78.00
Paper Balls	50	49	98.00
Coffee Cups	44	41	93.18
Accessories	50	47	94.00
Curling Stones	49	37	75.51
Rubik's Cube	49	46	93.88
Curling Stones - Tool Use	47	45	95.74
Accessories - Tool Use	47	40	85.11
Coffee Cups - Tool Use	44	44	100.00
Air Hockey - Tool Use	45	38	84.44
Kitchen Lemon - Tool Use	50	44	88.00
Paper Balls - Tool Use	50	47	94.00
Plant Pots - Tool Use	46	38	82.61
Soaps - Tool Use	48	42	87.50
Rubik's Cube - Tool Use	48	45	93.75

Table A8: **Visual planning results for all test scenes.** We also report the visual planning accuracy for diverse tool use scenarios, where success is defined as correctly using tools to achieve the specified goal force in the presence of distractors. The results show that the model achieves a high success rate in most cases across diverse and complex scenarios.

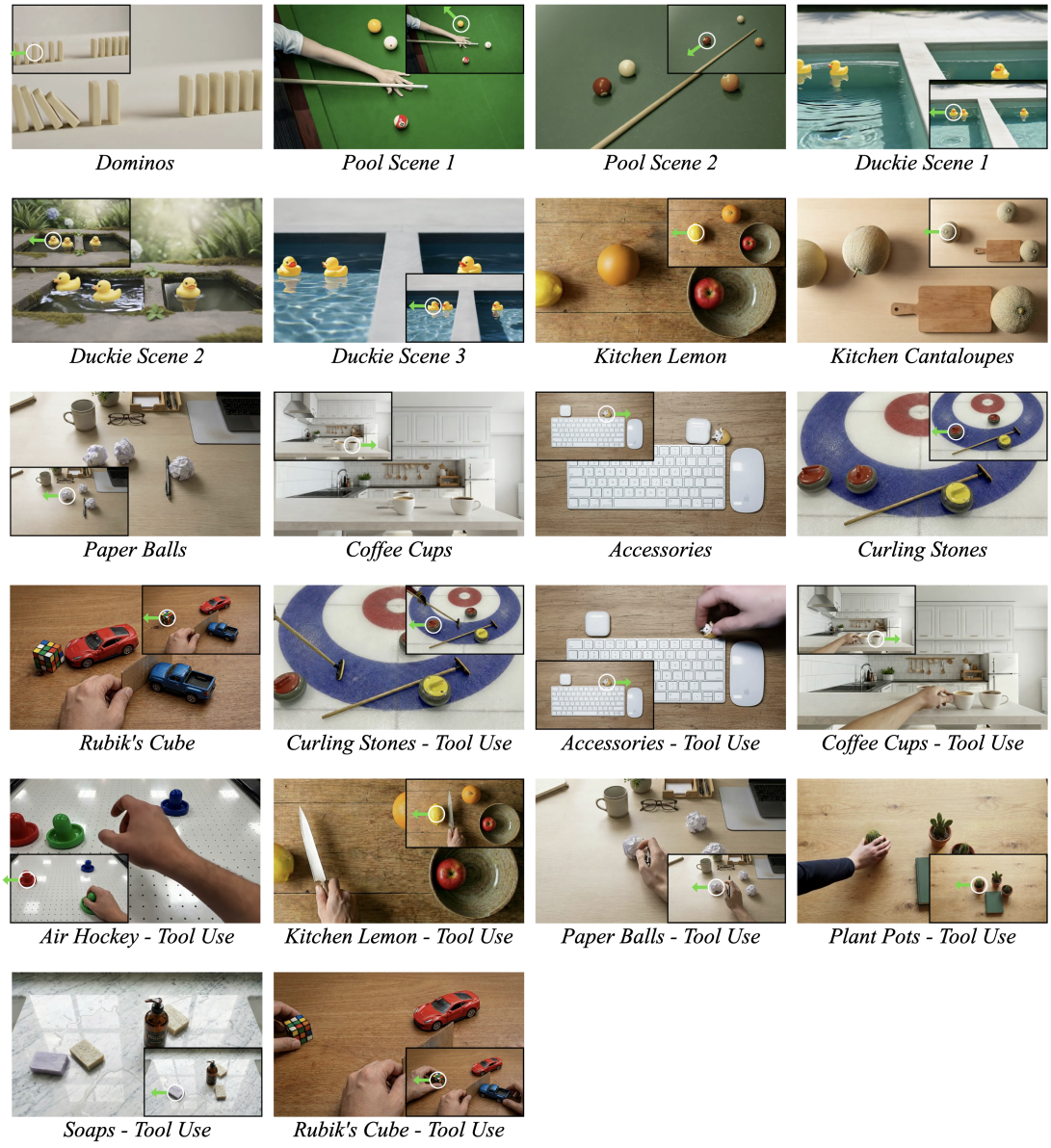


Figure A37: Visualization of all visual planning test scenes. The inset shows the initial state, where the green arrow indicates the goal force, while the larger image shows a valid outcome.