Humanoid Locomotion as Next Token Prediction

Ilija Radosavovic UC Berkeley
Bike Zhang* UC Berkeley
Baifeng Shi* UC Berkeley
Jathushan Rajasegaran* UC Berkeley

Sarthak Kamat UC Berkeley
Trevor Darrell† UC Berkeley
Koushil Sreenath† UC Berkeley
Jitendra Malik UC Berkeley

Abstract

We cast real-world humanoid control as a next token prediction problem, akin to predicting the next word in language. Our model is a causal transformer trained via autoregressive prediction of sensorimotor sequences. To account for the multi-modal nature of the data, we perform prediction in a modality-aligned way, and for each input token predict the next token from the same modality. This general formulation enables us to leverage data with missing modalities, such as videos without actions. We train our model on a dataset of sequences from a prior neural network policy, a model-based controller, motion capture, and YouTube videos of humans. We show that our model enables a real humanoid robot to walk in San Francisco zero-shot. Our model can transfer to the real world even when trained on only 27 hours of walking data, and can generalize to commands not seen during training. These findings suggest a promising path toward learning challenging real-world control tasks by generative modeling of sensorimotor sequences.

1 Introduction

The last decade of artificial intelligence (AI) has shown that large neural networks trained on diverse datasets from the Internet can lead to impressive results across different settings. The core enablers of this wave of AI have been large transformer models [37] trained by generative modeling of massive quantities of language data from the Internet [26, 6, 27, 4]. By predicting the next word, these models acquire rich representations of language that can be transferred to downstream tasks [26], perform multi-task learning [27, 25], and learn in a few-shot manner [4].

Are such modeling techniques exclusive to language? Can we learn powerful models of sensory and motor representations in a similar fashion? Indeed, we have seen that we can learn good representations of high-dimensional visual data by autoregressive modeling [5] and related masked modeling approaches [11]. While there has been positive signal on learning sensorimotor representations in the context of manipulation [28], this area remains largely unexplored.

In this paper, we cast humanoid control as data modeling of large collections of sensorimotor sequences. Like in language, we train a general transformer model to autoregressively predict the sequences. In contrast to language, the nature of data in robotics is different. It is high-dimensional and contains multiple input modalities. Different modalities include sensors, like joint encoders or inertial measurement units, as well as motor commands. These give rise to *sensorimotor sequences* which we view as the *sentences* of the physical world. Adopting this perspective suggests a simple instantiation of the language modeling framework in the robotics context. We tokenize the input sequences and train a causal transformer model to predict future tokens. Importantly, we predict *complete* input sequences, including both sensory and motor tokens. In other words, we are modeling the *joint* data distribution as opposed to the conditional action distribution.

*,† Authors with stars and daggers listed in arbitrary order. Author contributions listed at the end of the paper.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

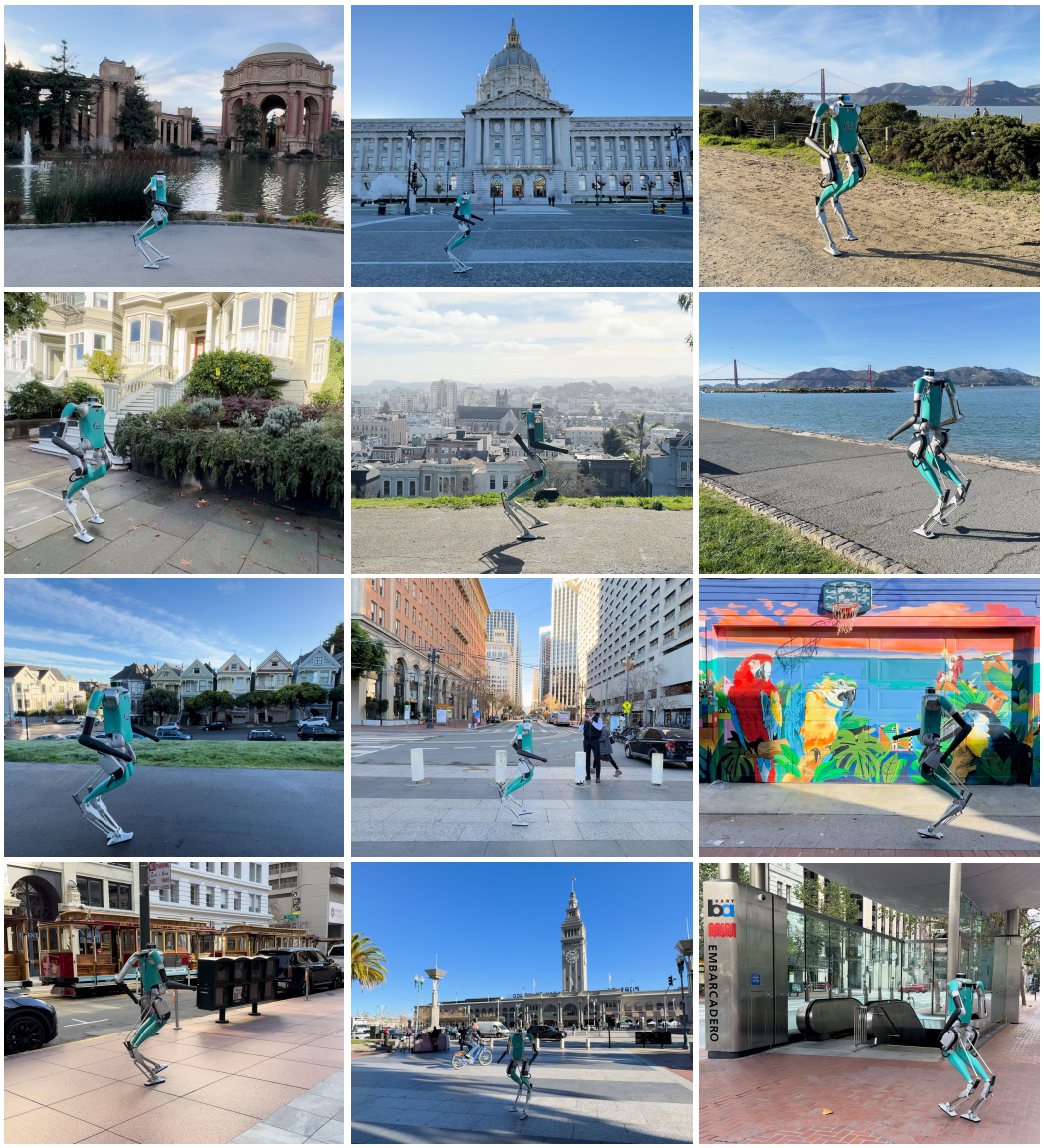

Figure 1: **A humanoid that walks in San Francisco.** We deploy our model to the robot and test it in various locations in San Francisco over the course of one week. Please see our project page for videos. We show that our model enables the humanoid walk over different surfaces including walkways, concrete, asphalt, tiled plazas, and sanded roads. Our model can follow omnidirectional velocity commands well and enables deployment in a challenging city environment like San Francisco.

Our core observation is that if a sequence is incomplete, *i.e.*, some of the sensory or motor information is missing, we can still learn from it by predicting whatever information is present and replacing the missing tokens with learnable mask tokens. The intuition is that if the model has learned to make good predictions, even in the absence of information, it will have acquired a better model of the physical world. An important source of such data are human videos from the Internet. Namely, we can observe human movement in videos but we do not get access to the motor commands or complete sensory inputs. We demonstrate that our method can learn from such data sources effectively.

To validate our method, we apply it to the challenging problem of real-world humanoid locomotion using a full-sized humanoid robot. We first construct a dataset of sensorimotor sequences in simulation. These include complete sequences from a neural network policy trained with reinforcement learning [29], as well as incomplete sequences from three different sources: (i) a model-based controller, (ii) motion capture of humans, and (iii) videos of humans. We reconstruct human videos using computer vision techniques and approximately retarget both the motion capture and video sequences via inverse kinematics. We then train an autoregressive transformer to model this dataset. At test time, we execute the predicted motor commands and ignore the sensory predictions.

We demonstrate that our model can be deployed in the real world zero-shot. Specifically, we deploy our model to a range of different locations and surfaces in San Francisco over the course of one week. Please see Figure 1 for example images and project page for videos. To quantitatively evaluate different aspects of our approach, we perform experiments in simulation. We find that our models trained via sequence modeling alone can be comparable to the state-of-the-art reinforcement learning approach [29] in the settings where data is available. We further find that our approach can benefit from incomplete data, has favorable scaling properties, and can generalize to unseen commands. We encourage the readers to see the arXiv version of this work for additional experiments.

These findings suggest a promising path toward learning challenging real-world robotic control tasks by generative modeling of large collections of sensorimotor sequences.

2 Related Work

Generative models. Generative models have been studied extensively, starting from Shannon's work [33] to the modern era of large language models [4]. Various such models emerged over the last decade. Examples include GAN [10] and Diffusion models [35, 14] for generating pixels, LSTM [15] and GPT [26] for generating language tokens. These models have been adopted for other modalities as well [23, 9, 38]. Among these, autoregressive transformer models became the front runner, due to the impressive scaling behaviors [17] and ability to learn from in-context examples [3]. They have been successfully extended to other modalities as well, such as vision [5] and vision-language [32]. We explore autoregressive generative models in the context of real-world humanoid locomotion.

Transformers in robotics. Following the success of transformer models [37] in natural language processing [26, 6, 27, 3] and computer vision [7, 11], over the last few years, there has been an increased interest in using transformer models in robotics. We have seen a number of works showing that transformers can be effective with behavior cloning. For example, [34] learns multi-task transformer policies with language, [2] trains language-conditioned manipulation policies from large-scale data, [8] trains language models with embodied data, and [1] trains policies conditioned on image goals. We have also seen that transformer models can be effective for large-scale reinforcement learning [29]. Related to ours, [28] learns sensorimotor representations with masked prediction. Like this body of work, we share the goal of using transformer models for robotics but focus on autoregressive modeling of sensorimotor sequences in the context of humanoid locomotion.

Humanoid locomotion. Legged locomotion has been a long-standing challenge in robotics. Humanoid locomotion is particularly challenging and has been studied extensively over the the past several decades [18, 13]. Stable locomotion behaviors have been demonstrated through model-based control approaches [30, 16], while optimization-based methods can further enable highly dynamic humanoid motions [19]. Although significant progress has been made, learning-based approaches have the potential to facilitate future progress due to their potential to improve from data and generalize to new environments. Recently, we have seen that a purely learning based approach trained with large-scale reinforcement learning in simulation can enable real-world humanoid locomotion [29]. We use the same architecture, an autoregressive transformer, but propose a different training procedure. Specifically, we train the model with sequence modeling instead of reinforcement learning.

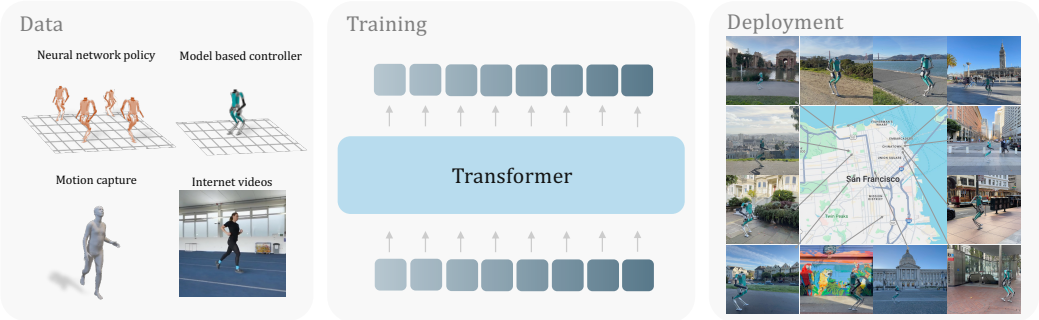

Figure 2: **Humanoid locomotion as next token prediction.** We collect a dataset on trajectories from various sources, such as from neural network policies, model-based controllers, human motion capture, and YouTube videos of humans. Then we use this dataset to train a transformer policy by autoregressive modeling of observations and actions. Our transformer allows a humanoid to walk zero-shot on various terrains around San Francisco. Please see our project page for video results.

3 Approach

In this section, we assume a dataset $\mathcal{D}$ of sensorimotor trajectories $\mathcal{T}$ and describe our approach.

3.1 Objective

Each sensorimotor trajectory is a sequence of observations, such as joint positions, and actions: $\mathcal{T} = (o_1, a_1, o_2, a_2, ..., o_T, a_T)$. We first tokenize the sequence into K tokens to obtain $t = (t_1, t_2, ..., t_K)$. Our goal is to train a neural network to model the density function $p(t)$ autoregressively:

$$p(t) = \prod_{k=1}^{K} p(t_k | t_{k-1}, ..., t_1) \tag{1}$$

We train our model by minimizing the negative log-likelihood over our trajectory dataset, assuming a Gaussian distribution with constant variance:

$$L = \sum_{t \in \mathcal{D}} -\log p(t) \tag{2}$$

Instead of regressing the continuous values of token elements, we could quantize each dimension into bins or perform vector quantization. However, we found the regression approach to work reasonably well in practice and opt for it in our experiments for simplicity.

3.2 Missing modalities

In the discussion so far we have assumed that each trajectory is a sequence of observations and actions. Next, we show how our framework can be generalized to sequences with missing modalities, like trajectories extracted from human videos that do not have actions. Suppose we are given a trajectory of observations without the actions $\mathcal{T} = (o_1, o_2, ..., o_T)$. Our core observation is that we can treat a trajectory without actions like a regular trajectory with actions masked. Namely, we can insert mask tokens `[M]` to obtain $\mathcal{T} = (o_1, \texttt{[M]}, o_2, \texttt{[M]}, ..., o_T, \texttt{[M]})$. This trajectory now has the same format as the regular trajectories and can thus be processed in a unified way. We ignore the loss for the predictions that correspond to the masked part of inputs. Note that this principle is not limited to actions and applies to any other modality as well, such as partially missing sensory observations.

3.3 Aligned prediction

Rather than predicting the next token in a modality-agnostic way, we make predictions in a modality-aligned way. Namely, instead of predicting the next token in the sequence, for each input token we predict the next token of the *same* modality. Please see Figure 3 for diagrams.

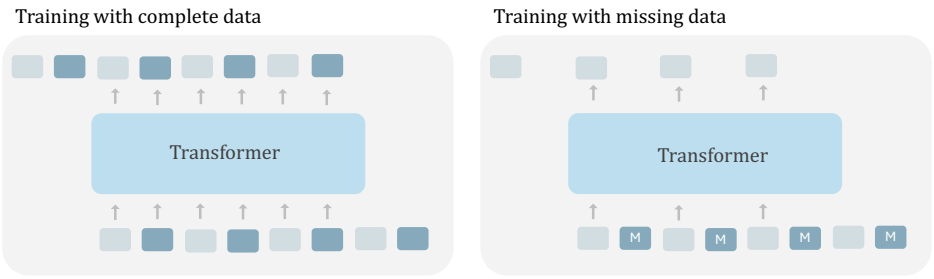

Figure 3: **A general framework for training with different data sources.** Our data modeling allows us to train our transformer with multiple modes of training. In the case of observation-action pairs being available, we train our transformer to predict the next pair of observation-action. When there is no action data available, with MoCap and internet data, we only train our transformer to predict the next observations by masking the actions with a mask token. These two models of training allow our model to utilize both types of data, and this enables us to scale our training in terms of data.

3.4 Joint training

We have two options for training on collections that contain diverse trajectories in terms of noise levels or modality subsets. We can either train jointly with all data at once, including complete and incomplete trajectories. Alternatively, we can first pre-train on noisy and incomplete trajectories. This can be viewed as providing a good initialization for then training on complete trajectories. And then fine-tune the model on complete data with actions. We find that both approaches work comparably in our setting and opt for joint training in the majority of the experiments for simplicity.

3.5 Model architecture

Our model is a transformer [37] with a fairly standard architecture. Given the trajectories from either complete or incomplete data, we first tokenize the trajectories into tokens. We learn separate linear projection layers for each modality but share weights across time. To encode the temporal information we use positional embeddings. Let us assume $o_i \in \mathcal{R}^m$ and $a_i \in \mathcal{R}^n$, then:

$$t_i = \texttt{concat}(o_i, a_i), \tag{3}$$

$$h_i^0 = W t_i, \tag{4}$$

where $W \in \mathcal{R}^{d \times (m+n)}$ is a linear projection layer to project concatenated observation and action modalities to d dimensional embedding vector. The superscript 0 indicates the embedding at 0-th layer, *i.e.*, the input layer. When action is unavailable, we use a mask token $\texttt{[M]} \in \mathcal{R}^n$ to replace a_i, and $\texttt{[M]}$ is initialized as a random vector and learned end-to-end with the whole model. The model takes the sequence of embedding vectors $H_0 = \{h_1^0, h_2^0, ..., h_t^0\}$ as input.

The transformer architecture contains L layers, each consisting of a multi-head self-attention module and an MLP module. Assume the output of the layer l is H_l, then the layer $l+1$ output is computed as follows:

$$\tilde{H}_l = \texttt{LayerNorm}(H_l) \tag{5}$$

$$\tilde{H}_l = \tilde{H}_l + MHSA(\tilde{H}_l) \tag{6}$$

$$H_{l+1} = \tilde{H}_l + MLP(\tilde{H}_l) \tag{7}$$

Here, the multi-head self-attention has causal masking, where the token only attends to itself and the past tokens. Once the tokens are processed through all the layers, we project the embedding to predicted states and actions, by learning a linear projection layer $\widehat{W} \in \mathcal{R}^{(m+n) \times d}$:

$$\widehat{t}_{i+1} = \widehat{W} h_i^L \tag{8}$$

$$\widehat{o}_{i+1} = (\widehat{t}_{i+1})_{0:m} \tag{9}$$

$$\widehat{a}_{i+1} = (\widehat{t}_{i+1})_{m:(m+n)} \tag{10}$$

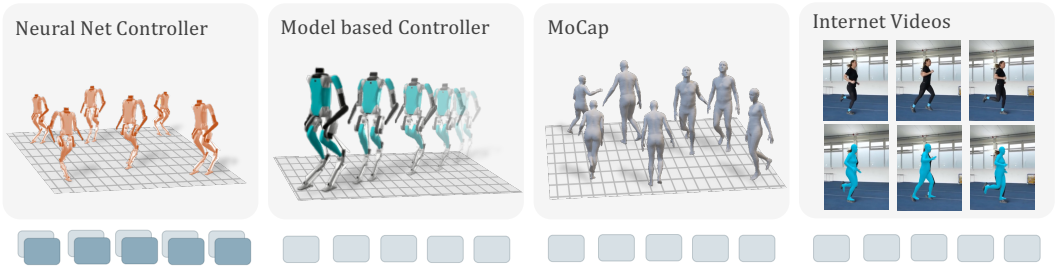

Figure 4: **Training dataset.** To train our model, we construct a dataset of trajectories coming from four different sources. *(i) neural network policy:* provides trajectories with complete observations and actions. *(ii) model-based controller:* produces trajectories from the same robot morphology but without actions. *(iii) motion capture of humans:* does not contain actions and is approximately retargeted onto the robot. *(iv) internet videos of humans:* sequences of human poses are first reconstructed via computer vision techniques and then approximately retargeted onto the robot.

Then we train the transformer with the objective in (2). In the cases where the token is masked, we do not apply any losses. We train our transformer with both types of data, as shown in Figure 3. This allows us to use various sources of data, thus enabling scaling in terms of data.

3.6 Model inference

At inference time, our transformer model always has access to observation-action pairs. In this setting, we apply our transformer model autoregressively for each observation-action pair token. By conditioning on past observations and actions, we predict the next action (or observation-action pair) and execute the predicted action. We then record the ground-truth observation from the robot and discard the predicted observation. We use the ground-truth observation and predicted action as the next set of tokens and append them to the past tokens to predict the next observation-action pair.

4 Dataset

Our approach motivates building a dataset of trajectories for training our model. We construct a dataset of trajectories from four different sources: (i) a neural network policy, (ii) a model-based controller, (iii) human motion capture, and (iv) human videos from YouTube. An illustration of trajectories from different data sources is shown in Figure 4. We describe each data source next.

4.1 Neural network trajectories

As the first source of training trajectories, we use a state-of-the-art neural network policy trained with large-scale reinforcement learning in simulation [29]. Specifically, this policy was trained with billions of samples from thousands of randomized environments in Isaac Gym [22]. We run this policy in the physics simulator developed by Agility Robotics and collect 10k trajectories of 10s each on flat ground, without domain randomization. Each trajectory is conditioned on a velocity command sampled from a clipped normal distribution as follows: linear velocity forward $[0.0, 1.0]$ m/s, linear velocity sideways $[-0.5, 0.5]$ m/s, and turning angular velocity $[-0.5, 0.5]$ rad/s.

Since we have access to the data generation policy, we are able to record complete observations as well as the exact actions that the model predicted. We use this set as our source of complete sensorimotor trajectories that have complete observations as well as ground truth actions.

4.2 Model-based trajectories

As the second source of trajectories, we use the model-based controller developed by Agility Robotics. It is the controller that is deployed on the Digit humanoid robot and available in the simulator provided by Agility Robotics as well. We collect 20k trajectories of walking on a flat ground of 10 seconds each, where we sample the velocity commands as follows: linear velocity forward $[-1.0, 1.0]$ m/s, linear velocity sideways $[-1.0, 1.0]$ m/s, and turning angular velocity $[-1.0, 1.0]$ rad/s.

The model-based controller outputs joint torques, which are not consistent with our joint position action space. Thus, we only record the observations without the actions. This data serves as a source of trajectories with accurate observations from the same robot morphology but without the actions.

4.3 Human motion capture trajectories

As the next source of trajectories, we use the motion capture (MoCap) recordings of humans from the KIT dataset [24] distributed via the AMASS repository [21]. This data was recorded using optical marker-based tracking in a laboratory setting. The dataset consists of ∼4k trajectories. We use a subset of ∼1k standing and walking trajectories. We exclude motions like dancing and jumping.

In addition to not containing the ground truth actions, the MoCap trajectories come with an additional challenge: different morphology. Namely, MoCap trajectories capture *human* keypoint positions in 3D and over time. In order to use these trajectories for training a robot, we solve an inverse kinematics problem to approximate the corresponding robot poses. Please see the arXiv for additional details.

4.4 Trajectories from YouTube videos

Internet videos of people doing various activities are potentially a vast source of training data for humanoid robots. However, the raw pixels have no information about the state and actions of the human. To recover this, we first we run a computer vision tracking algorithm PHALP [31] to extract sequences of 3D human poses. This provides an estimate of the 3D joints of the human body SMPL [20] parameters and a noisy estimate of the human joints in the world coordinates.

We use the human body joint positions to retarget the motion to the humanoid robot using the inverse kinematics optimization, like in the case of motion capture data discussed previously. After we retarget the motions from human videos to humanoid robot trajectories, we filter out the trajectories with high reconstruction error. Note that the scale of this data comes with the cost of being noisy.

5 Experiments

We evaluate our approach on the challenging task of humanoid locomotion. We perform outdoor experiments on real hardware and systematic evaluations in simulation. We encourage the readers to see the extended version of this work on arXiv, which includes additional experiments and ablations.

5.1 Experimental setup

Robot platform. We use the Digit humanoid robot developed by Agility Robotics. It is a full-sized humanoid that is 1.6m tall and weighs 45 kilograms. The robot has 36 degrees of freedom including the floating base, 20 of which are actuated. Due to its high dimensionality and the four-bar linkage structure, it is challenging to simulate accurately which makes it particularly interesting for learning approaches like ours that can learn from data, without requiring an explicit model or a simulator.

Model. Our model has a hidden size of 192 dimensions, with 4 layers of self-attention layers and MLP layers. Each self-attention has 4 heads. We use LayerNorm before each attention layer and ReLU activation after the MLP layer. We use a BatchNorm layer to process the input before the transformer model. When predicting a token at time k, to keep the context length at a reasonable size, we only keep the past 16 steps in input. In Section 5.6, we show the model is able to scale up to more parameters and longer context length and achieve higher performance. We train on 4 NVIDIA A100s.

5.2 Real-world deployment

We begin by reporting the results from the real-world experiments. Specifically, we deploy our model to the robot and evaluate it at various locations in San Francisco over the course of one week. Please see Figure 1 for examples and project page for videos. We find that our model is able to walk over a variety of surfaces including walkways, concrete, asphalt, tiled plazas, and dirt roads. Note that the deployment in a large city environment, like San Francisco, is more challenging than in constrained laboratory environments. The city environment is crowded with pedestrians, less controlled, and less forgiving. This makes the error tolerance low and requires a model that works consistently well.

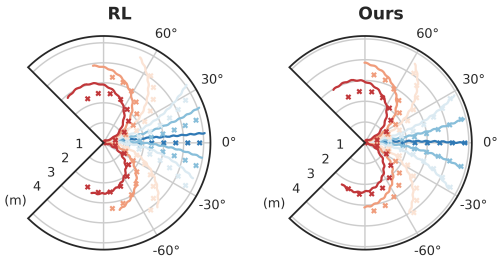

Figure 5: **Comparison to state of the art, trajectory adherence.** The robot is commanded to walk starting from the origin with a fixed heading command of 0.5 m/s and varying yaw commands in $[-0.4, 0.4]$ rad/s. We plot the desired (dotted) and actual (solid) trajectories for our policy and a reinforcement-learning trained policy (RL).

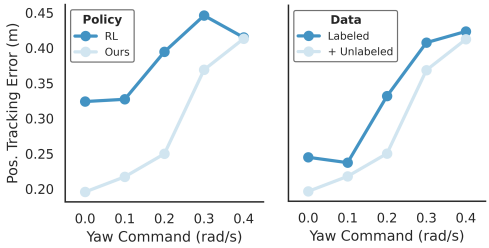

Figure 6: **Tracking error comparisons.** *Left:* We find that our model can follow yaw commands more accurately than the state-of-the-art RL policy [29]. *Right:* We see that our model can benefit from unlabeled trajectories without actions which is a promising signal for scaling our method to large datasets of diverse trajectories.

5.3 Evaluation metrics

Next, we evaluate the models in simulation under two metrics: *tracking error* and *prediction error*. Tracking error measures how accurately the robot follows a specific command. The prediction error is the next token prediction loss measured on a separate set of validation data. We introduce two metrics with details as follows and show that two metrics can consistently predict locomotion performance.

Tracking error. In all experiments, the robot starts from rest in a simulated environment and is issued a constant natural walking command consisting of a desired heading velocity sampled in $[0.35, 0.70]$ m/s, angular velocity sampled in $[-0.4, 0.4]$ rad/s, and zero lateral velocity. We compute $\mathbf{x}^*(t)$, the ideal robot base position trajectory that fully satisfies the velocity command $\mathbf{v}^*(t)$ at all time steps. To measure the accuracy of command tracking, we define the position tracking error as $\frac{1}{T}\sum_{t=0}^{T} \|\mathbf{x}(t) - \mathbf{x}^*(t)\|$. Each trajectory lasts for a duration of 10 seconds. For all evaluation experiments, we use the MuJoCo simulator [36] which is able to simulate the Digit robot accurately.

Prediction error. Since the model is trained with the next token prediction, we evaluate the prediction error on a set of validation data that is held out from training data and contains state-action trajectories collected from the RL policy. This is similar to the language modeling evaluation for large language models [12]. We test both state and action prediction errors and add them together as the error metric.

5.4 Comparison to the state of the art

Trajectory adherence. We compare our model to a neural network controller trained with reinforcement learning (RL) [29]. Figure 5 presents a visual comparison of the trajectory adherence of our controller against these state-of-the-art baselines. Starting with a robot at the origin, we plot the actual trajectory of the robot with eleven different yaw commands selected from $\{0.00, \pm 0.05, \pm 0.10, \pm 0.20, \pm 0.30, \pm 0.40\}$ rad/s. For each model, we jointly plot the desired and actual path traced by the robot base. Our model exhibits better tracking than the RL controller at all turning speeds, and achieves close to perfect tracking for straight-line walking.

Quantitative evaluation. In Figure 6, left, we repeat the above comparison to the RL controller $(N = 245)$, with the full range of heading and yaw velocities mentioned in Section 5.3. We plot the mean position tracking error, binned by the commanded angular yaw. While both models have lower tracking errors at lower yaw, ours consistently outperforms the baseline RL policy. Note that our model was trained on a dataset that includes the trajectories from the same baseline RL policy.

5.5 Training with action-free data

One of the benefits of our approach is that it can be applied to trajectories from diverse sources, including with missing information such as actions, like in the case of human videos. In Figure 6, right, we compare the performance of training only with complete trajectories to joint training with both complete and incomplete trajectories. We observe that including incomplete trajectories

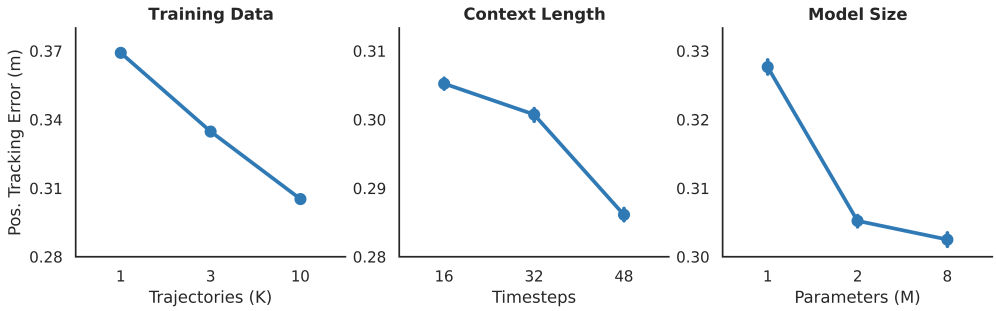

Figure 7: **Scaling studies.** We find that our approach has favorable scaling properties, and that the performance improves with the dataset size (left), context length (middle), and model size (right).

consistently leads to better performance. This suggests that incomplete trajectories can still provide useful signal and is a promising signal for scaling our approach to large datasets of diverse trajectories.

5.6 Scaling studies

Training data. In Figure 7, left, we report the performance of our model as the size of the training dataset increases. We find that training on more trajectories results in a considerably lower position tracking error. This is a promising signal for scaling our approach to larger datasets in future work.

Context length. In Figure 7, middle, we study the effect of increasing the number of tokens used in the context window of the transformer model. We observe that the larger context windows result in better performance which suggests that our model is able to benefit from the additional context.

Model size. In Figure 7, right, we compare the models of increasing size (1M, 2M, 8M) by varying the embedding dimension (144, 192, 384), number of attention heads (3, 4, 12), and number of transformer blocks (4, 4, 6). We see that the error decreases monotonically with model size.

5.7 Prediction error correlates with performance

We collect 14 models trained with different training recipes, model architectures, data size and types, and compute the tracking and prediction errors for each of them. In Figure 8, we report the tracking and prediction errors of all the models in a single scatter plot. We can see that tracking and prediction error are highly correlated with Pearson coefficient $r = 0.87$. This suggests that the prediction error is predictive of performance, and that models with lower prediction error on the validation set are likely to follow input commands with higher accuracy.

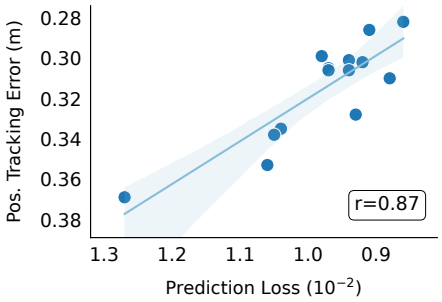

Figure 8: **Prediction error correlation.**

6 Discussion

Limitations. Our approach still lags behind state-of-the-art MPC controllers and is weaker than the state-of-the-art RL controllers in some regards, most notably robustness. In principle, we could scale our model to millions of YouTube trajectories. However, a major obstacle for doing this is the reliability of computer vision techniques which require a considerable amount of curation in practice.

Conclusion. We cast real-world humanoid locomotion as next token prediction. Our model is trained on sensorimotor sequences, which come from a neural network policy, a model-based controller, human motion capture, and videos of humans. We show that our model enables a humanoid robot to walk in the real world zero-shot. These findings suggest a promising path toward learning challenging real-world robotic control tasks by generative modeling of large collections of sensorimotor sequences.

Contributions

Ilija Radosavovic conceptualized and led the project. Generated robot datasets, implemented initial codebase and models, designed and performed experiments, and led the writing of the paper.

Bike Zhang worked on retargeting, performed experiments, and contributed to writing.

Baifeng Shi designed and implemented improved labeled training, trained and evaluated models, implemented improved unlabeled training, performed experiments, and contributed to writing.

Jathushan Rajasegaran designed and implemented improved model architectures, trained and evaluated models, generated human datasets, performed experiments, and contributed to writing.

Sarthak Kamat performed experiments and contributed to writing.

Trevor Darrell advised BS and SK and provided guidance.

Koushil Sreenath advised BZ, provided feedback, and robot equipment.

Jitendra Malik advised IR and JR and was the senior advisor to the project.

Acknowledgements

This work was supported in part by DARPA Machine Common Sense program and ONR MURI program (N00014-21-1-2801) for JM, a DGX A100 donation from NVIDIA, the Hong Kong Centre for Logistics Robotics and The AI Institute for KS, and BAIR's industrial alliance programs. We thank Saner Cakir and Vikas Ummadisetty for help with retargeting, and Alexei Efros for discussions.

References

[1] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv:2306.11706*, 2023.

[2] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv:2212.06817*, 2022.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[4] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020.

[5] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HCT*, 2019.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

[8] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv:2303.03378*, 2023.

[9] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *arXiv:1902.08710*, 2019.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[11] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv:2111.06377*, 2021.

[12] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv:2009.03300*, 2020.

[13] Kazuo Hirai, Masato Hirose, Yuji Haikawa, and Toru Takenaka. The development of honda humanoid robot. In *ICRA*, 1998.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[16] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. In *IROS*, 2001.

[17] 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:2001.08361*, 2020.

[18] Ichiro Kato. Development of wabot 1. *Biomechanism*, 1973.

[19] Scott Kuindersma. Recent progress on atlas, the world's most dynamic humanoid robot, 2020.

[20] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023.

[21] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *ICCV*, 2019.

[22] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. In *NeurIPS*, 2021.

[23] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

[24] Matthias Plappert, Christian Mandery, and Tamim Asfour. The KIT motion-language dataset. *Big Data*, 2016.

[25] 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 *ICML*, 2021.

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

[27] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. 2019.

[28] Ilija Radosavovic, Baifeng Shi, Letian Fu, Ken Goldberg, Trevor Darrell, and Jitendra Malik. Robot learning with sensorimotor pre-training. In *CoRL*, 2023.

[29] Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 2024.

[30] Marc H Raibert. *Legged robots that balance*. MIT press, 1986.

[31] Jathushan Rajasegaran, Georgios Pavlakos, Angjoo Kanazawa, and Jitendra Malik. Tracking people by predicting 3d appearance, location and pose. In *CVPR*, 2022.

[32] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021.

[33] Claude E Shannon. Prediction and entropy of printed english. *Bell system technical journal*, 1951.

[34] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *CoRL*, 2022.

[35] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

[36] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[38] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NeurIPS*, 2016.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification:

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.