2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Data-Driven Stochastic Motion Evaluation and Optimization with Image by Spatially-Aligned Temporal Encoding

Takeru Oba[1] and Norimichi Ukita[1]

***Abstract*— This paper proposes a probabilistic motion prediction method for long motions. The motion is predicted so that it accomplishes a task from the initial state observed in the given image. While our method evaluates the task achievability by the Energy-Based Model (EBM), previous EBMs are not designed for evaluating the consistency between different domains (i.e., image and motion in our method). Our method seamlessly integrates the image and motion data into the image feature domain by spatially-aligned temporal encoding so that features are extracted along the motion trajectory projected onto the image. Furthermore, this paper also proposes a data-driven motion optimization method, Deep Motion Optimizer (DMO), that works with EBM for motion prediction. Different from previous gradient-based optimizers, our self-supervised DMO alleviates the difficulty of hyper-parameter tuning to avoid local minima. The effectiveness of the proposed method is demonstrated with a variety of experiments with similar SOTA methods.**

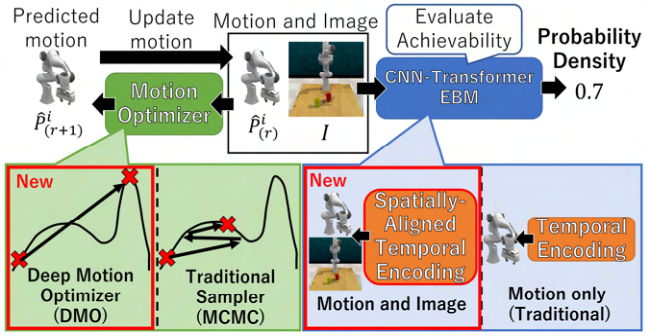

Fig. 1. Overview of our proposed method. CNN-Transformer EBM estimates the probability of a motion. The motion is efficiently optimized to improve its probability by Deep Motion Optimizer.

I. INTRODUCTION

One of the difficulties in motion prediction is that, for a certain situation, there might be a variety of appropriate motions, each of which accomplishes the task. It is not easy to train such one-to-many relationships compared to one-to-one relationships. In particular, a deterministic model is not good at training the one-to-many relationships. For representing the one-to-many relationships for motion prediction, probabilistic generative models are commonly used [1], [2].

An Energy-Based Model (EBM) is governed by an energy function representing the probability density of a state. EBM is more flexible than other generative models such as VAE [3], [4] and Normalizing Flow (NF) [5], [6] in terms of the model architecture and the prior distribution. As shown in the right part of Fig. 1, given an image observing the initial situation of a robot and its surrounding environment, EBM estimates the probability density of consistency between the initial image and motion. The probability density becomes higher, if the motion is considered to be appropriate for accomplishing the task from the initial situation.

However, different from other generative models, EBM predicts no motion but estimates only the probability density of each motion. With the probability density estimated by EBM, appropriate motions can be predicted using sampling such as MCMC. Accordingly, the accuracy of the predicted motion depends on EBM as well as sampling. However, many sampling methods are controlled by hyper-parameters, and it is difficult to manually tune them to avoid local minima in high-dimensional distributions such as long sequences.

To resolve the aforementioned problem of parameter tuning, our proposed Deep Motion Optimizer (DMO) rectifies each motion with a data-driven manner using a set of training motions. DMO is trained to directly rectify the motion to its ideal one, while sampling-based methods iteratively search for the ideal one based on the gradients of the probability density distribution, as illustrated in the left of Fig. 1.

Another difference between our EBM and previous EBMs for motion prediction is the types of domains fed into EBM. Since previous EBMs [7] evaluate the probability density of consistency between the same domain data (i.e., the observed and predicted motions), it is relatively easy to evaluate the consistency. In our EBM, on the other hand, the consistency is evaluated between data of different domains (i.e., the initial image and the predicted motion). While the image is useful for motion prediction based on the surrounding environment, we have to properly evaluate the consistency between the image and each predicted motion. To facilitate this evaluation in our method, the image and the predicted motion are seamlessly integrated into the image domain so that image features are extracted along the motion trajectory for spatially-aligned temporal encoding.

Our novel contributions are summarized as follows:

- Different from previous EBMs evaluating only motions, our proposed EBM properly evaluates the consistency between the image and motion domains by efficient spatially-aligned temporal encoding.
- Instead of hand-crafted hyper-parameters for tuning iterative optimizers, the data-driven framework for training DMO is proposed to directly optimize the motion.

*This work was not supported by any organization.

[1]The authors are with Graduate School of Engineering, Toyota Technological Institute, 2-12-1 Hisakata, Tempaku, Nagoya, 468-8511 Japan, sd21502@toyota-ti.ac.jp

II. Related work

A. EBM

It is not easy to represent a complex distribution in a high-dimensional space by EBM using sampling such as MCMC. To tackle this problem, in Score matching [8], [9] where no sampling is needed, given the gradient of the probability density of a real sample [8], EBM is trained so that the gradient estimated by EBM gets close to the given real gradient. In Noise Contrastive Estimation [10], [11], given a simple distribution where sampling is easy, EBM is trained to discriminate between this simple distribution and the distribution of real samples. Even with these training methods [8], [9], [10], [11], however, EBM inference still need to estimate the complex high-dimensional distribution by sampling.

Another solution is to reduce the sampling difficulty both in the training and inference stages by using other generative models such as VAE [7], [12] and NF [13], [14]. In [7] and [12], sampling is achieved in the low-dimensional space induced by VAE. In [14], initial inaccurate samples are produced fast by NF, and then these initial samples are given to EBM for more accurate sampling. Even with these methods, however, accurate sampling is still difficult to predict the accurate high-dimensional long-term motions.

B. Motion Prediction

Difficulties in our motion prediction problem include (i) long-term error accumulation, (ii) stochasticity of motions, and (iii) multi-domain data processing.

Even with recent LSTM [15], [16], [17], [18] and GRU [19], [20], it is difficult to predict a long-time complex motion due to gradient vanishing. This problem is alleviated by Transformer [21], [22], [23] using an attention mechanism [24], which keeps the gradient stable [25].

Stochastic motions can be predicted by probabilistic generative models such as VAE [26], [27], [2] and GAN [28], [29], [30]. However, these models do not explicitly estimate the probability density of a motion. This disadvantage prevents us from optimizing the motion using maximum likelihood estimation using the probability density as the likelihood. This problem is resolved by enabling probabilistic models to explicitly estimate the probability density, for example, NF [31], [32] and EBM [7], [33]. However, NF and EBM still have several problems as described in Sec. II-A.

For multi-domain data processing such as our method using image and motion data, one of the simplest ways is to concatenate these data. The expressiveness of the concatenated data can be improved by feature extraction such as convolution and pooling for the image data [34], [35], [36]. However, spatial pooling makes is difficult to precisely localize the motion in the image [37]. Inconsistency between the coordinate systems of the image and motion data is also problematic. For the former problem, we propose efficient high-dimensional feature extraction that focuses on the motion trajectory on the image. Since the latter problem can be resolved by coordinate rearrangement (e.g., motion heatmapping in the image [2]), our method also employs it.

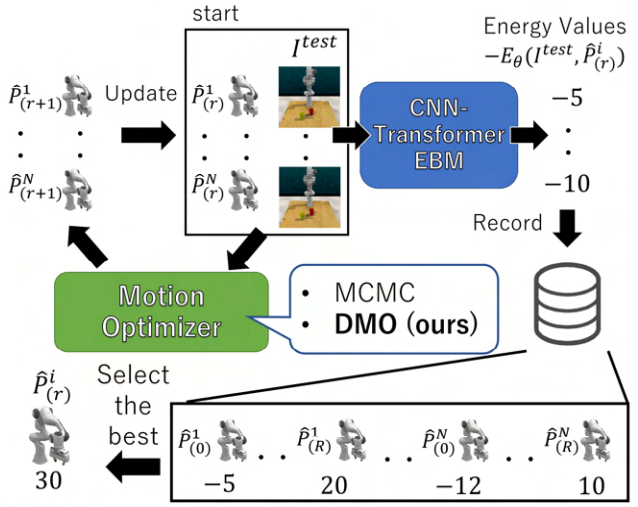

Fig. 2. Motion prediction in the inference stage. The image and motions are fed into EBM and the Motion Optimizer. EBM estimates the energy of each motion. Motion Optimizer updates the motion to increase its energy. Finally, the motion having the highest energy is selected as the predicted motion.

III. Inference and Training Methods

A. Notations

An image and a motion sequence are denoted by I and P, respectively. P is a set of $(T+1)$ temporal pose vectors $\boldsymbol{p}_t$ where $t \in \{0, \cdots, T\}$. In our experiments, $\boldsymbol{p}_t$ is a 11-D vector consisting of the 3D location of the robot hand ($\boldsymbol{x}_t$), its orientation ($\boldsymbol{r}_t$), its status (s_t), and a time stamp t. $\boldsymbol{x}_t$, $\boldsymbol{r}_t$, and s_t are a 3D positional vector in the camera coordinate system, a 6D vector representing the 3D orientation [38], and a scalar value between 0 and 1 representing the open (1) and close (0) status of the hand gripper, respectively. I^i and P^i denote the i-th data, where $i \in \{1, \cdots, N\}$, in our training dataset. In each of all pairs in the training dataset, P^i can accomplish the task from I^i.

B. Motion Prediction using EBM and DMO

Motion prediction from the initial state represented by the image is illustrated in Fig. 2. In our method, I and P are fed into EBM (denoted by E_θ) to estimate the probability density $p(P|I)$ of the consistency between I and P:

$$p(P|I) = \frac{\exp(-E_\theta(I,P))}{Z(I)}, \tag{1}$$

where θ denotes the parameters in EBM. Z, which depends only on I, is defined as follows for normalization:

$$Z(I) = \int \exp(-E_\theta(I,P))dP \tag{2}$$

In the inference stage, given I^{test} and any motion P, EBM is required to estimate the consistency between I^{test} and P. While $Z(I)$ in Eq. (1) is computationally intractable, $Z(I)$ is not changed because I is fixed to I^{test}. Instead of the probability density, therefore, the energy term $-E_\theta(I^{test}, P)$ is used as the output of EBM.

To increase $-E_\theta(I^{test},P)$, DMO updates P in order to bring P to its ideal motion (denoted by P^{test} paired with I^{test}), which is not available in the inference stage, as follows:

$$\hat{P}_{(1)} = DMO(I,P), \tag{3}$$

where I^{test} is substituted to I in the inference stage. Since it is not guaranteed that $\hat{P}_{(1)}$ is sufficiently close to P^{test}, DMO is recurrently used by rewriting Eq. (3) to Eq. (4):

$$\hat{P}_{(r+1)} = DMO(I,\hat{P}_{(r)}), \tag{4}$$

where the initial motion P in Eq. (3) is denoted by $\hat{P}_0$, and r is an integer greater than or equal to zero.

As initial motions each of which is fed into DMO in Eq. (4), $\{P^1,\cdots,P^N\}$ in the training dataset are reused in our method so that $\hat{P}^i_{(r+1)} = DMO(I^{test},\hat{P}^i_{(r)})$. Among all $\{P^1,\cdots,P^N\}$, several of them are close to P^{test}, while the remaining ones are far from P^{test}. For efficiency, only the former motions are used. These motions are determined by selecting the top n energy values computed by EBM, and fed into DMO expressed by Eq. (4).

In our method, both EBM and DMO are implemented as neural networks trained as described in Secs. III-C and III-D, respectively.

C. Training of EBM

If previous EBM training methods [33] are used for estimating E_θ, θ is optimized by maximizing the log likelihood of the following loss function $\mathscr{L}_{EBM}(\theta)$:

$$\mathscr{L}_{EBM}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(-E_\theta(I^i,P^i) - \log Z_\theta(I^i)\right) \tag{5}$$

The gradient of $\mathscr{L}_{EBM}(\theta)$ is expressed as follows:

$$\nabla\mathscr{L}_{EBM}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(-\nabla E_\theta(I^i,P^i) + \mathbb{E}_{\check{P}^i\sim p(P|I^i)}\nabla E_\theta(I^i,\check{P}^i)\right), \tag{6}$$

where $\check{P}^i$ is a motion synthesized by sampling based on $p(P|I^i)$ given by EBM. This sampling process is not easy in high dimension, as described in Sec. II-A.

To consider how to avoid the difficulty in this sampling, we focus on Eq. (6). With $-\nabla E_\theta(I^i,P^i)$ in Eq. (6), $p(P^i|I^i)$ as the probability density of a positive sample where P^i accomplishes the task from I^i is increased. On the other hand, $\nabla E_\theta(I^i,\check{P}^i)$ decreases $p(\check{P}^i|I^i)$ as the probability density of a negative sample where $\check{P}^i$ cannot achieve the task from I^i. In our method, these positive and negative samples are selected from the training dataset as follows. As described in Sec. III-B, $\{P^1,\cdots,P^N\}$ in the training dataset are reused as initial motions fed into DMO in the inference stage. To make this inference succeed, P^i is expected to be updated towards P^{test}, if I^i is the closest to I^{test}. On the other hand, P^j where $j\neq i$ is expected not to be updated to P^i. These relationships are represented in EBM so that (I^i,P^i) and (I^i,P^j) are trained as the positive and negative samples, respectively.

In addition to the positive and negative samples directly borrowed from the training dataset, our method augments

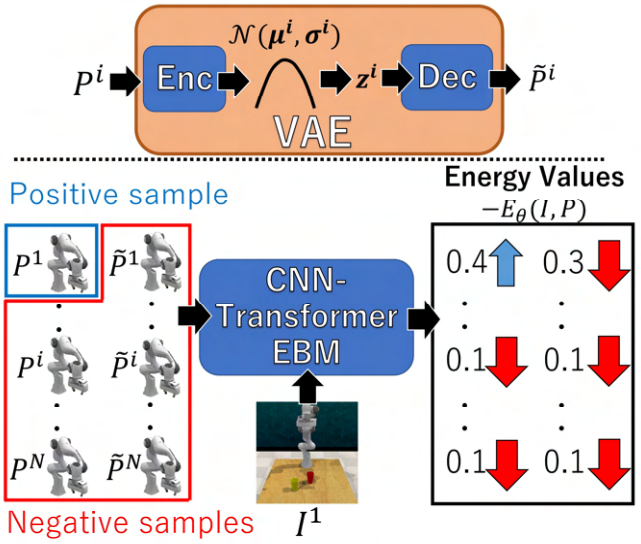

Fig. 3. Training procedure of our EBM. VAE reconstructs the motions to acquire various negative motions. The image and motions are fed into the EBM. EBM estimates the energy of each motion. This EBM is trained to improve the energy of the positive sample and decrease that of each negative sample.

samples for training EBM so that P close to P^j should be also a negative sample. This data augmentation is done by simple low-dimensional sampling via VAE. VAE trains an embedding space from a set of all N motion sequences (i.e., $P^1,\cdots,P^N$) in the training dataset. The trained VAE is denoted by V. P^j is fed into V to get $\mathcal{N}(\boldsymbol{\mu}^j,\boldsymbol{\sigma}^j)$, as shown in the upper part of Fig. 3. Let $\boldsymbol{z}^j$ be sampled from $\mathcal{N}(\boldsymbol{\mu}^j,\boldsymbol{\sigma}^j)$. The motion decoded from $\boldsymbol{z}^j$ (denoted by $\tilde{P}^j$) is close to its original motion P^j Since it is easy to represent this simple Gaussian distribution in a low-dimensional embedding space, sampling from this distribution in V allows us to stably synthesize $\tilde{P}^j$ close to P^j.

As shown in the lower part of Fig. 3, with the positive and negative samples prepared by the aforementioned manner, EBM is trained with the following loss function:

$$\nabla\mathscr{L}_{EBM}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(-\nabla E_\theta(I^i,P^i) + \sum_{j\neq i}\nabla E_\theta(I^i,P^j) + \sum_{j=1}^{N}\mathbb{E}_{\tilde{P}^j\sim\mathcal{N}(\boldsymbol{\mu}^j,\boldsymbol{\sigma}^j)}\nabla E_\theta(I^i,\tilde{P}^j)\right) \tag{7}$$

D. Training of DMO

Different from general motion optimizers such as MCMC sampling [39], DMO alleviates the difficulty of hyper-parameter tuning by being trained in a data-driven manner as follows. To train DMO, (I^i,P^i) in which the task can be accomplished is given as the ground-truth positive sample in the training dataset. DMO is trained so that any P close to P^i is updated toward P^i. In reality, however, P is not randomly distributed but should be limited within realistic robot motions (e.g., within range of articulated motion).

In our method, a set of such realistic P around P^i is synthesized via the trained VAE, V, as done in the training

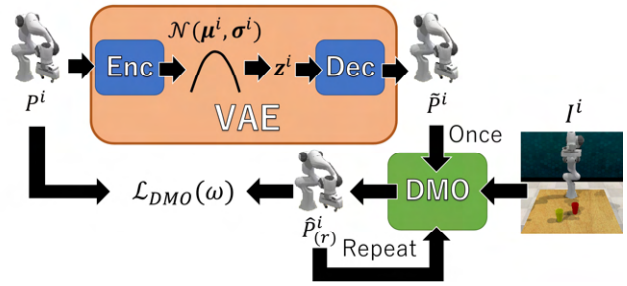

Fig. 4. Training procedure of our Deep Motion Optimizer (DMO). VAE reconstructs the motion from hidden vector z^i which include small noise. DMO is optimized to refine this small difference to precisely predict the ground truth motion.

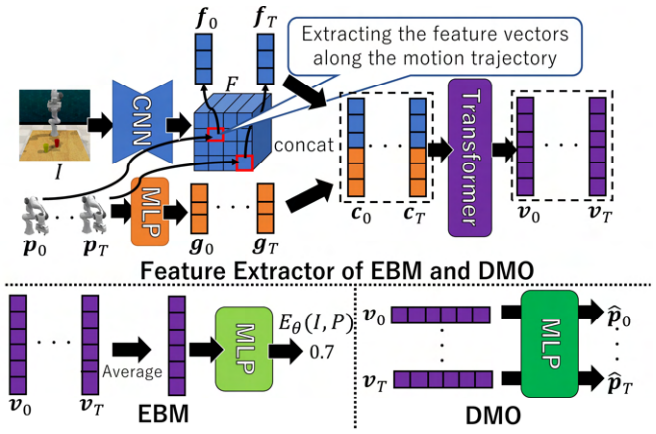

Fig. 5. The architectures of EBM and DMO. EBM and DMO have the same feature extractor but the parameters are not shared. CNN has the UNet-shaped architecture to create the feature map F. The pose features are extracted by MLP. To merge the image and pose features, the image features $\boldsymbol{f}_k$ are extracted from F along the motion trajectory, and concatenated with the pose features $\boldsymbol{g}_k$. These features are fed into Transformer to provide the features $\boldsymbol{v}_k$ to EBM and DMO.

of EBM. This training procedure is illustrated in Fig. 4. P^i is fed into V to get the motion sampled from V. This sampled motion (denoted by $\tilde{P}^i$) is close to its original motion P^i, as described in Sec. III-C.

By substituting $(I^i, \tilde{P}^i)$ to $(I, \hat{P}_{(r)})$ where $r = 0$ in Eq. (4), DMO gets the updated motion $\hat{P}^i_{(1)}$. As with the inference stage, DMO expressed by Eq. (4) is recurrently used to get $\{\hat{P}^i_{(2)}, \cdots, \hat{P}^i_{(R+1)}\}$.

With $\{\hat{P}^i_{(1)}, \cdots, \hat{P}^i_{(R+1)}\}$, the parameters of DMO denoted as ω are optimized with the following loss in which the difference between $\hat{P}^i_{(r)}$ and its ideal motion P^i is minimized:

$$\mathscr{L}_{DMO}(\omega) = \frac{1}{N}\sum_{i=1}^{N}\sum_{r=1}^{R+1} ||P^i - \hat{P}^i_{(r)}||_2, \tag{8}$$

where R denotes the number of recurrently-updated motions. In $\mathscr{L}_{DMO}(\omega)$, motion data augmentation is achieved by training all $(R+1)$ motions (i.e., $\hat{P}^i_{(1)}, \cdots, \hat{P}^i_{(R+1)}$) instead of training only $\hat{P}^i_{(R+1)}$.

IV. Network Architectures of EBM and DMO

EBM and DMO have the same architecture for feature extraction (as shown in the upper part of Fig. 5) but are trained independently, while their last layers are differently designed (as shown in the lower part of Fig. 5). The UNet-shaped image feature extractor [40] using CNN [41] extracts the feature maps (denoted by F) from I, as indicated by blue in Fig. 5. The spatial dimensions of I and F are equal. Each pose vector $\boldsymbol{p}_t$ where $t \in \{0, \cdots, T\}$ is independently fed into the MLP-based pose feature extractor, as indicated by orange in Fig. 5. Each extracted pose feature is denoted by $\boldsymbol{g}_t$.

F and $\boldsymbol{g}_t$ are fed into Transformer for extracting informative features by merging F and $\boldsymbol{g}_t$. While all the features in F can be theoretically fed into this Transformer, many redundant image features such as those of the background are less useful for motion prediction. Such useless features make it difficult to properly train Transformer. In our method, this problem is resolved by extracting only meaningful features from F. Based on the idea that the most meaningful features for motion prediction are located along the motion trajectory, our method extracts the features in the temporal image coordinates of the robot hand, each of which is denoted by $\boldsymbol{u}_t$ at $t \in \{0, \cdots, T\}$. $\boldsymbol{u}_t$ is the 2D projection of $\boldsymbol{x}_t$, which is the 3D position of the robot hand. Given the real coordinates $\boldsymbol{u}_t$, the feature in $\boldsymbol{u}_t$ (denoted by $\boldsymbol{f}_t$) is computed using bilinear interpolation in F. $\boldsymbol{f}_t$ is concatenated with $\boldsymbol{g}_t$, and all the temporal concatenated features (each of which is denoted by $\boldsymbol{c}_t$) is fed into Transformer. From $\{\boldsymbol{c}_0, \cdots, \boldsymbol{c}_T\}$, Transformer extracts the temporal features $\{\boldsymbol{v}_0, \cdots, \boldsymbol{v}_T\}$, as indicated by purple in Fig. 5.

The advantages of the aforementioned concatenated features are summarized as follows:

1) By spatially aligning the image and pose coordinate systems, $\{\boldsymbol{f}_0, \cdots, \boldsymbol{f}_T\}$ are extracted from the 2D positions consisting of each motion trajectory in F. Since $\{\boldsymbol{f}_0, \cdots, \boldsymbol{f}_T\}$ are temporal data, this encoding can be regarded as spatially-aligned temporal encoding.
2) Data of two different domains (i.e., image and motion) are integrated into the image feature domain.
3) The computational cost and training difficulty are reduced by decreasing the data size fed into Transformer.

Given $\{\boldsymbol{v}_0, \cdots, \boldsymbol{v}_T\}$, EBM and DMO are designed as follows:

- **EBM:** EBM evaluates the likelihood of temporal data as the energy (denoted by $E_\theta(I,P)$ in Fig. 5). This temporal evaluation is done by simultaneously feeding $\{\boldsymbol{v}_0, \cdots, \boldsymbol{v}_T\}$ into the evaluation network, which is implemented by MLP in our method. For simultaneous input of $\{\boldsymbol{v}_0, \cdots, \boldsymbol{v}_T\}$, their mean vector is fed into MLP for more efficient processing than other temporal processing such as recurrent networks.
- **DMO:** As with other temporal synthesis methods using Transformer, DMO separately accepts $\{\boldsymbol{v}_0, \cdots, \boldsymbol{v}_T\}$ to independently acquire $\hat{\boldsymbol{p}}_t$ from $\boldsymbol{v}_t$.

Our code is available at github.com/Obat2343/DMOEBM.

TABLE I

TASK SUCCESS RATE OF PREDICTION METHODS. **RED** AND **BLUE** VALUES INDICATE THE **BEST** AND **SECOND-BEST**, RESPECTIVELY.

	CB	PC	PB	PK	PR	PU	RT	SW	TP
(a) Ours-5	**97**	**89**	**85**	**51**	**85**	**25**	**32**	**74**	**94**
(b) Ours-1	**95**	**89**	**87**	**38**	**77**	**19**	31	**85**	**94**
(c) Langevin	71	77	83	0	46	11	0	18	77
(d) GD	54	78	61	18	40	5	**36**	48	81
(e) VAEBM-L	64	88	81	27	35	10	**32**	48	77
(f) VAEBM-S	62	82	79	27	40	9	**32**	50	74

TABLE II

TASK SUCCESS RATE OF VARIOUS ARCHITECTURES. (a) AND (b) USE OUR PROPOSED SPATIALLY-ALIGNED TEMPORAL ENCODING.

	CB	PC	PB	PK	PR	PU	RT	SW	TP
(a) Ours	**97**	**89**	**85**	**51**	**85**	**25**	**32**	**74**	**94**
(b) w/o concat	**96**	**85**	**94**	**55**	**91**	**13**	**26**	**85**	**93**
(c) with GAP	5	3	0	4	1	0	3	5	0
(d) with ViT	0	0	0	1	1	1	4	0	2

TABLE III

THE NUMBER OF MODEL PARAMETERS AND COMPUTATIONAL COST.

	All		*Reuse*	
	Params (M)	Cost (GMac)	Params	Cost
(a) Ours	35.84	23.62	1.00	0.11
(b) Ours w/o concat	35.68	23.71	0.84	0.19
(c) Ours with GAP	35.68	23.61	0.84	0.10
(d) Ours with ViT	35.87	24.97	0.84	0.38

V. EXPERIMENTAL RESULTS

A. Dataset

We evaluate our method using the RLBench dataset [42] in which a robot performs various tasks on the simulator, Pyrep [43]. The nine tasks used in our experiments are shown in Fig. 6. For each task, 1,000 and 100 training and test sequences were generated in the simulator. In all the sequences, the Franka Panda robot was used. The initial state was captured as an image in each sequence. Each image consists of RGB channels and a depth channel. The image size is 256×256 pixels. As VAE, ACTOR [44] was used.

B. Motion Optimizers

Our proposed motion optimizer (DMO) is evaluated with other motion optimizers so that each of the following optimizers is substituted for "Motion Optimizer" in Fig. 2:

(a) Ours-5: DMO proposed in Sec. III-D. $R = 5$ and $R = 1$ in the training and prediction stages, respectively.

(b) Ours-1: DMO proposed in Sec. III-D. Both in the training and prediction stages, $R = 1$.

(c) Langevin [39]: Langevin MCMC is widely used with EBM for prediction. For motion prediction, Langevin MCMC updates a motion using the gradient estimated by EBM and additive noise. Two hyper-parameters, the step size and the number of iterations, are 0.001 and 100, respectively.

(d) GD [45]: Different from Langevin MCMC, gradient descent (GD) optimization is not affected by noise. As with (c) Langevin MCMC, the step size and the number of iterations, are 0.001 and 100, respectively.

(e) VAEBM-L [12]: Langevin MCMC is achieved in the embedding space trained by VAE. The step size and the number of iterations, are 0.01 and 100, respectively.

(f) VAEBM-S [12]: This is same with (e) except that the step size is 0.001.

The success rate of each optimizer is shown in Table I. In all tasks except RT, our methods (a) and (b) outperform the others. This verifies the superiority of the proposed combination of probabilistic and deterministic methods over the probabilistic methods (c), (d), (e), and (f). The examples of motions predicted by (a) Ours-1 and (e) VAEBM-L, which is the best among the others, are shown in Fig. 7. We can see that Ours-1 can finely rectify the predicted motions for accomplishing the tasks. In comparison between (a) Ours-5 and (b) Ours-1, Ours-5 is a bit better. This might be because of motion data augmentation by iterative motion updates.

More examples are shown in the supplementary video.

C. Model Architectures of Feature Extractor

The following four variants of the feature extractor in EBM and DMO are evaluated.

(a) Ours: The feature extractor with the proposed spatially-aligned temporal encoding (upper part of Fig. 5).

(b) Ours w/o concat: Same with (a) except that $\{\boldsymbol{g}_0, \cdots, \boldsymbol{g}_T\}$ are not concatenated with $\{\boldsymbol{f}_0, \cdots, \boldsymbol{f}_T\}$. All the features are directly fed into Transformer, as shown in Fig. 8 (b).

(c) Ours with GAP: F is reduced to $\bar{\boldsymbol{f}}$ by Global Average Pooling [46] for dimensionality reduction. $\bar{\boldsymbol{f}}$ and $\{\boldsymbol{g}_0, \cdots, \boldsymbol{g}_T\}$, which are not concatenated, are fed into Transformer, as shown in Fig. 8 (c).

(d) Ours with ViT feature: As with ViT [47], F is spatially divided to patch features denoted by $\{\boldsymbol{f}_0, \cdots, \boldsymbol{f}_l\}$ where $l = 256$ in our experiments. The patch features are positionally encoded as done in generic Transformers [24], and then fed into Transformer with the pose vectors $\{\boldsymbol{g}_0, \cdots, \boldsymbol{g}_T\}$ without concatenation, as shown in Fig. 8 (d).

Their task success rates are shown in Table II. Our spatially-aligned temporal coding (a) and (b) outperform (c) and (d), while the effectiveness of the feature concatenation done in (a) is not validated.

Table III shows the cost (i.e., the number of parameters and the computational cost) of the four variants. Table III shows only the results of EBM because the dominant part is shared by EBM and DMO. Furthermore, when multiple motions are evaluated and updated with the same image by EBM and DMO, respectively, we can reuse F for reducing the cost. The results in the cases where F is reused and not reused are shown in *Reuse* and *All* columns, respectively.

The costs are almost same between (a), (b), (c), and (d) because the cost in CNN is larger than the costs in Transformer and MLP. On the other hand, it can be confirmed that the costs in *Reuse* are much smaller than those in *All*. This property is important because the performance is gained as the number of input motions (i.e., n) increases, as demonstrated later in Table V. If the number of input motions becomes n-fold, the total cost in Reuse also becomes n-fold.

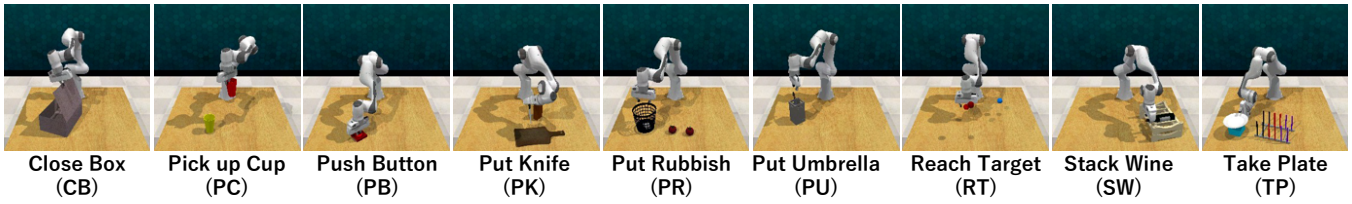

Fig. 6. List of the nine tasks. The position of the camera observing the initial image is fixed. Objects are randomly placed.

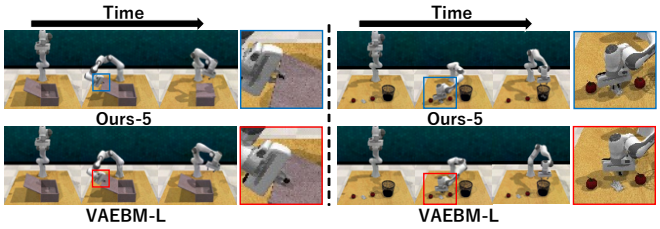

Fig. 7. Examples of predicted motions. Left: CB task. Right: PR task.

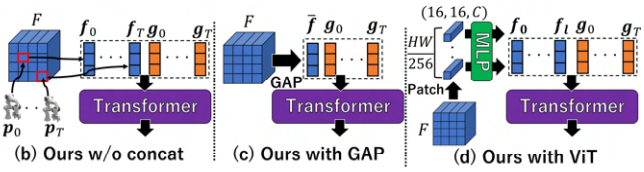

Fig. 8. Model architectures modified from our model.

TABLE IV

TASK SUCCESS RATES OF TWO VARIANTS OF OUR PREDICTION MODEL.

	CB	PC	PB	PK	PR	PU	RT	SW	TP
(a) Ours	95	86	73	56	84	13	29	79	74
(b) Only VAE	0	3	5	3	2	0	0	0	0

TABLE V

TASK SUCCESS RATES OF OUR METHOD WITH DIFFERENT n.

	CB	PC	PB	PK	PR	PU	RT	SW	TP
R=1, n=1	**99**	83	**84**	37	54	14	**32**	57	84
R=1, n=8	**97**	**89**	**85**	**51**	**85**	**25**	**32**	**74**	**94**
R=1, n=1000	**97**	**88**	68	**71**	**86**	**26**	**32**	**83**	**95**

TABLE VI

TASK SUCCESS RATES OF OUR METHOD WITH DIFFERENT R.

	CB	PC	PB	PK	PR	PU	RT	SW	TP
R=1, n=8	**97**	**89**	**85**	51	**85**	**25**	**32**	74	**94**
R=5, n=8	**95**	86	73	**56**	**84**	13	**29**	**79**	74
R=10, n=8	**95**	**88**	**76**	**57**	78	**16**	7	**79**	**92**

D. EBM Training Methods

The following two variants of the EBM training methods are evaluated, while their prediction method is fixed to be that described in Sec. III-B.

(a) Ours: The training procedure shown in Fig. 3. For making a negative pair with I^i, either of P^j where $j \neq i$ or $\tilde{P}^j$ synthesized by VAE is paired with I^i.

(b) Ours with only VAE: This is the same with (a) except that only $\tilde{P}^j$ synthesized by VAE is paired with I^i for making negative pairs, as done in [12] and [13].

The task success rates of these two variants are shown in Table IV. The scores in (b) are significantly worse. We interpret this performance drop in what follows.

In (b), the loss is expressed by $\sum_{j\neq i} \nabla E_\theta(I^i, P^j)$ is removed from Eq. (7). This loss can be minimized not only if any (I, P) can be correctly discriminated between positive and negative data but also if $\tilde{P}$ synthesized by V is discriminated from P^i independently of I^i. This latter case might be caused because the loss is decreased to almost zero, while the task success rate is low. This problem can be resolved by synthesizing training data that cannot be discriminated from P^i. While a possible way to synthesize such realistic training data is gradient-based adversarial training such as GAN [48], it is unstable [49], [50]. We also confirmed that EBM training methods using Langevin MCMC [39] and VAEBM [12], both of which exploit gradient information, are unstable. Instead of synthesizing realistic data, on the other hand, real training data is used as negative samples, as expressed in Eq. (7).

E. Parameters in DMO Inference

In Table V and VI, the performance changes with the changes in n and R are shown, respectively. By increasing n, the performance is gained in many tasks such as PC, PK, PR, PU, SW, TP, while n has less impact in CB and RT. Since n is the number of motions evaluated in EBM and rectified in DMO, it is natural that the task success rate can be increased. By increasing R, on the other hand, the performance change is fluctuating. This might be caused because a limited number of training data might be insufficient, if the task is achievable by a variety of motions. In such a case, since the training samples are sparsely distributed, overfitting is caused by increasing the number of iterative updates (i.e., R).

VI. Concluding Remarks

This paper proposed a robot motion prediction method using a given image representing the initial situation of the environment including the robot. Our method modifies EBM for evaluating the consistency between data of different domains (i.e., image and motion data). In addition, a data-driven motion optimizer called DMO is proposed for improving the task achievability in a high-dimensional motion space.

Future work includes broader motion augmentation required for predicting complex motions, while motion samples augmented by VAE in our method are locally distributed. The diffusion model [51] might be a prospective solution. For realistic scenarios, physical constraints [52] are also useful.

REFERENCES

[1] M. Hassan, D. Ceylan, R. Villegas, J. Saito, J. Yang, Y. Zhou, and M. J. Black, "Stochastic scene-aware motion prediction," in *ICCV*, 2021, pp. 11 354–11 364.

[2] Z. Cao, H. Gao, K. Mangalam, Q.-Z. Cai, M. Vo, and J. Malik, "Long-term human motion prediction with scene context," in *ECCV*, 2020.

[3] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[4] A. Vahdat and J. Kautz, "NVAE: A deep hierarchical variational autoencoder," in *NeurIPS 2020*, 2020.

[5] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[6] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," *NeurIPS*, vol. 30, 2017.

[7] B. Pang, T. Zhao, X. Xie, and Y. N. Wu, "Trajectory prediction with latent belief energy-based model," in *CVPR*, 2021, pp. 11 814–11 824.

[8] A. Hyvärinen and P. Dayan, "Estimation of non-normalized statistical models by score matching." *Journal of Machine Learning Research*, vol. 6, no. 4, 2005.

[9] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," *NeurIPS*, vol. 33, pp. 12 438–12 448, 2020.

[10] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *AISTATS*. JMLR Workshop and Conference Proceedings, 2010, pp. 297–304.

[11] A. J. Bose, H. Ling, and Y. Cao, "Adversarial contrastive estimation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 1021–1032.

[12] Z. Xiao, K. Kreis, J. Kautz, and A. Vahdat, "VAEBM: A symbiosis between variational autoencoders and energy-based models," in *ICLR*, 2021.

[13] R. Gao, E. Nijkamp, D. P. Kingma, Z. Xu, A. M. Dai, and Y. N. Wu, "Flow contrastive estimation of energy-based models," in *CVPR*, 2020, pp. 7518–7528.

[14] J. Xie, Y. Zhu, J. Li, and P. Li, "A tale of two flows: Cooperative learning of langevin flow and normalizing flow toward energy-based model," in *ICLR*, 2021.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *ICRA*, 2018.

[17] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments," in *ICRA*, 2018, pp. 5921–5928.

[18] B. Wang, E. Adeli, H.-k. Chiu, D.-A. Huang, and J. C. Niebles, "Imitation learning for human pose prediction," in *ICCV*, 2019, pp. 7124–7133.

[19] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[20] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," in *CVPR*, 2017.

[21] E. Aksan, M. Kaufmann, P. Cao, and O. Hilliges, "A spatio-temporal transformer for 3d human motion prediction," in *3DV*, 2021, pp. 565–574.

[22] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, "Transformer networks for trajectory forecasting," in *ICPR*, 2021, pp. 10 335–10 342.

[23] H. Kim, Y. Ohmura, and Y. Kuniyoshi, "Transformer-based deep imitation learning for dual-arm robot manipulation," in *IROS*, 2021, pp. 8965–8972.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, vol. 30, 2017.

[25] G. Kerg, B. Kanuparthi, A. G. ALIAS PARTH GOYAL, K. Goyette, Y. Bengio, and G. Lajoie, "Untangling tradeoffs between recurrence and self-attention in artificial neural networks," *NeurIPS*, vol. 33, pp. 19 443–19 454, 2020.

[26] S. Aliakbarian, F. Saleh, L. Petersson, S. Gould, and M. Salzmann, "Contextually plausible and diverse 3d human motion prediction," in *ICCV*, 2021, pp. 11 333–11 342.

[27] Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess, "Robust imitation of diverse behaviors," *NeurIPS*, vol. 30, 2017.

[28] E. Barsoum, J. Kender, and Z. Liu, "Hp-gan: Probabilistic 3d human motion prediction via gan," in *CVPR*, 2018, pp. 1418–1427.

[29] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NeurIPS*, 2016.

[30] A. Hernandez, J. Gall, and F. Moreno-Noguer, "Human motion prediction via spatio-temporal inpainting," in *ICCV*, 2019, pp. 7134–7143.

[31] C. Schöller and A. Knoll, "Flomo: Tractable motion prediction with normalizing flows," in *IROS*, 2021, pp. 7977–7984.

[32] W. Mao, M. Liu, and M. Salzmann, "Generating smooth pose sequences for diverse human motion prediction," in *ICCV*, 2021, pp. 13 309–13 318.

[33] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *CoRL*, 2022.

[34] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *ICRA*, 2018.

[35] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *ICRA*, 2018, pp. 4693–4700.

[36] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *ICCV*, 2019.

[37] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski, "An intriguing failing of convolutional neural networks and the coordconv solution," *NeurIPS*, vol. 31, 2018.

[38] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *CVPR*, 2019.

[39] G. O. Roberts and R. L. Tweedie, "Exponential convergence of langevin distributions and their discrete approximations," *Bernoulli*, pp. 341–363, 1996.

[40] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*. Springer, 2015, pp. 234–241.

[41] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *CVPR*, 2022.

[42] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics Autom. Lett.*, vol. 5, no. 2, pp. 3019–3026, 2020.

[43] S. James, M. Freese, and A. J. Davison, "Pyrep: Bringing V-REP to deep robot learning," *arXiv:1906.11176*, 2019.

[44] M. Petrovich, M. J. Black, and G. Varol, "Action-conditioned 3d human motion synthesis with transformer vae," in *ICCV*, 2021, pp. 10 985–10 995.

[45] D. R. Wilson and T. R. Martinez, "The general inefficiency of batch training for gradient descent learning," *Neural networks*, vol. 16, no. 10, pp. 1429–1451, 2003.

[46] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[47] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *ICLR*, 2021.

[48] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[49] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *NeurIPS*, 2017.

[50] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018.

[51] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *NeurIPS*, vol. 33, pp. 6840–6851, 2020.

[52] T. Maeda and N. Ukita, "Motionaug: Augmentation with physical correction for human motion prediction," in *CVPR*, 2022.