Masked Trajectory Models for Prediction, Representation, and Control

Anonymous Authors[1]

Abstract

We introduce Masked Trajectory Models (MTM) as a generic abstraction for sequential decision making. MTM takes a trajectory and aims to reconstruct the trajectory conditioned on random subsets of the same trajectory. By training with a highly randomized masking pattern, MTM learns versatile networks that can take on different roles or capabilities, by simply choosing appropriate masks at inference time. For example, the same MTM network can be used as a forward dynamics model, inverse dynamics model, or even an offline RL agent. Through extensive experiments in several continuous control tasks, we show that the same MTM network – i.e. same weights – can match or outperform specialized networks trained for the aforementioned capabilities. Additionally, we find that state representations learned by MTM can significantly accelerate the learning speed of traditional RL algorithms.

1. Introduction

We propose the use of Masked Trajectory Models (`MTM`) as a generic abstraction and framework for prediction, representation, and control. Our approach draws inspiration from two recent trends in Artificial Intelligence. The first is the success of masked prediction, also known as masked autoencoding, as a simple yet effective self-supervised learning objective in NLP (Devlin et al., 2018; Liu et al., 2019; Brown et al., 2020) and computer vision (He et al., 2021). This task of masked prediction not only forces the model to learn good representations but also develops its conditional generative modeling capabilities. The second trend that inspires our work is the recent success of transformer sequence models, such as decision transformers, for reinforcement (Chen et al., 2021; Janner et al., 2021) and imitation learning (Reed et al., 2022). See Appendix A for additional background and related work.

Conceptually, `MTM` is trained to take a trajectory sequence of the form: $\boldsymbol{\tau} := (\mathbf{s}_k, \mathbf{a}_k, \mathbf{s}_{k+1}, \mathbf{a}_{k+1}, \ldots \mathbf{s}_t, \mathbf{a}_t)$ and reconstruct it given a masked view of the same, i.e. $\hat{\boldsymbol{\tau}} = \boldsymbol{h}_\theta(\texttt{Masked}(\boldsymbol{\tau}))$, where $\boldsymbol{h}_\theta(\cdot)$ is a bi-directional transformer and $\texttt{Masked}(\boldsymbol{\tau})$ is a masked view of $\boldsymbol{\tau}$. For example, one masked view of the above sequence could be: $(\mathbf{s}_k, \_\_, \_\_, \mathbf{a}_{k+1}, \_\_, \ldots, \mathbf{s}_t, \_\_)$ where $\_\_$ denotes a masked element. In this case, `MTM` must infill intermediate states and actions in the trajectory as well as predict the next action in the sequence. A visual illustration of our paradigm is shown in Figure 1.

Our Contributions Our main contribution is the proposal of `MTM` as a versatile modeling paradigm and pretraining method. We empirically investigate the capabilities of `MTM` on several continuous control tasks including planar locomotion (Fu et al., 2020) and dexterous hand manipulation (Rajeswaran et al., 2018). We find `MTM` has many unique capabilities such as the the ability for a single model to perform many tasks, training on heteromodal data, learning good representations, and more.

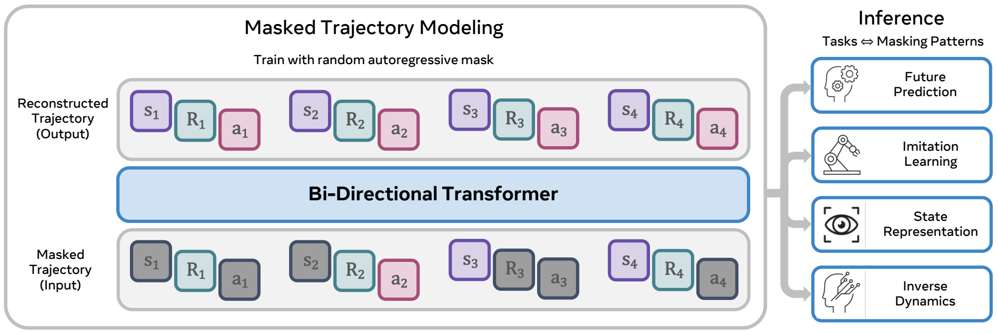

Masked Trajectory Modeling
Train with random autoregressive mask
Reconstructed Trajectory (Output)
Bi-Directional Transformer
Masked Trajectory (Input)
s_1 R_1 a_1 s_2 R_2 a_2 s_3 R_3 a_3 s_4 R_4 a_4
Inference
Tasks ⇔ Masking Patterns
Future Prediction
Imitation Learning
State Representation
Inverse Dynamics

Figure 1. **Masked Trajectory Modeling (`MTM`) Framework.** (Left) Depiction of the training process where the model is learned to reconstruct trajectory segments from a randomly masked view of the same. We use a bi-directional encoder and decoder to perform this masked autoencoding. (Right) A trained `MTM` model can enable several downstream use-cases by simply changing the masking pattern at inference time. See Section 2 for discussion on training and inference masking patterns.

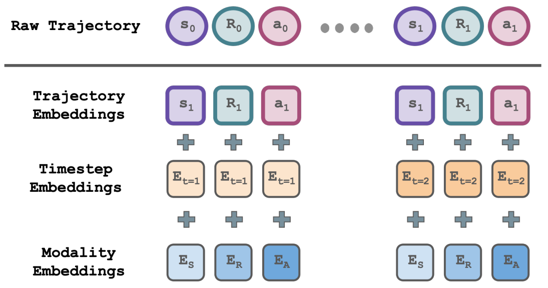


Figure 2. **Tokenization of the trajectory sequence** comprises three components. A modality specific encoder lifts from the raw modality space to a common representation space, where we additionally add timestep embeddings and modality embeddings. This allows the transformer model to distinguish between different elements in the sequence.

2. Masked Trajectory Modeling

2.1. Trajectory Datasets

`MTM` is designed to operate on trajectory datasets that we encounter in decision making domains. Taking the example of robotics, a trajectory comprised proprioceptive states, camera observations, control actions, task/goal commands, and so on. We can denote such a trajectory comprising of M different modalities as

$$\boldsymbol{\tau} = \left\{ \left(\mathbf{x}_1^1, \mathbf{x}_1^2, \dots \mathbf{x}_1^M \right), \; \dots \left(\mathbf{x}_T^1, \mathbf{x}_T^2, \dots \mathbf{x}_T^M \right) \right\}, \qquad (1)$$

where $\mathbf{x}_t^m$ refers to the m^{th} modality in the t^{th} timestep. In our empirical investigations, following prior work (Chen et al., 2021; Janner et al., 2021), we use state, action, and return-to-go (RTG) sequences as the different data modalities. Note that in-principle, our mathematical formulation is generic and can handle any modality.

2.2. Architecture and Masked Modeling

To perform masked trajectory modeling, we first "tokenize" the different elements in the raw trajectory sequence, by lifting them to a common representation space using modality-specific encoders. Formally, we compute

$$\mathbf{z}_t^m = E_\theta^m(\mathbf{x}_t^m) \quad \forall t \in [1, T], \; m \in [1, M],$$

where E_θ^m is the encoder corresponding to modality m. We subsequently arrange the embeddings in a 1-D sequence of length $N = M \times T$ as: $\boldsymbol{\tau} = \left(\mathbf{z}_1^1, \mathbf{z}_1^2, \dots \mathbf{z}_1^M, \dots \mathbf{z}_t^m, \dots \mathbf{z}_T^M \right)$.

The self-supervised learning task in `MTM` is to reconstruct the above sequence conditioned on a masked view of the same. We denote the latter with $\texttt{Masked}(\boldsymbol{\tau})$, where we randomly drop or "mask" a subset of elements in the sequence. The final self-supervised objective is given by:

$$\max_\theta \; \mathbb{E}_{\boldsymbol{\tau}} \sum_{t=1}^{T} \sum_{m=1}^{M} \log P_\theta \left(\mathbf{z}_t^m \middle| \texttt{Masked}(\boldsymbol{\tau}) \right), \qquad (2)$$

where P_θ is the prediction of the model. We train MTM with a random autoregressive mask, which is described in more detail in Appendix B.

Architecture and Embeddings We adopt an encoder-decoder architecture similar to He et al. (2021) and Liu et al. (2022), where both the encoder and decoder are bi-directional transformers. We use a modality-specific encoder to lift the raw trajectory inputs to a common representation space for tokens. Further, to allow the transformer to disambiguate between different elements in the sequence, a fixed sinusoidal timestep encoding and a learnable mode-specific encoding are added, as illustrated in Figure 2. The resulting sequence is then flattened and fed into the transformer encoder where only unmasked tokens are processed. The decoder processes the full trajectory sequence, and uses values from the encoder when available, or a mode-specific mask token when not. The decoder is trained to predict the original sequence, including the unmasked tokens, using an MSE loss (He et al., 2021). Additional model hyperparameters are detailed in Appendix C

2.3. `MTM` as a generic abstraction for RL

The primary benefit of `MTM` is its versatility. Once trained, the `MTM` network can take on different roles, by simply using different masking patterns at inference time. For example, `MTM` can be used as a stand-alone algorithm for offline RL, by utilizing a return-conditioned behavior cloning (RCBC) mask at inference time, analogous to DT (Chen et al., 2021). Alternatively, `MTM` can be used to recover various components that routinely feature in traditional RL pipelines, such as providing a state representation that accelerates the learning of traditional RL algorithms or acting as a world model for model-based RL algorithms.

3. Experiments

Through detailed empirical evaluations, we aim to study the following questions.

1. Is `MTM` a versatile learner? Can the same network trained with `MTM` be used for different capabilities without additional training?
2. Is `MTM` an effective heteromodal learner? Can it consume heteromodal datasets, like state-only and state-action trajectories, and effectively use such a dataset to improve performance?
3. Can `MTM` learn good state representations that accelerate downstream learning with standard RL algorithms?

To help answer the aforementioned questions, we draw upon a variety of continuous control tasks and datasets that leverage the MuJoCo simulator (Todorov et al., 2012). Specifically we use D4RL (Fu et al., 2020), Adroit (Rajeswaran

Table 1. **Evaluation of various MTM capabilities.** MTM refers to the model trained with the random autoregressive mask, and evaluated using the appropriate mask at inference time. S-MTM ("Specialized") refers to the model that uses the appropriate mask both during training and inference time. We also compare with a specialized MLP baseline trained separately for each capability. Note that higher is better for BC and RCBC, while lower is better for FD and ID. We find that MTM is often comparable or better than training on specialized masking patterns, or training specialized MLPs. We use a box outline to indicate that a single model was used for all the evaluations within it. The right most column indicates if MTM is comparable or better than S-MTM, and we find this to be true in most cases.

Domain	Dataset	Task	MLP	S-MTM (Ours)	MTM (Ours)	(MTM) $\gtrsim$ (S-MTM)?
D4RL Hopper	Medium Replay	BC	43.05 ± 6.10	31.51 ± 3.03	19.78 ± 4.04	✗
	Medium Replay	RCBC	91.57 ± 3.13	90.12 ± 2.61	92.32 ± 4.56	✓
	Medium Replay	ID	0.077 ± 0.009	0.156 ± 0.025	0.358 ± 0.087	✗
	Medium Replay	FD	1.018 ± 0.048	4.868 ± 0.119	0.521 ± 0.154	✓
Adroit Door	Medium Replay	BC	25.21 ± 6.20	24.41 ± 4.01	32.49 ± 4.75	✓
	Medium Replay	RCBC	58.96 ± 8.41	44.88 ± 20.11	60.73 ± 17.30	✓
	Medium Replay	ID	9.763 ± 0.031	0.604 ± 0.026	0.666 ± 0.009	✓
	Medium Replay	FD	2.199 ± 0.031	3.370 ± 1.009	2.442 ± 0.504	✓

et al., 2018), and ExORL (Yarats et al., 2022). Additional environment details can be found in Appendix D.

3.1. MTM Capabilities

We next study if MTM is a versatile learner by evaluating it across four different capabilities on Adroit and D4RL datasets. We test these capabilities for a single MTM-model (i.e. same weights) by simply altering the masking pattern during inference time. These capabilities are:

1. **Behavior Cloning (BC)**: Predict next action given state-action history.
2. **Return Conditioned Behavior Cloning (RCBC)** is similar to BC, but additionally conditions on the desired Return-to-Go. See Appendix F for additional offline RL results that leverage MTM's RCBC capability.
3. **Inverse Dynamics (ID)**, where we predict the action using the current and future desired state. This can be viewed as a 1-step goal-reaching policy.
4. **Forward Dynamics (FD)**, where we predict the next state given history and current action.

We consider two variations of MTM. The first variant, S-MTM, trains a specialized model for each capability using the corresponding masking pattern at *train time*. The second variant, denoted simply as MTM, trains a single model using the random autoregressive mask specified in Section 2. Subsequently, the same model (i.e. same set of weights) is evaluated for all the four capabilities. We also compare our results with specialized MLP models for each capability. We evaluate the best checkpoint across all models and report mean and standard deviation across 4 seeds, taking the average of 50 trajectory executions per seed. For all experiments we train on 95% of the dataset and reserve 5% of the data for evaluation. For BC and RCBC results, we report the normalized score obtained during evaluation rollouts. For ID and FD, we report normalized loss values on the aforementioned 5% held-out data.

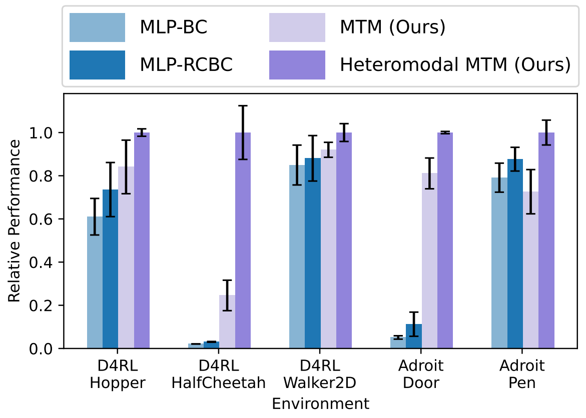
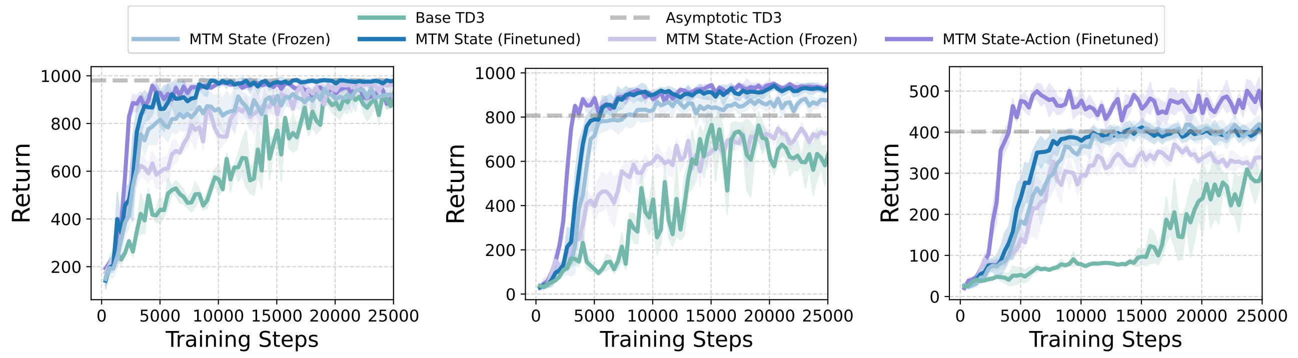


Figure 3. **MTM can effectively learn from heteromodal datasets.** Real world data may not always contain action labels. We simulate this setting by training a MTM models on Expert datasets across domains where only a small fraction of the data have action labels. Our Heteromodal MTM model is able to effectively improve task with the additional data over baseline MTM and MLP that train on only the subset of data with actions. Y-axis normalized with respect to performance of Heteromodal MTM.

A snapshot of our results are presented in Table 1. Please see Appendix for detailed results on all the environments. We find that MTM is comparable or even better than specialized masks, and also matches the performance of specialized MLP models. We suspect that specialized masks may require additional tuning of parameters to prevent overfitting or underfitting, whereas random autoregressive masking is more robust across tasks and hyperparameters.

3.2. Heteromodal Datasets

MTM is uniquely capable of learning from heteromodal datasets. This is enabled by the training procedure, where any missing data can be treated as if it were masked. During



(a) DM-Control Walker2D Stand Task. (b) DM-Control Walker2D Walk Task (c) DM-Control Walker2D Run Task

Figure 4. **State representations from MTM enable faster learning.** The plot visualizes an the walker agent's performance as it is trained using TD3 on different state representations across 3 tasks (stand, walk, run). The agent is trained completely offline using data from the ExORL dataset. The learning curves show that MTM state representations enable the agent to more quickly learn the task at hand, reaching or exceeding the asymptotic performance of TD3 on raw states. Both frozen MTM features and fine tuned MTM features are comparable in terms of learning speed and performance. In addition, we see that using state-action representations from MTM (as input to the critic of TD3) can also benefit learning. We also show the asymptotic performance reached by TD3 on raw states and actions after training for 100000 iterations and plot the average of 5 seeds.

training we apply the loss only to modes that exist in the dataset. For these experiments we take the `Expert` subset of our trajectory data and remove action labels from the majority of the dataset. The training data consists of 1% of the data with all modes (states, actions, return-to-go) and 95% percent of the data with no action labels. As is done in all experiments, the remainder is reserved for testing.

From our initial experiments, we found that naively adding in the state only data during training, and evaluating with the RCBC mask did not always result in improved performance, despite an improvement in forward dynamics prediction. Based on this observation, we propose a two-stage action inference procedure. First, we predict future states given current state and desired returns. This can be thought of as a forward dynamics pass where the desired returns are used instead of actions, which are masked out. Next, we predict actions using the current state and predicted future state using the inverse dynamics mask. We refer to this model trained on heteromodal data, along with the two stage inference procedure, as Heteromodal MTM. We present the results in Figure 3, where we find that Heteromodal MTM consistently improves performance over the baseline MLP and MTM that are trained only on the subset of data with action labels. We also provide sample efficiency trends of MTM and Heteromodal MTM in Appendix G.

3.3. State Representations of MTM

MTM can also be interpreted as an offline pre-training exercise to learn representations for downstream RL. To instantiate this in practice, we consider the setting of offline RL using TD3 on the ExORL dataset. The baseline method is to simply run TD3 on this dataset using the raw state as input to the TD3 algorithm. We compare this to our proposed approach of using MTM state representations for TD3. To do this, we pretrain an MTM model on state-action sequences in the ExORL dataset. Subsequently, to use the state representations, we simply use the MTM encoder to tokenize and encode each state. This representation of the state can be used in the place of raw states for the TD3 algorithm. We compare training TD3 on raw states to training TD3 with representation from (a) a frozen MTM model, and (b) finetuning the MTM model with the offline RL loss (i.e. TD3 objective). In addition, we also experiment with using MTM state-action representations for the critic of TD3.

Figure 4 depicts the learning curves for the aforementioned experiment. In all cases we see significant improvement in training efficiency by using MTM state representations – both in frozen mode and when finetuned. In the `Walk` task, we note it actually *improves* over the asymptotic performance of the base TD3 (Fujimoto et al., 2018a) algorithm within 10% of training budget. Additionally, we find that the state-action representation from MTM also provides significant benefits, with finetuned state-action representation leading to better asymptotic performance on the `Run` task.

4. Summary

In this paper, we introduced MTM as a versatile and effective approach for sequential decision making. We found that a single pretrained model (i.e. same weights) can be used for different downstream purposes like inverse dynamics, forward dynamics, imitation learning, offline RL, and representation learning. Future work includes incorporating training in online learning algorithms for more sample efficient learning, scaling MTM to longer trajectory sequences, and more complex modalities like videos.

References

Baker, B., Akkaya, I., Zhokhov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *ArXiv*, abs/2206.11795, 2022.

Bao, H., Dong, L., and Wei, F. Beit: Bert pre-training of image transformers. *ArXiv*, abs/2106.08254, 2021.

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

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Carroll, M., Paradise, O., Lin, J., Georgescu, R., Sun, M., Bignell, D., Milani, S., Hofmann, K., Hausknecht, M. J., Dragan, A. D., and Devlin, S. Unimask: Unified inference in sequential decision problems. *ArXiv*, abs/2211.10869, 2022.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. 02 2018a.

Fujimoto, S., Meger, D., and Precup, D. Off-Policy Deep Reinforcement Learning without Exploration. *CoRR*, abs/1812.02900, 2018b.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Hansen, N., Lin, Y., Su, H., Wang, X., Kumar, V., and Rajeswaran, A. Modem: Accelerating visual model-based reinforcement learning with demonstrations. *arXiv preprint*, 2022a.

Hansen, N., Wang, X., and Su, H. Temporal difference learning for model predictive control. In *ICML*, 2022b.

He, K., Chen, X., Xie, S., Li, Y., Doll'ar, P., and Girshick, R. B. Masked autoencoders are scalable vision learners. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15979–15988, 2021.

Janner, M., Li, Q., and Levine, S. Reinforcement learning as one big sequence modeling problem. *arXiv preprint arXiv:2106.02039*, 2021.

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2022a.

Jiang, Z., Zhang, T., Janner, M., Li, Y., Rocktäschel, T., Grefenstette, E., and Tian, Y. Efficient planning in a compact latent action space. *arXiv preprint arXiv:2208.10291*, 2022b.

Jolliffe, I. T. and Cadima, J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374, 2016.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. MOReL : Model-Based Offline Reinforcement Learning. In *NeurIPS*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. 2021.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-Learning for Offline Reinforcement Learning. *ArXiv*, abs/2006.04779, 2020.

Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. In *Reinforcement learning*, pp. 45–73. Springer, 2012.

Laskin, M., Yarats, D., Liu, H., Lee, K., Zhan, A., Lu, K., Cang, C., Pinto, L., and Abbeel, P. Urlb: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv:2110.15191*, 2021.

Liu, F., Liu, H., Grover, A., and Abbeel, P. Masked autoencoding for scalable and generalizable decision making. *ArXiv*, abs/2211.12740, 2022.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, 2019.

Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. Provably good batch off-policy reinforcement learning without great exploration. In *Neural Information Processing Systems*, 2020.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019.

Nair, S., Rajeswaran, A., Kumar, V., Finn, C., and Gupta, A. R3m: A universal visual representation for robot manipulation. *ArXiv*, abs/2203.12601, 2022.

Parisi, S., Rajeswaran, A., Purushwalkam, S., and Gupta, A. K. The unsurprising effectiveness of pre-trained vision models for control. In *ICML*, 2022.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A. D., Heess, N. M. O., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. A generalist agent. *ArXiv*, abs/2205.06175, 2022.

Schmidhuber, J. Reinforcement learning upside down: Don't predict rewards – just map them to actions, 2019.

Seo, Y., Hafner, D., Liu, H., Liu, F., James, S., Lee, K., and Abbeel, P. Masked world models for visual control. *ArXiv*, abs/2206.14244, 2022.

Shafiullah, N. M. M., Cui, Z. J., Altanzaya, A., and Pinto, L. Behavior transformers: Cloning k modes with one stone. *ArXiv*, abs/2206.11251, 2022.

Srivastava, R. K., Shyam, P., Mutz, F., Jaśkowski, W., and Schmidhuber, J. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa, Y. dm$_control : Software and tasks for continuous control, 2020. ISSN 2665-9638. URL$.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, 2008.

Xiao, T., Radosavovic, I., Darrell, T., and Malik, J. Masked visual pre-training for motor control. *ArXiv*, abs/2203.06173, 2022.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Reinforcement learning with prototypical representations. 2021.

Yarats, D., Brandfonbrener, D., Liu, H., Laskin, M., Abbeel, P., Lazaric, A., and Pinto, L. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. Combo: Conservative offline model-based policy optimization. In *NeurIPS*, 2021.

Zhou, A., Kumar, V., Finn, C., and Rajeswaran, A. Policy architectures for compositional generalization in control. *arXiv preprint arXiv:2203.05960*, 2022.

A. Related Work

Autoencoders and Masked Prediction. Autoencoders have found several applications in machine learning. The classical PCA (Jolliffe & Cadima, 2016) can be viewed as a linear autoencoder. Denoising autoencoders (Vincent et al., 2008) learn to reconstruct inputs from noise corrupted versions of the same. Masked autoencoding has found recent success in domains like NLP (Devlin et al., 2018; Brown et al., 2020) and computer vision (He et al., 2021; Bao et al., 2021). Our work explores the use of masked prediction as a self-supervised learning paradigm for RL.

Offline Learning for Control Our work primarily studies the offline setting for decision making, where policies are learned from static datasets. This broadly falls under the paradigm of offline RL (Lange et al., 2012). A large class of offline RL algorithms modify their online counterparts by incorporating regularization to guard against distribution shift that stems from the mismatch between offline training and online evaluation (Kumar et al., 2020; Kidambi et al., 2020; Fujimoto et al., 2018b; Yu et al., 2021; Liu et al., 2020). In contrast, our work proposes a generic self-supervised pre-training paradigm for decision making, where the resulting model can be directly repurposed for offline RL.

Self-Supervised Learning for Control The broad idea of self-supervision has been incorporated into RL in two ways. The first is self-supervised **data collection**, such as task-agnostic and reward-free exploration (Pathak et al., 2017; Laskin et al., 2021; Burda et al., 2018). The second is concerned with self-supervised **learning** for control, which is closer to our work. Prior works typically employ self-supervised learning to obtain state representations (Parisi et al., 2022; Nair et al., 2022; Xiao et al., 2022) or world models (Hafner et al., 2020; Hansen et al., 2022a;b; Seo et al., 2022), for subsequent use in standard RL pipelines. In contrast, `MTM` uses self-supervised learning to train a single versatile model that can exhibit multiple capabilities.

Transformers and Attention in RL Our work is inspired by the recent advances in AI enabled by transformers (Vaswani et al., 2017), especially in offline RL (Chen et al., 2021; Janner et al., 2021; Jiang et al., 2022b) and imitation learning (Reed et al., 2022; Shafiullah et al., 2022; Brohan et al., 2022; Jiang et al., 2022a; Zhou et al., 2022). Of particular relevance are works that utilize transformers in innovative ways beyond the standard RL paradigm. Decision Transformers and related methods (Schmidhuber, 2019; Srivastava et al., 2019; Chen et al., 2021) use return-conditioned imitation learning, which we also adopt in this work. However, in contrast to Chen et al. (2021) and Janner et al. (2021) who use next token prediction as the self-supervised task, we use a bi-directional masked prediction objective. This masking pattern enables the learning of versatile models that can take on different roles based on inference-time masking pattern.

Recently, Liu et al. (2022) and Carroll et al. (2022) explore the use of bi-directional transformers for RL. In contrast to Liu et al. (2022), we show that a single MTM model is capable of diverse capabilities without any specific finetuning or custom masking patterns. In contrast to Carroll et al. (2022), we study broader capabilities of our model on several high-dimensional control tasks. VPT (Baker et al., 2022) also tackles sequential decision making using transformers, focusing primarily on extracting action labels with a separate inverse dynamics model using semi-supervised learning. Furthermore, unlike prior work, we also demonstrate that our model has unique and favorable properties like data efficiency, heteromodality, and the capability to learn good state representations.

B. Masking Patterns

Intuitively, we can mask elements in the sequence randomly with a high enough mask ratio to make the self-supervised task difficult. This has found success in computer vision (He et al., 2021). We propose to use a variation of this – a random autoregressive masking pattern. This pattern requires at least one token in the input sequence to be autoregressive, meaning it must be predicted based only on previous tokens, and all future tokens are masked. This means the last element in each sampled trajectory segment is necessarily masked. See Figure B.1 for an illustration. We note that the autoregressive mask in our context is **not** using a causal mask in attention weights, but instead corresponds to masking at the input and output token level, similar to MAE.

In the case of computer vision and NLP, the entire image or sentence is often available at inference time. However, in the case of RL, the sequence data is generated as the agent interacts with the environment. As a result, at inference time, the model is forced to be causal (i.e. use only the past tokens). By using our random autoregressive masking pattern, the model both learns the underlying temporal dependencies in the data, as well as the ability to perform inference on past events. We find that this simple modification is helpful in most tasks we study.

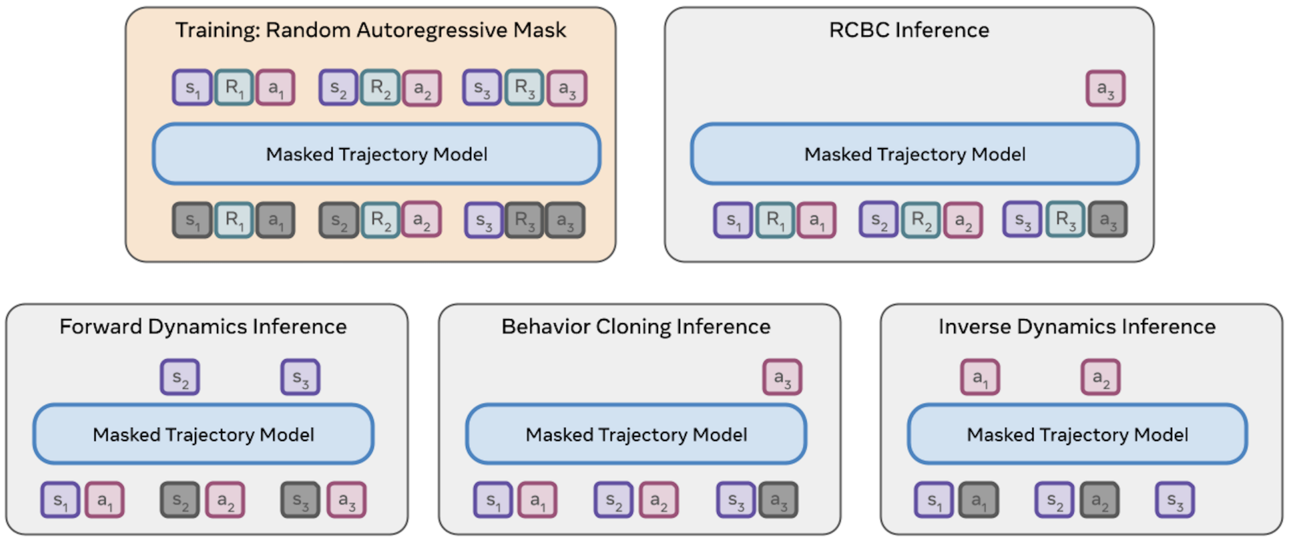

Training: Random Autoregressive Mask
RCBC Inference
Forward Dynamics Inference
Behavior Cloning Inference
Inverse Dynamics Inference
Masked Trajectory Model
s_1 R_1 a_1 s_2 R_2 a_2 s_3 R_3 a_3

Figure B.1. **Masking Pattern for Training and Inference.** (Training: box in orange) `MTM` is trained to reconstruct trajectory segments conditioned on a masked view of the same. We use a random autoregressive masking pattern, where elements in the input sequence are randomly masked, with the added constraint that at least one masked token must have no future unmasked tokens. This means the last element in the sequence must necessarily be masked. We note that the input sequence can start and end on arbitrary modalities. In this illustrated example, R_3 is the masked token that satisfies the autoregressive constraint. That is the prediction of R_3 is conditioned on no future tokens in the sequence. (Inference: boxes in gray) By changing the masking pattern at inference time, `MTM` can either be used directly for offline RL using RCBC (Chen et al., 2021), or be used as a component in traditional RL pipelines as a state representation, dynamics model, policy initialization, and more. These different capabilities are shown in gray. Modes not shown at the input are masked out and modes not shown at the output are not directly relevant for the task of interest.

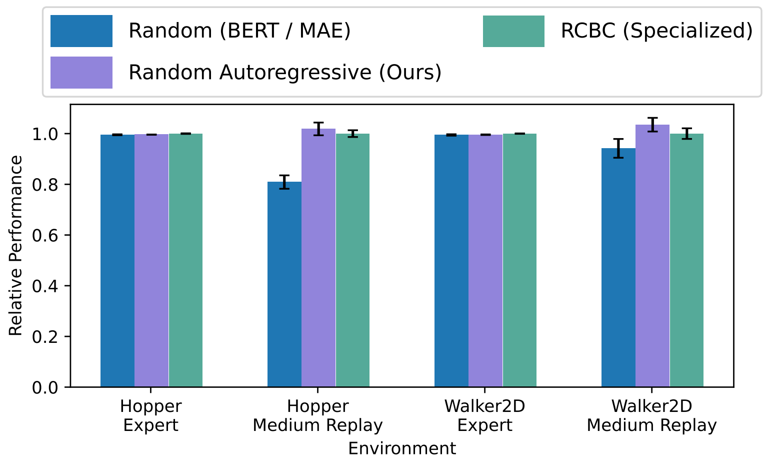


Figure B.2. **Impact of Masking Patterns.** This plot shows MTM RCBC performance trained with three different masking patterns, random, random autoregressive, and a specialized RCBC mask. We find that autoregressive random often outperforms random, and in most cases is even competitive with the specialized (or oracle) RCBC mask. Y-axis normalized with using RCBC mask.

We study if the masking pattern influences the capabilities of the learned model. Figure B.2 shows that random autoregressive masking matches or outperforms purely random masking on RCBC for a spread of environments for offline RL. We note that pure random masking, as done in MAE and BERT, which focuses on only learning good representations, can lead to diminished performance for downstream capabilities. Random autoregressive masking mitigates these issues by allowing the learning of a single versatile model while still matching or even exceeding the performance of specialized masks, as seen in Table 1.

C. Model and Training Details

C.1. MLP Baseline Hyperparameters

Table C.1. MLP Hyperparameters

	Hyperparameter	Value
MLP		
	Nonlinearity	GELU
	Batch Size	1024
	Embedding Dim	512
	# of Layers	2
Adam Optimizer		
	Learning Rate	0.0001
	Weight Decay	0.01
	Warmup Steps	40000
	Training Steps	140000
	Scheduler	cosine decay

C.2. MTM Model Hyperparameters

Table C.2. MTM Hyperparameters

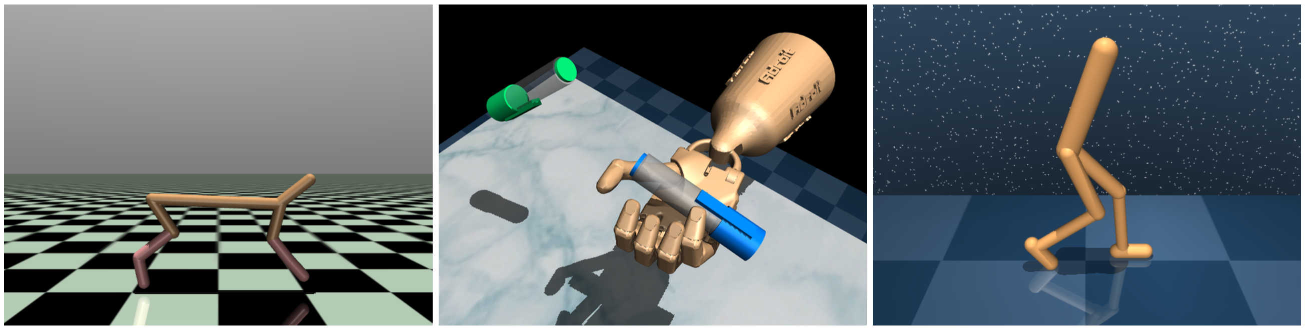
	Hyperparameter	Value
General		
	Nonlinearity	GELU
	Batch Size	1024
	Trajectory-Segment Length	4
	Scheduler	cosine decay
	Warmup Steps	40000
	Training Steps	140000
	Weight Decay	0.01
Bidirectional Transformer		
	# of Encoder Layers	2
	# Decoder Layers	1
	# Heads	4
	Embedding Dim	512
Mode Decoding Head		
	Number of Layers	2
	Embedding Dim	512
D4Rl Specific		
	Learning Rate	0.0001
	Mask Ratios	[0.7, 0.8, 0.85, 0.9, 0.95, 1.0]
Adroit Specific		
	Learning Rate	0.002
	Mask Ratios	[0.4, 0.5, 0.6, 0.7, 0.8, 0.85, 0.9, 0.95, 1.0]

C.3. MTM Training Details

In this section, we specify additional details of MTM for reproduction. Numerical values of the hyperparamters are found in table C.1. The architecture follows the structure of (He et al., 2021) and (Liu et al., 2022), which involves a bidirectional transformer encoder and a bidirectional transformer decoder. For each input modality there is a learned projection into the embedding space. In addition we add a 1D sinusoidal encoding for time step. The encoder only processes unmasked tokens. The decoder process the full trajectory sequence, replacing the masked out tokens with mode specific mask tokens. At the output of the decoder, we use a 2 Layer MLP. For training the model we use the AdamW optimizer (Kingma & Ba, 2017; Loshchilov & Hutter, 2019) with a warm up period and cosine learning rate decay.

As we rely on the generative capabilities of `MTM` which can be conditioned on a variety of different input tokens at inference time, we train `MTM` with a range of mask ratios that are randomly sampled. This is specified in the "Mask Ratios" in the hyperparameter. Our random autogressive masking scheme also requires that at least one token is predicted without future context. This is done by randomly sampling a time step and token, and masking out future tokens.

D. Additional Environment Details



(a) D4RL: HalfCheetah Task (b) Adroit: Pen Task (c) DM-Control: Walker2D Task

Figure D.1. **Continues Control evaluation settings.**

Here we provide additional details on each experiment setting. In general, our empirical evaluations are based on the standard versions of D4RL, Adroit, and ExORL. These benchmarks and setups are widely used in the community for studying various aspects of offline learning. The raw state space provided by these benchmarks typically comprise a mix of positions and velocities of different joints, bodies, and objects in the environment. We preprocess each dataset by normalizing the data before training.

Adroit (Rajeswaran et al., 2018) is a collection of dexterous manipulation tasks with a simulated five-fingered. Our `MTM` experiments use the `Pen`, and `Door` tasks. To match the setup of D4RL, we collect `Medium-Replay` and `Expert` trajectories for each task. This is done by training an expert policy. The `Expert` dataset comprises of rollout of the converged policy with a small amount of action noise. The `Medium-Replay` dataset is a collection of trajectory rollouts from various checkpoints during training of the expert policy, before policy convergence. The original Adroit environment provides a dense reward and a sparse measure of task completion. For `MTM` experiments, we use the task completion signal as an alternative to reward, which provides a more grounded signal of task performance (a measure of how many time steps in the episode is the task complete).

ExORL (Yarats et al., 2022) dataset consists of trajectories collected using various unsupervised exploration algorithms. ExORL leverages dm_control developed by Tunyasuvunakool et al. (2020). We use data collected by a ProtoRL agent (Yarats et al., 2021) in the Walker2D environment to evaluate the effectiveness of `MTM` representations on three different tasks: `Stand`, `Walk`, and `Run`. As the pretraining dataset has not extrinsic reward, `MTM` is trained with only states and actions. During downstream TD3 (Fujimoto et al., 2018a) learning, all trajectories are relabeled with the task reward.

E. Additional MTM Results

Table E.1. Evaluation of MTM capabilities on D4RL.

Domain	Dataset	Task	MLP	S-MTM (Ours)	MTM (Ours)
D4RL Hopper	Expert	BC	111.50 ± 0.14	111.20 ± 0.17	110.63 ± 0.34
	Expert	RCBC	111.75 ± 0.09	111.37 ± 0.19	111.01 ± 0.13
	Expert	ID	0.004 ± 0.001	0.008 ± 0.001	0.028 ± 0.006
	Expert	FD	0.063 ± 0.001	0.939 ± 0.122	0.026 ± 0.005
D4RL Hopper	Medium Expert	BC	60.61 ± 5.67	82.09 ± 17.35	53.82 ± 1.17
	Medium Expert	RCBC	111.91 ± 0.15	111.23 ± 0.31	110.95 ± 0.34
	Medium Expert	ID	0.008 ± 0.001	0.010 ± 0.001	0.043 ± 0.013
	Medium Expert	FD	0.102 ± 0.005	1.094 ± 0.181	0.039 ± 0.019
D4RL Hopper	Medium	BC	59.06 ± 0.19	58.53 ± 5.58	55.58 ± 2.05
	Medium	RCBC	67.96 ± 2.73	61.62 ± 1.00	64.07 ± 4.61
	Medium	ID	0.015 ± 0.003	0.018 ± 0.001	0.085 ± 0.019
	Medium	FD	0.158 ± 0.023	1.500 ± 0.289	0.070 ± 0.024
D4RL Hopper	Medium Replay	BC	43.05 ± 6.10	31.51 ± 3.03	19.78 ± 4.04
	Medium Replay	RCBC	91.57 ± 3.13	90.12 ± 2.61	92.32 ± 4.56
	Medium Replay	ID	0.077 ± 0.009	0.156 ± 0.025	0.358 ± 0.087
	Medium Replay	FD	1.018 ± 0.048	4.868 ± 0.119	0.521 ± 0.154
D4RL Walker2D	Expert	BC	109.15 ± 0.12	109.07 ± 0.29	108.94 ± 0.32
	Expert	RCBC	110.32 ± 0.15	109.01 ± 0.38	109.07 ± 0.26
	Expert	ID	0.009 ± 0.002	0.020 ± 0.003	0.108 ± 0.036
	Expert	FD	0.079 ± 0.008	1.913 ± 0.563	0.066 ± 0.030
D4RL Walker2D	Medium Expert	BC	109.06 ± 0.15	108.17 ± 0.46	92.78 ± 15.57
	Medium Expert	RCBC	110.53 ± 0.22	110.00 ± 0.58	109.53 ± 0.30
	Medium Expert	ID	0.018 ± 0.006	0.020 ± 0.014	0.202 ± 0.089
	Medium Expert	FD	0.108 ± 0.018	0.388 ± 0.133	0.144 ± 0.075
D4RL Walker2D	Medium	BC	74.26 ± 0.58	73.17 ± 0.93	60.42 ± 10.64
	Medium	RCBC	78.00 ± 1.44	75.11 ± 1.75	76.95 ± 0.51
	Medium	ID	0.022 ± 0.005	0.029 ± 0.010	0.150 ± 0.068
	Medium	FD	0.122 ± 0.014	0.390 ± 0.051	0.116 ± 0.090
D4RL Walker2D	Medium Replay	BC	29.75 ± 2.83	30.80 ± 2.74	23.33 ± 8.85
	Medium Replay	RCBC	77.08 ± 2.10	73.34 ± 1.31	78.31 ± 4.08
	Medium Replay	ID	0.452 ± 0.100	0.630 ± 0.228	1.407 ± 0.452
	Medium Replay	FD	1.814 ± 0.868	1.291 ± 0.407	0.634 ± 0.261
D4RL HalfCheetah	Expert	BC	93.28 ± 0.20	92.94 ± 0.63	93.23 ± 0.07
	Expert	RCBC	91.41 ± 0.54	93.58 ± 0.49	93.18 ± 0.42
	Expert	ID	0.001 ± 0.000	0.005 ± 0.001	0.005 ± 0.001
	Expert	FD	0.010 ± 0.001	0.301 ± 0.062	0.003 ± 0.001
D4RL HalfCheetah	Medium Expert	BC	76.15 ± 1.82	71.16 ± 4.00	59.81 ± 5.19
	Medium Expert	RCBC	79.66 ± 2.53	93.76 ± 0.24	91.57 ± 0.83
	Medium Expert	ID	0.001 ± 0.000	0.003 ± 0.001	0.014 ± 0.005
	Medium Expert	FD	0.018 ± 0.007	0.200 ± 0.083	0.010 ± 0.004
D4RL HalfCheetah	Medium	BC	42.84 ± 0.10	42.87 ± 0.08	42.06 ± 0.34
	Medium	RCBC	44.39 ± 0.14	43.63 ± 0.25	43.34 ± 0.31
	Medium	ID	0.000 ± 0.000	0.003 ± 0.001	0.030 ± 0.043
	Medium	FD	0.020 ± 0.001	0.454 ± 0.162	0.011 ± 0.007
D4RL HalfCheetah	Medium Replay	BC	36.78 ± 0.45	36.40 ± 1.06	31.32 ± 8.48
	Medium Replay	RCBC	40.27 ± 0.25	41.29 ± 0.36	42.03 ± 0.50
	Medium Replay	ID	0.003 ± 0.001	0.004 ± 0.001	0.039 ± 0.013
	Medium Replay	FD	0.099 ± 0.050	0.187 ± 0.037	0.026 ± 0.009

Table E.2. Evaluation of MTM capabilities on Adroit.

Domain	Dataset	Task	MLP	S-MTM (Ours)	MTM (Ours)
Adroit Door	Expert	BC	148.20 ± 0.08	144.32 ± 2.01	144.40 ± 3.51
	Expert	RCBC	149.19 ± 0.33	147.30 ± 0.54	149.25 ± 1.91
	Expert	ID	3.801 ± 0.004	0.426 ± 0.020	0.562 ± 0.036
	Expert	FD	0.072 ± 0.001	1.227 ± 0.652	0.880 ± 0.321
Adroit Door	Medium Replay	BC	25.21 ± 6.20	24.41 ± 4.01	32.49 ± 4.75
	Medium Replay	RCBC	58.96 ± 8.41	44.88 ± 20.11	60.73 ± 17.30
	Medium Replay	ID	9.763 ± 0.031	0.604 ± 0.026	0.666 ± 0.009
	Medium Replay	FD	2.199 ± 0.031	3.370 ± 1.009	2.442 ± 0.504
Adroit Pen	Expert	BC	60.08 ± 1.95	53.09 ± 2.24	47.23 ± 10.02
	Expert	RCBC	63.97 ± 1.84	55.62 ± 4.13	59.22 ± 0.94
	Expert	ID	13.247 ± 0.026	0.503 ± 0.182	1.172 ± 0.199
	Expert	FD	0.047 ± 0.001	0.787 ± 0.203	0.436 ± 0.019
Adroit Pen	Medium Replay	BC	30.55 ± 4.86	53.55 ± 6.50	43.27 ± 3.21
	Medium Replay	RCBC	38.71 ± 3.36	44.41 ± 3.58	53.80 ± 1.20
	Medium Replay	ID	0.966 ± 0.002	0.174 ± 0.157	0.034 ± 0.004
	Medium Replay	FD	0.679 ± 0.047	0.586 ± 0.059	0.472 ± 0.070

Table F.1. **Results in D4RL locomotion tasks.** We find that MTM outperforms RvS and DT, which also use RCBC for offline RL.

Environment	Dataset	BC	CQL	IQL	TT	MOPO	RsV	DT	**MTM (Ours)**
HalfCheetah	Medium-Replay	36.6	45.5	44.2	41.9	42.3	38.0	36.6	42.0
Hopper	Medium-Replay	18.1	95.0	94.7	91.5	28.0	73.5	82.7	92.3
Walker2d	Medium-Replay	26.0	77.2	73.9	82.6	17.8	60.6	66.6	78.3
HalfCheetah	Medium	42.6	44.0	47.4	46.9	53.1	41.6	42.0	43.3
Hopper	Medium	52.9	58.5	66.3	61.1	67.5	60.2	67.6	64.1
Walker2d	Medium	75.3	72.5	78.3	79.0	39.0	71.7	74.0	77.0
HalfCheetah	Medium-Expert	55.2	91.6	86.7	95.0	63.7	92.2	86.8	91.6
Hopper	Medium-Expert	52.5	105.4	91.5	110.0	23.7	101.7	107.6	110.9
Walker2d	Medium-Expert	107.5	108.8	109.6	101.9	44.6	106.0	108.1	109.5
Average		51.9	77.6	77.0	78.9	42.2	71.7	74.7	78.8

F. Offline RL results

We test the capability of MTM to learn policies in the standard offline RL setting. To do so, we train MTM with the random autoregressive masking pattern as described in Section 2. Subsequently, we use the Return Conditioned Behavior Cloning (RCBC) mask at inference time for evaluation. This is inspired by DT (Chen et al., 2021) which uses a similar RCBC approach, but with a GPT model.

Our empirical results are presented in Table F.1. We find that MTM outperforms the closest algorithms of DT and RvS, suggesting that masked prediction is an effective pre-training task for offline RL when using RCBC inference mask. More surprisingly, MTM is competitive with highly specialized and state-of-the-art offline RL algorithms like CQL (Kumar et al., 2020) and IQL (Kostrikov et al., 2021) despite training with a purely self-supervised learning objective without any explicit RL components.

G. Data Efficiency

We find that `MTM` is able to achieve higher performance than baseline (specialized) MLPs in the low data regimes. To explicitly test the data efficiency of `MTM`, we study the performance as a function of the training dataset size, and present results in Figure G.1. We observe that `MTM` is more sample efficient and achieves higher performance for any given dataset size. Heteromodal `MTM` also outperforms `MTM` throughout, with the performance gap being quite substantial in the low-data regime. We hypothesize that the data efficiency of `MTM` is due to better usage of the data. Specifically, since the model encounters various masks during training, it must learn general relationships between different elements. As a result, `MTM` may be able to squeeze out more learning signal from any given trajectory.

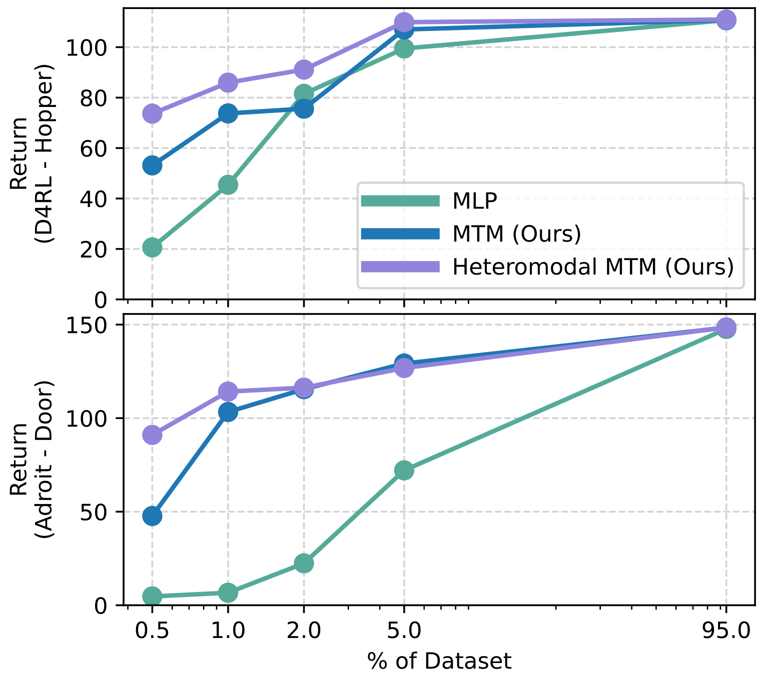


Figure G.1. **Dataset efficiency.** We train `MTM` in the D4RL Hopper and Adroit Door environments across a range of dataset sizes, measured by the percent of the original dataset ($\approx$ 1 million transitions). We see that `MTM` is able to consistently outperform specialized MLP models in the low data regime. Furthermore, we see that Heteromodal `MTM` (i.e. `MTM` trained on heteromodal data containing both state-only and state-action trajectories) is further able to provide performance improvement in low data regimes.