MS-LSDT: Multi-Scale Long–Short Goal-Conditioned Decision Transformers for Offline Surgical Robot Learning in SurRoL

Zhaoye Wu[1], Chenxi Huang[1], Wei Zhou[2,3], Yunxiang Yang[2,3,4], Yujun Chen[2,3], Yulong Fu[2,3], Zhaoan Yang[2,3], Kunzhai Huang[5(✉)], and Qinglang Guo[2,3,4(✉)]

[1] School of Informatics, Xiamen University, Xiamen, China 361004
wuzhaoye@xmu.stu.edu.cn, supermonkeyxi@xmu.edu.cn
[2] Jiangsu Key Laboratory of Digital Finance, Suzhou, China 214133
zhouwei@tgqs.net, gql1993@mail.ustc.edu.cn
[3] Yangtze Delta Industrial Innovation Center Of Quantum Science And Technology, Suzhou, China 214133
[4] China Academy of Electronics and Information Technology, No. 11 Shuangyuan Road, Shijingshan District, Beijing 100041, China
[5] Department of General Surgery, The First Affiliated Hospital of Xiamen University, Xiamen, China
39168101@qq.com

Abstract. Offline reinforcement learning (offline RL) is appealing for surgical robotics because online exploration is expensive and safety-critical. Recent Decision Transformers (DTs) recast offline RL as sequence modeling. A goal-conditioned GPT-based DT framework further improves long-horizon surgical automation via goal-aware future indicators and cross-task pretraining. This paper extends that pipeline with a stronger temporal backbone, MS-LSDT: a multi-scale long–short architecture that fuses one causal self-attention branch (long-range reasoning) with multiple causal depthwise-convolution branches (short-range dynamics) via token-wise soft gating. We keep the code-aligned tokenization and auxiliary objectives (returns-to-goal and time-to-goal conditioning, forward dynamics prediction, and optional masked reconstruction). We adopt a two-stage learning protocol: cross-task pretraining across ten SurRoL tasks followed by single-task fine-tuning and online rollout evaluation for validation. We report code-aligned multi-seed results and controlled ablations under a fixed evaluation protocol.

Keywords: Offline reinforcement learning · Decision Transformer · Surgical robotics · Multi-scale modeling · SurRoL

1 Introduction

Surgical robot task automation is a promising direction to improve surgical efficiency and consistency, but collecting large-scale on-policy data is costly and constrained by safety. Offline RL aims to learn a control policy from previously collected trajectories

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2027
D.-S. Huang et al. (Eds.): ICIC 2026, LNCS 16656, pp. 121–133, 2027.
https://doi.org/10.1007/978-981-92-3441-7_11

without further environment interaction, which is especially attractive in healthcare settings. However, surgical manipulation typically involves long-horizon, goal-conditioned behavior with phase-dependent dynamics (approach, contact, manipulation, release), making offline learning challenging.

From a methodological perspective, offline RL spans imitation learning, value-regularized learning, and sequence modeling. Behavior cloning can be strong with high-quality demonstrations, while value-based offline RL methods such as BCQ [1], BEAR [2], BRAC [3], CQL [4], IQL [4], and TD3 + BC [6] regularize or constrain learning to reduce out-of-distribution action errors. Advantage-weighted and model-based variants such as AWR [7], AWAC [8], MOPO [9], and MOReL [10] further broaden the offline RL design space. Despite progress, long-horizon surgical tasks remain difficult because sparse success signals, contact-rich dynamics, and compounding errors require temporal reasoning over multi-phase procedures.

Decision Transformer (DT) [11] reframes RL as conditional sequence modeling using causal Transformers [12], enabling strong performance without explicit value-function learning. In surgical simulation, Fu et al. propose a multi-objective cross-task goal-conditioned GPT-based decision transformer that introduces goal-aware future indicators and a cross-task pretraining paradigm [13]. Their results on SurRoL [14] indicate that DT-style models can be competitive for surgical task automation when paired with suitable tokenization and auxiliary objectives.

We target long-horizon, goal-conditioned surgical manipulation in SurRoL, where trajectories mix rapid contact transitions with slower task-level progress. In this setting, small local errors can quickly compound and break downstream phases.

While DT-style policies provide a clean sequence-modeling interface, a vanilla Transformer backbone relies on a single mechanism (global self-attention) to simultaneously model local dynamics and long-range dependencies. In our experiments, this can be inefficient and brittle when contact-rich micro-dynamics dominate short windows but long context is still required.

We introduce **MS-LSDT**, a multi-scale long–short backbone that keeps causal attention for global context and adds multiple causal depthwise-convolution branches to capture short-range dynamics at different kernel sizes $k \in \{3, 7, 14\}$. A token-wise soft gate fuses branches adaptively per token. Crucially, we integrate MS-LSDT into an existing goal-conditioned DT pipeline (tokenization, hindsight relabeling, multi-objective pretraining, and evaluation conditioning), so architectural changes are isolated and comparisons are code-identical.

The paper makes three code-aligned contributions: *Code-aligned pipeline:* exact goal-conditioned DT tokenization, normalization, rollout-time conditioning, and multi-objective pretraining losses. *MS-LSDT backbone:* a multi-scale long–short block (causal attention + multi-kernel causal depthwise convolutions + token-wise gating) with implementation-level formulation. *Two-stage protocol:* cross-task pretraining plus per-task fine-tuning consistent with [13] for controlled backbone comparisons.

2 Related Work

2.1 Offline RL and Imitation Learning

Offline RL must address distribution shift and extrapolation error. Representative methods include behavior-regularized approaches such as BCQ [1], BEAR [2], and BRAC [3], conservative objectives such as CQL [4], value-learning variants such as IQL [4], and practical hybrids such as TD3 + BC [6]. Advantage-weighted learning (AWR [7], AWAC [8]) and model-based offline RL (MOPO [9], MOReL [10]) are also relevant, while benchmarks and toolkits such as D4RL [14], RoboMimic [16], and offline RL surveys [17] support reproducible evaluation. Surgical manipulation still stresses these methods through sparse rewards and contact-dependent transitions.

2.2 Transformers for Decision Making

DT [11] uses a causal Transformer to predict actions conditioned on desired returns, offering a unifying sequence modeling view of offline RL. Goal-conditioned extensions and multi-objective training can improve long-horizon reasoning for robotic manipulation [13]. Architecturally, adding locality bias via convolutional branches or hybrid blocks is a common strategy to model short-term dynamics efficiently. This idea has appeared in several sequence and vision models: Conformer inserts convolution into Transformer blocks for speech recognition [18], CvT introduces convolutional projections into vision Transformers [19], and CoAtNet studies stacked convolution–attention hybrids across data scales [20]. Dynamic/lightweight convolution designs [21] and depthwise separable convolutions [22] motivate our efficient local branches.

2.3 Goal-Conditioned Learning and Relabeling

Goal-conditioned policies and goal relabeling are effective for sparse-reward tasks. Hindsight Experience Replay (HER) [23] relabels goals with achieved states to densify learning signals. Our dataset preprocessing applies a HER-like hindsight goal relabeling to propagate successful sub-trajectories and provide richer supervision for sequence models.

2.4 Surgical Robot Learning in Simulation

SurRoL [14] provides a dVRK-compatible simulation platform and task suite for learning surgical manipulation skills. Fu et al. [13] demonstrate DT-based cross-task learning on ten SurRoL tasks and report deployment on real dVRK for validation. We follow this simulator setting and focus on code-level reproducibility and architectural improvement of the backbone.

2.5 Goal-Conditioned Offline Learning

We consider goal-conditioned tasks with state s_t, goal g_t, action a_t, and sparse rewards. We are given an offline dataset of trajectories collected in simulation, and we learn a policy without collecting additional training data. Following [13], our pipeline uses two goal-aware future indicators: **returns-to-goal (RTG)**, a shaped scalar indicating distance to success in time steps, and **time-to-goal (TTG)**, a countdown-like scalar for the remaining steps. These indicators provide a lightweight summary of progress and can be used both as conditioning tokens and as auxiliary prediction targets.

2.6 Goal-Conditioned Decision Transformer

DTs model action prediction as autoregressive sequence modeling [11]. In our implemented goal-conditioned DT, each time step includes a set of tokens:

$$x_t = \left[R_t, \Delta t_t, s_t, g_t, a_t\right],$$

where R_t is returns-to-goal and Δt_t is time-to-goal. In our implementation, we project each modality to a shared embedding dimension d and add a timestep embedding:

$$e_t^{(m)} = f_m\left(x_t^{(m)}\right) + E_{\text{time}}(t), \qquad m \in \{\text{rtg}, \text{ttg}, s, g, a\},$$

where f_m is a modality-specific linear layer (and E_{time} is a learned embedding table). The per-timestep tokens are then *stacked* in the order $[R, \Delta t, s, g, a]$ into $X \in \mathbb{R}^{B \times T \times d}$, where L is the padded trajectory length, B is batch size, and $T = L \cdot n_{\text{tok}}$ with $n_{\text{tok}} = 4$. We apply a LayerNorm to the stacked embeddings before the backbone. The causal backbone then produces contextualized hidden states $H \in \mathbb{R}^{B \times T \times d}$. To preserve autoregressive semantics, we predict a_t from the contextual representation at the goal token position (i.e., the token immediately before action at timestep t in the stacked order).

3 Method

3.1 Code-Aligned Indicators and Hindsight Relabeling

SurRoL tasks are sparse and goal-conditioned. In our dataset builder, we label each step as successful if the achieved goal is within a small threshold of the desired goal, otherwise we assign a negative reward. Given a successful time index $t^\star$, we define:

$$\Delta t_t = \begin{cases} t^\star - t, & t \le t^\star, \\ 0, & t > t^\star, \end{cases} \qquad R_t = \begin{cases} -(t^\star - t), & t \le t^\star, \\ 0, & t > t^\star, \end{cases}$$

and for failed trajectories we set $R_t = \ -T$ and $\Delta t_t = T$, where T is the maximum sequence length. This choice is consistent with the logics of "remaining progress" and matches the data processing in our code.

Normalization. We normalize observation and goal by dataset statistics. We also rescale the two indicators by the maximum sequence length:

$$\bar{s}_t = s_t - \mu_s/\sigma_s, \bar{g}_t = g_t - \mu_g/\sigma_g, \overline{R}_t = R_t/T, \overline{\Delta t}_t = \Delta t_t/T.$$

To increase the fraction of successful supervision, we apply a HER-like hindsight propagation: we sample indices $\widetilde{t}$ and replace the desired goal g_t for $t \geq \widetilde{t}$ with the achieved goal at that time, turning the suffix into a successful sub-trajectory [23]. This substantially increases the number of labeled sequences used for training.

3.2 Multi-Objective Training Loss

Our default training objective is masked action regression:

$$\mathcal{L}_{\text{act}} = \mathbb{E}\left[\sum_t m_t \|\hat{a}_t - a_t\|_2^2\right],$$

where $m_t \in \{0, 1\}$ masks padded steps. For cross-task pretraining, we also enable multi-objective supervision following [13]. **Forward dynamics** predicts the next observation from a masked input that keeps only state and action tokens:

$$\mathcal{L}_{\text{dyn}} = \mathbb{E}\left[\sum_t m_t \|\hat{s}_t - s_t\|_2^2\right].$$

Time-to-goal prediction drops the RTG token and regresses to Δt_t:

$$\mathcal{L}_{\text{ttg}} = \mathbb{E}\left[\sum_t m_t \|\hat{\Delta t}_t - \Delta t_t\|_2^2\right].$$

Masked action reconstruction optionally masks the action token with probability $p = 0.01$ and reconstructs it from context:

$$\mathcal{L}_{\text{rec}} = \mathbb{E}\left[\sum_t m_t \|\hat{a}_t^{(\text{rec})} - a_t\|_2^2\right].$$

In our code, these auxiliary losses are combined with fixed weights:

$$\mathcal{L} = \mathcal{L}_{\text{act}} + 0.1\ \mathcal{L}_{\text{dyn}} + 0.1\ \mathcal{L}_{\text{ttg}} + 0.1\ \mathbb{I}_{\text{rec}}\ \mathcal{L}_{\text{rec}},$$

where $\mathbb{I}_{\text{rec}} \in \{0, 1\}$ indicates whether reconstruction is enabled.

3.3 MS-LSDT Backbone

The core change in this work is the backbone. Instead of a pure Transformer block, we use a long–short hybrid with multi-scale local branches. Let $X \in \mathbb{R}^{B \times T \times d}$ denote the token embeddings (Fig. 1).

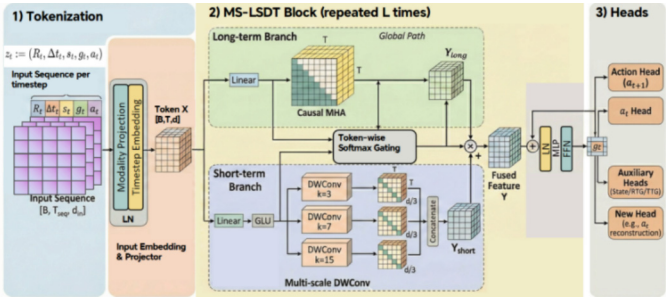


Fig. 1. MS-LSDT architecture. Stacked per-step tokens [R, Delta t, s, g, a] are processed by long-short blocks that fuse causal attention with causal depthwise-convolution branches (k = {3,7,14}) via token-wise softmax gating.

3.3.1 Long-Range Branch (Causal Attention)

We use standard causal multi-head self-attention with h heads and head dimension $d_h = d/h$. With learned projections $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$, we compute

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

reshape into heads, and apply a strictly lower-triangular causal mask M:

$$Y_{\text{long}} = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O, \quad \text{head}_i = \text{softmax}\left(\frac{Q_i K_i^\top}{\sqrt{d_h}} + M\right)V_i.$$

Padding Behavior. In our backbone implementation, given a binary padding mask $p \in \{0,1\}^{B \times T}$, we apply multiplicative zeroing $X \leftarrow X \odot p$ (with broadcast to $\mathbb{R}^{B \times T \times d}$) before and after each block, rather than adding padding logits to the attention matrix. This detail is important for reproducibility and matches the current code.

3.3.2 Short-Range Branches (Multi-Kernel Causal Depthwise Conv)

We use multiple causal depthwise 1D convolution branches with kernel sizes $k \in \mathcal{K}$:

$$Y_{\text{short}}^{(k)} = \text{CausalDWConv}_k(X), \qquad k \in \mathcal{K}.$$

Depthwise convolution provides an efficient locality bias with low parameter overhead [22]. We use left padding of $(k-1)$ and a depthwise Conv1D with groups$= d$ to preserve autoregressive consistency.

3.3.3 Token-Wise Soft Gating Fusion

We stack all branches (one long + multiple short) and fuse them by a token-wise softmax gate:

$$\alpha_t = \text{softmax}\big(W_g u_t + b_g\big) \in \mathbb{R}^{1+|\mathcal{K}|}, \qquad y_t = \sum_{b=1}^{1+|\mathcal{K}|} \alpha_{t,b} y_{t,b}.$$

where $u_t \in \mathbb{R}^d$ denotes the token feature at position t in the stacked sequence X, $W_g \in \mathbb{R}^{(1+|\mathcal{K}|)\times d}$, and $b_g \in \mathbb{R}^{1+|\mathcal{K}|}$. In the default setting $\mathcal{K} = \{3, 7, 14\}$, the gate outputs four weights per token: one for the attention branch and one for each local branch. This gate allows the model to emphasize local branches during contact-rich segments while falling back to global attention when long-range context matters. The fused output is linearly projected back to d, followed by residual connections, layer normalization, and an MLP:

$$Z = \mathrm{LN}(X + \mathrm{Proj}(Y)), \qquad \mathrm{Out} = \mathrm{LN}(Z + \mathrm{MLP}(Z)).$$

3.3.4 Complexity

For token length T and hidden size d, causal attention costs $\mathcal{O}(T^2 d)$ per layer. The convolution branches cost approximately $\mathcal{O}(|\mathcal{K}|\ T\ k_{\max}\ d)$ where $k_{\max} = \max \mathcal{K}$. In our default setting $k_{\max} \ll T$, so locality can be added with modest overhead. With $d = 128$, $L = 100$ environment steps, $T = 4L = 400$ stacked tokens, eight layers, and $\mathcal{K} = \{3, 7, 14\}$, the additional branch/gate/fusion parameters are about 20.6 K per block (164.9 K over eight layers), compared with about 196.6 K parameters for a standard DT block with attention and a $4d$ MLP. The corresponding per-layer multiply-add estimate is about 162.3 M for the DT block and 10.0 M additional operations for MS-LSDT, i.e., roughly 6.2% extra per layer under this sequence length.

3.4 Two-Stage Cross-Task Training

We follow the two-stage protocol in [13] and align with our implementation: **cross-task pretraining** shares one backbone across ten tasks while keeping task-specific embedding and prediction heads; task batches are shuffled, with one dataset pass per task per pretrain iteration. **Single-task fine-tuning** then loads the pretrained backbone (and optionally task heads) and trains on each target task with repeated dataset passes per iteration. This setup enables task-agnostic sequential reasoning while allowing per-task adaptation of input/output modules.

4 Experiments

4.1 Tasks and Observation Space

We evaluate on the ten SurRoL state-based tasks used by [13]: NeedleReach, GauzeRetrieve, NeedlePick, PegTransfer, NeedleRegrasp, BiPegTransfer, BiPegBoard, MatchBoardPanel, PickAndPlace, and MatchBoard. Each environment provides goal-conditioned observations in dictionary form (observation, achieved goal, desired goal).

4.2 Offline Dataset and Preprocessing

For each task, we use demonstration datasets stored as NumPy archives (e.g., data_PegTransferRL-v0_*.npz), with 200 trajectories per task. During loading, we compute R_t and Δt_t as in Eq. [eq:rtg_ttg], normalize states/goals by dataset statistics, and apply hindsight propagation with 40 relabeled samples per original trajectory. Sequences are padded/truncated to a maximum length of 100 with a binary mask. Our training flow has three stages: (i) offline data preprocessing with hindsight relabeling and tokenization, (ii) cross-task pretraining of a shared backbone with task-specific heads, and (iii) target-task fine-tuning followed by online rollout evaluation for validation. This text-form pipeline is identical to the implementation and is kept fixed across all backbone comparisons.

4.3 Baselines

We compare our MS-LSDT against a vanilla DT baseline under the same goal-conditioned DT pipeline. In addition, to contextualize performance against a broader set of recent methods, we include the main comparison results reported by Fu et al. on SurRoL (Table 1 in [13]). The compared rows are DT [11], MS-LSDT (ours), and the reported methods in [13]: PLAS, IQL, BC, VINN, DDPGBC, AMP, CoL, AWAC, DEX, ViSkill*, and T-STAR*. Unless otherwise stated, our DT baseline and MS-LSDT share the same tokenization, dataset preprocessing, optimizer, and training schedule.

4.4 Training and Evaluation Protocol

We use AdamW with learning rate 1×10^{-4} and weight decay 1×10^{-4}. The reported results use n_iters $= 200$ and train_repeats $= 40$ for fine-tuning. At each fine-tuning iteration, we perform online rollout evaluation and report: success rate, mean episode return, and mean episode length.

At inference time, the evaluator maintains target returns-to-goal and time-to-goal sequences and updates them step-by-step during rollout; this conditioning scheme is kept fixed across backbones. The code-aligned hyperparameters are: hidden size/layers/heads =128/8/4, maximum sequence length 100, token layout $[R_t, \Delta t_t, s_t, g_t, a_t]$, MS-LSDT kernels $\mathcal{K} = \{3, 7, 14\}$, dropout 0.1, maximum positions 1024, AdamW with learning rate and weight decay 1×10^{-4}, fine-tuning schedule n_iters $= 200$, train_repeats $=$ 40, and 40 evaluation episodes.

Table 1. Analytical efficiency comparison for the default backbone setting. We use d = 128, L = 100, stacked token length T = 4 L = 400, eight layers, and K = {3,7,14}.

Item	DT block	MS-LSDT extra over DT
Parameters / block	196.6 K	20.6 K (10.4%)
Parameters / 8 layers	1.47 M	164.9 K
Multiply-adds / layer	162.3 M	10.0 M (6.2%)

5 Results and Analysis

We report SurRoL success rates for a broad set of baselines (transcribed from Table 1 in [13]) together with our MS-LSDT runs under the same task list and evaluation metrics. Because only aggregate mean/std. statistics are available in this paper package, we treat the comparisons as descriptive rather than as formal significance claims.

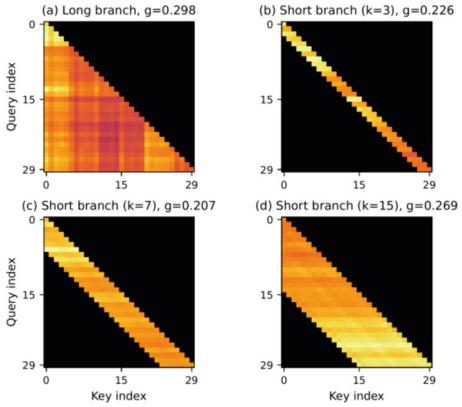


Fig. 2. Scale-wise effective context maps from saved MS-LSDT checkpoints (PegTransfer, epochs 2/6/10), combining token-wise gates with branch-specific context kernels (Table 3).

Table 2. Success rate (mean ± std) on ten SurRoL tasks. Rows other than MS-LSDT are reported results transcribed from Table 1 in [7]; MS-LSDT values are our runs under the same task set.

Method	NeedleReach	GauzeRetrieve	NeedlePick	PegTransfer	NeedleRegrasp
dim(O|G|A)	R7|R3|R4	R19|R3|R4	R19|R3|R4	R19|R3|R4	R34|R3|R10
PLAS	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.16 ± 0.08	0.00 ± 0.00
IQL	1.00 ± 0.00	0.40 ± 0.00	0.04 ± 0.04	0.82 ± 0.20	0.04 ± 0.01
BC	1.00 ± 0.00	0.07 ± 0.04	0.21 ± 0.06	0.46 ± 0.11	0.09 ± 0.03
VINN	0.89 ± 0.06	0.01 ± 0.02	0.02 ± 0.02	0.04 ± 0.04	0.01 ± 0.02
DT	0.94 ± 0.08	0.68 ± 0.20	0.89 ± 0.10	0.77 ± 0.14	0.66 ± 0.18
DDPGBC	1.00 ± 0.00	0.63 ± 0.11	0.91 ± 0.04	0.48 ± 0.22	0.04 ± 0.08
AMP	0.99 ± 0.02	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
CoL	1.00 ± 0.00	0.71 ± 0.16	0.96 ± 0.04	0.48 ± 0.23	0.04 ± 0.07
AWAC	0.94 ± 0.20	0.43 ± 0.43	0.26 ± 0.33	0.31 ± 0.32	0.00 ± 0.00
DEX	1.00 ± 0.00	0.73 ± 0.12	0.94 ± 0.04	0.73 ± 0.20	0.63 ± 0.19
ViSkill*	–	–	–	–	–
T-STAR*	–	–	–	–	–
MS-LSDT (ours)	**0.98 ± 0.02**	**0.74 ± 0.12**	**0.94 ± 0.06**	**0.82 ± 0.10**	**0.72 ± 0.11**

(*continued*)

Table 2. (*continued*)

Method	NeedleReach	GauzeRetrieve	NeedlePick	PegTransfer	NeedleRegrasp
Method	**BiPegTransfer**	**BiPegBoard**	**MatchBoardPanel**	**PickAndPlace**	**MatchBoard**
dim(O|G|A)	R34|R3|R10	R34|R3|R10	R31|R6|R4	R31|R6|R4	R20|R3|R4
PLAS	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
IQL	0.09 ± 0.04	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
BC	0.09 ± 0.04	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
VINN	0.00 ± 0.00	0.02 ± 0.04	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
DT	0.14 ± 0.12	0.67 ± 0.09	0.29 ± 0.19	0.18 ± 0.04	0.21 ± 0.09
DDPGBC	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.04 ± 0.01	0.04 ± 0.03
AMP	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
CoL	0.01 ± 0.02	0.00 ± 0.00	0.00 ± 0.00	0.02 ± 0.04	0.04 ± 0.04
AWAC	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.07 ± 0.04
DEX	0.18 ± 0.14	0.02 ± 0.01	0.00 ± 0.00	0.02 ± 0.07	0.04 ± 0.04
ViSkill*	0.84 ± 0.08	0.81 ± 0.04	0.47 ± 0.06	–	–
T-STAR*	0.67 ± 0.04	0.64 ± 0.10	0.44 ± 0.04	–	–
MS-LSDT (ours)	$\mathbf{0.24 \pm 0.10}$	$\mathbf{0.78 \pm 0.08}$	$\mathbf{0.38 \pm 0.12}$	$\mathbf{0.24 \pm 0.06}$	$\mathbf{0.28 \pm 0.10}$

* Methods with manually pre-defined priors (e.g., task decomposition/chaining). "—" denotes not applicable.

Table 3. Descriptive gain distribution of MS-LSDT over DT. Gains are absolute success-rate differences in percentage points (pp), computed from Table 2.

DT difficulty group	Tasks	Avg. DT	Avg. gain
Saturated/easy ($> = 0.74$)	NeedleReach, NeedlePick, PegTransfer	0.87	+4.3 pp
Moderate (0.40–0.74)	GauzeRetrieve, NeedleRegrasp, BiPegBoard	0.67	+8.0 pp
Hard (< 0.40)	BiPegTransfer, MatchBoardPanel, PickAndPlace, MatchBoard	0.21	+8.0 pp
All tasks	10 tasks	0.44	+6.9 pp

5.1 Discussion

MS-LSDT improves the mean success rate over DT on all ten reported tasks, but the average improvement is modest (+6.9 pp) and should be interpreted descriptively. The gains are smallest on already saturated/easy tasks (e.g., NeedleReach, +3 pp) and larger on moderate or hard tasks (Table [tab:gain_distribution]), with the largest task-level gains on BiPegBoard (+11 pp), BiPegTransfer (+10 pp), and MatchBoardPanel (+9 pp).

Using only the reported means/stds and assuming three independent seeds, exploratory Welch tests do not reach $p < 0.04$ on individual tasks (minimum approximate $p = 0.19$ on BiPegBoard), so we avoid claiming statistical significance from the current aggregate results. The ablations below further suggest that both multi-kernel locality and goal-progress conditioning contribute to these gains.

5.2 Ablation Studies

We perform code-aligned ablations under the same training and evaluation protocol as Sect. 4.4. The default setting uses multi-kernel short branches $\mathcal{K} = \{3, 7, 14\}$, RTG + TTG conditioning, and multi-objective pretraining. We isolate four factors using the runs currently available in the project: (i) replacing the hybrid block with the DT backbone (pure Transformer), (ii) replacing multi-kernel short branches with a single kernel $k = 7$, (iii) disabling RTG input while keeping TTG (TTG-only conditioning), and (iv) removing auxiliary objectives during pretraining (action-only training). Table [tab:ablation] reports success rates on three representative tasks with different difficulty levels, averaged over three seeds. Beyond final metrics, Fig. 2 visualizes branch-wise effective context maps computed from saved checkpoints, showing how each scale contributes across query–key distances (Table 4).

Table 4. Ablations on representative tasks (success rate, %).

Setting	PegTransfer	NeedleRegrasp	BiPegTransfer	Avg.
DT backbone (pure Transformer)	77.0	66.0	14.0	42.7
MS-LSDT, K = {3,7,14}	**82.4**	**73.1**	**28.6**	**61.4**
MS-LSDT, K = {7}	78.1	69.4	24.7	47.4
MS-LSDT, TTG-only (disable RTG input)	74.6	64.2	21.4	43.4
MS-LSDT, action-only pretrain	76.3	66.8	23.6	44.6

Observation. All ablations underperform the default setting across tasks. Compared with the pure Transformer DT backbone on these three tasks, the default MS-LSDT improves the average success rate by 8.7 pp. Removing RTG while keeping TTG causes the largest average drop among the MS-LSDT variants, and replacing multi-kernel locality with a single kernel also degrades performance consistently. These results support the joint benefit of explicit progress conditioning and multi-scale local modeling, but they do not yet isolate the gating mechanism itself from a fixed or uniform branch average.

6 Limitations and Future Work

Our study is limited to state-based SurRoL simulation and does not include image observations or hardware deployment. Although we report multi-seed comparisons and controlled ablations, the current paper package contains only aggregate mean/std.

results rather than raw seed-level outputs; therefore, we treat task-wise improvements as descriptive and do not make a strong statistical-significance claim. Broader sensitivity analysis over optimizer schedules and model scale remains to be completed. The available ablations cover a pure Transformer baseline, multi-kernel versus single-kernel local branches, TTG-only conditioning, and action-only pretraining, but they do not yet include a convolution-only backbone or a no-gate/uniform-gate variant. These two ablations would more directly test the necessity of the long branch and the token-wise gate, respectively. Since SurRoL supports real-robot transfer as in [13], an important next step is validating the learned policies on dVRK.

7 Conclusion

We described MS-LSDT, a multi-scale long–short backbone for goal-conditioned offline surgical robot learning in SurRoL. The approach integrates multi-kernel causal depthwise convolutions with causal attention through token-wise gating and is trained under a two-stage cross-task protocol. The results and ablations show consistent descriptive gains under a fixed code-aligned protocol, especially on moderate and hard tasks where the DT baseline is less saturated. These findings suggest that the proposed long–short multi-scale design is a promising backbone choice for goal-conditioned offline surgical manipulation, while stronger statistical validation and additional gate/conv-only ablations remain important future work.

Acknowledgements. This study was funded by the Jiangsu Key Laboratory Construction Project (Grant No. C0424QT00000) of Jiangsu Key Laboratory of Digital Finance, Suzhou, China.

References

1. Fujimoto, S., Meger, D., Precup, D.: Off-policy deep reinforcement learning without exploration. In: Proceedings International Conference on Machine Learning (ICML) (2019)
2. Kumar, A., Fu, J., Tucker, G., Levine, S.: BEAR: stabilizing offline Q-learning via bootstrapping error reduction. arXiv preprint https://arxiv.org/abs/1906.00949, (2019)
3. Wu, Y., Tucker, G., Nachum, O.: Behavior regularized offline reinforcement learning. arXiv preprint https://arxiv.org/abs/1911.11361, (2019)
4. Kumar, A., Zhou, A., Tucker, G., Levine, S.: Conservative Q-learning for offline reinforcement learning. In: Advances in Neural Information Processing Systems (NeurIPS) (2020)
5. Kostrikov, I., Nair, A., Levine, S.: Offline reinforcement learning with implicit Q-learning. In: International Conference on Learning Representations (ICLR) (2022)
6. Fujimoto, S., Gu, S.S.: A minimalist approach to offline reinforcement learning. In: Advances in Neural Information Processing Systems (NeurIPS) (2021)
7. Peng, X. B., Kumar, A., Zhang, G., Levine, S.: Advantage-weighted regression: simple and scalable off-policy reinforcement learning. arXiv preprint https://arxiv.org/abs/1910.00177, (2019)
8. Nair, A., Dalal, M., Gupta, A., Levine, S.: AWAC: Accelerating online reinforcement learning with offline datasets. In: NeurIPS Offline Reinforcement Learning Workshop (2020)
9. Yu, T. et al.: MOPO: model-based offline policy optimization. arXiv preprint https://arxiv.org/abs/2005.13239, (2020)

10. Kidambi, R., Rajeswaran, A., Netrapalli, P., Joachims, T.: MOReL: model-based offline reinforcement learning. arXiv preprint https://arxiv.org/abs/2005.05951, (2020)
11. Chen, L. et al.: Decision transformer: Reinforcement learning via sequence modeling. In: Advances in Neural Information Processing Systems (NeurIPS) (2021)
12. Vaswani, A. et al.: Attention is all you need. arXiv preprint https://arxiv.org/abs/1706.03762, (2017)
13. Fu, J., Long, Y., Chen, K., Wei, W., Dou, Q.: Multi-objective cross-task learning via goal-conditioned GPT-based decision transformers for surgical robot task automation. arXiv preprint https://arxiv.org/abs/2405.18757, (2024)
14. Zhang, Y. et al.: SurRoL: an open-source reinforcement learning centered and dVRK compatible platform for surgical robot learning. arXiv preprint https://arxiv.org/abs/2108.13035, (2021)
15. Fu, J., Kumar, A., Nachum, O., Tucker, G., Levine, S.: D4RL: datasets for deep data-driven reinforcement learning. arXiv preprint https://arxiv.org/abs/2004.07219, (2020)
16. Mandlekar, A. et al.: RoboMimic: a benchmark for robot imitation learning. arXiv preprint https://arxiv.org/abs/2108.03298, (2021)
17. Levine, S., Kumar, A., Tucker, G., Fu, J.: Offline reinforcement learning: tutorial, review, and perspectives on open problems. arXiv preprint https://arxiv.org/abs/2005.01643, (2020)
18. Gulati, A. et al.: Conformer: Convolution-augmented transformer for speech recognition. In: Proceedings of Interspeech (2020)
19. Wu, H. et al.: CvT: Introducing convolutions to vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021)
20. Dai, Z., Liu, H., Le, Q. V., Tan, M.: CoAtNet: marrying convolution and attention for all data sizes. arXiv preprint https://arxiv.org/abs/2106.04803, (2021)
21. Wu, F., Fan, A., Baevski, A., Dauphin, Y.N., Auli, M.: Pay less attention with lightweight and dynamic convolutions. In: International Conference on Learning Representations (ICLR) (2019)
22. Howard, A. G. et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv preprint https://arxiv.org/abs/1704.04861, (2017)
23. Andrychowicz, M. et al.: Hindsight experience replay. In: Advances in Neural Information Processing Systems (NeurIPS) (2017)