Rotary Positional Embeddings for Length Generalization in Decision Transformers

Lasse von Danwitz
Implementing Transformers Course
Winter Semester 2025/26

February 11, 2026

Abstract

This report presents a systematic evaluation of positional encoding strategies for length generalization in Decision Transformers. We compare three approaches: learned absolute embeddings, fixed sinusoidal embeddings, and Rotary Positional Embeddings (RoPE). Using a Key-Door maze navigation task, models are trained exclusively on 8×8 grids and evaluated on grids up to 20×20. Results demonstrate that RoPE achieves 86% success on 10×10 grids compared to 64% for learned embeddings and 40% for sinusoidal encodings. On 12×12 grids, RoPE maintains 43.3% success versus 18.3% and 1.7% respectively. Context length experiments (30–90 tokens) reveal stable performance across all models, indicating that RoPE's advantage stems from position extrapolation rather than context capacity. Statistical analysis confirms large effect sizes ($d > 0.8$) with significance at $p = 0.035$ for 12×12 extrapolation. These findings support RoPE's adoption in modern architectures requiring robust length generalization.

1 Introduction

Decision Transformers (DT) [3] and related architectures like Trajectory Transformers [4] treat reinforcement learning as a sequence modeling problem, predicting actions based on past states and desired returns. While effective within their training distribution, these models frequently struggle with *length generalization*, the ability to operate on trajectories longer than those seen during training [5]. This limitation is particularly relevant for offline reinforcement learning, where models trained on short, suboptimal demonstrations must extrapolate to longer, optimal paths during deployment.

A primary suspect for this limitation is the positional encoding scheme. The original Transformer architecture utilizes absolute positional encodings, which associate specific behaviors with specific indices. This rigid association fails when the inference sequence length exceeds the training horizon. Recently, Rotary Positional Embeddings (RoPE) [2] have gained traction in language modeling by encoding position via rotation rather than addition, theoretically enabling better extrapolation through relative position dependencies.

This report investigates the following research question: *Does RoPE improve length generalization in Decision Transformers, and what mechanisms enable this improvement?* We train models on 8×8 Key-Door maze grids and evaluate extrapolation to grids up to 20×20, comparing learned absolute embeddings, fixed sinusoidal embeddings, and RoPE.

2 Background

2.1 Transformer Architecture

The Transformer [1] architecture represents a paradigm shift in sequence modeling, replacing recurrent processing with self-attention mechanisms to process sequence data in parallel. This architecture serves as the backbone for our Decision Transformer implementation.

2.1.1 Attention Mechanism

The core of the Transformer is the Attention mechanism, which enables the network to weigh the relevance of different tokens in the context of one another. For input tokens projected into queries ($\mathbf{Q}$), keys ($\mathbf{K}$), and values ($\mathbf{V}$), the model computes a weighted sum of values. The attention scores are derived from the similarity between queries and keys, scaled by $\sqrt{d_k}$ for numerical stability:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V} \tag{1}$$

Here, $\mathbf{M}$ represents a causal mask, ensuring that the prediction for a specific timestep depends only on current and past context, a critical requirement for autoregressive generation.

2.1.2 Multi-Head Attention

To capture diverse relationships within the data, we employ Multi-Head Attention (MHA). This allows

the model to attend to information from different representation subspaces jointly. The input of dimension d_{model} is split into h heads, where each head i computes independent attention:

$$\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \quad (2)$$

These heads are concatenated and projected linearly to form the final output:

$$\text{MHA} = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O \quad (3)$$

2.1.3 Causal Masking

To preserve the autoregressive property required for sequence generation, we implement a causal mask $\mathbf{M}$ that prevents the model from "looking ahead" at future tokens. We add this mask matrix to the attention scores prior to the softmax operation.

For a sequence of length $T = 4$, the mask $\mathbf{M}$ is defined as:

$$\mathbf{M} = \begin{pmatrix} 0 & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (4)$$

The $-\infty$ entries result in zero probability after the softmax ($\exp(-\infty) = 0$), effectively forcing the attention mechanism at index i to aggregate information only from indices $j \in \{1, \ldots, i\}$. Formally:

$$\mathbf{M}_{ij} = \begin{cases} 0 & \text{if } i \geq j \\ -\infty & \text{otherwise} \end{cases} \quad (5)$$

2.1.4 Layer Components

Each transformer block follows a standard structure consisting of the attention mechanism followed by a Position-wise Feed-Forward Network (FFN). The FFN consists of two linear transformations with a GELU activation [6]:

$$\text{FFN}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2 \quad (6)$$

We employ a Pre-Layer Normalization (Pre-LN) architecture. In this setup, Layer Normalization is applied before the sub-layers (Attention and FFN), and residual connections are added afterwards, which improves training stability [7].

2.2 Decision Transformer Formulation

Our project utilizes the Decision Transformer, reframing Reinforcement Learning (RL) as autoregressive sequence modeling. Trajectories τ consist of return-to-go ($\hat{R}$), state (s), and action (a) triplets:

$$\tau = (\hat{R}_1, s_1, a_1, \ldots, \hat{R}_T, s_T, a_T) \quad (7)$$

Standard input embeddings combine modality and timestep:

$$\mathbf{x}_t = \text{Embed}(\text{modality}_t) + \text{Embed}_{\text{time}}(t) \quad (8)$$

Property	Sinusoidal	RoPE
Method	Addition	Rotation
Type	Absolute	Relative (implicit)
Application	Token emb.	Query/Key vectors
Translation Inv.	No	Yes

Table 1: Comparison of positional encoding approaches.

However, explicit absolute timestep embeddings hinder generalization when the inference horizon t exceeds training ranges, motivating alternative positional encodings.

2.3 Positional Encoding Strategies

We compare two methods to inject order information into the permutation-equivariant self-attention mechanism: Sinusoidal encoding and Rotary Positional Embeddings (RoPE).

2.3.1 Sinusoidal Positional Embeddings

The original Transformer [1] adds absolute encodings to inputs. For position m and dimension i:

$$PE(m, 2i) = \sin\left(\frac{m}{10000^{2i/d}}\right) \quad (9)$$

$$PE(m, 2i+1) = \cos\left(\frac{m}{10000^{2i/d}}\right) \quad (10)$$

Limitations in RL include reliance on absolute positions, poor length generalization (distribution shift), and potential interference with semantic information due to additive mixing.

2.3.2 Rotary Positional Embeddings (RoPE)

RoPE addresses these by rotating query and key vectors instead of adding to inputs. For embedding pairs, rotating by $\theta_i \cdot m$ encodes position m:

$$R_{\theta,m} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix} \quad (11)$$

RoPE naturally encodes relative distance. The inner product of query q (pos m) and key k (pos n) becomes:

$$q_m'^T k_n' = q_m^T R_{\theta,m}^T R_{\theta,n} k_n = q_m^T R_{\theta,(n-m)} k_n \quad (12)$$

Eq. 12 shows attention depends only on relative distance $(n-m)$, enabling translation invariance and better generalization to unseen trajectory lengths.

Efficient Implementation We apply rotation component-wise to avoid matrix multiplication:

$$x'_{rot} = x \odot \cos(\Theta) + \text{rotate\_half}(x) \odot \sin(\Theta) \quad (13)$$

where $\text{rotate\_half}([x_1, x_2, \ldots]) = [-x_2, x_1, \ldots]$ and $\theta_i = \text{base}^{-2i/d}$.

3 Experimental Setup

3.1 Environment and Data

We utilize a Key-Door Maze environment (Figure 1) requiring sequential logic: collect key → open door → reach goal.

- **Data Generation:** 5,000 trajectories on fixed 8×8 grids (70% optimal BFS planner, 30% sub-optimal).
- **Generalization Task:** Models trained *only* on 8×8 grids, evaluated on sizes up to 20×20.

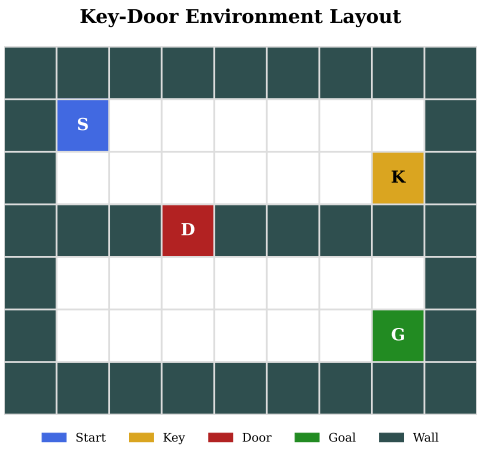

Figure 1: Key-Door Maze environment. The agent (2) must navigate walls (1) to find the key (4), unlock the door (5), and reach the goal (3).

3.2 Model Architecture

We adopt the Decision Transformer architecture with specific modifications for grid-world generalization (Table 2).

Parameter	Value
Embedding dim. (d_{model})	320
Attention heads	10
Head dimension	32
Transformer layers	8
Context length	30 timesteps
FFN hidden dim.	1280 (4×)
Dropout	0.1

Table 2: Decision Transformer architecture configuration.

Embeddings: To support inference on variable grid sizes, we employ a grid-agnostic State Encoder. Integer inputs are processed via item embeddings, followed by a 2 layer CNN (64 filters, 3×3 kernels, ReLU) and `AdaptiveAvgPool2d`. This reduces spatial dimensions to a fixed $(1, 1)$ output projected to d_{model}. Actions and returns-to-go utilize learned lookup tables and linear projections, respectively.

3.3 Positional Encoding Variants

We distinguish two types of positional information:

1. **Global Episode Timestep (t)**: Encoded using fixed sinusoidal embeddings in all variants to preserve horizon awareness.
2. **Local Context Index (k)**: Position within the context window.

Three strategies for encoding k:

- **Baseline**: Learned embeddings added to input.
- **Sinus**: Fixed sinusoidal encodings added to input.
- **RoPE**: Rotary embeddings applied to Query/Key vectors.

3.4 Training Protocol

- **Dataset**: 5,000 episodes (70/15/15 train/val/test split)
- **Optimization**: AdamW [8], lr $= 10^{-4}$ (no scheduler), weight decay 0.01
- **Training**: 100 epochs, batch size 64, gradient clipping 1.0
- **Seeds**: 3 random seeds (42, 123, 456)
- **Evaluation**: 100 episodes per grid size per seed

4 Results and Analysis

4.1 Training Performance

All three variants converge successfully on the training distribution (Table 3, Figure 2). RoPE achieves marginally lower training loss (0.356 vs 0.411–0.413) while maintaining comparable validation performance in terms of success rate. This suggests efficient optimization without overfitting.

4.2 The Loss Generalization Disconnect

Training metrics (Table 3) mask a fundamental disconnect between cross-entropy loss and true policy robustness. Because loss only measures single-timestep expert replication, it fails to distinguish between two strategies that yield identical in-distribution results:

Spurious Correlation: Memorizing absolute coordinates (e.g., "at index 5, move right").

Causal Mechanism: Learning relative navigation (e.g., "if goal is right of current position, move right").

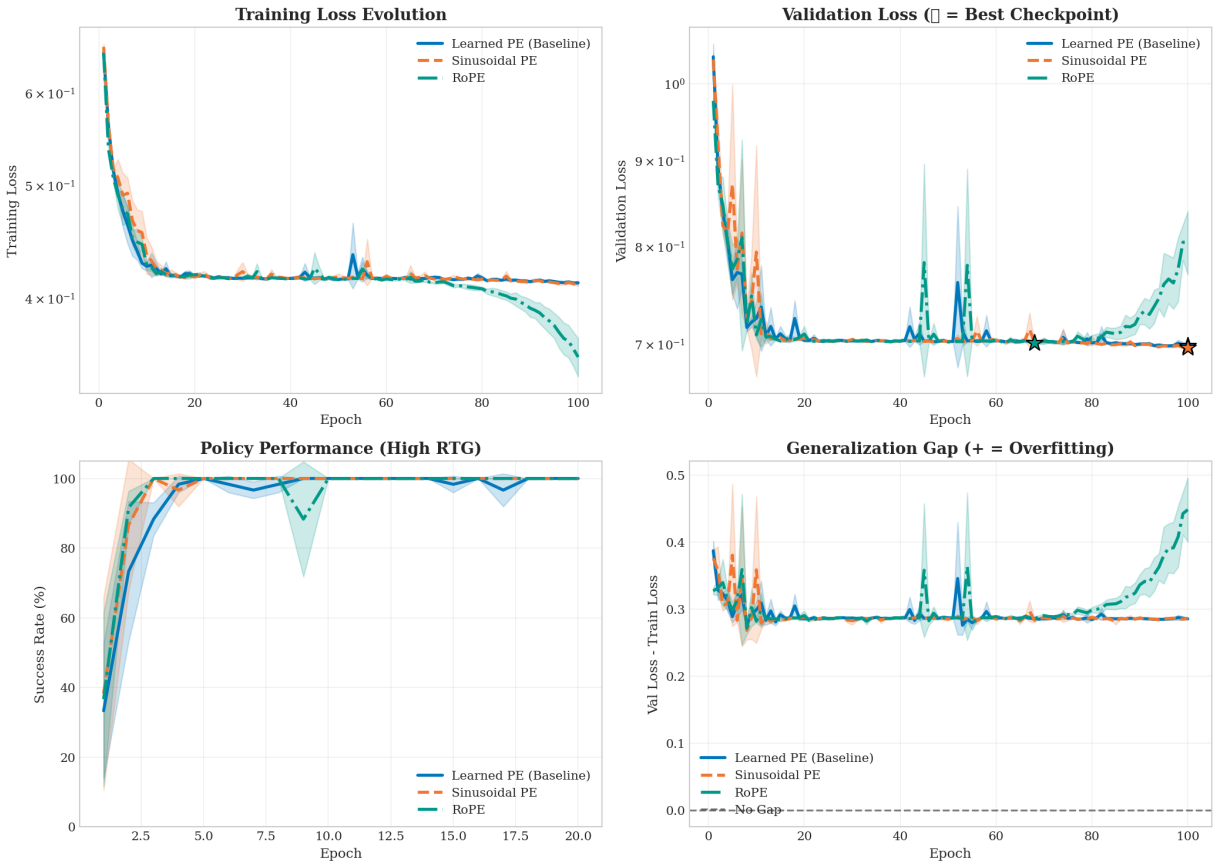

Figure 2: Training dynamics comparison. Top: training and validation loss converge similarly across models. Bottom: validation success rate and overfitting indicator show comparable in-distribution performance.

On 8×8 training grids, both strategies achieve low loss. The deficiency emerges only during extrapolation: the Baseline's coordinate-dependence triggers a performance collapse on 12×12 grids, whereas RoPE's relative encoding enables zero-shot generalization.

Thus, standard validation loss is an insufficient proxy for OOD performance. While it tracks convergence, **generalization must be explicitly verified on shifted test sets rather than inferred from in-distribution metrics.**

Model	Final Loss	Val Loss	Success (%)
Baseline	0.413	0.698	97.3
Sinus	0.411	0.697	98.7
RoPE	0.356	0.805	99.3

Table 3: Training metrics on 8×8 grids.

4.3 Length Generalization

Now we present the central findings of this report. Table 4 summarizes the results, visualized in Figure 3.

4.3.1 Key Findings

RoPE Advantage. RoPE achieves 86% success on 10×10 grids and 43.3% on 12×12, consistently outperforming both alternatives. At extreme extrapolation (20×20), RoPE maintains 5.3% success while alternatives reach 0–2.3%.

Learned vs. Fixed Encodings. Learned absolute embeddings (64% on 10×10) substantially outperform fixed sinusoidal versions (40%). This gap likely reflects optimization dynamics: learned weights start near-zero for task-specific specialization, whereas sinusoidal structures may impose rigid constraints that conflict with learned patterns. However, this advantage vanishes on larger grids, where both reach near-zero performance.

Variance Analysis. Standard deviations reveal instability in absolute encodings: Baseline shows ±25.2% on 10×10, Sinus shows ±40.6%, while RoPE maintains ±12.1%. This high variance indicates seed-dependent generalization when extrapolating beyond trained position indices.

4.3.2 Path Efficiency

Table 5 compares the average steps to reach the goal of the models per grid size.

The efficiency analysis highlights a crucial distinction: while RoPE degrades in success rate on larger grids, the successful episodes remain reasonably efficient (1.4 optimal on 10×10), whereas baseline models that succeed often do so via inefficient random walks.

Grid	Baseline	Sinus	RoPE	Δ vs Base	Δ vs Sinus
8×8	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	—	—
10×10	64.0 ± 25.2	40.0 ± 40.6	86.0 ± 12.1	+22.0	+46.0
12×12	18.3 ± 19.7	1.7 ± 1.2	43.3 ± 18.1	+25.0	+41.6
15×15	1.3 ± 0.9	0.0 ± 0.0	13.7 ± 12.9	+12.4	+13.7
20×20	2.3 ± 3.3	0.0 ± 0.0	5.3 ± 5.6	+3.0	+5.3

Table 4: Length generalization success rates (%, mean ± std across 3 seeds).

Grid	Baseline	Sinus	RoPE	Eff. Ratio
8×8	12.6	12.6	13.1	1.04
10×10	30.5	38.5	**22.3**	1.4
12×12	56.0	59.7	**49.1**	2.2
15×15	74.6	75.0	**70.0**	2.4
20×20	98.5	100.0	**98.7**	2.4

Table 5: Average steps to goal (lower is better).

4.4 Context Length Scaling

A critical experiment tested whether RoPE's advantage stems from better context window utilization. Models were evaluated with context lengths from 30 to 90 tokens (Table 6).

Context	Sinus	RoPE	RoPE Δ
30 (1.0×)	28.7	50.1	+21.3
45 (1.5×)	30.0	50.2	+21.2
60 (2.0×)	29.6	50.1	+20.5
90 (3.0×)	29.7	50.1	+20.4

Table 6: Average success rate (%) at varying context lengths across all grid sizes. RoPE's advantage remains constant.

4.4.1 Key Findings

Performance remains stable across context lengths for both models. RoPE's ∼21% advantage persists regardless of context window size. This finding reveals that RoPE's advantage is **not** about handling more context tokens, but about **extrapolating to unseen sequence positions**.

The distinction is worth emphasizing:

- **Context capacity**: How many tokens the model can attend to simultaneously.
- **Position extrapolation**: Whether patterns learned at positions 1–30 transfer to positions 50–80.

Crucially, as the networks were not explicitly trained to leverage extended sequences, the observed stability is expected. This effectively isolates the variable of interest: the performance gap stems not from context capacity, but from the ability to apply learned attention patterns to novel position indices (extrapolation).

4.5 Attention Pattern Analysis

Figure 4 visualizes attention weights from the final transformer layer. RoPE exhibits consistent banded

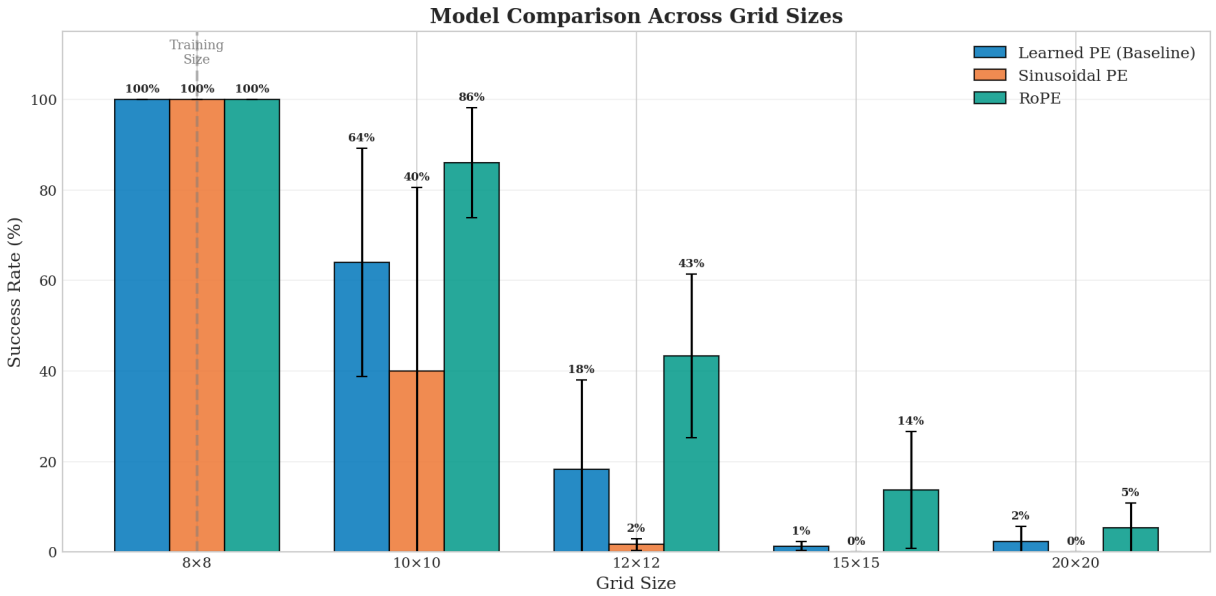

Figure 3: Length generalization comparison across grid sizes. RoPE maintains substantially higher success rates beyond training distribution. Error bars indicate standard deviation across 3 seeds.

patterns where attention concentrates at fixed relative offsets regardless of absolute position. In contrast, learned embeddings show position-specific patterns that fail to generalize beyond training indices.

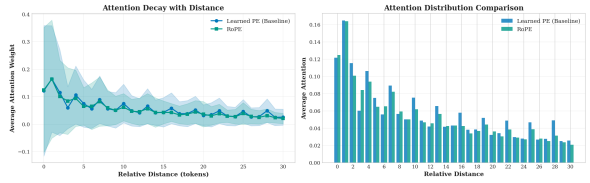

Figure 4: Attention distance decay analysis. RoPE maintains smoother decay with distance, reflecting its relative position encoding. Baseline shows irregular patterns indicating position-specific learned associations.

4.6 Translation Invariance Verification

RoPE's translation invariance was empirically validated by comparing attention patterns for sequences at positions $[0, 1, ..., L]$ versus $[100, 101, ..., 100 + L]$:

- Mean error: 0.014 ± 0.004
- Maximum error: 0.024

Errors below 3% confirm that RoPE maintains attention patterns regardless of absolute position, validating the theoretical property from Equation 12.

4.7 Statistical Analysis

Table 7 summarizes statistical comparisons, with effect sizes visualized in Figure 5.

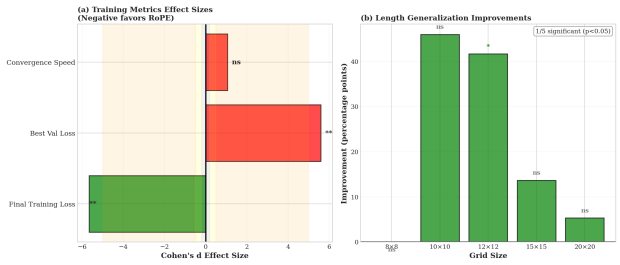

Figure 5: Effect sizes (Cohen's d) and significance levels. Large effects ($d > 0.8$) observed across extrapolation grid sizes.

Grid	Cohen's d	p-val	Signif.
10×10	1.23	0.208	No
12×12	2.56	0.035	Yes
15×15	1.21	0.213	No
20×20	1.18	0.223	No

Table 7: Statistical comparison (RoPE vs Sinus).

Effect sizes consistently exceed $d = 0.8$ (large effect). The 12×12 comparison achieves significance at $p = 0.035$. Non-significant results at other sizes reflect limited statistical power rather than absent effects.

4.8 Failure Mode Analysis

Examining unsuccessful 20×20 episodes reveals systematic failures at positions > 50 steps from episode start. RoPE's frequency progression (base $10000^{-2i/d}$) produces wavelengths up to ∼35,000 positions, theoretically sufficient for these lengths. The fact that empirical generalization degrades significantly earlier (beyond ∼3× training length) indicates that while the encoding capacity exists, the model struggles to utilize it effectively. This points to potential bottlenecks in how the agent processes increasingly complex spatial states, a hypothesis we explore in the Discussion.

5 Discussion

5.1 Mechanism Analysis

Why Learned Embeddings Partially Succeed. Learned embeddings can adjust representations so $PE_m - PE_n$ correlates with $(m - n)$, forming an implicit relative encoding. However, this is fragile: high variance (±25.2% on 10×10), rapid degradation beyond training ranges, and the lack of a principled mechanism for unseen indices highlight fundamental limitations.

The Role of Global Timesteps. All models utilize a separate, fixed sinusoidal encoding for global episode timestep t. This separates concerns: global embeddings signal trajectory progress, while local embeddings handle spatial context. This dual encoding explains the efficiency gap in Table 5. Baselines likely rely on the global horizon to spur "random walk" exploration, whereas RoPE's relative signals facilitate directed navigation ("goal is k steps away").

Why RoPE Provides Robust Generalization. RoPE structurally enforces $q'_m k'^T_n = f(m - n)$, making attention dependent solely on relative distance. Because absolute indices are never computed, a learned rule like "key seen two steps ago" transfers identically from positions $(5, 7)$ to $(105, 107)$ as both represent distance $+2$.

5.2 The Encoder Information Bottleneck

Despite RoPE's strengths, performance drops to 5.3% on 20×20 grids. The state encoder is a likely bottleneck: our CNN uses `AdaptiveAvgPool2d` to compress spatial maps to $(1, 1)$ regardless of input size.

Moving from 8×8 (64 pixels) to 20×20 (400 pixels) increases the compression ratio by ∼6×. This loss may cause "state aliasing," where distinct configurations map to identical latents. This suggests that even with infinite-range positional encodings, navigation fails if the visual encoder cannot distinguish

states on larger grids. Future work should explore FCNs or spatial attention to decouple representation quality from grid scale.

5.3 Implications for Offline RL

The results have significant implications for Offline Reinforcement Learning. A core challenge in Offline RL is combining parts of suboptimal, short trajectories to form optimal, long horizons. Our findings suggest that RoPE enhances this capability by allowing the attention mechanism to maintain temporal logic (e.g., cause-and-effect relationships) across time horizons significantly longer than the specific fragments seen in the static dataset.

5.4 Limitations

Statistical Power. Three seeds provide limited confidence. While effect sizes are consistently large, only the 12×12 comparison achieved $p < 0.05$. Therefore further research with additional seeds is necessary to confirm findings at all grid sizes.

Task Specificity. Key-Door maze results may not generalize to all domains. The task requires specific sequential dependencies (key→door→goal) that particularly benefit from relative position encoding. It might be worthwhile to evaluate on additional task types.

Extreme Extrapolation. Performance degrades substantially beyond 2× training length (from 86% at 10×10 to 5.3% at 20×20), suggesting fundamental limits even with relative encodings.

No complete Isolation. Each version of the model uses sinusoidal encoding for the timestep encoding. As such a complete generalization for non dual positional encoding is at best limited.

RoPE Variants. We tested RoPE for sequence positions with sinusoidal timestep encoding. Applying RoPE to timesteps may provide additional benefits and is an interesting direction for future work.

6 Conclusion

This report evaluated three positional encoding approaches for length generalization in Decision Transformers.

Findings:

1. **Generalization Hierarchy.** RoPE achieved 86% success on 10×10 and 43.3% on 12×12 grids when trained only on 8×8. Learned embeddings showed surprising resilience (64% on 10×10) compared to fixed sinusoidal (40%), but collapsed rapidly on larger grids.

2. **Variance Indicates Instability.** Learned embeddings exhibited high variance (±25.2% on 10×10) compared to RoPE (±12.1%), indicating seed-dependent generalization. Combined with longer paths to goal (30.5 vs 22.3 steps), this suggests learned relativity is fragile.

3. **Structural Guarantees Enable Robustness.** By encoding position through rotations such that $q'_m k'^T_n = f(m-n)$, RoPE ensures attention depends only on relative distances. Translation invariance (verified with error <3%) allows learned patterns to transfer to any position range.

4. **Position Extrapolation, Not Context Scaling.** Context length experiments (30–90 tokens) showed stable performance with RoPE's ∼21% advantage constant. This confirms RoPE addresses position index generalization rather than context capacity.

Implications: For offline RL applications where deployment environments may exceed training scales, reliance on absolute positional encodings is a primary bottleneck. RoPE offers a mathematically principled solution, requiring no additional parameters while providing robust extrapolation through learned relative patterns.

However, our analysis of extreme extrapolation (20×20) indicates that once positional ambiguity is resolved, **visual state representation becomes the limiting factor**. Future architectures must therefore address both sequence length (via RoPE) and spatial resolution (via scale-invariant encoders) to achieve unbounded generalization.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

[2] Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., & Liu, Y. (2021). RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

[3] Chen, L., Lu, K., Rajeswaran, A., et al. (2021). Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*, 34.

[4] Janner, M., Li, Q., & Levine, S. (2021). Offline reinforcement learning as one big sequence modeling problem. *Advances in Neural Information Processing Systems*, 34, 1273-1286.

[5] Press, O., Smith, N. A., & Lewis, M. (2021). Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

[6] Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*.

[7] Xiong, R., Yang, Y., He, D., et al. (2020). On layer normalization in the transformer architecture. *International Conference on Machine Learning*, 10524-10533.

[8] Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. *International Conference on Learning Representations*.

[9] Touvron, H., Lavril, T., Izacard, G., et al. (2023). LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

[10] Chowdhery, A., Narang, S., Devlin, J., et al. (2022). PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

[11] Black, S., Biderman, S., Hallahan, E., et al. (2022). GPT-NeoX-20B: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.