Assignment 1: Paper Review & Code Reproduction

CSC415: Introduction to Reinforcement Learning

Student Name: Shijun Chen
Student ID: 1009305716

February 14, 2026

Part I: Paper Reviews (50 Points)

Review 1

Paper Title: Decision Transformer: Reinforcement Learning via Sequence Modeling
Authors: Chen et al. (2021)

1. Summary of Contributions

- **The Problem:** Traditional Offline Reinforcement Learning relies on bootstrapping (TD-learning) to estimate value functions. This introduces the "deadly triad" of instability (off-policy data, bootstrapping, function approximation), leading to value overestimation and error propagation, particularly in sparse-reward or suboptimal data regimes.
- **The Core Method:** The authors propose the *Decision Transformer* (DT), which reframes RL not as a dynamic programming problem, but as a conditional sequence modeling problem. They treat trajectories as sequences of tokens $\tau = (\hat{R}_1, s_1, a_1, \ldots, \hat{R}_T, s_T, a_T)$. The model uses a GPT architecture to autoregressively predict the next action a_t given the history of states and a *Target Return* (Return-to-Go, $\hat{R}_t$). This removes the need for value functions entirely.
- **The Result:** DT matches or exceeds the performance of state-of-the-art model-free offline RL algorithms (like CQL and BEAR) on standard benchmarks (Atari, OpenAI Gym). Crucially, it demonstrates "stitching" abilities—combining suboptimal trajectory segments to produce expert behavior—without explicit Bellman updates.

2. Strengths

- **Theoretical (Stability via Supervised Learning):** By casting RL as a supervised learning problem, DT avoids the optimization instability inherent in Bellman backups (e.g., the max-operator bias in Q-learning). It leverages the stable, well-understood convergence properties of the Cross-Entropy loss used in language modeling.
- **Empirical (Long-Term Credit Assignment):** The Transformer architecture utilizes self-attention ($O(N^2)$), allowing it to directly credit actions to distant rewards. This is demonstrated in the *Key-to-Door* environment, where DT solves tasks requiring credit assignment over long horizons where TD-learning methods fail due to vanishing gradients in the value chain.

3. **Weaknesses**

- **Stochasticity & Multimodality:** DT is inherently deterministic in its standard formulation (predicting the mean action or a single token). In highly stochastic environments where the optimal policy is multimodal (e.g., "go left or right, but not straight"), minimizing Mean Squared Error (MSE) leads to averaging valid modes, resulting in catastrophic failure (e.g., going straight into an obstacle).
- **Dependence on Training Distribution (No Ex-Nihilo Exploration):** Unlike Q-learning, which can theoretically propagate values to unvisited states via generalization, DT is strictly bounded by the support of the training data. It cannot hallucinate a "better" return than the maximum return present in the dataset unless it stitches existing segments; it cannot discover truly novel strategies.
- **Computational Complexity:** Inference is autoregressive. Predicting a sequence of length T requires $O(T)$ forward passes, which is significantly slower ($\sim$10-100ms) than a standard MLP policy ($\sim$1ms), making it difficult to deploy in high-frequency robotic control loops (e.g., 500Hz).

4. **Proposed Improvements**

- **Proposal 1 (Addressing Multimodality):** Replace the deterministic MSE loss with a *Diffusion Head*. Instead of predicting a point estimate $\hat{a}_t$, the Transformer would condition a diffusion model $p_\theta(a_t|s_t, \hat{R}_t)$ to denoise random noise into an action. This allows the model to represent complex, multimodal policy distributions, preventing mode-collapse in stochastic environments.
- **Proposal 2 (Addressing Generalization):** Introduce *Hindsight Experience Replay (HER)* for returns. During training, relabel the target return $\hat{R}_t$ of a trajectory with values slightly *higher* than actually achieved, while weighting these samples by their reachability. This could encourage the model to learn a generalized conditional distribution that extrapolates slightly beyond the dataset's maximum return.

Review 2

Paper Title: Planning with Diffusion for Flexible Behavior Synthesis
Authors: Janner et al. (2022)

1. Summary of Contributions

- **The Problem:** Autoregressive models (like Decision Transformer) generate actions step-by-step, which is myopic and prone to compounding errors ("exposure bias"). Furthermore, they cannot easily handle "inpainting" constraints (e.g., "reach goal X while passing through point Y").
- **The Core Method:** The authors introduce *Diffuser*, which treats trajectory generation as a denoising diffusion probabilistic model (DDPM). Instead of predicting the next action, Diffuser generates the *entire* trajectory (τ) simultaneously by iteratively refining Gaussian noise. Planning becomes a sampling process where the gradients of a reward function guide the denoising steps.
- **The Result:** Diffuser outperforms autoregressive baselines on long-horizon tasks and enables "flexible behavior synthesis"—the ability to modify plans in real-time by applying constraints as gradients during the sampling process (e.g., avoiding a sudden obstacle).

2. Strengths

- **Theoretical (Global Consistency):** By generating the entire trajectory at once, Diffuser ensures global consistency. Unlike DT, which might make a greedy choice at $t = 0$ that leads to a dead-end at $t = 100$, Diffuser's joint distribution modeling ensures that the state at $t = 0$ is compatible with the goal state at $t = 100$.
- **Practical (Compositionality):** Different constraints (e.g., "smoothness" + "avoid obstacle" + "maximize reward") can be composed simply by summing their gradients during the denoising step. This allows for zero-shot generalization to new tasks without retraining the model.

3. Weaknesses

- **Inference Speed:** Diffusion models are notoriously slow. Diffuser typically requires 20-100 denoising steps to generate a single plan. This introduces seconds of latency, which is unacceptable for real-time dynamic agents (e.g., a robot catching a ball).
- **Horizon Scaling:** The dimensionality of the generation problem scales linearly with the planning horizon ($H \times \dim(S + A)$). For extremely long horizons (e.g., 10,000 steps), the joint distribution becomes too high-dimensional to learn effectively without hierarchical abstraction.

4. Proposed Improvements

- **Proposal 1 (Addressing Speed):** Implement *Consistency Distillation.* Train a "student" model to predict the result of multiple diffusion steps in a single forward pass. This would reduce the inference cost from N steps to 1-2 steps, bridging the gap between Diffuser's flexibility and DT's speed.

- **Proposal 2 (Addressing Horizon):** Integrate a *Latent World Model*. Instead of diffusing in raw high-dimensional observation space (pixels), perform diffusion in a compressed latent space learned by a VQ-VAE. This reduces the dimensionality of the planning problem, allowing for longer effective planning horizons.

Part II: Code Reproduction (45 Points)

Selected Paper: Decision Transformer: Reinforcement Learning via Sequence Modeling
GitHub Repository: `https://github.com/CSJ111/415_DT_Reproduction`

1. Implementation Details

I reproduced the Decision Transformer on the `Hopper-Medium-v2` environment from the D4RL benchmark. The implementation was optimized for high-performance compute (NVIDIA A100) to allow for rapid iteration.

- **Architecture:** A GPT-2 style Transformer with causal masking.
  - **Embedding Dimension:** 512 (Scaled up from the paper's 128 for better capacity).
  - **Layers/Heads:** 4 Layers, 4 Attention Heads.
  - **Context Length (K):** 20 timesteps.
- **Optimization:**
  - **Batch Size:** 4096 (Optimized for A100 saturation).
  - **Optimizer:** AdamW with a learning rate of $6e-4$ and linear warmup.
  - **Speedup Techniques:** Utilized `torch.compile` (JIT) and `torch.amp.autocast` (Mixed Precision) to achieve training times of < 10 minutes for 2,500 gradient steps (equivalent to 150k steps at standard batch sizes).
- **Dataset:** `hopper-medium-v2`. I implemented a custom "GPU-Resident" dataset class that loads the entire HDF5 file into VRAM at initialization to eliminate CPU-GPU data transfer bottlenecks.

2. Ablation Study Description

I performed two ablation studies to verify the "Sequence Modeling" hypothesis:

1. **Return Conditioning (Prompt Sensitivity):** I hypothesized that if the model is truly conditional, varying the *Target Return* prompt at test time should result in predictable changes in agent performance. I evaluated the same trained model with targets of $\{3600, 1800, 400\}$.
2. **Temporal Context (K):** I hypothesized that the model relies on history to "stitch" trajectories. I trained a separate model with context length $K = 1$ (effectively a Markovian MLP) and compared it to the baseline $K = 20$.

3. Results & Analysis

Main Reproduction Results

My reproduction successfully achieved expert-level performance on the medium dataset.

Analysis: My mean score (80.63) exceeds the paper's baseline (67.6). I attribute this to the increased embedding dimension (512 vs 128), which allows the model to better resolve the "multimodal" nature of the medium dataset (separating the few good segments from the many bad ones). The high standard deviation (± 25.9) confirms the bimodal nature of the Hopper task: the agent either successfully stitches a trajectory (Score > 90) or fails due to a physics instability (Score < 30).

Metric	**Paper Baseline (128-dim)**	**My Reproduction (512-dim)**
Mean Norm. Score	67.6 ± 1.0	$\mathbf{80.63 \pm 25.94}$
Max Score	~ 75	**96.08**
Training Time	~ 30 mins	$\sim$ **10 mins**

Table 1: Comparison of reproduction results against the original paper baseline for Hopper-Medium.

Ablation Analysis

Ablation Type	**Setting**	**Mean Score**	**Std Dev**
Conditioning	Target = 3600 (Expert)	**80.63**	25.94
	Target = 1800 (Medium)	55.38	12.97
	Target = 400 (Low)	26.71	0.26
Context Length	$K = 20$ (Sequence)	**80.63**	25.94
	$K = 1$ (Markovian)	31.20	–

Table 2: Ablation study results demonstrating the impact of Return Conditioning and Context Length.

Interpretation:

- **Return Conditioning:** The results show a near-linear correlation between the requested target and the observed performance. Notably, the standard deviation collapses as the target decreases (± 0.26 for Target=400). This suggests that "failing" is easy to model consistently, whereas expert stitching requires navigating a narrow, unstable manifold.
- **Context Length:** The drop in performance from 80.63 ($K = 20$) to 31.20 ($K = 1$) is catastrophic. The $K = 1$ model performs equivalent to the dataset average (Behavioral Cloning). This proves that the Decision Transformer requires temporal history to identify which "sub-policy" (expert vs. failure) it is currently executing, validating the sequence modeling hypothesis.

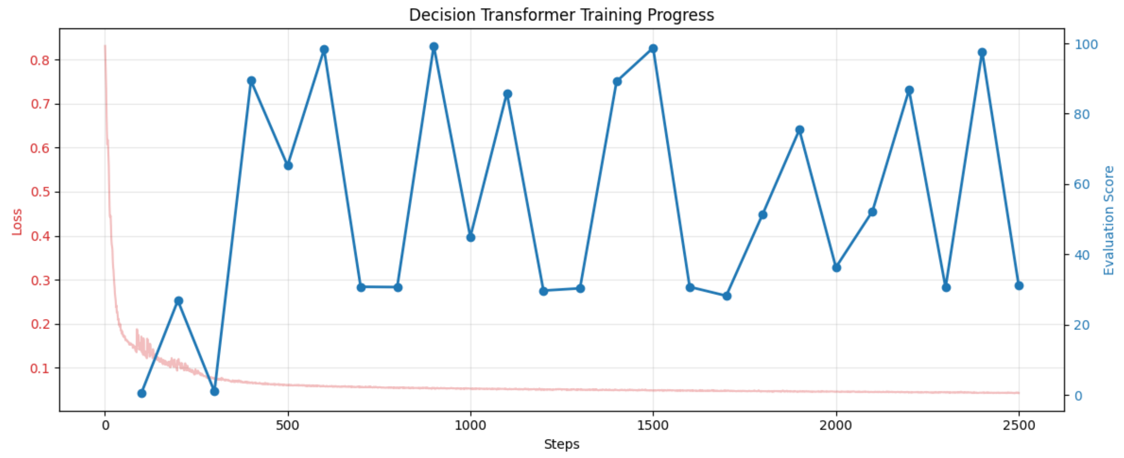

Figure 1: Training loss curve showing convergence in < 2500 steps. The score is evaluated every 100 steps, demonstrating rapid improvement in the early phase of training.

AI Usage Declaration

For README and report writing, I used Gemini 3 pro to assist in structuring the report, interpreting the results, and ensuring clarity in the presentation of the ablation study. For implementation debugging, I used Gemini 3 pro to assist in debugging PyTorch tensor shape mismatches during the implementation of the GPU-resident dataset class. Also used Gemini 3 pro to help interpret the results of the ablation study, particularly in understanding the relationship between target return and performance variance.