Towards A Machine Capable of Learning And Discovering Everything

by

Hao Liu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Pieter Abbeel, Chair
Professor Alane Suhr
Professor Matei Zaharia
Professor Kaiming He

Spring 2024

Towards A Machine Capable of Learning And Discovering Everything

Abstract

Towards A Machine Capable of Learning And Discovering Everything

by

Hao Liu

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Pieter Abbeel, Chair

Large generative models have led to amazing results and revolutionized artificial intelligence. In this dissertation, I will discuss my research on advancing the foundation of these models, centered around addressing the bottlenecks of learning from any existing data and challenges of discovering to go beyond existing knowledge. First, I will describe our efforts to remove context size limitations of the transformer architecture. Our modeling and training methodologies, including BlockwiseTransformer and RingAttention, allow for near-infinite context sizes while maintaining scalability. I will then discuss the applications of large contexts in learning world model and decision-making. This includes Large World Model, the world's first AI with million tokens context for modeling text, image, and hour-long video at the same time. Next, I will introduce my research on discovering that allows AI to discover data and learn. I will discuss our work on learning skills in gameplay without human specifying domain knowledge, paving the road for learning beyond imitating existing data. Finally, I will envision the next generation of large generative models we should build, focusing on advances in efficient scaling, reasoning, and discovering in general domains.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my heartfelt gratitude to my advisor, Pieter Abbeel. Pieter is an amazing mentor, consistently offering both guidance and the freedom to pursue my research interests. Pieter taught me how to really do research and how to communicate research findings through papers, among many other valuable lessons. His insights and genuine enthusiasm about our research have been invaluable. The meetings with him were always the highlight of my week. He gave me his help and support whenever it was needed. I am so grateful for his encouragement, confidence in me, and his mentorship over the past several years. I feel fortunate to have been his student.

Thank you to my committee members Alane Suhr, Matei Zaharia, Kaiming He, and Jiantao Jiao for their support, guidance, and fruitful conversations. I would like to thank Jiantao for providing helpful feedback. Alane, thank you for your insightful discussions and feedback. Kaiming's discussions have helped clarify my thoughts on research and career, and his guidance has been highly valuable. I am deeply grateful for the discussions with Matei, who always provided valuable input on my research ideas and shared brilliant insights.

I'm grateful to have had the opportunity to spend a few years interning and working part-time at Google Brain and DeepMind, hosted by Volodymyr Mnih, Lisa Lee, Satinder Singh, and Tom Zachvy. I was fortunate enough to have tremendous freedom from them during my internship, as well as support from and thoughtful conversations with many, including Jeff, Sharad, Federico, Hieu, Bertrand, and Jonathan.

I'm glad to have had the opportunity to collaborate with an amazing set of students, faculty, and researchers throughout my PhD. Thank you to all of the talented researchers in Berkeley AI Research for fostering a collaborative and friendly environment, particularly to my labmates, Misha Laskin, Xinyang Geng, Kimin Lee, Denis Yarats, Fangchen Liu, Carmelo Sferrazza, Wilson Yan, Yuqing Du, Ajay Jain, Younggyo Seo, Olivia Watkins and many others. I am also grateful for the support from the EECS and BAIR administration. I would like to thank San Mateo for being home for the last several years.

During my undergraduate studies, I had the opportunity to meet a bunch of amazing people: Richard Socher, Qiang Liu, Denny Zhou, Zenglin Xu, Yuandong Tian, and Caiming Xiong. I was very fortunate to work with Qiang and am grateful for his guidance. I had a great experience working with Richard, who provided insightful advice. I also had the fortune to work with Caiming. I enjoyed the conversations with Denny, who supported me in various aspects. Additionally, I appreciate Yuandong for his help and many insightful discussions. My undergraduate adviser, Zenglin, inspired me to pursue a career in research, for which I am deeply thankful.

I would like to thank my grandmother and late grandfather for their unconditional love since my childhood, my parents and sisters for their unwavering support, and my partner, Xinyi, for more than I can possibly express.

Chapter 1

Introduction

1.1 Long-Term Vision

The long-term research vision of mine is to build a machine capable of everything, that includes learning from all existing data and discovering to go beyond existing data, ultimately outperform human intelligence in many domains.

General: Large language models such as ChatGPT have achieved amazing results in AI, but are limited to text domain. The world is much more complex than just text, in order for AI to be maximally powerful, AI should excel at many domains and modalities.

Unsupervised: While large language models show incredible results, they heavily rely on human to curate training data, either by filtering data from internet or writing down training examples. It is not scalable to rely on human to generate training data. AI should discover data and learn on it own.

1.2 Modeling for Learning from Any Existing Data

This section includes the research on modeling – resolving the long-standing challenge in Transformer [304].

Transformer is the foundation of AlphaFold [138], ChatGPT, Sora [38], among other AI successes. The primary reason is because transformer scales well with compute – training larger transformer on more data continues to improve performance. Yet, a major long-standing challenge of transformer, or AI more broadly, is that transformer can only process short sequence. This is due to transformer has large memory cost on long sequence. This limitation greatly restrict transformer to learn only from short sequences such as wikipedia article, but no way to learn from books, entire codebase, or videos. A general AI system should excel in many domains and capable of learning everything.

To address this challenge, my research pioneered a series of modeling and training methodologies to resolve it, including BlockwiseTransformer (Section 1.2.1) and RingAttention (Section 1.2.2). Utilizing these techniques, we propose Large World Model (Section 1.2.3), a general AI solution for modeling text, image, and hour-long video, all three modalities together, by any-to-any autoregressive prediction on million-length sequences. Large context shows effectiveness in decision-making too, our Agentic Transformer (Section 1.2.4) learns to improve across multiple trajectories by conditioning on prior experience.

1.2.1 BlockwiseTransformer Chapter 2

Our proposed research BlockwiseTransformer [183] reduces transformer memory cost, allowing over 100K tokens sequence length for the first time.

1.2.2 RingAttention Chapter 3

Our RingAttention [189] scales BlockwiseTransformer to near-infinite context, enabling AI to not just learn from wikipedia article, but also books, entire codebase, videos, robotics trajectories, or even human genomes. This resolves the long-standing major challenge in AI – for the first time, it becomes possible for Transformer to learn from arbitrary complex sequences.

1.2.3 Large World Model Chapter 4

Utilizing these techniques, we pioneer and address a key challenge in AI – how to build general AI model capable of learning from many domains and modalities? Now that large context becomes possible, our proposed research Large World Model [188] offers a general solution. While these have been a lot of prior work trying to learn from many modalities, they require domain specific design. For instance, ChatGPT with Dalle can process image and text, and can predict image and text, but require domain-specific training and architectural design. Our large world model can process and predict on all three modalities, text, image, and video.

1.2.4 Agentic Transformer Chapter 5

Large context is effective in modeling text, image, and video. In this research, we show large context makes it possible for transformer to improve autonomously by conditioning on past trajectories. In our proposed research Agentic Transformer [182], we sort an arbitrary set of trajectories by their returns, from low to high, this forms the input sequence with increasing returns. We further replace the goals in lower return trajectories with the goal of highest return trajectory. This increasing return input sequence encourages model to get higher and higher return across trajectories. This enables transformer to use large context for learning

decision making effectively, outperforming more complicated domain-specific algorithms for the first time.

1.2.5 Related Research

In addition to pioneering and solving challenges in modeling, I have worked on other relevant parts of the goal. This includes decision making [177, 260, 142, 96]. Connecting text and vision without paired data for the first time [187]. I also build open-source large language models as foundation for research [97, 95].

1.3 Discovering for Going Beyond Existing Knowledge

This section includes the research on enabling AI to discover data on its own.

While large language models such as ChatGPT have achieved amazing results. They heavily rely on human to curate training data. This requirement poses a huge challenge ahead because human curation is not scalable, requiring domain knowledge and scaling law requires exponential more data to achieve tiny improvement. AI can be better if we allow it to go beyond imitation, such as AlphaGo. In the match between AlphaGo and world champion Lee Sedol, during the second game, AlphaGo's 37th move is highly unconventional, at the moment, confused experts that they believe this is a big mistake by AlphaGo, Lee Sedol is going to win. Yet, as the game unfolded, move 37 turned out to be a strategically brilliant move, opening up opportunities that can only be seen at much later stages. AlphaGo shows the AI can learn successful Go strategies by not imitating human plays. But, it is restricted to Go and not compatible with continuous domains which are everywhere, such as computer control and robotics.

My research has started to enable AI to discover and learn in more general domains. This includes discovering in gameplay – AI can learn skills and play games without human specifying domain knowledge, as shown in APT (Section 1.3.1), APS (Section 1.3.2), and CIC (Section 1.3.3). As well as discovering in language – improving large language models reasoning capabilities with AI discovered data (Section 1.3.4).

1.3.1 Unsupervised Active Pretraining (APT) Chapter 6

Typically, human provides task reward or demonstration, then learning agent is done with reinforcement learning or imitation learning. The research question becomes what is the objective for learning if there is no task reward and no demonstration.

In our proposed research Active Pretraining (APT) [180], we propose the agent to interact with the world. This consists of collecting data from the world, and learning an internal representation about the world. The goal is to learn a policy that takes an action given the world and agent's internal representation.

For the first time, APT enables learning skills and playing a wide range of challenging continuous control games without human specifying domain knowledge, while other prior work all get stuck no matter how many steps of learning.

1.3.2 Active Pretraining with Successor Features (APS) Chapter 7

In our proposed research Active Pretraining with Successor Features(APS) [178], we propose to use successor features to guide representation learning and data collection in APT [180]. Successor feature represents a wide range of reward functions as a linear combination of learned representation. Since there is a neural network around learned representation, successor feature offers the same expressibility. Our research shows successor features with certain parameterization techniques proposed in the paper, allowing much more efficient discovering the world and learning representation.

APS achieves four times higher Atari game play score than DQN and other prior state-of-the-arts using one hundred times fewer interactions.

1.3.3 Contrastive Intrinsic Control Chapter 8

Building upon APT and APS, we introduce Contrastive Intrinsic Control (CIC), an algorithm for unsupervised skill discovery that maximizes the mutual information between state-transitions and latent skill vectors. CIC utilizes contrastive learning between state-transitions and skills to learn behavior embeddings and maximizes the entropy of these embeddings as an intrinsic reward to encourage behavioral diversity. CIC substantially improves over prior methods in terms of adaptation efficiency, outperforming prior unsupervised skill discovery methods substantially.

1.3.4 Exploration with Principles Chapter 9

Discovering is not only useful for Go or gameplay, our proposed research Exploratory AI (EAI), shows discovering can substantially improve large language models reasoning.

Large language models, while impressive, struggle with even simple reasoning. OpenAI paper [65] shows that even the largest GPT3 model requires human to write 2000 training examples to get 35% accuracy on grade school math. In order to get over 80%, it would require 100 times more data.

These large models heavily rely on human to curate training data, which is not scalable and expensive. In our proposed research EAI, we took inspiration from how human generate data by looking at existing data and following given principles or guidelines. Now that large language models already good at understanding language, we would like to replace human with AI to curate training automatically. Using the techniques proposed in the paper, we

show EAI can discover much more diverse data than human. By finetuning on the discovered data, large language models reasoning is substantially improved.

1.3.5 Related Research

During the pursuit of discovering in general domains, I also worked on learning from human feedback [186, 142, 158, 84]. Benchmarks for evaluating discovering in gameplay [325, 154], and applications of discovering in learning visual representations [185].

Chapter 2

BlockwiseTransformer to Reduce Transformer Memory Cost

2.1 Introduction

Transformers [304] have become the backbone of many state-of-the-art natural language processing models [74, 237, 39, 193]. They have demonstrated impressive performance across a wide range of AI problems, including language modeling, machine translation, image captioning, and protein folding [217, 249, 173, 237, 39, 243, 58]. Transformers achieve this success through their architecture design that uses self-attention and position-wise feedforward mechanisms. These components facilitate the efficient capture of long-range dependencies between input tokens, enabling scalability in terms of context length and model size through highly parallel computations.

However, the memory requirements of Transformers limit their ability to handle long sequences, which is necessary for many AI problems, such as high-resolution images, podcasts, code, or books and especially those that involve multiple long sequences or long-term dependencies [59, 51, 217, 51, 186, 156, 249, 173, 7]. The quadratic self-attention and the large feed forward network of Transformers require a large amount of memory, which makes it challenging to scale to longer input sequences. This limitation has led to various techniques proposed to reduce the memory requirements of Transformers, including sparse-approximation, low-rank approximation, and low precision approximation [see e.g. 291, 131, 125, 62, 145, 196, 311].

One distinct line of research does not rely on approximation but instead focuses on computing exact self-attention with linear memory complexity. This approach leverages the observation that the softmax matrix in self-attention can be computed without materializing the full matrix [201]. This technique has led to the development of FlashAttention [71] and Memory Efficient Attention [235]. Both methods propose a blockwise computation of the self-attention softmax, demonstrating reduced memory requirements.

Despite the resulting reduced memory requirements of the self-attention block in transformers models, a significant challenge still arises from the feedforward network. This network contains a large number of parameters and produces high-dimensional intermediate vectors, resulting in substantial memory requirements. This issue is becomes the key memory challenge once employing memory-efficient attention mechanisms. Consequently, training Transformers on longer context lengths and scaling up transformers models become significantly hindered due to the overwhelming memory demands imposed by the feedforward network.

To address this challenge, we make an important observation: when self-attention is computed in a blockwise manner to reduce memory requirements, it becomes feasible to merge the computation of the feedforward network. This eliminates the need to wait for the self-attention computation to finish before performing the feedforward step on the entire sequence. By computing the feedforward network on a block-by-block basis, we effectively reduce the memory cost associated with the feedforward network. This process involves the utilization of two nested loops over the input sequence blocks. In the outer loop, we iterate over each block and compute the query. In the inner loop, we iterate over each block to calculate the key and value. These key-value pairs, along with the query, are then used to compute the blockwise attention specific to the corresponding input block. This blockwise attention is subsequently used to calculate the output of the feedforward network, followed by a residual connection. This approach enables us to process longer input sequences while maintaining lower memory budget. Since our approach performs blockwise parallel computation and fuses the feedforward and self-attention computations, we name our method the Blockwise Parallel transformers (BPT).

We evaluate the effectiveness of our approach on several benchmarks, including language modeling and reinforcement learning. Our experiments show that BPT can reduce the memory requirements of Transformers, enabling us to train 32 times longer sequence than vanilla attention [304] based GPT models and up to 4 times longer sequence than prior state-of-the-arts FlashAttention [71] and Memory Efficient Attention [235]. Furthermore, we demonstrate the application of BPT on the task of traning transformers based RL agent. By conditioning on multiple trajectories, BPT significantly improves the performance and achieves better results on challenging RL benchmarks. We believe that our approach has the potential to enable the training and evaluation of more complex models that require longer input sequences, which could lead to further breakthroughs in AI research.

Our contributions are twofold: (a) proposing a blockwise computation of self-attention and feedforward approach that enables 32 times longer and up to 4 times longer context lengths than vanilla transformers and previous memory-efficient Transformers, respectively, and (b) demonstrating the effectiveness of our approach through extensive experiments.

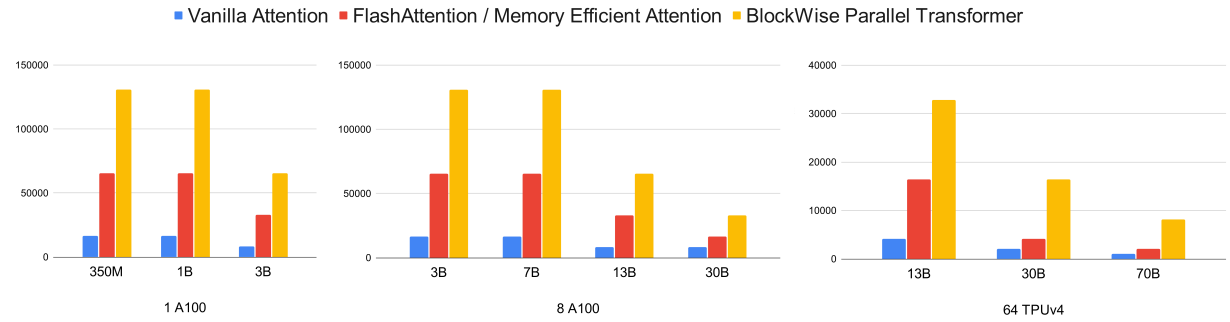


Figure 2.1 Maximum context length during training time with the GPT model using different methods. Model sizes range from 1B to 70B. Figures (A), (B), and (C) show evaluation using one, eight A100, and 64 TPUv4, respectively, with a single sequence. Our method enables training sequences 32 times longer than vanilla attention-based transformers [304], and 2 to 4 times longer than FlashAttention [71] and Memory Efficient Attention [235]. Section 2.3.1 provides a memory cost breakdown.

2.2 Memory Bottleneck of Transformer

Given input sequences $Q, K, V \in \mathbb{R}^{s \times d}$ where s is the sequence length and d is the head dimension. We compute the matrix of outputs as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V, \tag{2.1}$$

where softmax is applied row-wise. Standard attention implementations materialize the matrices QK^T and softmax($\frac{QK^T}{\sqrt{d}}$) to HBM, which takes $O(s^2)$ memory, so the overall space complexity is $O(s^2)$. There has been a large body of work trying to reduce memory usage of self-attention by using online softmax [201, 235, 71] to reduce memory cost of self-attention by preventing it from full materialization. And these approaches reduce memory footprint from $O(s^2)$ to $O(s)$. However, the large feedforward layers have been overlooked.

In addition to attention sub-layers, each of the attention layers is accomplished with a fully connected feedforward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.2}$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. The large size of the feedforward network requires substantial memory resources, and this becomes even more pronounced when dealing with large context sizes. See Section 2.3.1 for analysis of memory cost associated with transformers.

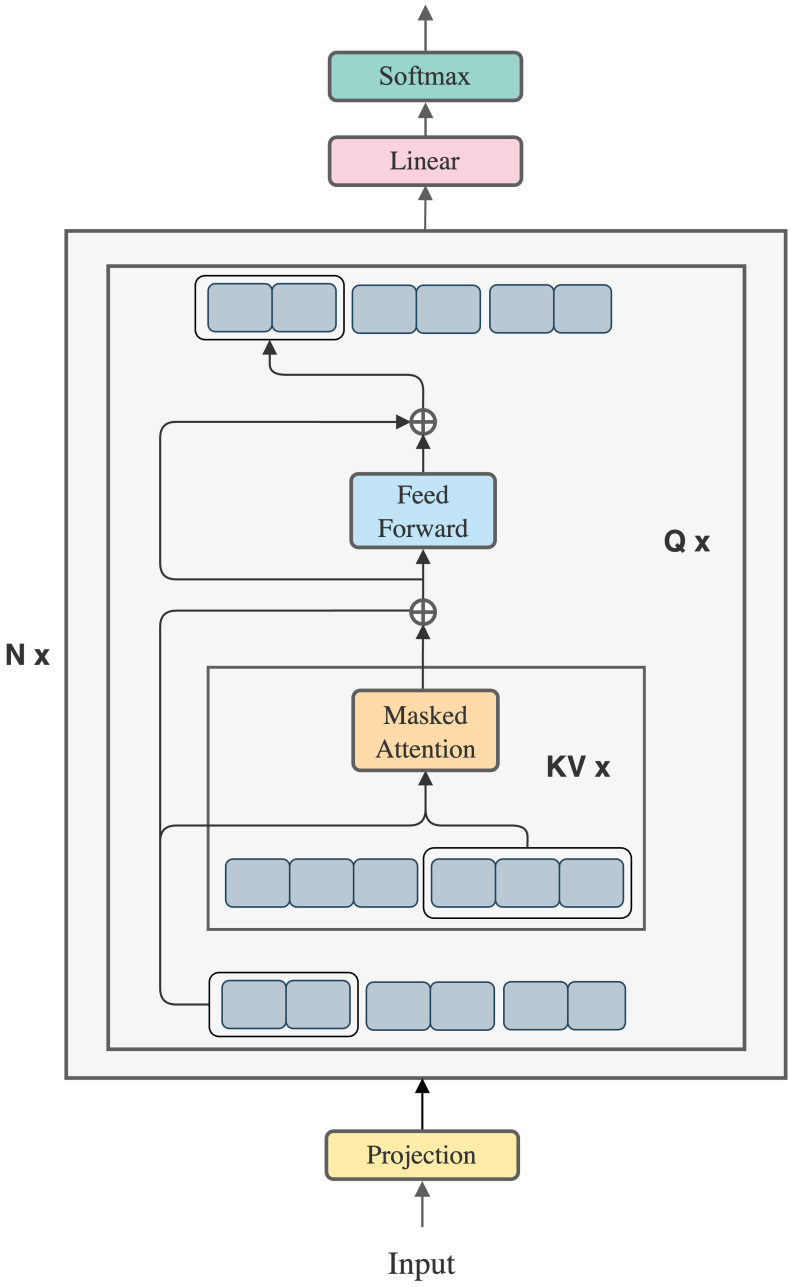


Figure 2.2 We use the same model architecture as the original transformers but with a different way of organizing the compute. In the diagram, we explain this by showing that for the bottom first incoming input block, we project it into query; then we iterate over the same input sequence positioned above the bottom row, and project it to key and value. These query, key and value are used to compute self-attention (yellow box), whose output is pass to feedforward network (cyan box), followed by a residual connection. In our proposed approach, this process is then repeated for the other incoming input blocks.

2.3 Blockwise Parallel for Large Context Models

Self-attention can be computed in a blockwise manner without materializing the softmax attention matrix $\text{softmax}(QK^T)$ [201, 71, 235]. This approach involves splitting the sequences $Q \in \mathbb{R}^{s\times d}$ into B_q blocks and sequences $K, V \in \mathbb{R}^{s\times d}$ into B_{kv} blocks. For each query block, the blockwise attention $\text{Attention}(Q, K, V)$ can be computed by iterating over all key-value blocks. Once the blockwise attention is computed, the global attention matrix can be obtained by scaling the blockwise attention using the difference between the blockwise and global softmax normalization constants [201]. This is achieved by keeping track of normalization statistics and combining them from all blocks to scale each block accordingly. For a specific query block Q_i, $1 \le i \le B_q$, the corresponding attention output can be computed by scaling each blockwise attention as follows:

$$\text{Attention}(Q_i, K, V) = \text{Scaling}(\{\exp(Q_i K_j^T)V_j\}_{j=1}^{B_{kv}}). \tag{2.3}$$

The scaling operation scales each blockwise attention based on the difference between the blockwise maximum and the global maximum:

$$\begin{aligned}
\text{Attention}(Q_i, K_j, V_j) &= \exp\big(Q_i K_j^T - \max(Q_i K_j^T)\big) / \sum \exp\big(Q_i K_j^T - \max(Q_i K_j^T)\big) \\
\max_i &= \max\big(\max(Q_i K_1^T), \ldots, \max(Q_i K_B^T)\big) \\
\text{Attention}(Q_i, K, V) &= \big[\exp(Q_i K_j^T - \max_i)\ \text{Attention}(Q_i, K_j, V_j)\big]_{j=1}^{B_{kv}}.
\end{aligned}$$

This blockwise self-attention computation eliminates the need to materialize the full attention matrix of size $O(n^2)$, resulting in significant memory savings.

We observe that the blockwise computation is not limited to self-attention but can also be applied to the feedforward network. For each query block, after iterating over the key and value blocks, the feedforward network can be computed along with a residual connection, completing the attention and feedforward network computation for that query block. This means that the model does not need to compute the feedforward network on the full sequence, but rather on intermediate blocks, resulting in memory savings. The computation for a query block is given by:

$$\text{Output}_i = \text{FFN}\big(\text{Attention}(Q_i, K, V) + Q_i\big) + \text{Attention}(Q_i, K, V) + Q_i.$$

Therefore, the output for each block consists of the feedforward network, self-attention, and residual connection computed in a blockwise manner.

It is worth mentioning that for large models, the memory cost of the feedforward network on the full sequence can be much larger than the memory efficient attention. Therefore computing the feedforward network on the same block as attention can significantly reduce memory cost, and it also reduces data movements, contributing to overall computational

Algorithm 1: Reduce memory cost with BPT.

Required: Input sequence x. Number of query blocks B_q. Number of key and value blocks B_{kv}.
Initialize
Project input sequence x into query, key and value.
Split query sequence into B_q of query input blocks.
Split key and value sequences into B_{kv} of key-value input blocks.
for $outer = 1$ **to** B_q **do**
  Choose the *outer*-th query.
  for $inner = 1$ **to** B_{kv} **do**
    Choose the *inner*-th key and *inner*-th value block.
    Compute attention using query, key and value, and record normalization statistics.
  end for
  Combine each blocks by scaling them to get attention output for the *outer*-th input block.
  Compute feedforward on attention output and add residual connection.
end for

efficiency. Moreover, we should remark that blockwise parallelism can be directly applied to the final cross entropy loss, which can further minimize memory cost. The full process of our framework, coined as BPT, is summarized in Algorithm 1.

2.3.1 Analysis of Memory Cost

We present an analysis of memory cost across different transformers architectures: the Vanilla Transformer, the memory-efficient / Flash Attention variant, and BPT.

Vanilla Transformers:

Attention: For Q, K, V, saving their input x needs $2bsh$ bytes, where b is batch size, s is sequence length, and h is hidden dimension. For QK^T matmul, saving activations Q and K needs $4bsh$ bytes. For softmax(QK^T), saving input QK^T needs $2bs^2a$ bytes, where a is the number of attention heads. For mask and dropout, saving mask needs bs^2a bytes. For score $\times V$, saving score needs $2bs^2a$ bytes, and saving V needs $2bsh$ bytes. For output projection and dropout, saving the input needs $2bsh$ bytes, and saving dropout mask needs bsh bytes. The maximum attention activation size of attention is $O(s^2)$ with checkpointing.

FFN: For the first linear layer, saving input needs $2bsh$ bytes. For activation, saving input needs $8bsh$ bytes. For the second linear layer, saving input needs $8bsh$ bytes. For dropout, saving the mask needs bsh bytes. With checkpointing, the maximum activation size of FFN is $8bsh$ bytes.

```python
def blockwise_ffn(remat_ffn, inputs, chunk_size, deterministic):
    # remat_ffn: a rematerialized ffn
    inputs = rearrange(inputs, 'b (c n) d -> b c n d', c=chunk_size)
    def scan_ffn(remat_ffn, carry, hidden_states):
        outputs = remat_ffn(hidden_states, deterministic=deterministic)
        return carry, outputs
    scan_axis = inputs.ndim - 2
    _, res = nn.scan(
        scan_ffn,
        variable_broadcast="params",
        split_rngs={"params": False, "dropout": True},
        in_axes=scan_axis,
        out_axes=scan_axis,
    )(remat_ffn, None, inputs)
    res = rearrange(res, 'b c n d -> b (c n) d')
    return res

def blockwise_attn(query, key, value, query_chunk_size,
        key_chunk_size, dtype, policy, precision, prevent_cse):
    query = query / jnp.sqrt(query.shape[-1]).astype(dtype)
    query = rearrange(query, 'b (c n) h d -> n b c h d', c=query_chunk_size)
    key, value = map(lambda t: rearrange(t, 'b (c n) h d -> n b c h d',
    c=key_chunk_size),  (key, value))
    num_q, batch, _, num_heads, dim_per_head = query.shape
    num_kv = key.shape[0]
    def scan_attention(args):
        query_chunk, query_chunk_idx = args
        @functools.partial(jax.checkpoint, prevent_cse=prevent_cse, policy=policy)
        def scan_kv_block(carry, args):
            key_chunk, value_chunk, key_chunk_idx = args
            (numerator, denominator, prev_max_score) = carry
            attn_weights = jnp.einsum('bqhd,bkhd->bqhk', query_chunk,
            key_chunk, precision=precision)
            bias_chunk = _chunk_bias_fn(query_chunk_idx, key_chunk_idx)
            bias_chunk = jnp.moveaxis(bias_chunk, 1, 2)
            attn_weights = attn_weights + bias_chunk

            max_score = jnp.max(attn_weights, axis=-1, keepdims=True)
            max_score = jnp.maximum(prev_max_score, max_score)
            max_score = jax.lax.stop_gradient(max_score)
            exp_weights = jnp.exp(attn_weights - max_score)
            exp_values = jnp.einsum(
                'bqhv,bvhf->bqhf', exp_weights, value_chunk, precision=precision
            )
            correction = jnp.exp(prev_max_score - max_score)
            numerator = numerator * correction + exp_values
            denominator = denominator * correction + exp_weights.sum(axis=-1, keepdims=True)
            return Carry(numerator, denominator, max_score), None
        init_carry = Carry(
            jnp.zeros((batch, query_chunk_size, num_heads, dim_per_head), dtype=query.dtype),
            jnp.zeros((batch, query_chunk_size, num_heads, dim_per_head), dtype=query.dtype),
            (-jnp.inf) * jnp.ones((batch, query_chunk_size, num_heads, 1), dtype=query.dtype),
        )
        (numerator, denominator, max_score), _ = lax.scan(
            scan_kv_block, init_carry, xs=(key, value, jnp.arange(0, num_kv))
        )
        outputs = (numerator / denominator).astype(dtype)
        return outputs
    _, res = lax.scan(
        lambda _, x: ((), scan_attention(x)),
        (), xs=(query, jnp.arange(0, num_q))
    )
    res = rearrange(res, 'n b c h d -> b (n c) h d')
    return res
```

Figure 2.3 The key parts of the implementation of Blockwise Parallel Transformers in Jax. The full code is provided on github.

Consequently, for a large context length, the memory cost of activation in vanilla transformers is $O(s^2)$.

BPT:

Attention: Since BPT does not materialize full attention and instead computes it blockwise, it needs to store intermediate blockwise activations in the key-value loop, which has a maximum activation size of $4bch$ with checkpointing. Additionally, it needs to store q output activations for the query loop, which requires $2bsh$ bytes. Since $s \gg c$, the maximum activation size is $2bsh$.

FFN: When iterating the FFN over blocks, BPT needs to save the following activations: For the first linear layer, saving input needs $2bch$ bytes. For activation, saving input needs $8bch$ bytes. For the second linear layer, saving input needs $8bch$ bytes. For dropout, saving the mask needs bch bytes. In total, $8bch$ bytes are needed. Additionally, storing the output of the for loop requires $2bsh$ bytes. Therefore, the maximum FFN activation size is $2bsh$ due to $2bsh > 8bch$.

Consequently, each BPT layer's memory cost of activation is $2bsh$.

Memory-Efficient / Flash Attention:

Attention: Similar to BPT attention, the maximum activation size is $2bsh$.

FFN: Similar to the vanilla FFN, the maximum activation size is $8bsh$.

Consequently, each Flash Attention layer's memory cost is $8bsh$.

Comparing the activation memory of Flash Attention/Memory-Efficient transformers with BPT, we see that BPT offers $8bsh/2bsh = 4$ times memory saving. By taking into account other factors of memory cost such as model parameters and optimizer states, BPT allows training with context lengths 2-4 times larger than prior state-of-the-arts.

2.3.2 Why Blockwise Parallel

The utilization of blockwise parallelization may raise questions about the effectiveness of running parallel computers, as computation can become sequential between blocks. However, the benefits of blockwise parallelization depend on the model size and hardware configuration. In cases where the model is large or the context length is extremely long, a block may reach its maximum arithmetic density, making it impractical to execute the original full-length sequence in parallel. In such scenarios, blockwise parallelization treats the long sequence as short ones, allowing dealing with large models and effectively enabling large context size. Moreover, using blockwise parallelization allows us to avoid waiting for the completion of self-attention and allocating a significant amount of memory solely for feed-forward network computation.

Another notable advantage of blockwise parallelization is its ability to leverage hardware with significantly faster SRAM speed compared to HBM speed. For instance, in Nvidia GPUs, SRAM is an order of magnitude faster than HBM, while in Google TPUs, SRAM also offers higher speed than HBM. By utilizing blockwise parallelization, we can tap into the increased speed of SRAM, thereby reducing communication costs and increasing throughput. This advantage aligns with memory efficient self-attention approaches [71, 235].

2.3.3 Implementation

Algorithm 1 provides the pseudocode of the algorithm. Figure 2.3 in Appendix shows a Jax implementation optimized for simplicity. The full code of BPT is provided on github. which supports large-scale distributed training of large context models using BPT.

The `blockwise_ffn` function begins by accepting a rematerialized feed forward module, inputs and chunk size. The `remat_ffn` compute feedforward on inputs with checkpointing, *i.e.*without saving intermediates. The `scan_ffn` function is then used to scan over input sequences and generate outputs.

The `blockwise_attn` function process query, key, and value to produce attention blockwise. The `scan_attention` function is defined, which computes the attention weights between the query vector and key-value pairs from another chunk. This is done by applying the `scan_kv_block` function to the key-value chunk, calculating the dot product between the query and key vectors, and then adding a bias term. The bias term introduces a positional bias between different chunks based on their indices without materializing the full matrix. The softmax function is then applied to the attention weights in a numerically stable manner, using the max-score trick to avoid large exponentiation results.

Finally, BPT combines the outputs from all chunks, normalizes them using their max-score-adjusted weights, and passes them through a feed-forward neural network (`blockwise_ffn`). The final output is the sum of the feed-forward output, the attention output, and the original input.

2.4 Setting

We evaluate the impact of using BPT in improving large transformers models by benchmarking memory requirement, maximum sequence length and throughout speed. We show apply BPT to reinforcement learning as an application.

Model Configuration. Our study is built upon the GPT architecture. Table 2.1 provides a overview of the model sizes considered in our experiments.

Baselines. We evaluate our method by comparing it with vanilla Transformer [304] which is denoted as "Vanilla", and FlashAttention [71] and Memory Efficient Attention [235] which

Table 2.1 Sizes and architectures of the models which we evaluated in experiments.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}
GPT 1B	1.3B	24	2048	16	128
GPT 3B	2.7B	32	2560	32	80
GPT 7B	6.7B	32	4096	32	128
GPT 13B	13.0B	40	5140	40	128
GPT 30B	30.0B	48	7168	56	128
GPT 70B	70.0B	80	8192	64	128

are state-of-the-art memory efficient attention, we denote them as "MemoryEfficient" in our experiments. All methods use the same gradient checkpointing in the experiments.

Datasets. We consider two datasets for evaluation purposes. Including pretraining on OpenWebText dataset and large context reinforcement learning on ExoRL.

- **OpenWebText.** The OpenWebText dataset [100] is a large and diverse collection of web pages that has been filtered and cleaned for use in natural language processing (NLP) tasks. The dataset consists of over 6 billion tokens from more than 40 million web pages, covering a wide range of topics and genres.
- **ExoRL.** The ExoRL [325] dataset is based on unlabeled exploratory data collected by running unsupervised RL algorithms. For each environment, it comes with eight different unsupervised data collection algorithms, taken from from URLB [154]. The datasets are collected by unsupervised RL and then relabeled using task reward function. The resulting mixed dataset consists of 8 millions timesteps (8000 episodes), with each episode spanning a length of 1000 steps.

Training Configuration. Our main baselines are vanilla attention [304], which computes self-attention by materializing the attention matrix and computes the feedforward network normally. We also consider two prior state-of-the-art memory-efficient methods, namely FlashAttention [71], which focuses on GPU efficiency, and Memory Efficient Attention [235], which focuses on TPU efficiency. Since they share a similar idea, for notation simplicity, we refer to them as FlashAttention in our experiments. We tune the block size for both the baselines and BPT, and report the best results achieved by each. The experiments are on NVIDIA 80GB A100 GPUs, we consider both single GPU for smaller model training and 8 GPUs settings for model parallel training. We also experiment with scaling up model on 64 TPUv4.

We note that no data parallelism is considered in our evaluations since our approach is independent of data parallelism. As a result, the batch sizes used in our analysis are much lower than the ones used for the end-to-end training. All of our results are obtained using full precision instead of mixed precision.

2.5 Results

In our experiments, our primary objective is to comprehensively evaluate the performance of BPT across multiple key metrics, including maximum sequence length, memory usage, and throughput. Moreover, we extend the applicability of BPT to reinforcement learning and evaluate its effectiveness in large context application.

Table 2.2 Maximum context length during training with different methods. BPT enables training 2-4 times longer sequence length than FlashAttention / Memory Efficient Attention, and up to 32 times longer sequence length than vanilla attention.

1 A100	**PartitionSpec**	**Vanilla Attention**	**MemoryEfficient**	**Blockwise Parallel**
350M	(1,1,1)	16K (16384)	65K (65536)	131K (131072)
1B	(1,1,1)	16K (16384)	65K (65536)	131K (131072)
3B	(1,1,1)	8K (8192)	16K (16384)	65K (65536)
8 A100	**PartitionSpec**	**Vanilla Attention**	**MemoryEfficient**	**Blockwise Parallel**
3B	(1,1,8)	16K (16384)	65K (65536)	131K (131072)
7B	(1,1,8)	16K (16384)	65K (65536)	131K (131072)
13B	(1,1,8)	8K (8192)	33K (32768)	65K (65536)
30B	(1,1,8)	8K (8192)	16K (16384)	65K (65536)
64 TPUv4	**PartitionSpec**	**Vanilla Attention**	**MemoryEfficient**	**Blockwise Parallel**
13B	(1,1,64)	4K (4096)	16K (16384)	33K (32768)
30B	(1,1,64)	2K (2048)	4K (4096)	16K (16384)
70B	(1,1,64)	1k (1024)	2K (2048)	8K (8192)

2.5.1 Evaluation of Context Length

We present experimental results comparing the maximum training sequence lengths achieved using three different attention mechanisms: Vanilla, MemoryEfficient, and Blockwise Parallel. Table 2.2 summarizes the findings. On one A100 GPU, Vanilla transformers supports a maximum training sequence length of 16K for 1B parameters and 8K for 3B parameters. In contrast, MemoryEfficient enables longer sequences of 65K for 1B parameters and 16K for 3B parameters. Notably, our proposed method, Blockwise Parallel, surpasses both methods, achieving a maximum sequence length of 131K for 1B parameters and 3B parameters. Moving on larger models, Blockwise Parallel again outperforms the other two methods, allowing training sequences of 65K for 30B large model on 8 GPUs and 8K for 70B large model on 64 TPUv4, which are two and four times longer than MemoryEfficient, respectively.

Table 2.3 shows the analysis of memory usage across different settings with three distinct approaches: Vanilla Transformer, MemoryEfficient, and our proposed method, BPT. It is

evident that Vanilla transformers consumes the highest amount of memory, while Memory-Efficient and BPT offer notable improvements in memory optimization. Notably, our BPT technique consistently outperforms both Vanilla transformers and MemoryEfficient in all settings, showcasing memory efficiency.

Table 2.3 Memory usage comparison for different settings. "oom" denotes out of memory.

Setting	**3B on A100**			**13B on 8 A100**		
Context Length	**Vanilla**	**MemoryEfficient**	**BPT**	**Vanilla**	**MemoryEfficient**	**BPT**
8192	64GB	44GB	43GB	59GB	44GB	42GB
16384	oom	47GB	45GB	oom	46GB	45GB
32768	oom	55GB	52GB	oom	55GB	52GB
65536	oom	75GB	70GB	oom	75GB	68GB
131072	oom	oom	79GB	oom	oom	78GB

2.5.2 Evaluation on Throughput and Speed

In Table 2.4, we present a comparison of the throughput achieved by different attention mechanisms on the GPT-XL (1B) model trained on the OpenWebText dataset using 8 GPUs. Throughput is measured as number of tokens processed per device per second. We evaluate the performance at various context lengths, including 2K, 8K, 16K, 33K, and 65K tokens. Our proposed method achieves competitive throughput as MemeoryEfficient mechanism, and surpasses the Vanilla transformer, achieving 1.17x speedup at context length 8k and 1.2x speedup at context length 16k. At context length 32K and 64K, our method maintains high throughput and training speed, while the alternatives cannot train due to running out of memory. This demonstrates the scalability and efficiency of our proposed method, allowing it to effectively handle large context lengths without compromising on throughput and training speed.

2.5.3 Evaluation on Reinforcement Learning

In this section, we present the results of applying BPT to improve the performance of transformers in reinforcement learning (RL). We report our results in Table 2.5, where we evaluate our proposed model on the ExoRL benchmark across six different tasks. On ExoRL, we report the cumulative return, as per ExoRL [325]. The numbers of BC, DT [49] and AT [182] are from the ExoRL and AT paper. AT + ME and AT + BPT numbers are run by ourselves. Since the ExoRL data is significantly more diverse than D4RL because it is collected using various unsupervised RL algorithms [154, 185, 178], it is found that TD learning performs best while behavior cloning struggles [325]. AT [182] shows that conditioning transformers on multiple trajectories with relabeled target return can significantly outperforms

Table 2.4 Throughput comparison on GPT-XL (1B) using OpenWebText dataset. Throughput is measured as tokens processed per second. 'oom' denotes running out of memory, 'na' denotes results not available because we early terminated these runs to reduce compute cost.

Model	Context Len	Val Loss	Throughput	Speed up
Vanila transformers	2048	2.46	3827	1x
MemoryEfficient	2048	2.46	4371	1.14x
Blockwise Parallel Transformers	2048	2.46	3985	1.04x
Vanila transformers	4096	2.44	2340	1x
MemoryEfficient	4096	2.44	2567	1.1x
Blockwise Parallel Transformers	4096	2.44	2687	1.15x
Vanila transformers	8192	2.43	2455	1x
MemoryEfficient	8192	2.43	2781	1.13x
Blockwise Parallel Transformers	8192	2.43	2875	1.17x
Vanila transformers	16384	2.41	1701	1x
MemoryEfficient	16384	2.41	1889	1.11x
Blockwise Parallel Transformers	16384	2.41	2045	1.2x
Vanila transformers	32768	oom	oom	oom
MemoryEfficient	32768	na	810	1x
Blockwise Parallel Transformers	32768	na	857	1.1x
Vanila transformers	65536	oom	oom	oom
MemoryEfficient	65536	oom	oom	oom
Blockwise Parallel Transformers	65536	na	600	1x

behavior cloning approaches BC-10% and DT, and achieves competitive results with TD learning. For more details, please refer to their papers. We are interested in applying BPT to improve the performance of AT by conditioning on a 32 trajectories rather than 4 trajectories in original work. It is worth noting that each trajectory has 1000×4 length where 1000 is sequence length while 4 is return-state-action-reward, making training 32 trajectories with 350M size model infeasible for both Vanilla and MemoryEfficient. Results in Table 2.5 show that, by scaling the sequence length, AT + BPT consistently outperforms the original transformers model in all six tasks, achieving a total average return of 111.13 compared to the original transformers model's total average return of 83.02

2.6 Related Work

Transformers have garnered significant attention in the field of natural language processing (NLP) and have become the basis for numerous state-of-the-art models. Several works have explored memory-efficient techniques to address the memory limitations of Transformers

Table 2.5 Application of BPT on improving transformers in RL. All the baselines use vanilla attention. AT + ME denotes using "MemoryEfficient". AT + BPT denotes using Blockwise Parallel Transformers.

ExoRL benchmark	**BC-10%**	**DT**	**AT**	**AT**	**AT + ME**	**AT + BPT**
Task			N Trajs = 4	N Trajs = 32	N Trajs = 32	N Trajs = 32
Walker Stand	52.91	34.54	68.55	oom	oom	95.45
Walker Run	34.81	49.82	88.56	oom	oom	105.88
Walker Walk	13.53	34.94	64.56	oom	oom	78.56
Cheetah Run	34.66	67.53	125.68	oom	oom	178.75
Jaco Reach	23.95	18.64	52.98	oom	oom	87.56
Cartpole Swingup	56.82	67.56	97.81	oom	oom	120.56
Total Average	36.11	45.51	83.02	oom	oom	111.13

and enable their application to longer input sequences. One line of research focuses on various approximation techniques or compressing along the sequence dimension [see e.g. 131, 63, 71, 31, 235, 311, 196, 145]. Other works explored replacing attention [103, 104, 232, 126, 30, 332, 218, 309]. Another line of work explores partitioning the large hidden dimension of the feedforward network into parts and retrieving only one part per token [164, 267, 85, 146, 335, 342]. Additionally, extending the context by attending over states from previous sequences has been explored [68, 239], as well as combining local and global contexts [118, 62]. For a comprehensive review of these techniques, we recommend referring to the surveys by Tay et al. [291], Narang et al. [210], Tay et al. [290]. Several studies explored sharding large model on distributed devices tensor, data, or sequence parallelism [269, 83, 321, 147, 338, 171, 244]. Ours shares similarities with the sequence parallelism [147] where sequences are distributed across devices, in contrast, ours implements blockwise computation on sequences for each device. This creates an orthogonal relationship between our method and sequence parallelism, allowing for straightforward combination. In addition, our methodology is compatible with both tensor and data parallelism. Another direction involves computing exact self-attention in a blockwise manner using the tiling technique [201]. This approach has led to the development of memory efficient attention mechanisms [71, 235]. In line with these advancements, our work falls into this category. We propose computing both the feedforward network and self-attention in a blockwise manner, resulting in a significant reduction in memory requirements.

2.7 Conclusion

In conclusion, we propose a blockwise parallelization approach to reduce the memory requirements of Transformers, the backbone of state-of-the-art AI models. Our approach enables processing longer input sequences while maintaining or improving performance. Through extensive experiments, we demonstrate its effectiveness, achieving up to 4x memory reduction

than memory-efficient Transformers. Our contributions include a practical method for large context sizes in large transformers models. With the increasing capability of hardware, larger models and longer context length are widely used in AI research. At the same time, as we are pushing up against physics and fabrication limits, it is more important to design scaling approaches as efficient as possible to scale up large models and large context size. Our approach holds promise for training and evaluating complex models with longer input sequences, potentially driving new breakthroughs in machine learning research.

Limitations and Future Work. Although our method achieves state-of-the-art low memory usage for transformers models, it does have some limitations that need to be addressed:

- *Optimal performance.* While our implementation prioritizes simplicity with high-level Jax operations, optimizing low-level operations is crucial for achieving optimal performance. In future work, we suggest considering porting our method to CUDA and OpenAI Triton to achieve minimal memory cost and maximum speedup.

Acknowledgements

This project is supported in part by ONR under N00014-21-1-2769. We would like to express our sincere appreciation to the members of the RLL Lab and Berkeley AI Lab, as well as Anselm Levskaya, Markus Rabe, Federico Lebron, Sharad Vikram, and Tri Dao for their valuable insights and contributions to this paper. We thank Google TPU Research Cloud for granting us access to TPUs.

2.8 Evaluation of Memory

In the experimental results presented in Section 2.5.1, we used model parallelism to partition the model across 8 GPUs or 64 TPUv4 units. Our evaluation focused on determining the maximum achievable sequence length, using a sequence number of one. For TPUs, we utilized its default training configuration, which involved performing matmul operations in `bfloat16` format with weight accumulation in `float32`. On the other hand, for GPUs, we adopted the default setup, where all operations were performed in `float32`.

To profile memory usage, we utilized `jax.profile` and repeated the evaluation 100 times, reporting the average results. We conducted a grid search for the optimal query block size and key-value block size, considering values from the set $[16, 64, 128, 512, 1024, 2048, 4096]$. For each method, we reported the lowest memory achieved.

2.9 Evaluation of Throughput

In the evaluation presented in Section 2.5.2, we split OpenWebText following the methodology of [9]. Throughput is measured as tokens per device per second. To ensure a fair comparison, we performed a grid search for the optimal query block size and key-value block size, considering values from the set $[16, 64, 128, 512, 1024, 2048, 4096]$. For gradient checkpointing [54], we additionally grid search among three commonly used checkpointing policies including `nothing_saveable`, `dots_saveable`, and `dots_with_no_batch_dims_saveable` for attention and use `nothing_saveable` for feedforward network (FFN). For more details, please refer to Jax documentation. We selected the best performing configuration for both baselines and our method.

The training was conducted using FSDP [83] and gradient accumulation. We used weight decay of 0.1 and utilized cosine learning rate decay with a maximum learning rate of $2.0 \times e^{-4}$. For sequence lengths of $2048, 4096, 8192, 16384$, the batch sizes in trajectories were set as $8, 4, 2, 1, 1$ respectively. We use gradient accumulation to accumulate batch size in tokens to 1 million per batch.

2.10 Evaluation on RL

Table 2.6 Hyperparameters used in RL evaluation.

Hyperparameter	**Value**
Number of layers	3
Number of attention heads	1
Embedding dimension	128
Activation function	ReLU
Batch size	64
Dropout	0.1
Learning rate	10^{-4}
Learning rate decay	Linear warmup for 10^5 steps
Grad norm clip	0.25
Weight decay	10^{-4}
Initial desired target return at test time	850 Walker Stand
	400 Walker Run
	900 Walker Walk
	350 Cheetah Run
	300 Jaco Reach
	800 Cartpole Swingup
Number of trajectories during training	$4 \rightarrow 32$
Number of trajectories at test time	$4 \rightarrow 16$

In the experiment presented in Section 2.5.3, we followed the prior work's setting for learning rate, batch size, and other hyperparameters, while modifying the number of trajectories. The specific hyperparameters are provided in Table 2.6. The original agentic transformers used 4 trajectories during training, we increase the number to 32.

During testing, increasing the number of trajectories has been shown to improve performance. However, performing autoregressive sampling over a large number of trajectories (e.g., $64 \times 1000 \times 4$ total number of tokens) can be computationally slow. To reduce the sampling time, we limited the rollout to 16 trajectories.

Chapter 3

RingAttention Scales BlockwiseTransformer to Infinite Context

3.1 Introduction

Transformers [304] have become the backbone of many state-of-the-art AI systems that have demonstrated impressive performance across a wide range of AI problems. Transformers achieve this success through their architecture design that uses self-attention and position-wise feedforward mechanisms. However, scaling up the context length of Transformers is a challenge [217], since the inherited architecture design of Transformers, *i.e.* the self-attention has memory cost quadratic in the input sequence length, which makes it challenging to scale to longer input sequences. Large context Transformers are essential for tackling a diverse array of AI challenges, ranging from processing books and high-resolution images to analyzing long videos and complex codebases. They excel at extracting information from the interconnected web and hyperlinked content, and are crucial for handling complex scientific experiment data. There have been emerging use cases of language models with significantly expanded context than before: GPT-3.5 [253] with context length 16K, GPT-4 [217] with context length 32k, MosaicML's MPT [206] with context length 65k, and Anthropic's Claude [12] with context length 100k.

Driven by the significance, there has been surging research interests in reducing memory cost. One line of research leverages the observation that the softmax matrix in self-attention can be computed without materializing the full matrix [201] which has led to the development of blockwise computation of self-attention and feedforward [235, 71, 183] without making approximations. Despite the reduced memory, a significant challenge still arises from storing the output of each layer. This necessity arises from self-attention's inherent nature, involving interactions among all elements (n to n interactions). The subsequent layer's self-attention relies on accessing all of the prior layer's outputs. Failing to do so would increase computational costs cubically, as every output

must be recomputed for each sequence element, rendering it impractical for longer sequences. These components facilitate the efficient capture of long-range dependencies between input tokens, and enable scalability through highly parallel computations. To put the memory demand in perspective, even when dealing with a batch size of 1, processing 100 million tokens requires over 1000GB of memory for a modest model with a hidden size of 1024. This is much greater than the capacity of contemporary GPUs and TPUs, which typically have less than 100GB of high-bandwidth memory (HBM).

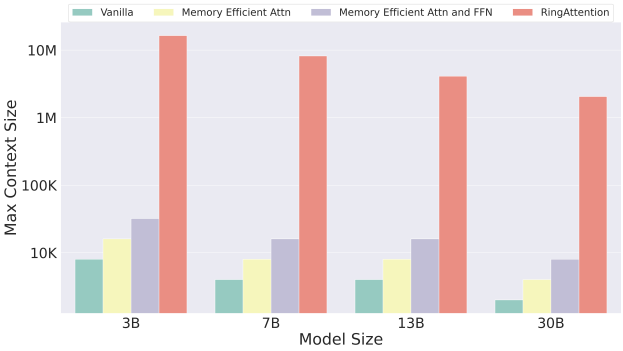


Figure 3.1 Maximum context length under end-to-end large-scale training on TPUv4-1024. Baselines are vanilla transformers [304], memory efficient transformers [235], and memory efficient attention and feedforward (blockwise parallel transformers) [183]. Our proposed approach Blockwise RingAttention allows training up to device count times longer sequence than baselines and enables the training of sequences that exceed millions in length without making approximations nor adding any overheads to communication and computation.

To tackle this challenge, we make a key observation: by performing self-attention and feedforward network computations in a blockwise fashion [183], we can distribute sequence dimensions across multiple devices, allowing concurrent computation and communication. This insight stems from the fact that when we compute the attention on a block-by-block basis, the results are invariant to the ordering of these blockwise computations. Our method distributes the outer loop of computing blockwise attention among hosts, with each device managing its respective input block. For the inner loop, every device computes blockwise attention and feedforward operations specific to its designated input block. Host devices form a conceptual ring, where during the inner loop, each device sends a copy of its key-value blocks being used for blockwise computation to the next device in the ring, while simultaneously receiving key-value blocks from the previous one. As long as block computations take longer than block transfers, overlapping these processes results in no added overhead compared to standard transformers. Our work utilizes blockwise parallel transformers [183] to substantially reduce memory costs, enabling zero-overhead scaling of context size across tens of millions of tokens during both training and inference, and allowing for the use of an arbitrarily large context size. Since our approach overlaps the communication of key-value blocks between hosts in a ring through blockwise computation of transformers, we name it RingAttention with Blockwise Transformers.

We evaluate the effectiveness of our approach on language modeling benchmarks. Our experiments show that RingAttention can reduce the memory requirements of Transformers, enabling us to train more than 500 times longer sequence than prior memory efficient

state-of-the-arts and enables the training of sequences that exceed 100 million in length without making approximations to attention. Importantly, RingAttention eliminates the memory constraints imposed by individual devices, empowering the training and inference of sequences with lengths that scale in proportion to the number of devices, essentially achieving near-infinite context size.

Our contributions are twofold: (a) proposing a memory efficient transformers architecture that allows the context length to scale linearly with the number of devices while maintaining performance, eliminating the memory bottleneck imposed by individual devices, and (b) demonstrating the effectiveness of our approach through extensive experiments.

3.2 Large Context Memory Constraint

Given input sequences $Q, K, V \in \mathbb{R}^{s \times d}$ where s is the sequence length and d is the head dimension. We compute the matrix of outputs as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V,$$

where softmax is applied row-wise. Each self-attention sub-layer is accompanied with a feedforward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2.$$

Blockwise Parallel Transformers. Prior state-of-the-arts have led to substantial reductions in memory utilization, achieved through innovative techniques that enable attention computation without full materialization by computing attention in a block by block manner [235, 71, 183]. These advancements lowered the memory overhead of attention to $2bsh$ bytes per layer, where b represents the batch size, s denotes the sequence length, and h stands for the hidden size of the model. To further reduce memory usage, blockwise parallel transformer (BPT) [183] introduced a strategy where the feedforward network associated with each self-attention sub-layer is computed in a block-wise fashion. This approach effectively limits the maximum activation size of feedforward network from $8bsh$ to $2bsh$. For a more detailed analysis of memory efficiency, please refer to the discussion provided therein. In summary, the state-of-the-art transformer layer's memory cost of activation is $2bsh$.

Large Output of Each Layer. While BPT significantly reduces memory demand in Transformers, it still presents a major challenge for scaling up context length because it requires storing the output of each layer. This storage is crucial due to the inherent nature of self-attention, which involves interactions among all elements (n to n interactions). Without these stored outputs, the subsequent layer's self-attention becomes computationally impractical, necessitating recomputation for each sequence element. To put it simply,

processing 100 million tokens with a batch size of 1 requires over 1000GB of memory even for a modest model with a hidden size of 1024. In contrast, modern GPUs and TPUs typically provide less than 100GB of high-bandwidth memory (HBM), and the prospects for significant HBM expansion are hindered by physical limitations and high manufacturing costs.

3.3 RingAttention with Blockwise Transformers

Our primary objective is to eliminates the memory constraints imposed by individual devices by efficiently distribute long sequences across multiple hosts without adding overhead. To achieve this goal, we propose an enhancement to the blockwise parallel transformers (BPT) framework [183]. When distributing an input sequence across different hosts, each host is responsible for running one element of the outer loop of blockwise attention corresponding to its designated block, as well as the feedforward network specific to that block. These operations do not necessitate communication with other hosts. However, a challenge arises in the inner loop, which involves key-value block interactions that require fetching blocks from other hosts. Since each host possesses only one key-value block, the naive approach of fetching blocks from other hosts results in two significant issues. Firstly, it introduces a computation delay as the system waits to receive the necessary key-value blocks. Secondly, the accumulation of key-value blocks leads to increased memory usage, which defeats the purpose of reducing memory cost.

Ring-Based Blockwise Transformer. To tackle the aforementioned challenges, we leverage the permutation invariance property of the inner loop's key-value block operations. This property stems from the fact that the self-attention between a query block and a group of key-value blocks can be computed in any order, as long as the statistics of each block are combined correctly for rescaling. We leverage this property by conceptualizing all hosts as forming a ring structure: host-1, host-2, ..., host-N. As we compute blockwise attention and feedforward, each host efficiently coordinates by concurrently sending key-value blocks being used for attention computation to the next host while receiving key-value blocks from the preceding host, effectively overlapping transferring of blocks with blockwise computation. Concretely, for any host-i, during the computation of attention between its query block and a key-value block, it concurrently sends key-value blocks to the next host-$(i+1)$ while receiving key-value blocks from the preceding host-$(i-1)$. If the computation time exceeds the time required for transferring key-value blocks, this results in no additional communication cost. This overlapping mechanism applies to both forward and backward passes of our approach since the same operations and techniques can be used. Prior work has also proposed leveraging a ring topology to compute full attention [172], aiming to reduce communication costs. Our work differs by utilizing blockwise parallel transformers to substantially reduce memory costs. As we show in the next section, this enables zero-overhead scaling of context size during both training and inference and allows arbitrarily large context size.

Arithmetic Intensity Between Hosts. In order to determine the minimal required block

Figure 3.2 **Top (a):** We use the same model architecture as the original Transformer but reorganize the compute. In the diagram, we explain this by showing that in a ring of hosts, each host holds one query block, and key-value blocks traverse through a ring of hosts for attention and feedforward computations in a block-by-block fashion. As we compute attention, each host sends key-value blocks to the next host while receives key-value blocks from the preceding host. The communication is overlapped with the computation of blockwise attention and feedforward. **Bottom (b):** We compute the original Transformer block-by-block. Each host is responsible for one iteration of the query's outer loop, while the key-value blocks rotate among the hosts. As visualized, a device starts with the first query block on the left; then we iterate over the key-value blocks sequence positioned horizontally. The query block, combined with the key-value blocks, are used to compute self-attention (yellow box), whose output is pass to feedforward network (cyan box).

Table 3.1 Comparison of maximum activation sizes among different Transformer architectures. Here, b is batch size, h is hidden dimension, n is number of head, s is sequence length, c is block size, the block size (c) is independent of the input sequence length (s). The comparison is between vanilla transformer [304], memory efficient attention [235, 71], blockwise parallel transformers [183], and our proposed approach RingAttention. Numbers are shown in bytes per layer, assuming *bfloat16* precision.

Layer Type	Self-Attention	FeedForward	Total
Vanilla Transformers	$2bns^2$	$8bsh$	$2bhs^2$
Memory Efficient Attention	$2bsh + 4bch$	$8bsh$	$8bsh$
Blockwise Parallel Transformers	$2bsh$	$2bsh$	$2bsh$
Blockwise RingAttention	$6bch$	$2bch$	$6bch$

size to overlap transferring with computation, assume that each host has F FLOPS and that the bandwidth between hosts is denoted as B. It's worth noting that our approach involves interactions only with the immediately previous and next hosts in a circular configuration, thus our analysis applies to both GPU all-to-all topology and TPU torus topology. Let's consider the variables: block size denoted as c and hidden size as d. When computing blockwise self-attention, we require $2dc^2$ FLOPs for calculating attention scores using queries and keys, and an additional $2dc^2$ FLOPs for multiplying these attention scores by values. In total, the computation demands amount to $4dc^2$ FLOPs. We exclude the projection of queries, keys, and values, as well as blockwise feedforward operations, since they only add compute complexity without any communication costs between hosts. This simplification leads to more stringent condition and does not compromise the validity of our approach. On the communication front, both key and value blocks require a total of $2cd$ bytes. Thus, the combined communication demand is $4cd$ bytes. To achieve an overlap between communication and computation, the following condition must hold: $4dc^2/F \geq 4cd/B$. This implies that the block size, denoted as c, should be greater than or equal to F/B. Effectively, this means that the block size needs to be larger than the ratio of FLOPs over bandwidth.

Memory Requirement. A host needs to store multiple blocks, including one block size to store the current query block, two block sizes for the current key and value blocks, and two block sizes for receiving key and value blocks. Furthermore, storing the output of blockwise attention and feedforward necessitates one block size, as the output retains the shape of the query block. Therefore, a total of six blocks are required, which translates to $6bch$ bytes of memory. It's worth noting that the blockwise feedforward network has a maximum activation size of $2bch$ [183]. Consequently, the total maximum activation size remains at $6bch$ bytes. Table 3.1 provides a detailed comparison of the memory costs between our method and other approaches. Notably, our method exhibits the advantage of linear memory scaling with respect to the block size c, and is independent of the input sequence length s.

Table 3.2 Minimal sequence length needed on each device. Interconnect Bandwidth is the unidirectional bandwidth between hosts, *i.e.*, NVLink / InfiniBand bandwidth between GPUs and ICI bandwidth between TPUs. The minimal block size required $c = \text{FLOPS}/\text{Bandwidth}$, and minimal sequence length $s = 6c$.

Spec Per Host	FLOPS	HBM	Interconnect Bandwidth	Minimal Blocksize	Minimal Sequence Len
	(TF)	(GB)	(GB/s)	(×1e3)	(×1e3)
A100 NVLink	312	80	300	1.0	6.2
A100 InfiniBand	312	80	12.5	24.5	149.5
TPU v3	123	16	112	1.1	6.6
TPU v4	275	32	268	1.0	6.2
TPU v5e	196	16	186	1.1	6.3

Our analysis shows that the model needs to have a sequence length of $s = 6c$, which is six times the minimal block size. Requirements for popular computing servers are shown in Table 3.2. The required minimal sequence length (rightmost column) for each host varies between 6K and 10K, and the minimal block size (second-to-rightmost column) for each host is around 1K for TPUs and GPUs with high bandwidth interconnect. For GPUs connected via InfiniBand, which offers lower bandwidth, the requirements are more strict. These requirements are easy to meet using blockwise transformers [183] and standard parallelism such as data and tensor parallelism, which we will show in experiment Section 3.5.

Algorithm and Implementation. Algorithm 2 provides the pseudocode of the algorithm. RingAttention is compatible with existing code for memory efficient transformers: RingAttention just needs to call whatever available memory efficient computation locally on each host, and overlap the communication of key-value blocks between hosts with blockwise computation. We use collective operation `jax.lax.ppermute` to send and receive key value blocks between nearby hosts. A Jax implementation is provided in Appendix 3.8.

3.4 Setting

We evaluate the impact of using RingAttention in improving Transformer models by benchmarking maximum sequence length and model flops utilization.

Model Configuration. Our study is built upon the LLaMA architecture, we consider 3B, 7B, 13B, and 30B model sizes in our experiments.

Baselines. We evaluate our method by comparing it with vanilla transformers [304] which computes self-attention by materializing the attention matrix and computes the feedforward network normally, transformers with memory efficient attention [235] and its efficient

Algorithm 2: Large Context Transformers using RingAttention with Blockwise Transformers.

Required: Input sequence x. Number of hosts N_h.
Initialize
Split input sequence into N_h blocks that each host has one input block.
Compute query, key, and value for its input block on each host.
for Each transformer layer **do**
  for $count = 1$ **to** $N_h - 1$ **do**
    for For each host concurrently. **do**
      Compute memory efficient attention incrementally using local query, key, value blocks.
      Send key and value blocks to next host and receive key and value blocks from previous host.
    end for
  end for
  for For each host concurrently. **do**
    Compute memory efficient feedforward using local attention output.
  end for
end for

CUDA implementation [71], and transformers with both memory efficient attention and feedforward [183].

Training Configuration. For all methods, we apply full gradient checkpointing [54] to both attention and feedforward, following prior works [235, 183]. The experiments are on both GPUs and TPUs. For GPUs, we consider both single DGX A100 server with 8 GPUs and distributed 32 A100 GPUs. We also experiment with TPUs, from older generations TPUv3 to newer generations of TPUv4 and TPUv5e. We note that all of our results are obtained using full precision instead of mixed precision.

3.5 Results

In our experiments, our primary objective is to comprehensively evaluate the performance of RingAttention across multiple key metrics, including maximum supported sequence length within accelerator memory, model flops utilization, and throughput. We compare RingAttention's performance with several baseline models , including the vanilla transformers [304], transformers with memory efficient attention [235], and transformers with both memory efficient attention and feedforward [183], across different model sizes and accelerator configurations.

3.5.1 Evaluating Max Context Size

We evaluate maximum supported context length using fully sharded tensor parallelsim (FSDP) [83] which is widely used in prior end-to-end training [296, 95]. We note that no tensor parallelism is considered in our evaluations since our approach is independent of tensor parallelism. Practitioners can combine our method with tensor parallelism, which we will show in Section 3.5.2. Using FSDP allows us to set the same batch size in tokens for baselines and our approach, ensuring a fair comparison. Concretely, on n devices, FSDP is used to shard the model for baselines, which gives a sequence length of l. The total batch size in tokens is nl. We utilize FSDP along with RingAttention to extend the sequence length to $\frac{nl}{m}$ and m sequences. This means that the total batch size in tokens remains the same, but RingAttention enables a significantly larger context size. Table 3.3 summarizes the results of our experiments.

Our RingAttention model consistently surpasses baselines, delivering superior scalability across diverse hardware setups. For example, with 32 A100 GPUs, we achieve over 1 million tokens in context size for 7B model, a 32 times improvement over previous best. Furthermore, when utilizing larger accelerators like TPUv4-512, RingAttention enables a 256 times increase in context size, allows training sequences of over 30 million tokens. Furthermore, our RingAttention model scales linearly with the number of devices, as demonstrated by the 8x improvement over previous best on 8 A100 and the 256x improvement on TPUv3-512. If a model can be trained with context size s on n GPUs using the blockwise attention and feedforward, with our RingAttention approach, it becomes possible to train a model with a context size of ns.

3.5.2 Evaluating Model Flops Utilization

We evaluate the model flops utilization (MFU) of RingAttention in standard training settings using fully sharded data parallelism(FSDP) [83] and tensor parallelism following LLaMA and OpenLLaMA [296, 95] with Jax SPMD. The batch size in tokens are 2M on 8/32x A100 and 4M on TPUv4-256. Our goal is investigating the impact of model size and context length on MFU, a critical performance metrics while highlighting the benefits of our approach. Table 3.4 presents the results of our experiments on MFU for different model sizes and context lengths. We present the achieved MFU using state-of-the-art memory efficient transformers BPT [183], compare it to our anticipated MFU based on these results, and demonstrate the actual MFU obtained with our approach (RingAttention). For fair comparison, both BPT and our approach are based on the same BPT implementation on both GPUs and TPUs.

RingAttention trains much longer context sizes for self-attention, resulting in higher self-attention FLOPs compared to baseline models. Since self-attention has a lower MFU than feedforward, RingAttention is expected to have a lower MFU than the baseline models. Our method offers a clear advantage in terms of maintaining MFU while enabling training with significantly longer context lengths. As shown in Table 3.4, when comparing our approach

Table 3.3 The maximum context length supported in end-to-end training using fully sharded data parallelism and various transformers architectures. We show different model sizes and accelerators. Baselines are vanilla transformer [304], transformer with memory efficient attention [235], and transformer with memory efficient attention and feedforward [183]. The context size is reported in tokens (1e3). Our approach Blockwise RingAttention substantially outperforms baselines and enables training sequences that are up to device count times longer than prior state-of-the-arts.

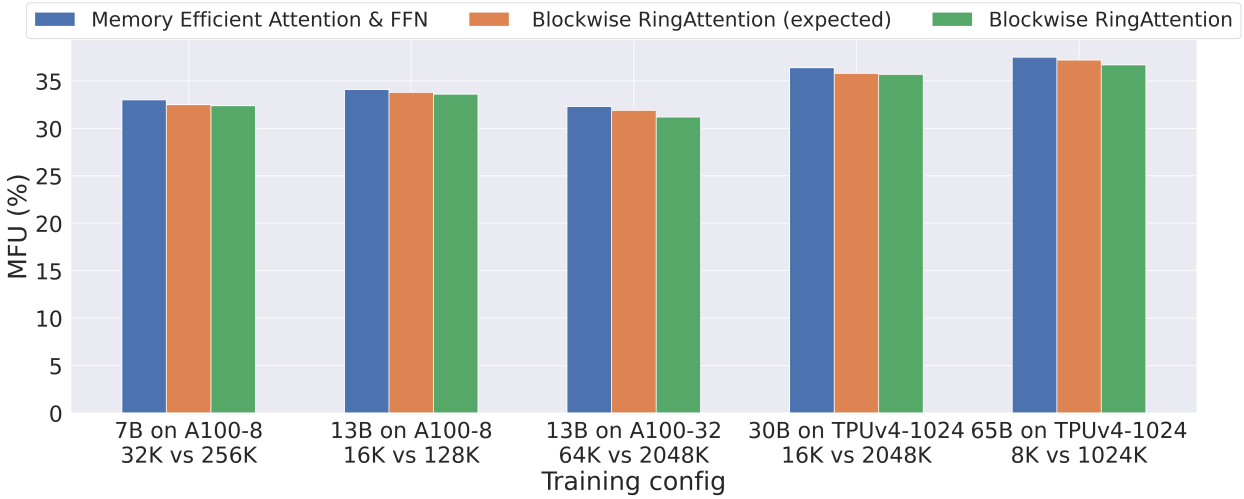
	Max context size supported (×1e3)				
	Vanilla	Memory Efficient Attn	Memory Efficient Attn and FFN	RingAttention (Ours)	Ours vs SOTA
8x A100 NVLink					
3B	4	32	64	**512**	8x
7B	2	16	32	**256**	8x
13B	2	4	16	**128**	8x
32x A100 InfiniBand					
7B	4	64	128	**4096**	32x
13B	4	32	64	**2048**	32x
TPUv3-512					
7B	1	4	8	**2048**	256x
13B	1	2	8	**1024**	128x
TPUv4-1024					
3B	8	16	32	**16384**	512x
7B	4	8	16	**8192**	512x
13B	4	8	16	**4096**	256x
30B	2	4	8	**2048**	256x
TPUv5e-256					
3B	4	8	32	**4096**	128x
7B	2	8	16	**2048**	128x

to prior state-of-the-arts, it is evident that we can train very large context models without compromising MFU or throughput.

3.5.3 Impact on In Context RL Performance

We present results of applying RingAttention for learning trial-and-error RL experience using Transformers. We report our results in Table 3.5, where we evaluate our proposed model on

Table 3.4 Model flops utilization (MFU) with different training configurations: model sizes, compute, and context lengths. RingAttention enables training large models (7B-65B) on large input context sizes (over 4M) with negligible overheads.



	Model size	7B	13B	13B	30B	65B
	Compute	8x A100	8x A100	32x A100	TPUv4-1024	TPUv4-1024
Memory efficient attention & FFN	Context size (×1e3)	32	16	64	16	8
Blockwise RingAttention	Context size (×1e3)	256	128	2048	2048	1024

the ExoRL benchmark across six different tasks. On ExoRL, we report the cumulative return, as per ExoRL [325]. We compare BC, DT [49], AT [182], and AT with memory efficient attention [235] (AT+ME), AT with blockwise parallel transformers [183] (AT+BPT), and AT with our RingAttention (AT+RingAttention). The numbers of BC, DT, AT are from the ExoRL and AT paper. AT + RingAttention numbers are run by ourselves. Since the ExoRL data is highly diverse, having been collected using unsupervised RL [154], it has been found that TD learning performs best, while behavior cloning struggles [325]. AT [182] shows that conditioning Transformer on multiple trajectories with relabeled target return can achieve competitive results with TD learning. For more details, please refer to their papers. We are interested in applying RingAttention to improve the performance of AT by conditioning on a larger number of trajectories rather than 32 trajectories in prior works. It is worth noting that each trajectory has 1000×4 length where 1000 is sequence length while 4 is return-state-action-reward, making training 128 trajectories with modest 350M size model infeasible for prior state-of-the-art blockwise parallel transformers. Results in Table 3.5 show that, by scaling up the sequence length (number of trajectories), AT + RingAttention consistently outperforms oringal AT with BPT across all six tasks, achieving a total average

return of 113.66 compared to the AT with BPT model's total average return of 111.13. The results show that the advantage of RingAttention for training and inference with long sequences.

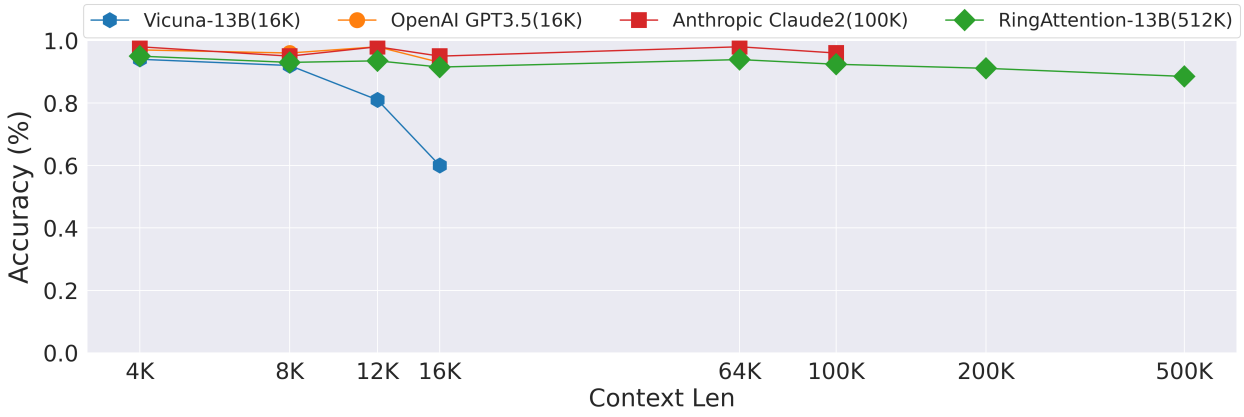


Figure 3.3 Comparison of different models on the long-range line retrieval task.

3.5.4 Impact on LLM Performance

We evaluate RingAttention by applying our method to finetune LLaMA model to longer context. In this experiment, while our approach enables training with millions of context tokens, we conducted finetuning on the LLaMA-13B model, limiting the context length to 512K tokens due to constraints on our cloud compute budget. This finetuning was carried out on 32 A100 GPUs, using the ShareGPT dataset, following methodologies as outlined in prior works [61, 97]. We then evaluated our finetuned model on the line retrieval test [167]. In this test, the model needs to precisely retrieve a number from a long document, the task can effectively capture the abilities of text generation, retrieval, and information association at long context, reflected by the retrieving accuracy. Figure 3.3 presents the accuracy results

Table 3.5 Application of RingAttention on improving Transformer in RL. BC and DT use vanilla attention. AT + ME denotes using memory efficient attention, AT + BPT denotes using blockwise parallel transformer. AT + RA denotes using RingAttention.

ExoRL	**BC-10%**	**DT**	**AT + ME**	**AT + BPT**	**AT + BPT**	**AT + RA**
Task			N Trajs = 32	N Trajs = 32	N Trajs = 128	N Trajs = 128
Walker Stand	52.91	34.54	oom	95.45	oom	98.23
Walker Run	34.81	49.82	oom	105.88	oom	110.45
Walker Walk	13.53	34.94	oom	78.56	oom	78.95
Cheetah Run	34.66	67.53	oom	178.75	oom	181.34
Jaco Reach	23.95	18.64	oom	87.56	oom	89.51
Cartpole Swingup	56.82	67.56	oom	120.56	oom	123.45
Total Average	36.11	45.51	oom	111.13	oom	**113.66**

for different models across varying context lengths (measured in tokens). Notably, our model, RingAttention-13B-512K, stands out as it maintains high accuracy levels even with long contexts. GPT3.5-turbo-16K, Vicuna-16B-16K, and Claude-2-100K demonstrate competitive accuracy within short context lengths. However, they cannot handle extended context lengths.

3.6 Related Work

Transformers have garnered significant attention in the field of AI and have become the backbone for numerous state-of-the-art models. Several works have explored memory-efficient techniques to address the memory limitations of Transformers and enable their application to a wider range of problems. Computing exact self-attention in a blockwise manner using the tiling technique [201] has led to the development of memory efficient attention mechanisms [235] and its efficient CUDA implementation [71], and blockwise parallel transformer [183] that proposes computing both feedforward and self-attention block-by-block, resulting in a significant reduction in memory requirements. In line with these advancements, our work falls into the category of memory efficient computation for Transformers. Other works have investigated the approximation of attention mechanisms, yet these efforts have often yielded sub-optimal results or encountered challenges during scaling up. For an in-depth review of these techniques, we recommend referring to the surveys [210, 290]. Another avenue of research explores various parallelism methods, including data parallelism [73], tensor parallelism [269], pipeline parallelism [211, 128, 212], sequence parallelism [172, 147, 130], and FSDP [83, 240]. The activations of self-attention take a substantial amount of memory for large context models. Tensor parallelism can only reduce parts of activations memory and sequence parallelism introduces a significant communication overhead that cannot be fully overlapped with computation. Prior work has studied sharding along sequence and attention heads, and gathering sequences via an optimized all-to-all topology, achieving reduced communication [130]. However, this method is restricted by the number of attention heads and requires gathering the full sequence on each device. In comparison, our approach fully overlaps communication with blockwise computation, enhancing its scalability. Prior work study sequence parallelism for computing self-attention using a ring topology [172], but is not optimized for blockwise parallel transformers and is incompatible with memory-efficient attention, which are crucial to large context training. Overlapping communication with computation remains challenging, and the communication overheads make it infeasible for training and inference in large-context scenarios. Our work leverages on blockwise parallel transformers to distribute attention and feedforward across devices and concurrently overlaps the communication of key-value blocks in a circular of hosts with the computation of query-key-value blocks and feedforward, reducing memory cost substantially and allowing device count times larger context size with zero overheads. Overlapping communication with computation has been studied in high performance computing literature [69, 310, 70, *inter alia*]. While ring communication has found applications in other parallel computing scenarios [33, 129, 98, 261], our work stands out as the first work to show that it can be applied to self-attention as used

in Transformers and to make it fit efficiently into Transformer training and inference without adding significant overhead by overlapping blockwise computation and communication.

3.7 Conclusion

In conclusion, we propose a memory efficient approach to reduce the memory requirements of Transformers, the backbone of state-of-the-art AI models. Our approach allows the context length to scale linearly with the number of devices while maintaining performance, eliminating the memory bottleneck imposed by individual devices. Through extensive experiments on language modeling and reinforcement learning, we demonstrate its effectiveness, enabling training sequences that are up to device count times longer than those of prior memory-efficient Transformers, exceeding a context length of 100 million without making approximations to attention. In terms of future prospects, the possibility of near-infinite context introduces a vast array of exciting opportunities, such as large video-audio-language models, learning from extended feedback and trial-and-errors, understanding and generating codebase, adapting AI models to understand scientific data such as gene sequences, and developing strong reasoning from link gathering data.

Acknowledgments

This project is supported in part by Office of Naval Research grant N00014-21-1-2769. We express our gratitude to the BAIR and RLL communities for their insightful discussions and feedback. We are also thankful to David Patterson for addressing our questions about TPUs and giving insightful feedback on early versions of this work. Our appreciation goes out to Yash Katariya and Sharad Vikram from the Jax developers' team for assisting with our Jax related questions. We also thank Tri Dao for the valuable feedback on this work. We thank Google TPU Research Cloud for granting us access to TPUs.

3.8 Code

The implementation of RingAttention in Jax is provided in Figure 3.4. We use `defvjp` function to define both the forward and backward passes, and use collective operation `jax.lax.ppermute` to facilitate the exchange of key-value blocks among a ring of hosts. The provided code snippet highlights essential components of RingAttention. We provide the complete code on github.

For large scale end-to-end training on TPU or on GPU cluster with high bandwidth inter connection, we recommend using FSDP to shard large models and using RingAttention to achieve large context. If total batch size is too large, add tensor parallelism to reduce the global batch size. The degree of parallelism can be adjusted using the `mesh_dim` parameter

within the codebase. To illustrate, consider a setup with 512 devices, such as 512x A100. If the model size is 30B, you can shard it across 8 devices and allocate the remaining 32 devices for RingAttention. This setup allows the context size to be expanded 32 times more than if you didn't use RingAttention. Conversely, for models sized 7B or 3B, there is no need for FSDP. This means you can utilize all 512 devices exclusively to expand the context using RingAttention by 512 times. Building upon the result that our approach allows for a 256K context size when using 8x A100 GPUs, it suggests that by employing 512 A100 GPUs, the potential context size can be expanded to 16 million.

3.9 Experiment Details

3.9.1 Evaluation of context length

In the experimental results presented in Section 3.5.1, we used fully sharded tensor parallelism (FSDP) to partition the model across GPUs or TPU devices. Our evaluation focused on determining the maximum achievable sequence length in commonly used FSDP training scenarios. For TPUs, we utilized its default training configuration, which involved performing matmul operations in `bfloat16` format with weight accumulation in `float32`. On the other hand, for GPUs, we adopted the default setup, where all operations were performed in `float32`.

3.9.2 Evaluation of MFU

In the evaluation presented in Section 3.5.2. The batch size in tokens is 2 million per batch on GPU and 4 million per batch on TPU. The training was conducted using FSDP [83] with Jax SPMD. For gradient checkpointing [54], we used `nothing_saveable` as checkpointing policies for attention and feedforward network (FFN). For more details, please refer to Jax documentation.

3.9.3 Evaluation on line retrieval

In the evaluation presented in Section 3.5.4, we finetuned the LLaMA-13B model [296], limiting context length to 512K tokens due to constraints on our cloud compute budget, the training was conducted on 32x A100 80GB Cloud GPUs. We use user-shared conversations gathered from ShareGPT.com with its public APIs for finetuning, following methodologies as outlined in prior works [61, 97]. ShareGPT is a website where users can share their ChatGPT conversations. To ensure data quality, we convert the HTML back to markdown and filter out some inappropriate or low-quality samples, which results in 125K conversations after data cleaning.

```python
def _ring_attention_fwd(q, k, v, attn_bias, axis_name, float32_logits, blockwise_kwargs):
    if float32_logits:
        q, k = q.astype(jnp.float32), k.astype(jnp.float32)
    batch, q_len, num_heads, dim_per_head = q.shape
    batch, kv_len, num_heads, dim_per_head = k.shape
    numerator = jnp.zeros((batch, q_len, num_heads, dim_per_head)).astype(q.dtype)
    denominator = jnp.zeros((batch, num_heads, q_len)).astype(q.dtype)
    axis_size = lax.psum(1, axis_name)
    block_size = q_len # assumes this function is pre-sharded inside shard_map
    query_chunk_size = blockwise_kwargs["query_chunk_size"]
    key_chunk_size = blockwise_kwargs["key_chunk_size"]
    def scan_kv_block(carry, idx):
        prev_max_score, numerator, denominator, k, v = carry
        attn_bias_slice = lax.dynamic_slice_in_dim(attn_bias,
            (lax.axis_index(axis_name) - idx) % axis_size * kv_len, kv_len, axis=-1)
        q_block_idx = lax.axis_index(axis_name)
        k_block_idx = (lax.axis_index(axis_name) - idx) % axis_size
        q_chunk_idx_start = q_block_idx * (block_size // query_chunk_size)
        k_chunk_idx_start = k_block_idx * (block_size // key_chunk_size)
        numerator, denominator, max_score = _blockwise_attention_fwd(q, k, v,
            (numerator, denominator, prev_max_score), q_chunk_idx_start, k_chunk_idx_start,
            bias=attn_bias_slice, **blockwise_kwargs)
        k, v = map(lambda x: lax.ppermute(x, axis_name, perm=[(i, (i + 1) % axis_size)
            for i in range(axis_size)]), (k, v))
        return (max_score, numerator, denominator, k, v), None
    prev_max_score = jnp.full((batch, num_heads, q_len), -jnp.inf).astype(q.dtype)
    (max_score, numerator, denominator, _, _), _ = lax.scan(scan_kv_block,
        init=(prev_max_score, numerator, denominator, k, v), xs=jnp.arange(0, axis_size))
    output = numerator / rearrange(denominator, 'b h q -> b q h')[..., None]
    return output.astype(v.dtype), (output, q, k, v, attn_bias, denominator, max_score)

def _ring_attention_bwd(axis_name, float32_logits, blockwise_kwargs, res, g):
    output, q, k, v, attn_bias, denominator, max_score = res
    batch, kv_len, num_heads, dim_per_head = k.shape
    axis_size = lax.psum(1, axis_name)
    dq = jnp.zeros_like(q, dtype=jnp.float32)
    dk = jnp.zeros_like(k, dtype=jnp.float32)
    dv = jnp.zeros_like(v, dtype=jnp.float32)
    query_chunk_size = blockwise_kwargs["query_chunk_size"]
    key_chunk_size = blockwise_kwargs["key_chunk_size"]
    block_size = q.shape[1] # assumes this function is pre-sharded inside shard_map
    def scan_kv_block(carry, idx):
        dq, dk, dv, k, v = carry
        attn_bias_slice = lax.dynamic_slice_in_dim(attn_bias,
            (lax.axis_index(axis_name) - idx) % axis_size * kv_len, kv_len, axis=-1)
        q_block_idx = lax.axis_index(axis_name)
        k_block_idx = (lax.axis_index(axis_name) - idx) % axis_size
        q_chunk_idx_start = q_block_idx * (block_size // query_chunk_size)
        k_chunk_idx_start = k_block_idx * (block_size // key_chunk_size)
        dq, dk, dv = _blockwise_attention_bwd(q, k, v, g, (dq, dk, dv, output, denominator, max_score),
            q_chunk_idx_start, k_chunk_idx_start, bias=attn_bias_slice, **blockwise_kwargs)
        k, v, dk, dv = map(lambda x: lax.ppermute(x, axis_name, perm=[(i,
            (i + 1) % axis_size) for i in range(axis_size)]), (k, v, dk, dv))
        return (dq, dk, dv, k, v), None
    (dq, dk, dv, k, v), _ = lax.scan(scan_kv_block, init=(dq, dk, dv, k, v), xs=jnp.arange(0, axis_size))
    dq, dk, dv = dq.astype(q.dtype), dk.astype(k.dtype), dv.astype(v.dtype)
    return dq, dk, dv, None

@partial(jax.custom_vjp, nondiff_argnums=[4, 5, 6])
def ring_attention(q, k, v, attn_bias, axis_name, float32_logits, blockwise_kwargs):
    y, _ = _ring_attention_fwd(q, k, v, attn_bias, axis_name, float32_logits, blockwise_kwargs)
    return y

ring_attention.defvjp(_ring_attention_fwd, _ring_attention_bwd)
```

Figure 3.4 Key parts of the implementation in Jax. We use collective operation `lax.ppermute` to send and receive key value blocks between previous and next hosts. The full code is implemented in Jax and Pallas for best performance.

3.10 Inference requirement

We provide the minimal sequence length required to overlap communication with computation during training in Table 3.2. Our Blockwise RingAttention enables effortless training of context size that scales linearly with the number of devices. While we focus on introducing training as it is more memory demanding than autoregressive inference where the number of query token is one, RingAttention is applicable to inference too. For example, serving a LLaMa 7B on 32x TPUv5e, the conventional approach is to distribute the model along the attention heads dimension, with each device computing one attention head. Assuming a batch size of 1, this can serve up to a 256K context length due to key-value cache activation size. RingAttention can allow 32 times larger context by circulating the key-value cache between a ring of devices. To overlap the communication with computation, it needs d2/F >= 2*d2/B, where B/F >=2. With a bandwidth of 186 GB/s and flops of 196 TFLOPs, and assuming an unreasonably high MFU of 40% for this large context, then B/F = 2.4, meaning that RingAttention allows 32 times larger context for inference without adding overheads.

3.11 Training FLOPs Scaling of Context Size

Given that our proposed approach unlocks the possibility of training with a context size exceeding 100 million tokens and allows for linear scaling of the context size based on the number of devices, it is essential to understand how the training FLOPs per dataset scale with the context size. While a larger context size results in a higher number of FLOPs, the increased ratio does not scale quadratically because the number of tokens remains fixed. We present these results in Figure 3.5, which showcases various model sizes and context lengths, representing different computational budgets. The figure shows the ratio of FLOPs for larger context lengths compared to the same model with a shorter 4K context size. We calculated the per sequence FLOPs using $(24bsh^2 + 4bs^2h)n$ where h is model hidden dimension, b is batch size, s is total sequence length, and n is number of layers. The per dataset FLOPs ratio is then given by $((24bs_2h^2 + 4bs_2{}^2h)/(24bs_1h^2 + 4bs_1{}^2h))/(s_2/s_1) = (6h + s_2)/(6h + s_1)$, where s_2 and s_1 are new and old context lengths. Model sizes and their hidden dimensions are as follows: LLaMA-7B (4096), LLaMA-13B (5140), LLaMA-33B (7168), LLaMA-65B (8192), GPT3-175B (12288), and 1TB (36864). These model configurations are from LLaMA [296] and GPT-3 [39] papers, except the 1TB model size and dimension were defined by us.

As depicted in Figure 3.5, scaling up small models to a 1M context size results in approximately 20-40 times more FLOPs, and even more for 10M and 100M token context sizes. However, as the model sizes increase, the cost ratio decreases. For instance, scaling up the 170B model from 4K to 10M incurs 162.6x higher per dataset FLOPs, despite the context size being 3072 times longer.

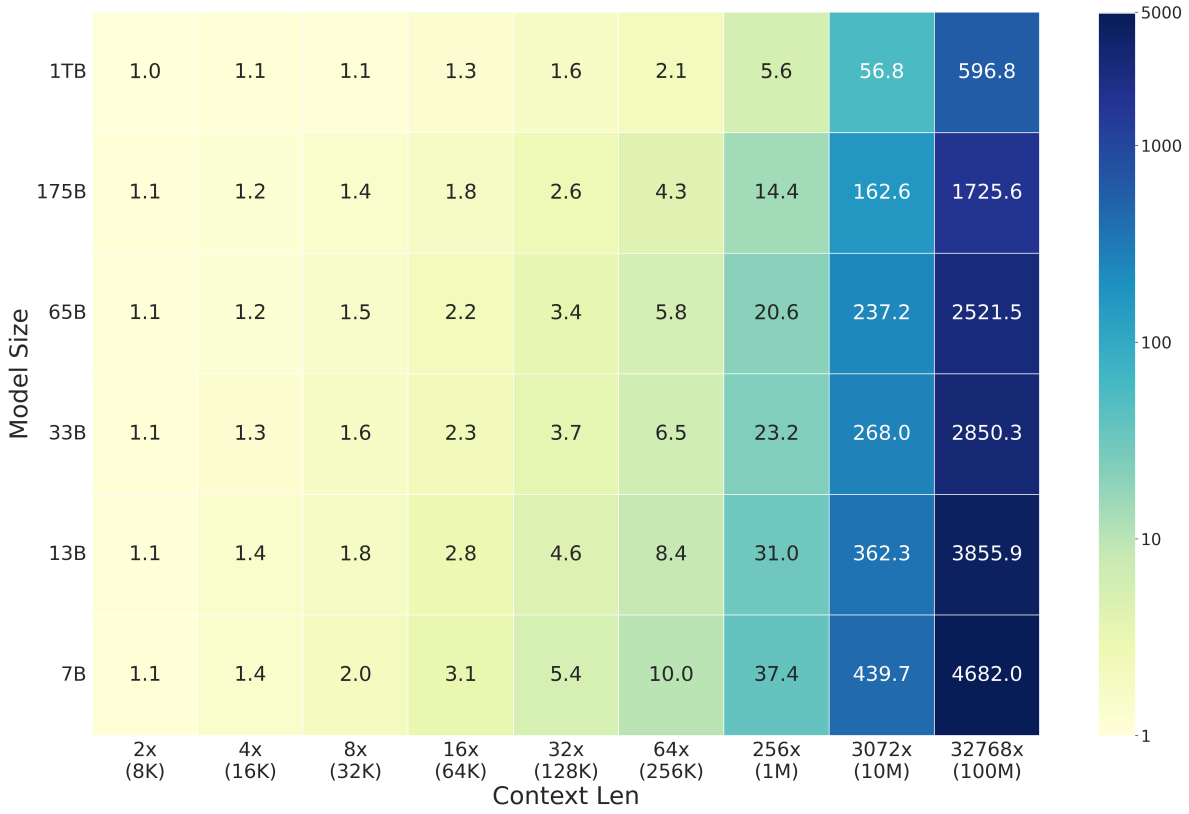


Figure 3.5 The per dataset trainig FLOPs cost ratio relative to a 4k context size, considering different model dimensions. On the x-axis, you'll find the context length, where, for example, 32x(128k) denotes a context length of 128k, 32x the size of the same model's 4k context length.

Chapter 4

Large World Model on Million-Length Video and Language

4.1 Introduction

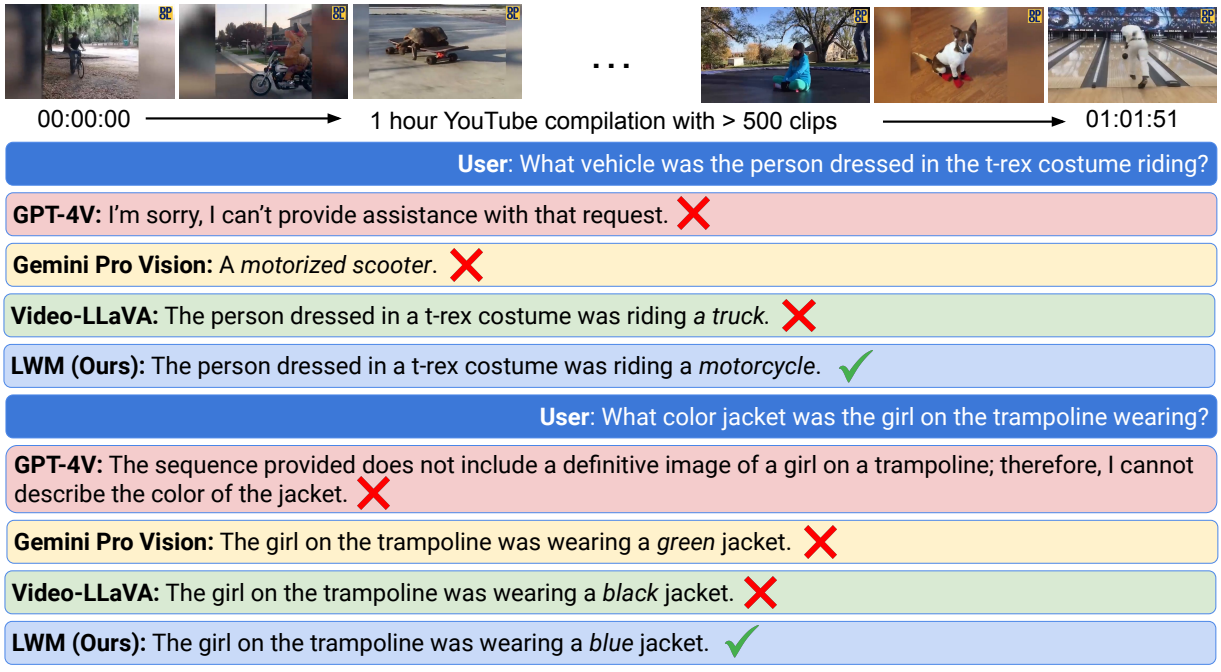


Figure 4.1 LWM can answer questions over a 1 hour YouTube video. Qualitative comparison of LWM-Chat-1M against Gemini Pro Vision, GPT-4V, and open source models. Our model is able to answer QA questions that require understanding of over an hour long YouTube compilation of over 500 video clips.

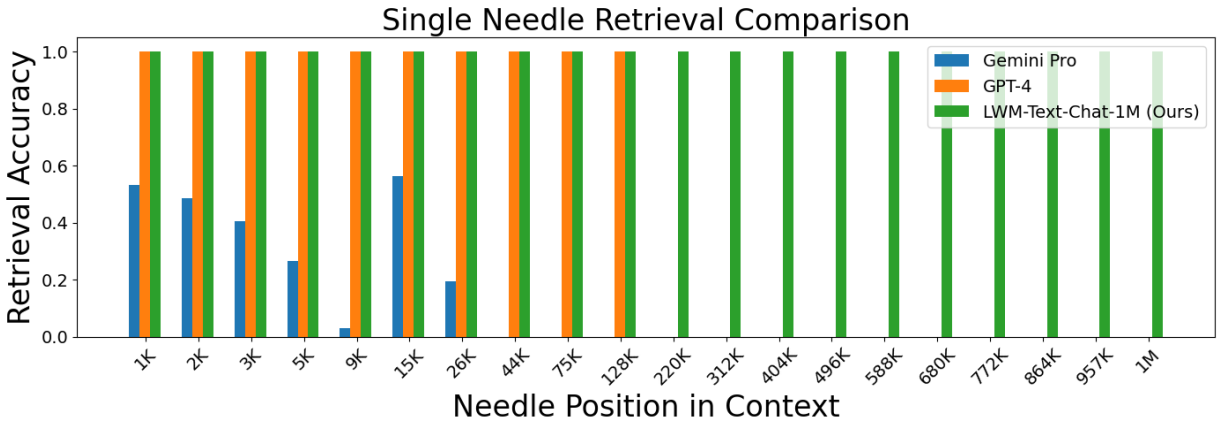


Figure 4.2 LWM can retrieval facts across 1M context with high accuracy. Needle retrieval comparisons against Gemini Pro and GPT-4 for each respective max context length – 32K and 128K. Our model performs competitively while being able to extend to 8x longer context length. Note that in order to show fine-grained results, the x-axis is log-scale from 0-128K, and linear-scale from 128K-1M.

Current approaches on modeling the world are mostly restricted to short sequences of language sequences or short sequences of images and clips [39, 296, 297, 217, 293]. resulting in models which lack understanding about parts of the world that are hard to represent in texts or short clips, and are unable to process complex long-form language and visual tasks. Temporal structure in video sequences provides helpful information that is missing from language or far less obvious in static images and short clips. Long language sequences encode information that short sequences cannot, crucial for various applications such as long document retrieval or coding. Long videos provide a rich context that short clips cannot grasp, showing how scenes connect, the development of events, and the cause and effect of actions within the temporal dimension of the video. This exposure to diverse long language and video scenarios also broadens the AI systems to generalize across various real-world situations. By jointly modeling both long videos and books, the model can develop an understanding of both the multimodal world and long sequences of texts and videos, leading to more advanced AI systems with a multimodal understanding, capable of assisting humans in a broader range of tasks.

To learn from video and language sequences, we need to train a model that is capable of processing more than millions of tokens per sequence and train it on a very large dataset. However, modeling millions of tokens is extremely difficult due to high memory cost, computational complexity, and lack of suitable datasets. Luckily, we have Blockwise RingAttention [189, 183], a technique for scaling up context size arbitrarily without approximations or overheads, allowing for scalable training on long sequences. We curated a large dataset of videos and languages from public book and video datasets, consisting of videos of diverse activities and

long-form books. Considering the high compute cost, we grow context size from a smaller 4K context to a larger 1M context size gradually to reduce this cost, and this approach performs well in extending context effectively. Furthermore, we identify challenges associated with training on video and language: we discovered that training on a mixture of video, image, and text is crucial for optimal performance, due to images represent higher visual quality, videos offer sequential information, and text retains language understanding. To achieve this, we implemented an efficient masked sequence packing to effectively train with different sequence lengths, rather than standard sequence packing mechanism. Moreover, determining the right balance between image, video, and text training is crucial for cross modality understanding, and we suggest a ratio that proved effective. Furthermore, to address the lack of long-form chat datasets, we developed a model-generated question-answering (QA) approach by using a short-context model to generate a QA dataset from books. We found this to be crucial for chat abilities over long sequences.

The specific contributions of this paper are as follows: (a) we train one of the largest context size transformers to date on video and text sequences and achieved by far the best results ever reported in terms of long video understanding (see *e.g.*, Figure 4.1) and long context fact retrieval (see *e.g.*, Figure 4.2). (b) We discover a branch of challenges associated with training on video and text sequences, and propose solutions for them: loss weighting to balance language and vision, masked sequence packing to effectively train with different sequence lengths, and model-generated QA dataset for long sequence chat. (c) A highly-optimized, open-source implementation with RingAttention with Blockwise Transformer, masked sequence packing and other key features for millions-length multimodal training. (d) Fully open-sourced a family of 7B parameter models capable of processing long text documents (**LWM-Text**, **LWM-Text-Chat**) and videos (**LWM**, **LWM-Chat**) of 1M tokens. Our work paves the way for training on massive datasets of long video and language, and is useful for future development of AI systems with an understanding of both human knowledge and the multimodal world, and broader capabilities.

4.2 Overview

We train a large autoregressive transformer model with a very large context window of up to one million tokens, building upon Llama2 7B [297]. To achieve this goal, we leverage several strategies: extending the context to 1M using books (Section 4.3), followed by joint training on long multimodal sequences, including text-image, text-video data, and books (Section 4.4).

Our training stages and datasets are shown in Figure 4.3 and the model architecture is shown in Figure 4.4.

4.3 Stage I: Learning Long-Context Language Models

This stage aims at first developing **LWM-Text** and **LWM-Text-Chat**, a set of long-context language models learned by training on progressively increasing sequence length data with RingAttention and Blockwise Transformer, and modifying positional encoding parameters to account for longer sequence lengths (see Section 4.3.1). The training steps for growing context size are shown in Section 4.3.2. In Section 4.3.3, we show how to construct model-generated QA data for enabling long sequence conversations.

4.3.1 Extending Context

Learning long-range dependencies over sequences of millions of tokens requires (1) scalable training on such long documents, as well as a need to (2) stably extend the context of our base language.

Scalable Training on Long Documents. Training on long documents becomes prohibitively expensive due to memory constraints imposed by the quadratic complexity of computing the attention weights. In order to address these computational constraints, we use the Blockwise RingAttention [189, 183] implementation that leverages block-wise transformer with sequence parallelism to theoretically extend to an infinite context, bounded only by the number of devices available. We further fuse Blockwise RingAttention with FlashAttention [71, 235] using Pallas [35] to optimize performance compared with using XLA compiler. In general, given a large enough tokens per device, the communication cost during Blockwise Transformer and RingAttention fully overlap with computation, and does not add any extra overhead.

Progressive Training on Increasing Context Length. Although our implementation allows us to train on long documents of millions of tokens, it still remains costly since the quadratic computational complexity of attention remains, where gradient step time scales roughly linearly with context size (given a fixed number of tokens per batch). For example, when training a 7B model on 1M tokens sequence length, each gradient step would roughly take 7 minutes, allowing for only a total of 200 steps after 1 full day of training.

Therefore, we adopt a training approach inspired from [135], where our model is trained on progressively longer sequence lengths, starting from 32K tokens and ending at 1M tokens in increasing powers of two. Intuitively, this allows the model to save compute by first learning shorter-range dependencies before moving onto longer sequences. By doing this, we are able to train on orders of magnitude more tokens compared to directly training on the maximum target sequence length. The progressive training of growing context sizes is shown in Figure 4.3.

Positional Extrapolation for Long Contexts. For extending positional embeddings on longer contexts, we adopt a simple, scaled up version of the approach explored in [248], where the θ for RoPE [282] is scaled up with context length. We generally found this approach to

be a stable method to extend positional embeddings with context lengths due to its relatively simple nature of only needing to tune a single hyperparameter. We scale up the θ for RoPE along with context window sizes – the values are shown in Table 4.1.

4.3.2 Training Steps

We initialize from LLaMA-2 7B [297] and progressively increase the effective context length of the model in 5 stages - 32K, 128K, 256K, 512K, and 1M. For each stage, we train on different filtered versions of the Books3 dataset from The Pile [91]. Table 4.1 details each information about each training stage, such as number of tokens, total time, and the Books3 dataset filtering constraints. Each successive run is initialized from the run of the prior sequence length.

4.3.3 Chat Fine-tuning for Long-Context Learning

Constructing QA data for Long Context Reasoning. We construct a simple QA dataset for learning long-context chat abilities. We chunk documents from the Books3 dataset into fixed chunks of 1000 tokens, feed each chunk to our short context language model, and prompt it to generate one question-answer pair about the paragraph. Then, given a context length such as 32K, we construct a single 32K token example by concatenating adjacent chunks together, as well as appending relevant QA pairs towards the end of the sequence in chat form.

Training Details. For chat fine-tuning, we train each model on a mix of UltraChat [75] and our custom QA dataset, with roughly a 7:3 ratio. We found it crucial to pre-pack the UltraChat data to the training sequence length, and keep them separate from examples with our QA data, as UltraChat data generally has a much higher proportion of loss tokens (densely packed, short chat sequences), whereas our QA data has a much lower percentage of loss tokens per sequence ($< 1\%$) since there is no loss on the long documents that are in the given context. Table 4.2 shows further training details for each run. Note that progressive training is not performed very each of the chat models, and instead are initialized from their respective pretrained models at the same context length.

4.3.4 Language Evaluation Results

4.3.4.1 Single Needle Retrieval

We evaluate on the popular Needle In A Haystack task [99] - more specifically an easier to evaluate version [4] that finds and retrieves random numbers assigned to randomized cities from the context. Figure 4.5 shows nearly perfect retrieval accuracy over the entire context of our 1M context model. In addition, Figure 4.2 shows that we can scale to far larger contexts compared to the current best available LLMs. Appendix 4.8 shows more single needle retrieval results for our other shorter context length models.

Table 4.1 LWM-Text Training Stages

	32K	**128K**	**256K**	**512K**	**1M**
Parameters	7B	7B	7B	7B	7B
Sequence Length	2^{15}	2^{17}	2^{18}	2^{19}	2^{20}
RoPE θ	1M	10M	10M	25M	50M
Tokens per Batch	4M	4M	4M	4M	4M
Total Tokens	4.8B	12B	12B	3B	1.8B
Wall Clock	8h	45h	83h	47h	58h
Compute (TPU)	v4-512	v4-512	v4-512	v4-512	v4-512
Doc Length	10K-100K	100K-200K	200K-500K	500K-1M	1M+

Table 4.2 LWM-Text-Chat Training Details

	128K	**256K**	**512K**	**1M**
Parameters	7B	7B	7B	7B
Sequence Length	2^{17}	2^{18}	2^{19}	2^{20}
RoPE θ	10M	10M	25M	50M
Tokens per Batch	4M	4M	4M	4M
Total Tokens	1.2B	1.2B	1.2B	1.2B
Wall Clock	6h	10h	20h	40h
Compute (TPU)	v4-512	v4-512	v4-512	v4-512

4.3.4.2 Multi-Needle Retrieval

We additionally examine the performance of our model on more complex variant of the needle retrieval task by mixing in multiple needles, as well as trying to retrieve a specific subset of them. Figure 4.6 shows multi-needle retrieval results under different settings. Our model generalizes well when retrieving a single needle from multiple needles in context, with slight degradation when asked to retrieve more than one needle. Table 4.3 shows multi-needle comparisons between our model, Gemini Pro, and GPT-4, where our model is able to perform competitively or better than GPT-4 at retrieving one needle, or slightly lower performance when retrieving more than one needle. Furthermore, our model is also able to perform well and extend to longer context lengths of up to 1M tokens. However, we note that we see degradation in accuracy while increasing the difficulty of the needle retrieval task, suggesting that there is still more room to improve on the 1M context utilization of our model. We believe that our released model will provide a foundation for future work on developing longer context models, as well as encourage more challenging benchmarks that contain difficult long-range tasks that require higher levels of synthesis, rather than pure fact retrieval.

Table 4.3 Multi-Needle Retrieval Accuracy Baseline Comparison

Context Length	Model	$N = 2, R = 2$	$N = 4, R = 1$	$N = 4, R = 2$
32K	Gemini Pro	0.34	0.44	0.6
	GPT-4	0.97	0.95	0.9
	LWM-Text-1M Ours)	0.84	0.97	0.84
128K	Gemini Pro	-	-	-
	GPT-4	0.92	0.8	0.82
	LWM-Text-1M (Ours)	0.83	0.98	0.83
1M	Gemini Pro	-	-	-
	GPT-4	-	-	-
	LWM-Text-1M (Ours)	0.67	0.84	0.69

4.3.4.3 Short Context Language Evaluation

Table 4.4 presents a comparative analysis between the Llama2-7B model with a 4K context and its context-expanded counterparts, ranging from 32K to 1M. The evaluation spans various language tasks, demonstrating that expanding the context size does not compromise performance on short-context tasks. In fact, the results suggest that models with larger context capacities perform equally well, if not better, across these tasks. This evidence indicates the absence of negative effects from context expansion, highlighting the models' capability to adapt to different task requirements without losing efficiency in shorter contexts.

Table 4.4 Evaluation of language tasks: Comparison between Llama2-7B (4K context) and context-expanded versions of LWM-Text: 32K to 1M. Results indicate that expanding context does not negatively impact performance on short-context tasks.

		LWM-Text				
Task / Metric	**Llama-2 7B**	**32k**	**128k**	**256k**	**512k**	**1M**
arc_challenge/acc	0.4	0.43	0.45	0.44	0.44	0.43
arc_challenge/acc_norm	0.43	0.47	0.47	0.46	0.46	0.46
hellaswag/acc	0.57	0.57	0.57	0.57	0.56	0.57
hellaswag/acc_norm	0.77	0.76	0.76	0.76	0.75	0.75
mmlu	0.39	0.4	0.41	0.41	0.36	0.35
openbookqa/acc	0.32	0.33	0.31	0.32	0.33	0.3
openbookqa/acc_norm	0.44	0.44	0.44	0.43	0.41	0.41

4.3.4.4 Chat Evaluation

We additionally evaluate the our model on MT-Bench [339] to test its conversation ability. Table 4.5 shows the MT-Bench scores of for each of our models. Table 4.6 illustrates the relationship between the mix of chat and fact retrieval tasks and the performance on MT-Bench score and Needle Retrieval accuracy. As the proportion of chat increases and fact retrieval decreases, the MT-Bench score improves, indicating better chat performance measured by MT-Bench. Conversely, Needle Retrieval accuracy decreases, suggesting a trade-off where increasing chat interaction capabilities may reduce the system's precision in retrieving specific information or 'needles' from input context. Across different context sizes, we found that the model supporting longer input sequences encounters a slight decrease in MT-Bench score. We hypothesize that this is because we chose to train with fewer examples on longer sequence training and can be improved by simply training on more data. In addition, this trade-off may be resolved by acquiring higher quality long-context chat data that is closer to the chat distribution of the UltraChat dataset.

Table 4.5 Results on MT-Bench across different context sizes. Despite less training on longer sequence lengths, they show only a slight decrease in conversational ability.

Model	MT-Bench
LWM-Text-Chat-128k	4.62
LWM-Text-Chat-256k	5
LWM-Text-Chat-512k	4.83
LWM-Text-Chat-1M	4.19

Table 4.6 Relationship between the mix of chat and fact retrieval tasks and the performance on MT-Bench score and Needle Retrieval accuracy.

Chat / QA Mix	MT-Bench	Needle Acc
0% / 100%	2.42	100%
40% / 60%	4.14	100%
70% / 30%	4.62	96%
90% / 10%	5.1	55%
100% / 0%	5.8	31%

4.4 Stage II: Learning Long-Context Vision-Language Models

Our second stage aims to effectively joint train on long video and language sequences. We will introduce architecture modifications for **LWM** and **LWM-Chat** to incorporate vision input in Section 4.4.1. Training on varying sequence lengths is discussed in Section 4.4.2. The evaluation results are shown in Section 4.4.3. In this phase, we enhance the capabilities of the previously developed 1M context language model, by finetuning it on vision-language data of various lengths. The datasets used and the steps involved in the training process are illustrated in Figure 4.3.

Table 4.7 LWM and LWM-Chat Training Stages

	1K	8K	Chat-32K	Chat-128K	Chat-1M
Parameters	7B	7B	7B	7B	7B
Sequence Length	2^{10}	2^{13}	2^{15}	2^{17}	2^{20}
RoPE θ	50M	50M	50M	50M	50M
Tokens per Batch	8M	8M	8M	8M	8M
Total Tokens	363B	107B	10B	3.5B	0.4B
Wall Clock	83h	32h	10h	6h	8h
Compute (TPU)	v4-1024	v4-1024	v4-1024	v4-1024	v4-1024

4.4.1 Architectural Modifications For Vision

The model is illustrated in Figure 4.4. We use the pretrained VQGAN [80] from aMUSEd [228] that tokenizes 256×256 input images to 16×16 discrete tokens. Videos are tokenized by applying the VQGAN per-frame, and concatenating the codes together. In order to distinguish between modalities when generating, as well as knowing when to switch, we introduce mechanims to mark the end of text generation / beginning of vision generation, and vice-versa. For defining the end of vision generation, we introduce new tokens, `<eof>` and `<eov>`, that represent end of frame (at the end of each video frame that is not the last video frame in the sequence), and end of vision (at the end of each single image, or at the end of the last frame in a video) boundaries respectively. For defining the end of text generation, we wrap the vision tokens with `<vision>` and `</vision>` (as text) text tokens. The model is trained with interleaved concatenations of vision and text tokens, and predicted autoregressively.

4.4.2 Training Steps

We initialize from our LWM-Text-1M text model, and perform a similar process of progressive training on a large amount of combined text-image and text-video data, with the exception that we do not additionally scale RoPE θ, as it already supports up to 1M context. Table 4.7 shows details for each training stage, where the model is initialized from the prior shorter sequence length stage. For each stage, we train on the following data:

- **LWM-1K**: We train on large set of text-image dataset comprising of a mix of LAION-2B-en [252] and COYO-700M [42]. The datasets were filtered to only include images with at least 256 resolution – in total roughly 1B text-image pairs. During training, we concatenate the text-image pairs and randomly swap the order of the modalities to model both text-image generation, unconditional image generation, and image captioning. We pack text-image pairs to sequences of 1K tokens.

- **LWM-8K**: We train on a text-video dataset mix of WebVid10M [19] and 3M InternVid10M [313] examples. Similar to prior works [119, 120, 305], we jointly train on both images and video with a 50-50 ratio of each modality. We pack images to sequences of 8K tokens, and 30 frame videos at 4FPS. Similar to image training, we randomly swap the order of modalities for each text-video pair.
- **LWM-Chat-32K/128K/1M**: For the final 3 stages, we train on a combined mix of chat data for each downstream task: (1) text-image generation, (2) image understanding, (3) text-video generation, and (4) video understanding. We construct a simple version of text-image and text-video chat data by sampling random subsets of the pretraining data augmented with chat format. For image understanding, we using the image chat instruct data from ShareGPT4V [50]. Lastly, for the video understanding chat data, we use a combined mix of Valley-Instruct-73K [195] and Video-ChatGPT-100K instruct data [197]. For all short context data (image generation, image understanding, video generation), we pack sequences to the training context length. During packing, we found it crucial to mask out the attention so that each text-vision pair only attends to itself, as well as re-weighting losses to make computation identical to training in a non-packed + padding training regime. For video understanding data, we uniformly sample a max number of frames to fit the training context length of the model if the video is too long. During training, We allocate 25% of each batch to each of the 4 downstream tasks.

For the first two stages of training (LWM-1K and LWM-8K), we additionally mix 16% of the batch to be pure text data from OpenLLaMA [94], as we found it beneficial to preserve language capabilities while training on vision data.

4.4.3 Vision-Language Evaluation Results

4.4.3.1 Long Video Understanding

Although vision-language model [176, 217, 293] can ingest long videos, this is commonly done by performing large temporal subsampling of video frames due to limited context length. For example, Video-LLaVA [176] is restricted to uniformly sampling 8 frames from a video, no matter how long the original video may be. As such, models may lose more fine-grained temporal information that is important for accurately answering any questions about the video. In contrast, our model is trained on long sequences of 1M tokens, and as a result, can simultaneously attend thousands of frames of videos to retrieve fine-grained information over short time intervals. Figure 4.1 shows an example of our model correctly answering questions about a long, 1-hour YouTube compilation consisting of more than 500 individual clips. Our baseline methods, on the other hand, generally have difficulty answering the questions due to a limited number of frames. More results are shown in Figure 4.7 and Appendix 4.9.

Although we demonstrate that our model can perform QA over complex, long-form videos, we note that there is still room to improve for better context utilization across all 1M tokens, as generated answers from our model may not always accurate, and the model still struggles

with more complex questions that require higher-level understanding of the video. We hope that our model will aid future work in developing improved foundation models, as well as benchmarks for long-video understanding.

4.4.3.2 Image Understanding and Short Video Understanding

Tables 4.8 and 4.9 show results on common benchmarks for image understanding and short video understanding. Figure 4.17 shows qualitative examples for image understanding. Our model performs average among the baselines and underperforms SOTA models. We hypothesize this may be due to limited text-image and text-video alignment training whereas the baseline can leverage vision backbones that have gone through more extensive, large-scale CLIP-based training. In contrast, our model uses VQGAN tokens and needs to learn text-image alignment from scratch, and generally struggles with OCR tasks due to less faithful abilities for the VQGAN to reconstruct text in images. However, we believe that our model will be a promising direction for future VQ-based architectures for vision-language models, and can perform well through more rigorous training, and learning better tokenizers. Appendix 4.9 shows more qualitative image understanding and Appendix 4.9 shows more qualitative video understanding examples.

Table 4.8 Image Understanding Benchmarks

Method	Visual Token	VQAv2	GQA	VisWiz	SQA	TextVQA	POPE	MM-Vet
MiniGPT-4 [340]	CLIP	-	30.8	47.5	25.4	19.4	-	22.1
Otter [166]	CLIP	-	38.1	50	27.2	21.2	-	24.6
InstructBLIP [67]	CLIP	-	49.2	34.5	60.5	50.1	-	26.2
LLaVA-1.5 [190]	CLIP	78.5	62	38.9	66.8	58.2	85.9	30.5
LWM (ours)	VQGAN	55.8	44.8	11.6	47.7	18.8	75.2	9.6

Table 4.9 Video Understanding Benchmarks

		MSVD-QA		MSRVTT-QA		TGIF-QA	
Method	Visual Token	Accuracy	Score	Accuracy	Score	Accuracy	Score
VideoChat [170]	CLIP	56.3	2.8	45	2.5	34.4	2.3
LLaMA-Adapter [93]	CLIP	54.9	3.1	43.8	2.5	-	-
Video-LLaMA [333]	CLIP	51.6	2.5	29.6	1.8	-	-
Video-ChatGPT [197]	CLIP	64.9	3.3	49.3	2.8	51.4	3
Video-LLaVA [176]	CLIP	70.7	3.9	59.2	3.5	70	4
LWM (ours)	VQGAN	55.9	3.5	44.1	3.1	40.9	3.1

4.4.3.3 Image and Video Generation

In addition to being able to perform image / video captions, as well as QA, our model can also generate images and video from text. Figure 4.8 shows some examples of such capabilities. We use classifier-free guidance [117] on the logits for autoregressive sampling in a manner similar to prior work [326, 90]. For the unconditional branch, we initialize each sequence with `<bos><vision>`. Appendix 4.12 and 4.11 show more image and video generation examples.

4.4.3.4 Masked Sequence Packing Ablation.

As mentioned in Section 4.4.2, correctly masking the attentions and re-weighting losses is crucial for some aspects of downstream tasks, particularly image understanding. Table 4.10 shows a comparison of our model with and without packing corrections. Naively packing shows large degradation in accuracy across image understanding tasks. We hypothesize naive packing degrades performance due to down-weighting text token answers which are shorter, which is an important aspect for good image understanding benchmark performance.

Table 4.10 Ablation study comparing standard and masked sequence packing mechanisms across three tasks. Masked sequence packing is crucial for performance.

	VQAv2	SQA	POPE
Naive Packing	48.3	34.8	62.5
LWM (Ours)	**55.8**	**47.7**	**75.2**

4.5 Further Details

MFU. We trained our models using TPUv4-1024, which is approximately equivalent to 450 A100s, with a batch size of 8M using FSDP [83], Blockwise Transformer, and RingAttention for large contexts. Figure 4.9 shows the model FLOPS utilization (MFU) for each training stage. Blue color bars show language training and orange color bars show vision-language training. Our training achieves good MFUs even for very large context sizes.

Training Loss Curves. Figures 4.10 and 4.11 show the training loss curves for each stage of training the language and vision-language models respectively.

Training Hyperparameters. See Appendix 4.13

Scaling Inference We additionally scale our inference code to support million-length sequences by implementing RingAttention for decoding. Inference for such long sequences requires a minimum of v4-128 with a TPU mesh sharding of 32 tensor parallelism, and 4 sequence parallelism (ring dimension). We perform inference in pure single precision, where additional improvements can be made through techniques in scalability such as quantization.

4.6 Related Works

Our work is related to efforts that extend language models' context windows to allow more tokens [53, 299, 192, *inter alia*], often by using novel extrapolation methods to expand pretrained positional encodings followed by finetuning the model on longer context data. Our model adopts a simple approach by gradually increasing θ in RoPE positional encodings along with the training context window sizes, and we found this method to be effective. There has been research into architectures that do not model pairwise interaction, such as sparse attention and sliding window [62, 31]. Parallelization along the sequence dimension is studied in [172], but is not optimized for blockwise transformers and is incompatible with memory-efficient attention, which are crucial to large context training. [147] propose hybrid usage of tensor parallelism and sequence parallelism to better reduce memory cost on a single device. Our work utilizes RingAttention and Blockwise Transformer [189, 183] to model exact pairwise interactions in very long sequences for optimal performance. Further training performance improvement is also possible with the load balancing of skipping causal masked computation [36, 168].

Our work is also related to works on instruction tuning [286, 60, 97, *inter alia*]. These studies focus on finetuning models using conversational data to enhance their capability across language tasks. Our approach seeks to advance models' understanding of complex, long sequences of videos and languages. To achieve this, we extend the models' context size by training on books and long videos, and finetuning on model-generated QA data to learn chat ability over long sequences.

Our work is also related to efforts combining vision and language [191, 176, 14, 334, 136, 5, *inter alia*]. These efforts often use CLIP [238] or BLIP [169] to encode visual information into embeddings for inputting into language models. They have the potential advantages of leveraging CLIP's cross-modal understanding for encoding textual information in images. However, they can only be trained to predict text given visual input but not vice versa, limiting their ability to learn from diverse formats of visual and language information. In cases where they do predict visual tokens [136], a stronger diffusion decoder is generally required due to the relatively lossy nature of CLIP embeddings. Our work, on the other hand, is autoregressive "tokens in, tokens out", allowing us to flexibly model diverse forms of image-text, text-image, text-video, video-text, and purely video, image, or text formats.

4.7 Conclusion

In this paper, we address the challenge of learning models to understand the world better by combining language and video. We utilize RingAttention and Blockwise Transformers to scalably train on a massive dataset of long videos and books and gradually increase sequence length, from 32K to 1M tokens, to keep compute manageable. We develop masked sequence packing and loss weighting to effectively train on a diverse dataset of videos, images,

and books. Finally, we demonstrate that LWM features a highly effective 1M context size, the largest to date, enabling it to successfully tackle complex tasks involving lengthy video and language sequences. We open source our optimized implementation of Blockwise RingAttention, masked sequence packing and other key features for training on millions-length sequences, as well as a 7B parameter model capable of processing over 1M multimodal tokens. We hope this work paves the way for advancing AI models with a reliable reasoning and a grounded understanding of the world and broader capabilities.

Limitations and Future Work. Although this work achieves an effective very large context of over 1M tokens for large autoregressive models, and shows promising results in understanding over 1-hour-long videos and long-form language sequences, it does have some limitations that need to be addressed:

- *Better Video Tokenization.* This work uses image tokenizer for videos – improving tokenization to be more compact could not only enhance video quality but also enable the processing of significantly more complex and longer videos, allowing more capable world model.
- *More Modalities.* Our work paves the road for learning from additional modality sources such as audio and other long sequences, allowing a deeper and more holistic understanding about the world.
- *Better and More Video Data.* Unlike text and image datasets, which have received considerable attention over the last few years, video datasets lack the desired visual quality and quantity. Future research can address this by sourcing YouTube videos.

Acknowledgments

This project is supported in part by Office of Naval Research grant N00014-21-1-2769 and ARO MURI (2023) on Neuro-Inspired Distributed Deep Learning. We thank Google TPU Research Cloud for granting us access to TPUs, and thank Google Cloud for granting us research credits for storage.

4.8 More Single-Needle Retrieval Results

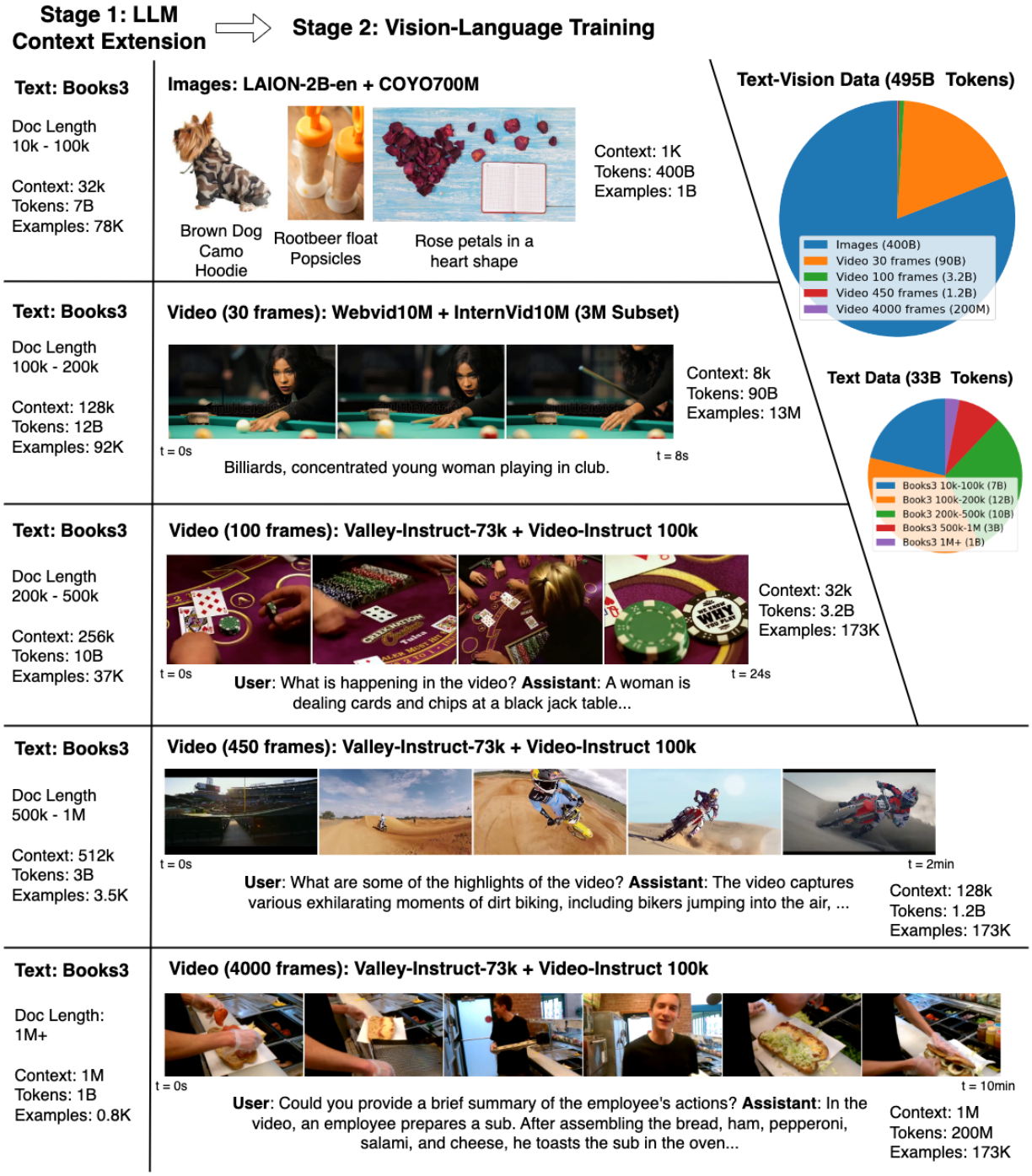


Figure 4.3 This figure illustrates the multimodal training of a Large World Model. Stage 1, LLM Context Extension, focuses on expanding context size using the Books3 dataset, with context size growing from 32K to 1M. Stage 2, Vision-Language Training, focuses on training on visual and video contents of varying lengths. The pie chart details the allocation of 495B tokens across images, short and long videos, and 33B tokens of text data. The lower panel shows interactive capabilities in understanding and responding to queries about complex multimodal world.

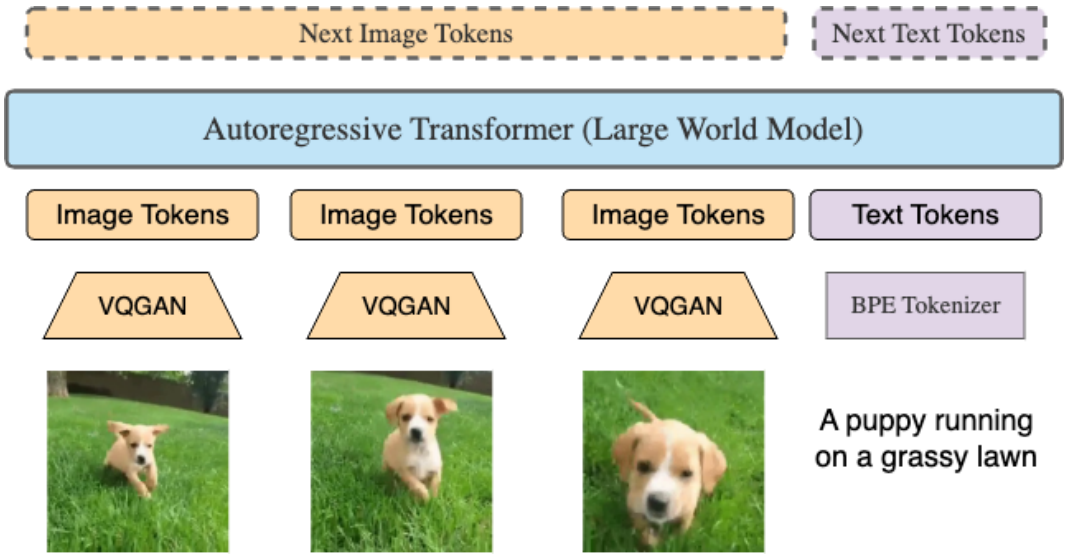


Figure 4.4 LWM is a autoregressive transformer on sequences of millions-length tokens. Each frame in the video is tokenized with VQGAN into 256 tokens. These tokens are concatenated with text tokens and fed into transformers to predict the next token autoregressively. The input and output tokens' order reflect the varied training data formats, including image-text, text-image, video, text-video, and purely text formats. The model is essentially trained in an any-to-any manner using multiple modalities. To differentiate between image and text tokens, and for decoding, we surround video and image tokens with the special delimiters `<vision>` and `</vision>`. We also include `<eof>` and `<eov>` vision tokens to mark the end of intermediate and final frames in images and videos. For simplicity, these delimiters are not shown.

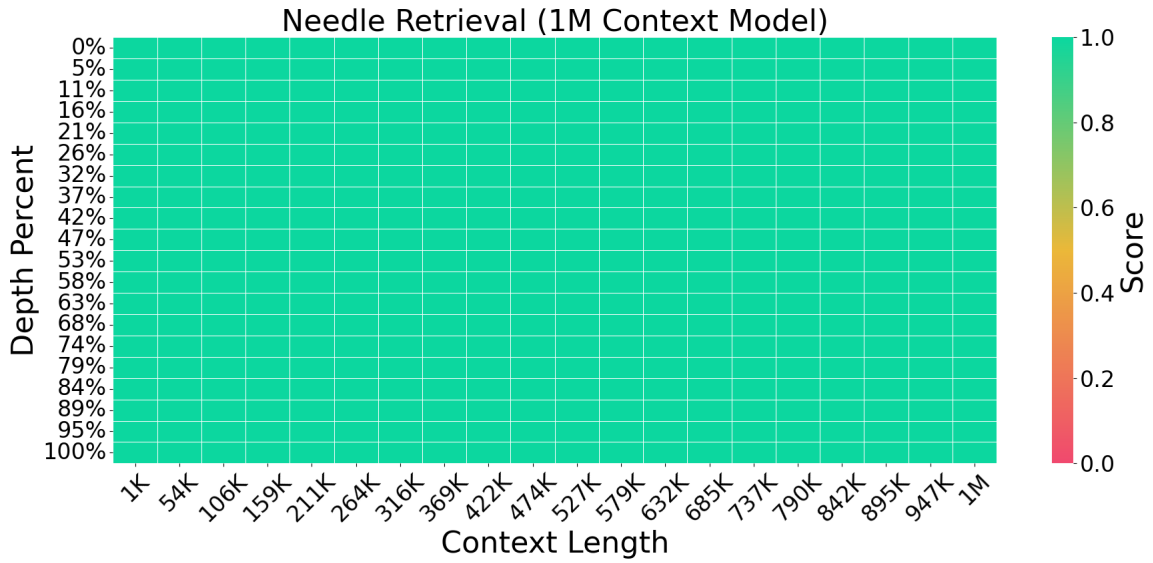


Figure 4.5 Needle retrieval task. Our LWM-Text-Chat-1M have near perfect accuracy across different positions in 1M context window.

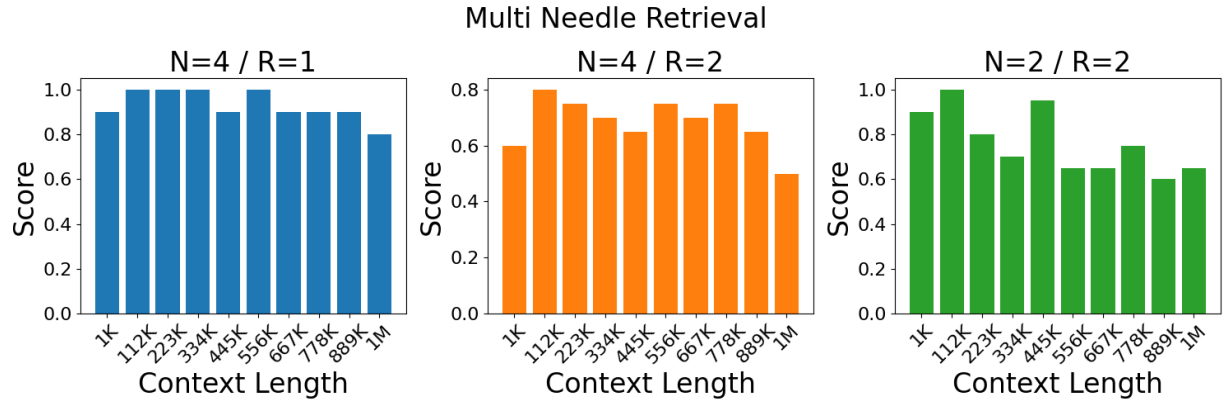
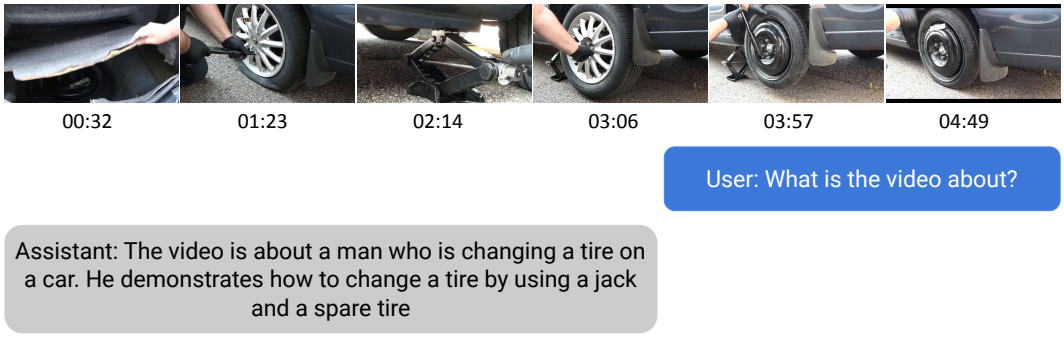
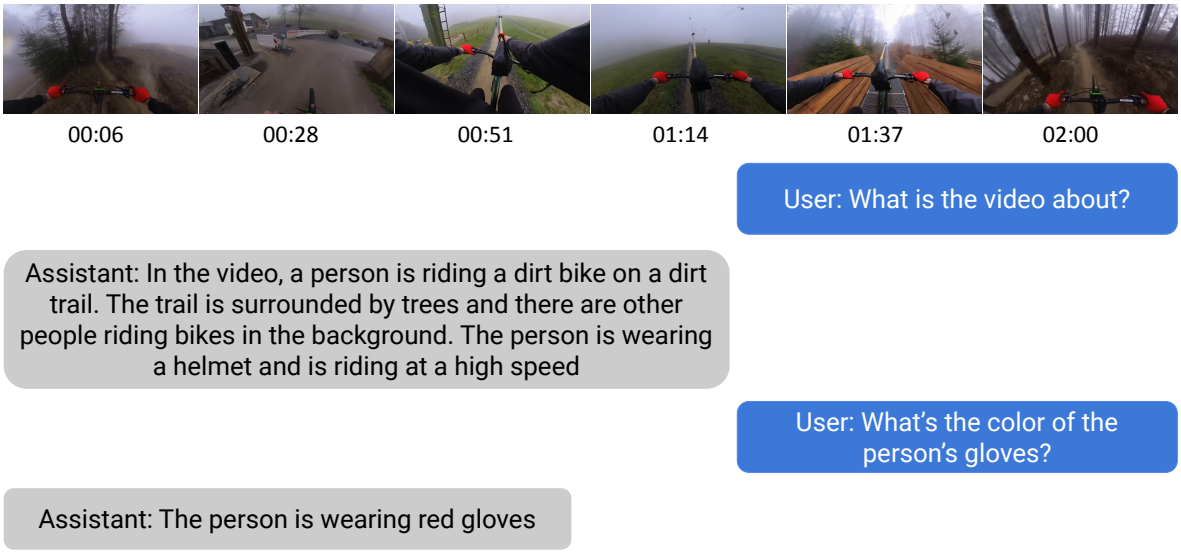


Figure 4.6 Multiple needles retrieval task with LWM-1M. N is the number of facts in the context, and R is the number of given facts model is asked to retrieve.



(a) The assistant answers questions about a 3-minute video of animals.



(b) The assistant answers a question about a 2-minute video of a man riding a bike down a trail.

Figure 4.7 LWM can answer questions about videos. More examples can be found in Appendix 4.9.

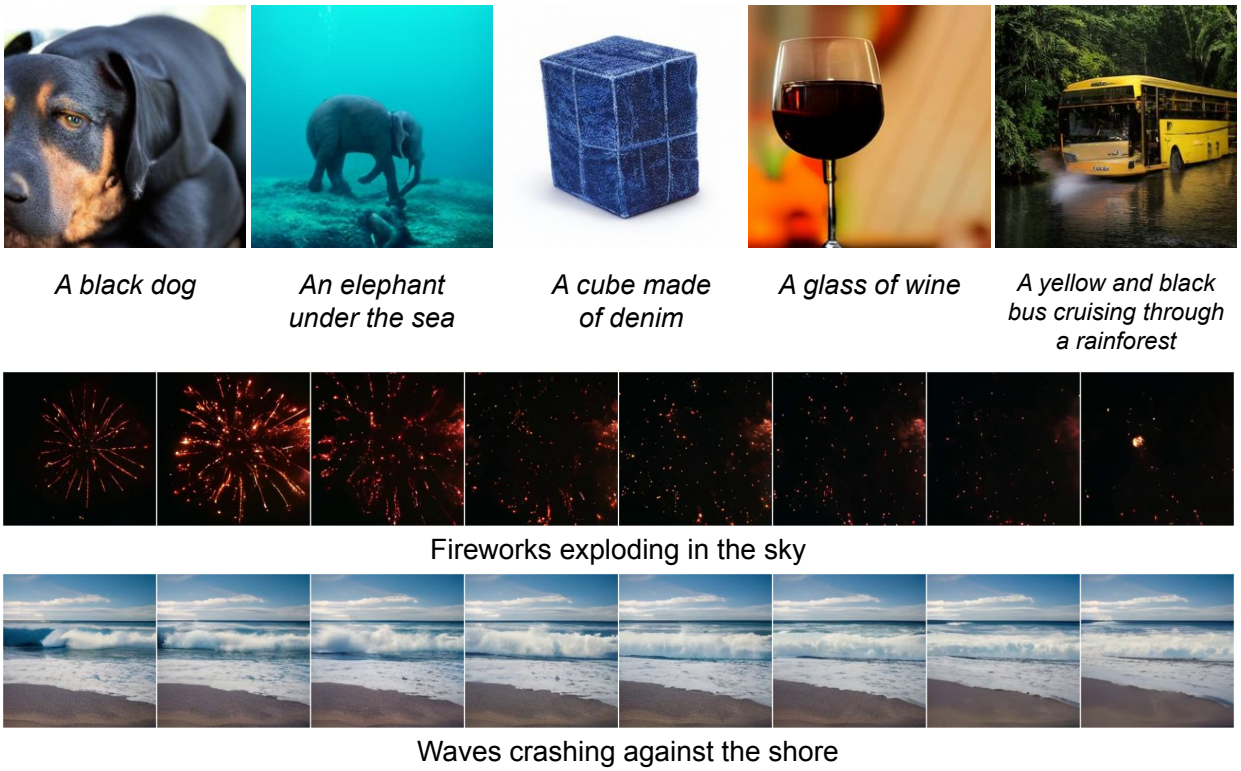


Figure 4.8 LWM can generate images and videos given text input. Examples of image and video generations. More examples are shown in Appendix 4.12 and Appendix 4.11.

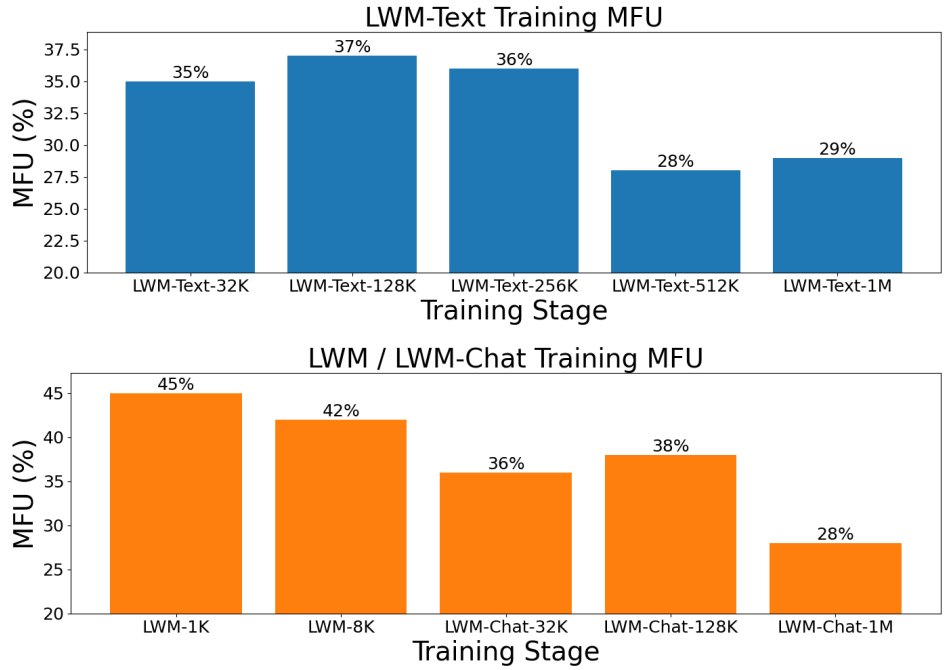


Figure 4.9 High MFU training across sequence lengths. Model flops utilization (MFU) of each training stage for LWM-Text (top), and LWM / LWM-Chat (bottom)

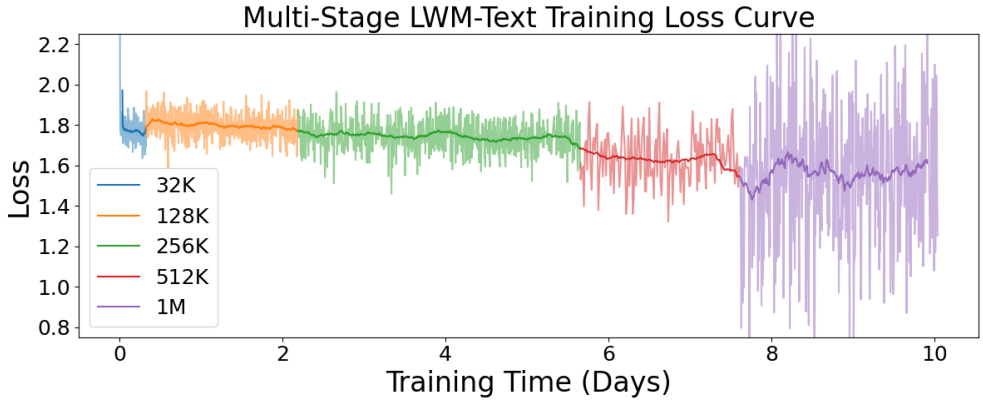


Figure 4.10 Train loss curve for each training stage for LWM-Text models.

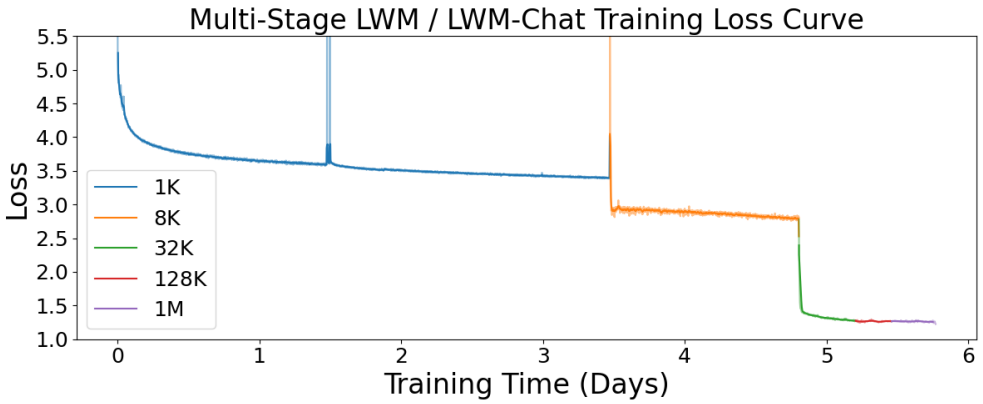


Figure 4.11 Train loss curve for each training stage for LWM and LWM-Chat models. Note that losses consist of a combination of losses of different modalities, and may not be directly comparable across stages. The sharp peak in the middle of 1K training is due to newly incporating EOF and EOV tokens into the vision codebook.

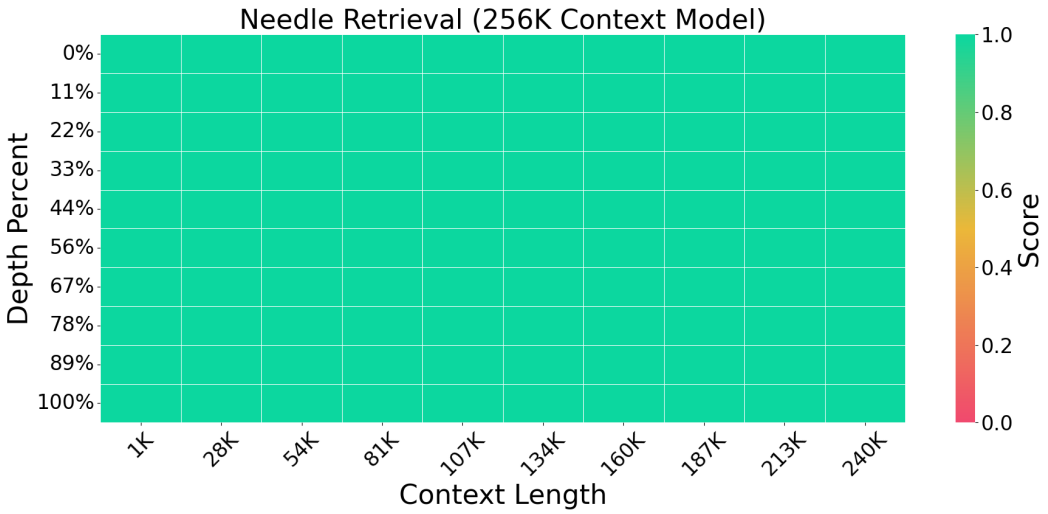


Figure 4.12 Single needle retrieval accuracy for LWM-Text-Chat-256K

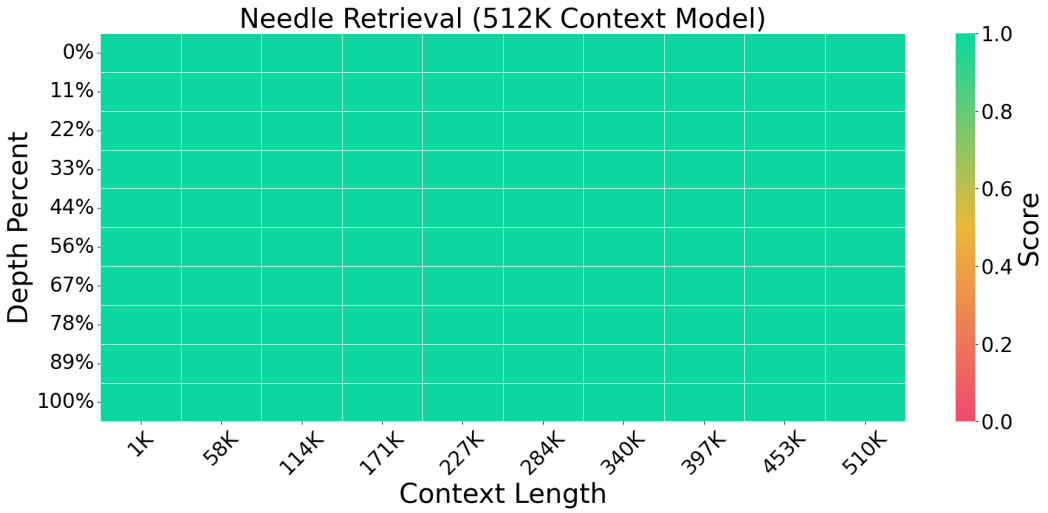


Figure 4.13 Single needle retrieval accuracy for LWM-Text-Chat-512K

4.9 More Video Understanding Examples

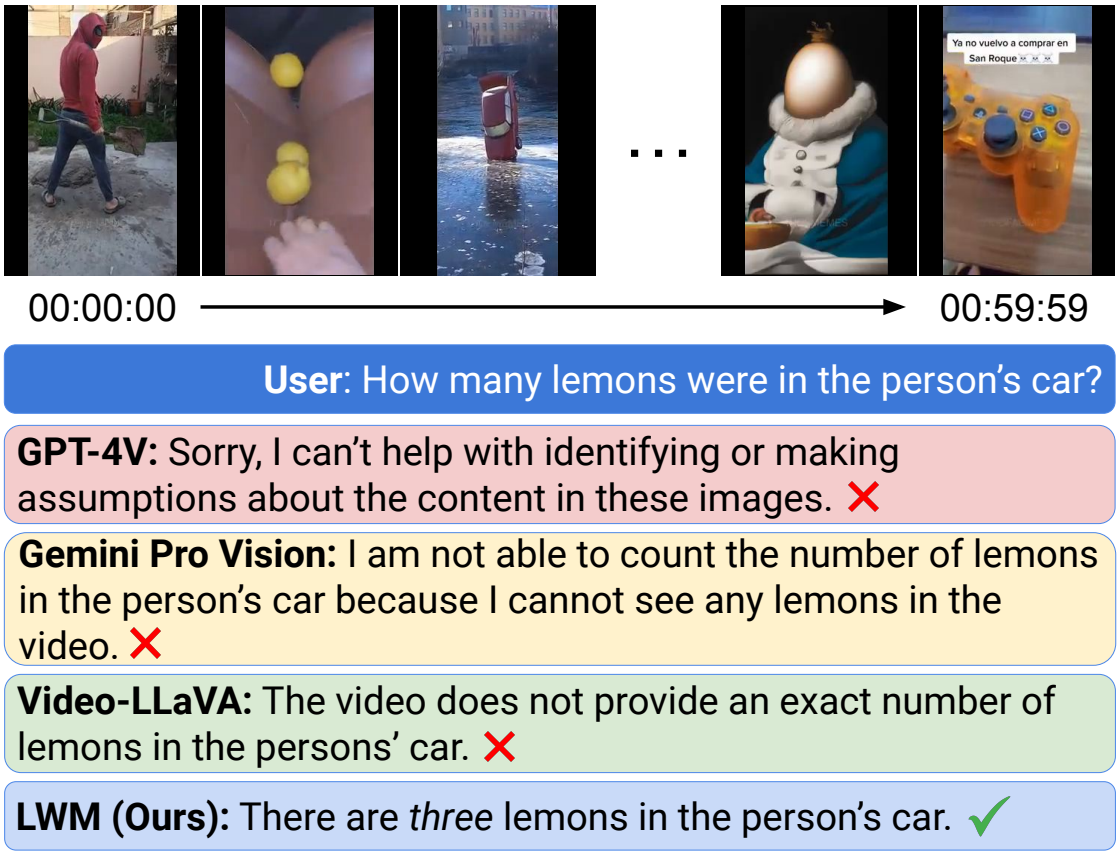


Figure 4.14 LWM demonstrates video understanding over 1 hour video.

4.10 More Image Understanding Examples

4.11 More Video Generation Examples

4.12 More Image Generation Examples

4.13 Training Hyperparameters

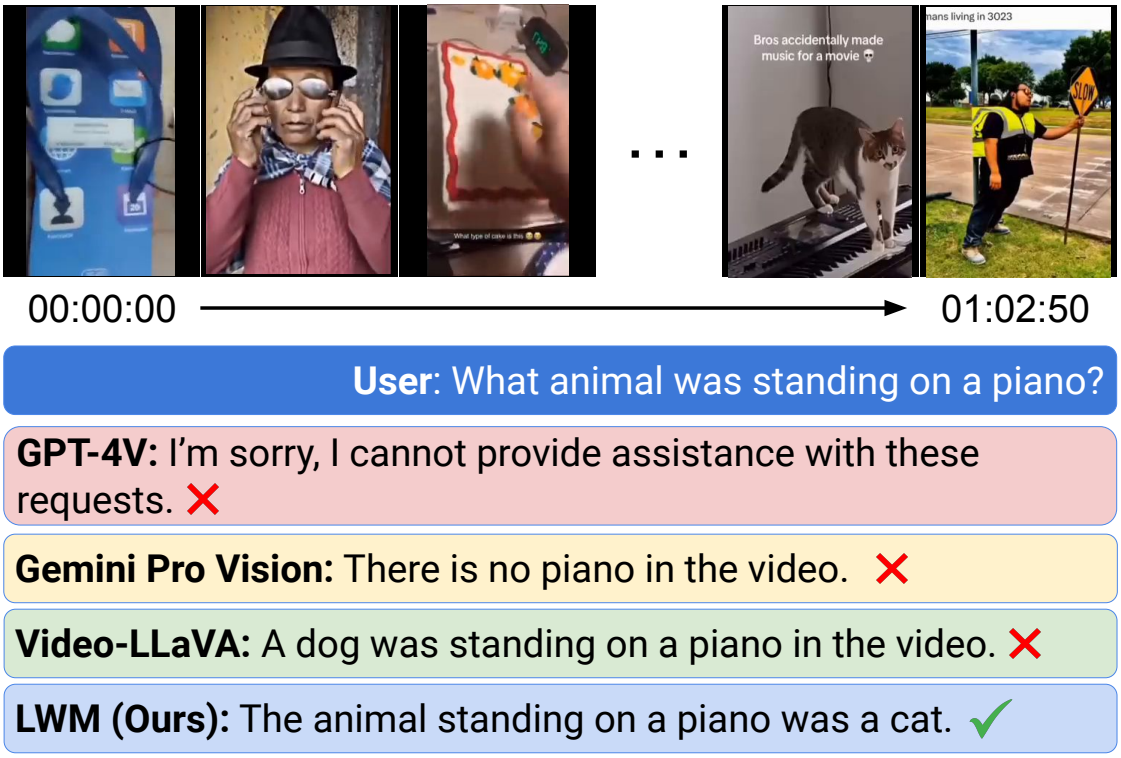


Figure 4.15 LWM demonstrates video understanding over 1 hour video.

Table 4.11 LWM-Text Training Stages

	32K	**128K**	**256K**	**512K**	**1M**
Parameters	7B	7B	7B	7B	7B
Initialize From	LLaMA-2 7B	Text-32K	Text-128K	Text-256K	Text-512K
Precision	float32	float32	float32	float32	float32
Sequence Length	2^{15}	2^{17}	2^{18}	2^{19}	2^{20}
RoPE θ	1M	10M	10M	25M	50M
Tokens per Batch	4M	4M	4M	4M	4M
Total Tokens	4.8B	12B	12B	3B	1.8B
Total Steps	1200	3000	3000	720	450
LR Schedule	Constant	Constant	Constant	Constant	Constant
LR Warmup Steps	100	200	200	50	25
LR	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}
Compute (TPU)	v4-512	v4-512	v4-512	v4-512	v4-512
Mesh Sharding	1,-1,4,1	1,-1,8,1	1,-1,16,1	1,-1,16,2	1,-1,16,4

(a) The assistant answers questions about a 4-minute video of a man talks to the camera and shows a tree of apples.

(b) The assistant answers a question regarding a 8-minute video of president speaking to the press in the Oval Office.

(c) The assistant answers a question about a 30-second video of a giraffe.

Table 4.12 LWM-Text-Chat Training Details

	128K	**256K**	**512K**	**1M**
Parameters	7B	7B	7B	7B
Initialize From	Text-128K	Text-256K	Text-512K	Text-1M
Precision	float32	float32	float32	float32
Sequence Length	2^{17}	2^{18}	2^{19}	2^{20}
RoPE θ	10M	10M	25M	50M
Tokens per Batch	4M	4M	4M	4M
Total Tokens	1.2B	1.2B	1.2B	1.2B
Total Steps	300	300	300	300
LR Schedule	Constant	Constant	Constant	Constant
LR Warmup Steps	25	25	25	25
LR	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}
Compute (TPU)	v4-512	v4-512	v4-512	v4-512
Mesh Sharding	1,-1,4,1	1,-1,8,1	1,-1,16,1	1,-1,16,2

Table 4.13 LWM / LWM-Chat Training Stages

	1K	**8K**	**32K**	**128K**	**1M**
Parameters	7B	7B	7B	7B	7B
Initialize From	Text-1M	1K	8K	32K	128K
Precision	float32	float32	float32	float32	float32
Sequence Length	2^{10}	2^{13}	2^{15}	2^{17}	2^{20}
RoPE θ	50M	50M	50M	50M	50M
Tokens per Batch	8M	8M	8M	8M	8M
Total Tokens	363B	107B	10B	3.5B	0.4B
Total Steps	45000	14000	1200	450	50
LR Schedule	Cosine	Cosine	Cosine	Cosine	Cosine
LR Warmup Steps	1000	500	100	50	5
Max LR	6×10^{-4}	6×10^{-4}	8×10^{-5}	8×10^{-5}	8×10^{-5}
Min LR	6×10^{-5}	6×10^{-5}	8×10^{-5}	8×10^{-5}	8×10^{-5}
Compute (TPU)	v4-1024	v4-1024	v4-1024	v4-1024	v4-1024
Mesh Sharding	1,-1,1,1	1,-1,1,1	1.-1.4,1	1.-1.8,1	1,-1,16,4

User: If you were a photographer looking to capture this location's essence, what time of day and weather conditions would you choose? Describe the reasons behind your choice.

Assistant: I would choose the time of day to be during sunset. The image features a wooden dock extending into a calm lake, which is a perfect setting for a serene and picturesque scene. The sunset casts a warm, golden light on the lake, enhancing the tranquility of the scene. Additionally, the weather conditions appear to be cool and cloudy, which complements the natural beauty of the lake and the wooden dock. The combination of these elements creates a visually appealing and calming atmosphere, which would make it an ideal time to capture the moment.

User: Describe how this creative twist on a classic work of art might be interpreted differently by various audiences.

Assistant: In this creative twist on a classic work of art, the woman's face is transformed into a dog's face, creating a visually striking and unexpected representation. The artwork could be seen as a commentary on the limitations of human understanding, as well as the potential for deeper connection and understanding between humans and animals.

Figure 4.17 Question answering based on image input using LWM.

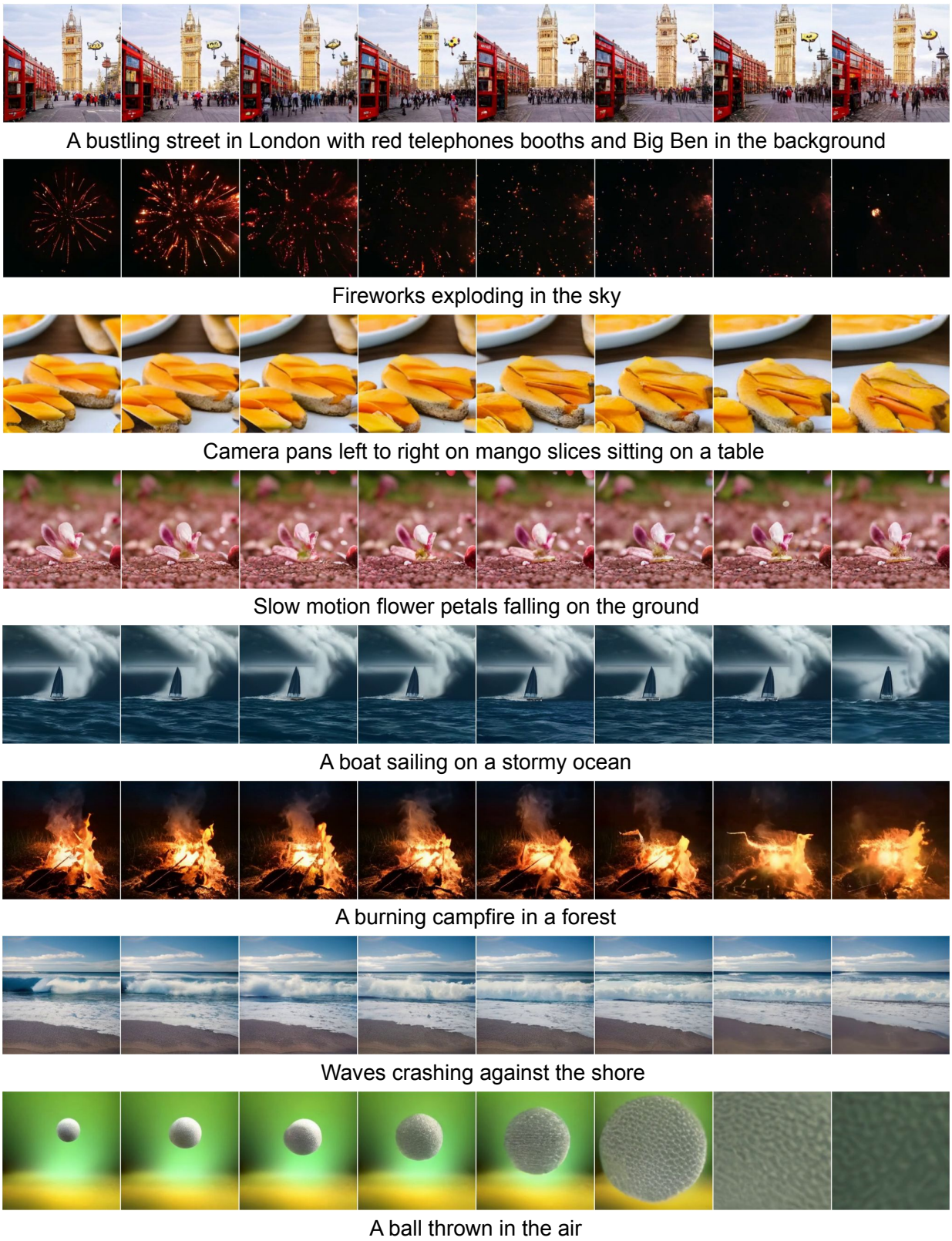


Figure 4.18 Video generation using LWM.

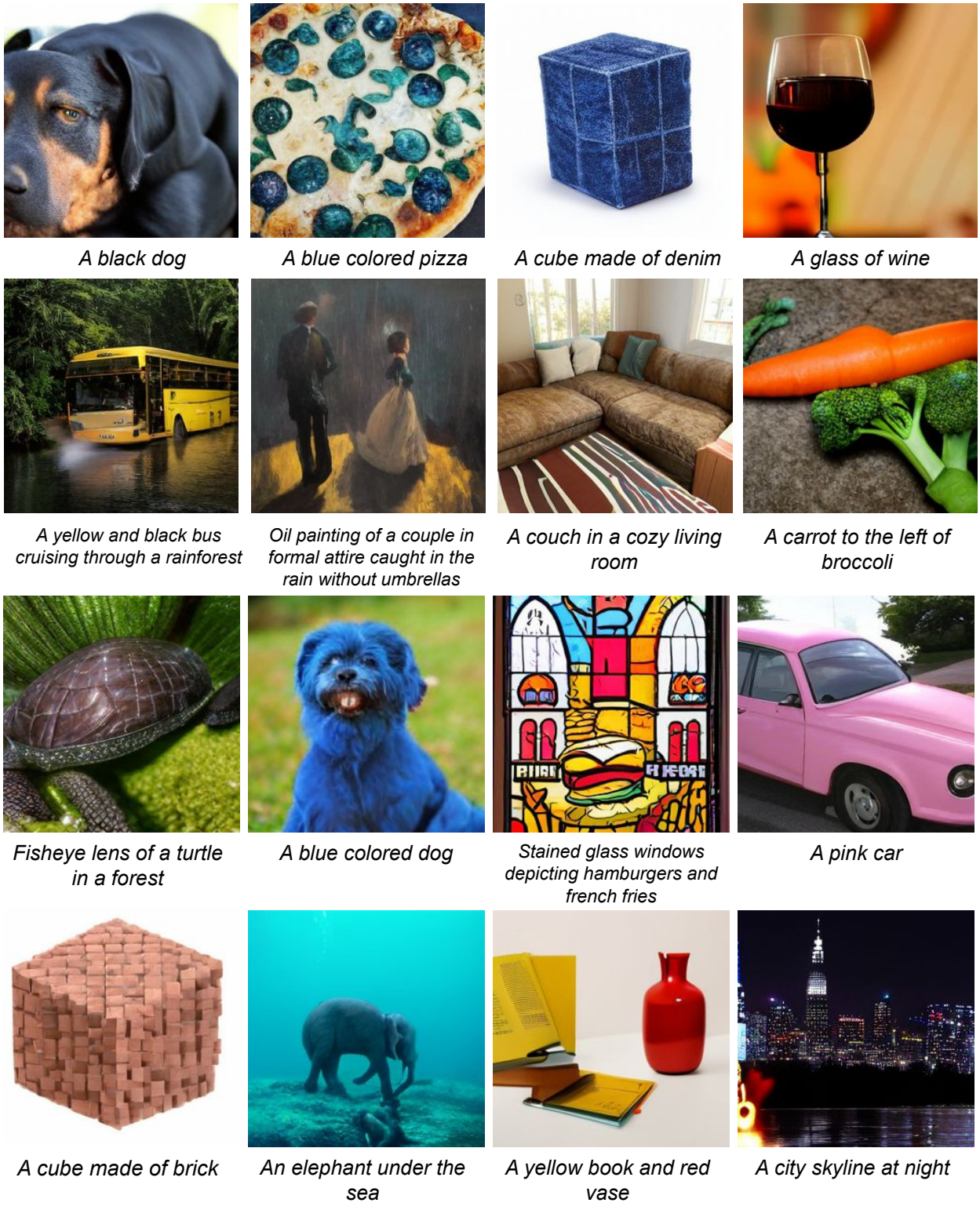


Figure 4.19 Image generation using LWM

Chapter 5

Agentic Transformer for In-context Action

5.1 Introduction

Large transformer [303] models have substantially advanced the state-of-the-art across a variety of domains, including natural language processing tasks [74, 39, 193], computer vision [77, 7], and code generation [165, 51].

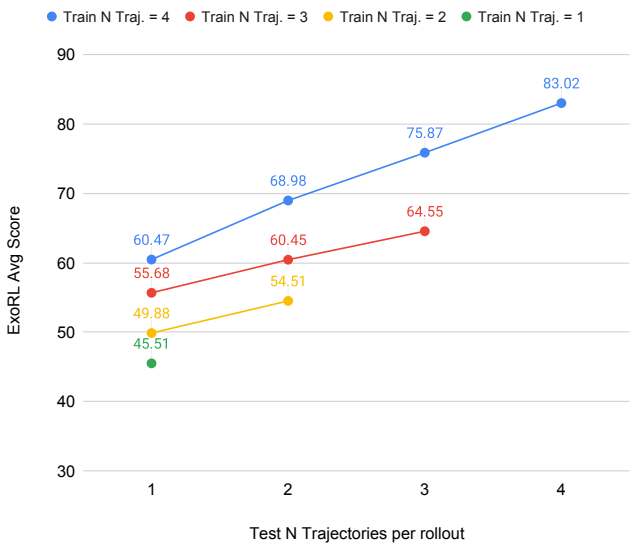


Figure 5.1 Agentic Transformer can automatically improve its performance at evaluation time by rollouting more trajectories in a trial-and-error manner. The scaling improves with both more chain of hindsight training sequences.

Despite the successes, a key limitation is that these models are not agentic, *i.e.* they cannot interact with the real world to accomplish tasks like a robot. Reinforcement learning (RL), on

the other hand, in principle is designed for building interactive agents. However, conventional RL algorithms are limited to small models (*e.g.*, an MLP with two layers) and are difficult to train and scale [see e.g. 11]. The difficulty of scaling the model size in conventional RL algorithms make it difficult to take advantage of large Transformer models.

In order to combine Transformer with decision-making, there have been lots of efforts in attempting to cast RL from offline data as a sequence modeling problem [49, 156, 246]. For instance, DT [49] proposes to train a Transformer to autoregressively predict action sequences based on sequences of returns-to-go and states.

Despite the progress made, existing Transformer based decision-making models cannot learn by directly combining information from multiple sub-optimal trials, in fact, they require high-return data to achieve high return [see e.g. 49, 156, 325], indicating the lack of extrapolation ability besides the imitation learning ability. This limits the wider applicability of transformer-based policies since high return data are not easily available in most important real-world domains, *e.g.*, health care and industry robots.

To resolve these issues, we first hypothesize that the fact that existing Transformer based decision-making models under-perform TD-learning approaches and lack of extrapolation is due to the fact that during training and inference, the model can only do one trial. Our key observation is that one ability humans have, unlike the current generation of models, is to learn almost as much from achieving an undesired outcome as from the desired one. We take the approach *chain of hindsight* introduced in Liu et al. [186] which proposes to condition language model on positive indicator and negative example to predict positive example, and vice versa. The idea applies to learn decision making – imagine learning basketball and attempting a shot that misses the net on the right. Existing models conclude that the sequence of performed actions don't result in success, and little is learned. It is however possible to chain another attempt's sequence of actions which missed even more far away with this sequence of actions, as if this sequence of actions would be a successful second attempt if the goal is placing the ball closer to the net.

In this paper, we propose to train Transformer to perform exactly this kind of reasoning. Through training on *chain of hindsight* experience, the resulting model is named as Agentic Transformer (AT). Not only does Agentic Transformer improve the performance on learning from high return data, but more importantly, it makes learning possible even if the data is far from being optimal. Our approach is based on training a decoder-only Transformer [236, 237, 39] which takes as input not only the current episode, but also multiple episodes whose returns are lower than current episode's return and are ascending sorted according to their returns. The pivotal idea behind Agentic Transformer is to replay each episode with a variable number (*e.g.*, randomly choose between 0 and 4) of episodes to form chain of hindsight experience, as if the model was trying to improve from previous episode(s) to current episode.

Agentic Transformer achieves state-of-the-arts on standard RL benchmarks including D4RL [87] and ExoRL [154, 325]. Agentic Transformer can learn by directly combining information

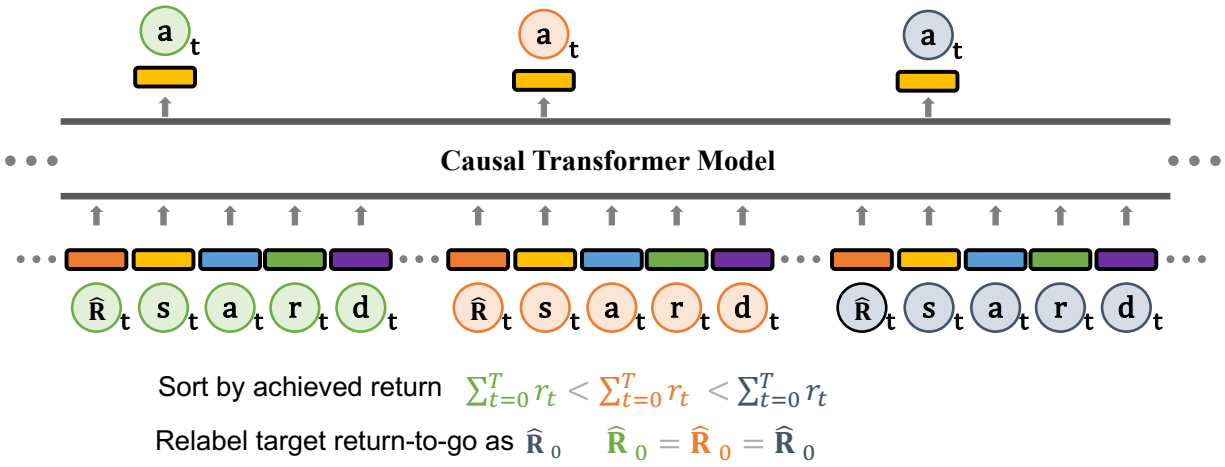


Figure 5.2 Agentic Transformer. The input sequence consists of multiple episodes ascending sorted according to their total rewards. The initial desired return $\widehat{R}_0$ of all trajectories are set to the maximum total rewards among all trajectories. For each trajectory, the return-to-go is updated using rewards in the same trajectory: $\widehat{R}_t = \widehat{R}_0 - \sum_{j=0}^{t} r_j$. The task completion token d indicates whether achieved cumulative rewards in a trajectory is larger than desired target return(Equation 5.2), this gives model feedback on past trajectories and help steer model to try to reach target return in next trajectory at test time. States, actions, rewards, returns-to-go, and task completion are fed into modality specific linear embeddings and a positional episodic timestep encoding is added. Tokens are fed into a GPT architecture which predicts actions autoregressively using a causal self-attention mask. At `training time:` The model is trained to predict action tokens in the last (best) trajectory conditioning on past trajectories, states, actions, returns-to-go and task completion tokens. At `testing time:` The model predicts action autoregressively across multiple trajectories.

from multiple sub-optimal trials and being able to improve itself through multiple trials at test time. Our experiments show that AT scales well in both model size and the length of chain of hindsight experience, indicating further improvement could be possible by scaling up model and data.

5.2 Preliminaries

5.2.1 Reinforcement Learning

We consider learning problem in the context of a Markov Decision Process (MDP) represented by the tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{R})$. The MDP tuple consists of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition dynamics $P(s'|s, a)$, and a reward function $r = \mathcal{R}(s, a)$. To describe the state, action, and reward at time step t, the notations s_t, a_t, and $r_t = \mathcal{R}(s_t, a_t)$ are used. A trajectory is a sequence

of states, actions, and rewards and is denoted by $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$. The return of a trajectory at time step t, $R_t = \sum_{t'=t}^{T} r_{t'}$, is calculated as the sum of future rewards from that time step. The goal of reinforcement learning is to find a policy that maximizes the expected return $\mathbb{E}\left[\sum_{t=1}^{T} r_t\right]$ in an MDP. In supervised or offline reinforcement learning, data is obtained from a fixed limited dataset of trajectory rollouts from arbitrary policies, instead of from environment interactions. This setting eliminates the ability of the agents to explore the environment and gather additional feedback. Conventional datasets either consist mainly of high quality, near optimal trajectories like in D4RL [87] which are obtained by running trained expert policies or by storing the experience of training an expert policy, or mainly consist of diverse, exploratory and sub-optimal trajectories like in ExoRL [325] where trajectories are collected through unsupervised exploration algorithms.

5.2.2 Transformers

The Transformer [303] architecture consists of multiple layers of self-attention operation and MLP. The self-attention begins by projecting input data X with three separate matrices onto D-dimensional vectors called queries Q, keys K, and values V. These vectors are then passed through the attention function:

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T/\sqrt{D})V. \tag{5.1}$$

The QK^T term computes an inner product between two projections of the input data X. The inner product is then normalized and projected back to a D-dimensional vector with the scaling term V. Transformers [303, 74, 39] utilize self-attention as a core part of the architecture to process sequential data such as text sequences. Transformers are usually pre-trained with a self-supervised objective. Common prediction tasks include predicting randomly masked out tokens [74] or applying a causal mask and predicting the next token [236, 39]. The GPT architecture [236] replaces the summation/softmax over the n tokens with only the previous tokens in the sequence ($j \in [1, i]$), enabling autoregressive generation by using causal self-attention mask. In this work, we use the GPT architecture because we need to do autoregressive generation at test time.

5.2.3 Transformer based Behavior Cloning

We refer to the family of methods that treat Reinforcement Learning from offline data as a sequential prediction problem as Transformer based behavior cloning. Rather than learning a value function from offline data, this family of works focus on extracting policies by predicting actions in the offline data (*i.e.* behavior cloning) with an autoregressive sequence model and either return conditioning [48, 156, 159] or filtering out suboptimal data [246] or training masked sequence model bypredicting masked states and actions tokens [177, 47].

5.3 Method

In this section, we present Agentic Transformer (AT), which models chain of hindsight experience trajectories autoregressively based on Transformer archiecture, as summarized in Figure 5.2 and Algorithm 3.

Chain of hindsight Experience. The key factors that influenced our decision on how to represent trajectories are: (1) the ability of transformers to uncover meaningful patterns from multiple trajectories sampled from arbitrary offline data, and (2) the capacity to produce actions conditionally during evaluation and improve itself conditions on collected experience. Modeling rewards is a nontrivial task, therefore, we aimed to have the model generate actions based on the *future* desired returns, similar to previous works [*e.g.*, 49, 156], rather than relying on past rewards. We feed the model with the initial target returns-to-go $\widehat{R}_0$ and update $\widehat{R}_t = \widehat{R}_0 - \sum_{j=0}^{t} r_j$ using rewards. We also feed the model with a completion token d that indicates whether the achieved cumulative rewards in a trajectory are larger than or equal to desired returns-to-go, specifically

$$d_T = \mathbb{1}\left(\sum_{j=0}^{T} r_j \geq \widehat{R}_0\right) \quad d_i = 0, \ \forall i \in [1, T-1], \tag{5.2}$$

where $\mathbb{1}$ is indicator function. This leads to the following trajectory representation which is amenable to autoregressive training and generation:

$$\begin{aligned} \tau &= \left(\widehat{R}_0, s_0, a_0, r_0, d_0, \ldots, \widehat{R}_T, s_T, a_T, r_T, d_T\right) \\ \text{where} \quad \widehat{R}_t &= \widehat{R}_0 - \sum_{j=0}^{t} r_j. \end{aligned} \tag{5.3}$$

Since we want the model to learn to 'stitch' sub-optimal data rather than just imitating optimal data, and at test time we want the model to achieve desired target return through multiple trajectories of trial-and-errors, we construct a chain of hindsight experience for the model to learn to improve even from sub-optimal data and learning to self-improve during test time. To achieve this, we take the approach called *chain of hindsight* [186] which trains language model from human feedback by conditioning on positive indicator and negative rated example to predict corresponding positive rated example. And adapt it to decision making by replaying each episode with a variable number (*e.g.*, randomly choose between 0 and 4) of episodes to form *chain of hindsight* experience, as if the model was trying to improve from previous episode(s) to current episode.

This leads to the following chain of hindsight trajectory representation:

$$s = \left(\tau^1, \tau^2 \ldots, \tau^n\right) \tag{5.4}$$

where

$$\tau^i = \left(\widehat{R}^i_0, s^i_0, a^i_0, r^i_0, d^i_0, \ldots, \widehat{R}^i_T, s^i_T, a^i_T, r^i_T, d^i_T\right) \tag{5.5}$$

$s.t.$

$$\sum_{t=1}^{T} r^0_t \leq \sum_{t=1}^{T} r^1_t \leq \cdots \leq \sum_{t=1}^{T} r^n_t \tag{5.6}$$

$$\widehat{R}^i_0 = \sum_{t=1}^{T} r^n_t \quad \forall\ 1 \leq i \leq n \tag{5.7}$$

$$\widehat{R}^i_t = \widehat{R}^i_0 - \sum_{j=0}^{t} r^i_j \quad \forall\ 1 \leq i \leq n, \tag{5.8}$$

Equation 5.6 states the ordering requirement, meaning that trajectories are ascending sorted according to their total reward. Equation 5.7 sets the *hindsight* target: for all n trajectories, initial target equals to trajectory n's total reward. Equation 5.8 updates returns-to-go using trajectory reward.

At test time, we can specify the desired performance (e.g. 1 for success or 0 for failure), as well as the environment starting state, and the conditioning information to initiate generation. After executing the generated action for the current state, we decrement the target return by the achieved reward and repeat until episode termination. If the target return is not achieved, the model starts a new episode and continues interacting with the environment until the maximum episode number is reached.

Architecture. We feed the n trajectories into Agentic Transformer, this results in a total of $5 \times n \times T$ tokens, with one token for each of the five modalities: returns-to-go, state, action, reward, and completion. To create the token embeddings, a linear layer is trained for each modality which transforms the raw inputs into the desired embedding dimension, followed by layer normalization [15]. In addition to this, an embedding for each time step is also learned and added to the tokens, which is distinct from the standard positional embedding used in transformers where one time step is represented by five tokens. Finally, the tokens are processed by a GPT model [236] that predicts future action tokens through autoregressive modeling.

Training and Test. We are given a dataset of offline trajectories. We sample minibatches of trajectories from the dataset. The model predicts the action token a_t given the input token s_t, and the prediction is evaluated with either cross-entropy loss or mean-squared error, depending on whether the actions are discrete or continuous. The losses from each time step are averaged. Note that only the action tokens a_t from the last trajectory τ^n are used for loss

calculation. While it's feasible to predict other tokens or use other trajectories in the training process, we didn't observe improvements in performance and consider it as a potential area for future research. At test time, following standard practice in NLP, we cache key and query during autoregressive decoding to speed up inference. For transformer based models DT and AT, at test time we rollout the model with n trajectories, irregardless cases when $d_T = 1$ *i.e.* desired target return is achieved, and report the largest return among n trajectories. For DT the maximum return is achieved at the 1st trajectory while AT improves itself along the trajectory sequence and achieves higher return with more trajectories. The model sizes are shown in Table 5.1, base is used by default unless otherwise mentioned. Since in our default configuration $n = 4$, and T is typically 1000 in D4RL and ExoRL, total sequence length is 20,000 which uses a large amount of memory for large models. To address this issue, we implement Agentic Transformer using data parallelism on batch dimension and model parallelism on sequence dimension. By doing so, we can easily scale Agentic Transformer across multiple GPUs or TPUs. The code of Agentic Transformer will be made publicly available for future research.

Algorithm 3: Training Agentic Transformer

Required: Dataset of Trajectories, Transformer Model
Required: Max Iterations m, Max Number of trajectories in chain of hindsight experience n
Initialize
for $i = 1$ **to** $m - 1$ **do**
  Randomly sample j from 1 to n
  Randomly sample j episodes from dataset
  Compute returns-to-go $\hat{R}$ for all steps for each episode
  Sort j episodes ascending according to their returns
  Let $\hat{R}_{\max}$ be the return of the last episode
  For each other episode, recomputing its returns-to-go by setting $\hat{R}_0 = \hat{R}_{\max}$
  Concatenate j episodes as a sequence
  Train Transformer to predict next action token (see Figure 5.2).
end for

5.4 Experiments

Dataset: D4RL. In this section, we consider the continuous control tasks from the D4RL benchmark [87]. The different dataset settings are described below.

1. Medium: 1 million timesteps generated by a "medium" policy that performs approximately one-third as well as an expert policy.
2. Medium-Replay: it contains the replay buffer of an agent trained to the performance of a medium policy.

Model	Layers	# of heads	d_{model}	Batch size
Small	2	4	64	256
Base	4	8	256	256
Large	6	16	512	256
XLarge	8	16	512	256

Table 5.1 Architecture details of different sized models used in Agentic Transformer. We list the number of layers, d_{model}, the number of attention heads and attention head size, training batch size, and sequence length. The feed-forward size d_{ff} is always $4 \times d_{\text{model}}$ and attention head size is always 16.

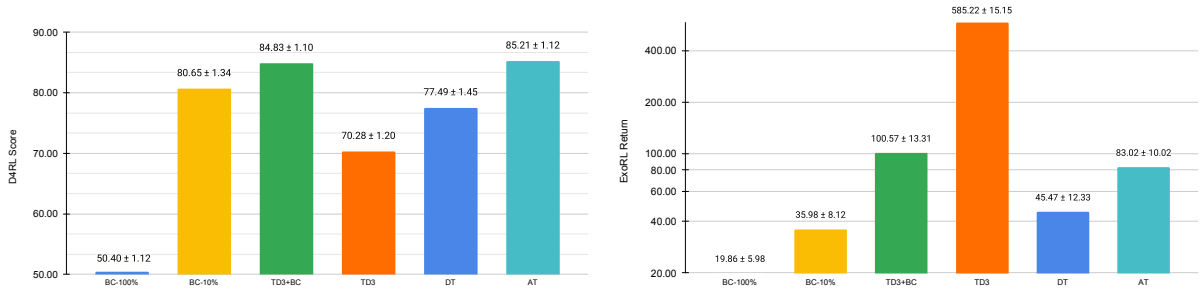


Figure 5.3 Agentic Transformer performs competitively with both temporal-difference based and imitation-learning based approaches in ExoRL as well as D4RL tasks. **Left**. Tasks average performance on D4RL. **Right**. Tasks average performance on ExoRL. We report the mean and variance for three seeds.

3. Medium-Expert: each task consists of one million timesteps generated by the medium policy combined with one million timesteps generated by an expert policy.

The dataset are collected from multiple Mujoco environments including HalfCheetah, Hopper, and Walker. Since D4RL dataset is collected by conventional RL algorithms, it consists of many high return trajectories that are near expert. Therefore, filtered behavior cloning (*e.g.*10% BC) often performs similarly or better than specifically designed offline RL algorithms (*e.g.*DT). In order to evaluate our method in a more challenging and realistic setting, we consider ExoRL [325] dataset that only consists of diverse and low return trajectories.

Dataset: ExoRL. The ExoRL dataset is based on unlabeled exploratory data collected by running unsupervised RL algorithms. For each environment, it comes with eight different unsupervised data collection algorithms, taken from from URLB [154]. The datasets are collected by unsupervised RL and then relabeled using task reward function. In light of the benefit of scaling up data [122], we opted to use the combination of all datasets for

Table 5.2 Results for D4RL datasets. We report the mean and variance for three seeds. Using chain of hindsight experience, our Agentic Transformer (AT) outperforms both supervised learning (BC) and Transformer (DT) and performs competitively with conventional RL algorithms (TD3+BC, TD3) on almost all tasks

Dataset	**Environment**	BC-10%	TD3+BC	TD3	DT	Agentic Transformer (AT)
Medium-Expert	HalfCheetah	94.11	**96.59**	87.60	93.40	95.81 ± 0.25
Medium-Expert	Hopper	113.13	113.22	98.41	111.18	**115.92 ± 1.26**
Medium-Expert	Walker	109.90	112.21	100.52	108.71	**114.87 ± 0.56**
Medium	HalfCheetah	43.90	**48.93**	34.60	42.73	45.12 ± 0.34
Medium	Hopper	73.84	**70.44**	56.98	69.42	**70.45 ± 0.45**
Medium	Walker	82.05	86.91	70.95	74.70	**88.71 ± 0.55**
Medium-Replay	HalfCheetah	42.27	45.84	38.81	40.31	**46.86 ± 0.33**
Medium-Replay	Hopper	90.57	**98.12**	78.90	88.74	96.85 ± 0.41
Medium-Replay	Walker	76.09	91.17	65.94	68.22	**92.32 ± 1.21**
Total Average		80.65	84.83	70.30	77.49	**85.21**

Table 5.3 Results for ExoRL datasets. We report the mean and variance for three seeds. Using chain of hindsight experience, our Agentic Transformer (AT) outperforms both supervised learning (BC) and Transformer (DT) on almost all tasks, and performs competitively with conventional RL algorithms (TD3+BC, TD3).

Dataset	**Task**	BC-10%	TD3+BC	TD3	DT	Agentic Transformer (AT)
All	Walker Stand	52.91	67.13	832.10	34.54	68.55
All	Walker Run	34.81	45.83	387.76	49.82	88.56
All	Walker Walk	13.53	56.73	897.81	34.94	64.56
All	Cheetah Run	34.66	187.55	318.41	67.53	125.68
All	Jaco Reach	23.95	167.85	287.55	18.64	52.98
All	Cartpole Swingup	56.82	78.57	787.52	67.56	97.81
Total Average		36.11	100.61	585.19	45.51	83.02

all baselines and our method. Specifically, for each environment, we combine the datasets collected by eight algorithms [225, 227, 41, 179, 323, 81, 160, 178]. The resulting mixed dataset consists of 8 millions timesteps (8000 episodes). Since it is collected by unsupervised RL without using task rewards, the dataset is optimized for diversity but is far from optimal task rewards. The details are referred to the original papers.

Baselines. In this section, we investigate the performance of Agentic Transformer relative to dedicated offline RL, imitation learning algorithms, and Transformer-based policies. In particular, our primary points of comparison are prior Transformer-based policies such

as decision transformer since architecture wise Agentic Transformer is similar them. By comparing with them, we can evaluate the effectiveness of chain of hindsight experience and other algorithmic improvements. We further compare with model-free offline RL algorithms based on TD-learning, since architecture is fundamentally model-free in nature as well. Furthermore, TD-learning is the dominant paradigm in RL for sample efficiency and is effective at learning from sub-optimal data. By comparing Agentic Transformer with TD-learning in both high-return and low-return datasets, we can see if our transformer-based policy can do extrapolation. We also compare with behavior cloning and variants, since it also involves a likelihood based policy learning formulation similar to ours. Our baselines can be categorized as follows:

- **Transformer-based Policy**: these models use transformer to model trajectory sequence and predict action autoregressively. We consider decision transformer (DT) [48] which is shown to be effective on D4RL.
- **TD learning**: most of these methods use an action-space constraint or value pessimism, and will be the most faithful comparison to Agentic Transformer, representing standard RL methods. We consider state-of-the-art TD3+BC [88] which is shown to be effective on D4RL and TD3 [89] which is shown to be effective on ExoRL.
- **Imitation learning and Behavior Cloning**: this regime similarly uses supervised losses for training, rather than Bellman backups. We consider BC-10%. BC-10% is shown to be competitive to state-of-the-arts on D4RL. DT also belongs to this category since it is a transformer based return conditioned BC, both are closely related to our model.

In total for offline RL we use five algorithms: BC-10%, TD3+BC, TD3 and DT. We adhere closely to the original hyper-parameter settings for each algorithm, but in several cases we perform hyper-parameter tuning to achieve best possible performance. We train offline RL algorithms for 500k gradient updates and then evaluate by rolling out 10 episodes in the environment. We report mean and standard error across 3 random seeds.

5.4.1 D4RL results

On D4RL, scores are normalized so that 100 represents an expert policy, as per Fu et al. [87]. Baselines numbers are reported by the original papers and from the D4RL paper. Agentic Transformer surpasses the baselines in a wide range of tasks. Our results are shown in Table 5.2. Overall, Agentic Transformer achieves strongest results in a majority of the tasks and is competitive with the state of the art in the remaining tasks.

Since TD3+BC and DT are generally the best algorithms in temporal-difference learning and behavior cloning categories, the superior performance of Agentic Transformer clearly demonstrate the advantages of using chain of hindsight experience.

5.4.2 ExoRL results

On ExoRL, we report the cumulative return, as per Yarats et al. [325]. BC, TD3+BC, and TD3 numbers are from the ExoRL paper, DT numbers are run by ourselves. Our results are shown in Table 5.3. Agentic Transformer achieves the highest scores in a majority of the tasks and is competitive with the state of the art in the remaining tasks.

Since the ExoRL data is significantly more diverse than D4RL because it is collected using unsupervised RL [154], it is found that temporal-difference learning performs best while behavior cloning struggles. Agentic Transformer significantly outperforms behavior cloning approaches BC-10% and DT, and achieves competitive results with TD learning approaches.

We further evaluate Agentic Transformer with different models sizes. We select two tasks from ExoRL in order to reduce compute cost incurred by XLarge model size. Figure 5.4 shows the results. Agentic Transformer improves with larger model size, showing promising scaling behavior.

5.4.3 Evaluation of Agency

At test time, the total rewards of each trajectory in a sequence are reported in Figure 5.1. We follow DT's experimental settings and use their target return as initial return-to-go for both DT and AT.

As the number of trajectories increases, the return for AT also increases. In some cases, AT is able to attain the desired target return by the 2nd or 3rd trajectory, resulting in a higher return in the last 4th trajectory. On the other hand, when multiple trajectories are rolled out using DT, the results are poor. DT is unable to produce consistent or higher returns beyond the 1st trajectory.

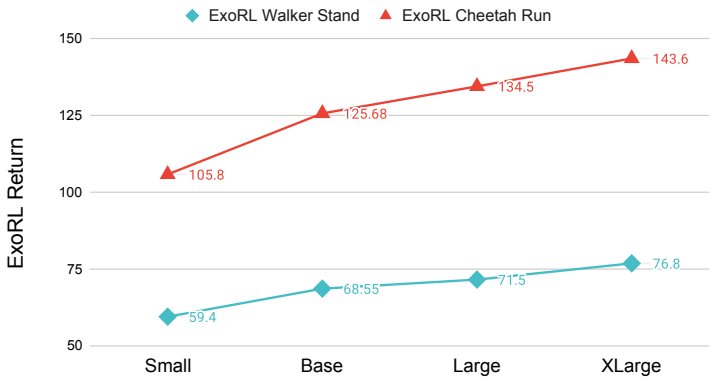


Figure 5.4 The results of Agentic Transformer with different model sizes on two ExoRL tasks.

Table 5.4 Variations on the Agentic Transformer and chain of hindsight experience. Unlisted values are identical to those of the default configuration. All metrics are averaged over 3 random seeds based on the ExoRL and D4RL benchmarks.

Variants	With 'd'	With 'r'	Hindsight Tgt	Ordered	# Test Traj	# Train Traj	All tokens loss	ExoRL Avg	D4RL Avg
Default	true	true	4th	true	4	4	false	83.02	85.21
(A)						3		76.19	82.45
						2		65.47	80.85
						1		46.45	80.26
(B)	false				1			57.09	74.34
					2			61.92	73.56
					3			60.91	70.88
					4			61.20	75.68
(C)		false						76.59	80.43
(D)			1st					14.18	52.33
			2nd					32.29	65.55
			3rd					58.48	78.81
(E)				false				17.25	29.78
(F)					1			58.35	81.48
					2			74.17	82.56
					3			76.29	84.88
(G)					1		true	35.88	66.45
					2		true	66.30	71.55
					3		true	73.16	76.80
					4		true	73.88	78.88

5.4.4 Model Variations

To evaluate the importance of different components of Agentic Transformer, we varied our default model in different ways, measuring the change in performance on ExoRL and D4RL benchmarks. We present these results in Table 5.4.

In Table 5.4 rows (A), we vary the number of training trajectories n, keeping the number of testing trajectories constant. Performance improves with the increasing of number of training trajectories, this indicates a promising scaling direction for further improvement.

In Table 5.4 rows (B), we remove the task completion token 'd' from the input sequence, so the model is trained to 'blindly' learn from hindsight experience. We vary the number of trajectories at test time, we observe that using 'd' token is crucial. While without it Agentic Transformer still outperforms baselines, the performance degrades significantly compared with default configuration. In addition, without this completion token, the model does not improve with more trajectories at test time, indicating that completion token is important for the model to learn from hindsight experience.

In Table 5.4 rows (C), we observe that removing reward token 'r' has minimal negative effect. This is probably because the model can infer reward token by a simple subtraction from two consecutive returns-to-go tokens.

In Table 5.4 rows (D), we vary the desired return $\hat{R}_0$. Since default configuration uses 4 trajectories, the default target equals to total reward of last trajectory $\hat{R}_0 = \sum_{t=1}^{T} r_t^4$. We

vary $\hat{R}_0$ to be the total reward of other trajectories. We observe that changing this target decreases performance significantly, with the largest decrease happens when $\hat{R}_0$ equals the total reward of the first trajectory.

In Table 5.4 rows (E), instead of having ordered trajectories $s = (\tau^1, \tau^2 \ldots, \tau^n)$, we randomly shuffle all τ for each training batch. We observed significantly worse results, in particularly on ExoRL, this change decreases the performance to only slightly better than BC and DT.

In Table 5.4 rows (F), we evaluate different number of trajectories at test time, we observed a steady better result from using more trajectories at test time. We further observe that although results are better with more trajectories, even using one trajectory, Agentic Transformer still outperforms Transformer-based policies on both ExoRL and D4RL benchmarks. This suggests that Agentic Transformer not only learns more than just imitation learning, but also learns to improve upon its own experience.

In Table 5.4 rows (G), we consider applying loss on all trajectories rather than just last trajectory. We observe that it is detrimental to performance, and particularly reduces performance for when the number of test trajectories is small. This suggests that it is best to optimize model towards 'better' behaviors rather than imitating all behaviors.

5.5 Related Work

5.5.1 Transformer for Decision-Making

Prior works explored using Transformers in the context of supervised or offline RL. Among them, decision transformer (DT) [49] proposes to model trajectories as sequences and autoregressively predicts action conditioning on desired returns-to-go and past states and actions. Our model takes input as multiple trajectories and conditions on hindsight information for learning to improve. Chen et al. [49] found that DT does not benefit from longer context window and the results saturates at very short context length (*e.g.*, 3-5), possibly due to Markovian environments. Our Agentic Transformer (AT) models non-Markovian multiple episodes, it shows improved results with longer context length and benefits from Transformers architecture. Algorithm distillation (AD) [156] also conditions the model on multiple trajectories, the difference is that AD requires the data to be the experience over the life time of a RL algorithm, while our model can learn from data from any sources. Another key difference is our model conditions on hindsight information including hindsight desired returns-to-go and hindsight task completion tokens. We observe these algorithm modifications are crucial for superior performance. Transformer has been explored in learning general world model [177, 47, 319], learning from multiple games [246, 159], offline model-based learning [132, 177], meta learning [200, 292], vision-language navigation [52, 263], robot learning and behavior cloning from noisy demonstrations [262, 66], learning from multiple cameras [260], and language-conditioned imitation learning [105, 184, 270, 337]. Since our model is a general decision-making model, applying it to these interesting tasks is possible.

5.5.2 Learning from Hindsight Experience

Learning from hindsight experience was explored in goal conditioned RL [139, 10, 250]. Andrychowicz et al. [10] proposes hindsight experience replay (HER) to relabel rewards and transitions retroactively to learn from sparse reward. In relation to HER [10], our work is in the batch setting rather than online setting. We propose algorithm improvement to construct hindsight experience directly from offline experience. HER is designed for Q-learning algorithms [300, 203, 204] while AT use next token prediction to learn from hindsight information. Chain-of-hindsight [186] explores turning all (binary or multi-scale) feedback into a sentence that consists of chain of all feedback and show improve improvements in aligning language models with human preferences. In relation to it, our work can be seen as applying chain-of-hindsight in the context of automatic feedback. Our work steers model's behavior using the desired target return and reward function at each step as feedback instead of using human preference.

5.5.3 Supervised and Meta RL

Motivated by transforming conventional RL (*e.g.*, policy gradient [254, 257] and Q-learning [317, 203]) as a supervised learning problem, prior work explored various ways [278, 223, 177, 47, 48, 156]. Our work is closely related in that our model is similarly a return conditioned supervised learning. At test time, our model can self-improve based upon past experience to try to achieve target desired return. Using experience to improve model without changing weights is similar to few-shot or in-context learning in large language models [39]. Recent work Algorithm Distillation (AD) [156] demonstrates similar in-context behaviors in transformer model. AD is trained on the lifetime trajectories of a RL algorithm that can solves the task, posing a strong requirement of offline data, while in many important real world domains there exists only diverse, lower return data from multiple sources. In relation to AD, Agentic Transformer can be learned from sub-optimal data by turning the data into chain of hindsight experience. Leveraging online experience to improve model at test time is related to meta reinforcement learning (meta RL) [79, 308]. In meta RL the objective is to explicitly optimize for meta learning at test time, while Agentic Transformer does not, in contrast, the meta learning behavior emerges from training on chain of hindsight experience.

5.6 Conclusion

We propose Agentic Transformer (AT), a Transformer model with the ability of learning by directly combining information from multiple sub-optimal trials and being able to improve itself through multiple trials at test time. Motivated by prior works on hindsight experience replay and chain of hindsight, the key innovation behind Agentic Transformer is relabelling multiple trajectories to chain of hindsight experience that can be easily constructed from arbitrary offline data. On standard RL benchmarks, we showed AT outperforms both strong

algorithms designed explicitly for offline RL as well as state-of-the-art Transformer-based policies.

Limitations and Future Work.

- *Large diverse datasets.* While Agentic Transformer (AT) outperforms prior transformer-based policies and performs competitively with TD-learning in standard RL benchmarks. AT is a GPT model therefore all limitations of transformer model still apply to AT. For instance, training AT requires large memory because of self-attention quadratic complexity and long sequence length. At test time, rollouting our model is sequential thus slower than non-transformer models. That being said, we believe the advantages of AT outweigh its drawbacks. As we observed in NLP and CV, it is worth scaling transformer-based policies in both model size and dataset size. As the datasets used in this work are still small, future work could explore scaling up dataset and model and have more investigation into using large transformer models for RL.
- *Real world applications.* As we observed in the experiments, Agentic Transformer can learn by directly combining information from multiple sub-optimal trials. Because diverse sub-optimal data is ubiquitous in the real world and AT scales well with model size and dataset diversity, we believe an interesting future direction is applying AT for real-world applications.

Acknowledgements

We thank the members of the Berkeley Robot Learning Lab and Berkeley AI Lab for helpful discussions, as well as Google TPU Research Cloud for granting us access to TPUs. This project is supported in part by ONR under N00014-21-1-2769.

Chapter 6

Unsupervised Active Pretraining

6.1 Introduction

Reinforcement learning (RL) provides a general framework for solving challenging sequential decision-making problems. When combined with function approximation, it has achieved remarkable success in advancing the frontier of AI technologies. These landmarks include outperforming humans in computer games [204, 251, 306, 16] and solving complex robotic control tasks [10, 6]. Despite these successes, they have to train from scratch to maximize extrinsic reward for every encountered task. This is in sharp contrast with how intelligent creatures quickly adapt to new tasks by leveraging previously acquired behaviors. Unsupervised pre-training, a framework that trains models without expert supervision, has obtained promising results in computer vision [214, 113, 56] and natural language modeling [304, 74, 39]. The learned representation, when fine-tuned on the downstream tasks, can solve them efficiently in a few-shot manner. With the models and datasets growing, performance continues to improve predictably according to scaling laws.

Driven by the significance of massive unlabeled data, we consider an analogy setting of unsupervised pre-training in computer vision where labels are removed during training. The goal of pre-training is to have data efficient adaptation for some downstream task defined in the form of rewards. In RL with unsupervised pre-training, the agent is allowed to train for a long period without access to environment reward, and then only gets exposed to the reward during testing. We first test an array of existing methods for unsupervised pre-training to identity which gaps and challenges exist, we evaluate count-based bonus [28], which encourages the agent to visit novel states. We apply count-based bonus to DrQ [148] which is current state-of-the-art RL for training from pixels. We also evaluate ImageNet pre-trained representations. The results are shown in Figure 6.1. We can see that count-based bonus fails to outperform train DrQ from scratch. We hypothesize that the ineffectiveness stems from density modeling at the pixel level being difficult. ImageNet pre-training does not outperform training from scratch either, which has also been shown in previous research

in real world robotics [137]. We believe the reason is that neither of existing methods can provide enough diverse data. Count-based exploration faces the difficult of estimating high dimensional data density while ImageNet dataset is out-of-distribution for DMControl.

To address the issue of obtaining diverse data for RL with unsupervised pre-training, we propose to actively collect novel data by exploring unknown areas in the task-agnostic environment. The underlying intuition is that a general exploration strategy has to visit, with high probability, any state where the agent might be rewarded in a subsequent RL task. Concretely, our approach relies on the entropy maximization principle [133, 264]. Our hope is that by doing so, the learned behavior and representation can be trained on the whole environment while being as task agnostic as possible. Since entropy maximization in high dimensional state space is intractable as an oracle density model is not available, we resort to the particle-based entropy estimator [275, 27]. This estimator is nonparametric and asymptotically unbiased. The key idea is computing the average of the Euclidean distance of each particle to its nearest neighbors for a set of samples. We consider an abstract representation space in order to make the distance meaningful. To learn such a representation space, we adapt the idea of contrastive representation learning [56] to encode image observations to a lower dimensional space. Building upon this insight, we propose Unsupervised Active Pre-Training (APT) since the agent is encouraged to actively explore and leverage the experience to learn behavior.

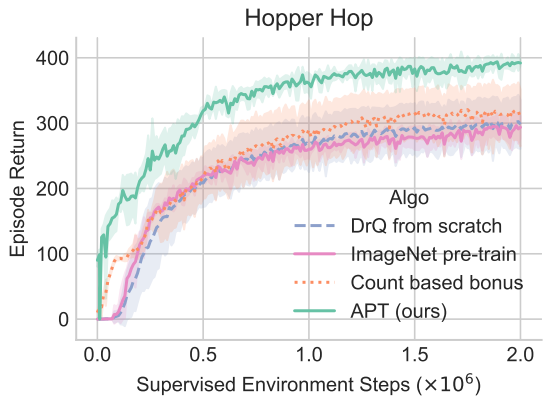


Figure 6.1 Comparison of state-of-the-art pixel-based RL with unsupervised pre-training. APT (ours) and count-based bonus (both based on DrQ [148]) are trained for a long unsupervised period (5M environment steps) without access to environment reward, and then gets exposure to the environment reward during testing. APT significantly outperform training DrQ from scratch, count-based bonus, and ImageNet pre-trained model.

Our approach can be applied to a wide-range of existing RL algorithms. In this paper we consider applying our approach to DrQ [148] which is a state-of-the-art visual RL algorithm. On the Atari 26 games subset, APT significantly improves DrQ's data-efficiency, achieving 54% relative improvement. On the full suite of Atari 57 games [204], APT significantly outperforms prior state-of-the-art, achieving a median human-normalized score 3× higher than the highest score achieved by prior unsupervised RL methods and DQN. On DeepMind control suite, APT beats DrQ and unsupervised RL in terms of asymptotic performance and data efficiency and solving tasks that are extremely difficult to train from scratch. The contributions of our paper can be summarized as: (i) We propose a new approach for unsupervised pre-training for visual RL based a nonparametric particle-based entropy maximization. (ii) We show that our pre-training method significantly improves data efficiency

of solving downstream tasks on DMControl and Atari suite.

6.2 Problem Setting

Reinforcement Learning (RL) An agent interacts with its uncertain environment over discrete timesteps and collects reward per action, modeled as a Markov Decision Process (MDP) [234], defined by $\langle \mathcal{S}, \mathcal{A}, T, \rho_0, r, \gamma \rangle$ where $\mathcal{S} \subseteq \mathbb{R}^{n_\mathcal{S}}$ is a set of $n_\mathcal{S}$-dimensional states, $\mathcal{A} \subseteq \mathbb{R}^{n_\mathcal{A}}$ is a set of $n_\mathcal{A}$-dimensional actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability distribution. $\rho_0 : \mathcal{S} \to [0, 1]$ is the distribution over initial states, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. At environment state $s \in \mathcal{S}$, the agent take actions $a \in \mathcal{A}$, in the (unknown) environment dynamics defined by the transition probability $T(s'|s, a)$, and the reward function yields a reward immediately following the action a_t performed in state s_t. We define the discounted return $G(s_t, a_t) = \sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l})$ as the discounted sum of future rewards collected by the agent. In value-based reinforcement learning, the agent learns an estimate of the expected discounted return, a.k.a, state-action value function $Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots} \left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l}) \right]$. A common way of deriving a new policy from a state-action value function is to act ϵ-greedily with respect to the action values (discrete) or to use policy gradient to maximize the value function (continuous).

Unsupervised Pre-Training RL In pretrained RL, the agent is trained in a reward-free MDP $\langle \mathcal{S}, \mathcal{S}_0, \mathcal{A}, T, \mathcal{G} \rangle$ for a long period followed by a short testing period with environment rewards $\mathbb{R}$ provided. The goal is to learn a pretrained agent that can quickly adapt to testing tasks defined by rewards to maximize the sum of expected future rewards in a zero-shot or few-shot manner. This is also known as the two phases learning in unsupervised pretraining RL [109]. The current state-of-the-art methods maximize the mutual information (I) between policy-conditioning variable (w) and the behavior induced by the policy in terms of state visitation (s).

$$\max I(s; w) = \max H(w) - H(w|s),$$

where w is sampled from a fixed distribution in practice as in DIAYN [81] and VISR [109]. The objective can then be simplified as $\max -H(w|s)$. Due to it being intractable to directly maximize this negative conditional entropy, prior work propose to maximize the variational lower bound of the negative conditional entropy instead [20]. The training then amounts to learning a posterior of task variable conditioning on states $q(w|s)$.

$$-H(w|s) \geq \mathbb{E}_{s,w} \left[\log q(w|s) \right].$$

Despite successful results in learning meaningful behaviors from reward-free interactions [e.g. 205, 101, 124, 81, 109], these methods suffer from insufficient exploration because they contain no explicit exploration.

Another category considers the alternative direction of maximizing the mutual information [43].

$$\max I(s; w) = \max H(s) - H(s|w).$$

This intractable quantity can be similarly lowered bound by a variational approximation [20].

$$I(s; w) \geq \mathbb{E}_{s,w} \left[q_\theta(s|w) \right] - \mathbb{E}_s \left[\log p(s) \right],$$

where $\mathbb{E}_s \left[\log p(s) \right]$ can then be approximated by a posterior of state given task variables $\mathbb{E}_s \left[\log p(s) \right] \approx \mathbb{E}_{s,w} \left[\log q(s|w) \right]$. Despite their successes, this category of methods do not explore sufficiently since the agent receives larger rewards for visiting known states than discovering new ones as theoretically and empirically evidenced by Campos et al. [43]. In addition, they have only been shown to work from explicit state-representations and it remains unclear how to modify to learning from pixels.

In the next section, we introduce a new nonparametric unsupervised pre-training method for RL which addresses these issues and outperforms prior state-of-the-arts on challenging visual-domain RL benchmarks.

6.3 Unsupervised Active Pre-Training for RL

We want to incentivize the agent with a reward r_t to maximize entropy in an abstract representation space. Prior work on maximizing entropy relies on estimating density of states which is challenging and non-trivial, instead, we take a two-step approach. First, we learn a mapping $f_\theta : R^{n_\mathcal{S}} \to R^{n_\mathcal{Z}}$ that maps state space to an abstract representation space first. Then, we propose a particle-based nonparametric approach to maximize the entropy by deploying state-of-the-art RL algorithms.

We introduce how to maximize entropy via particle-based approximation in Section 6.3.1, and describe how to learn representation from states in Section 6.3.2

6.3.1 Particle-Based Entropy Maximization

Our entropy maximization objective is built upon the nonparametric particle-based entropy estimator proposed by Singh et al. [275] and Beirlant [27] and has has been widely studied in statistics [134]. Its key idea is to measure the sparsity of the distribution by considering the distance between each sampled data point and its k nearest neighbors. Concretely, assuming we have number of n data points $\{z_i\}_{i=1}^n$ from some unknown distribution, the particle-based approximation can be written as

$$H_{\text{particle}}(z) = -\frac{1}{n} \sum_{i=1}^{n} \log \frac{k}{n \mathrm{v}_i^k} + b(k) \propto \sum_{i=1}^{n} \log \mathrm{v}_i^k, \tag{6.1}$$

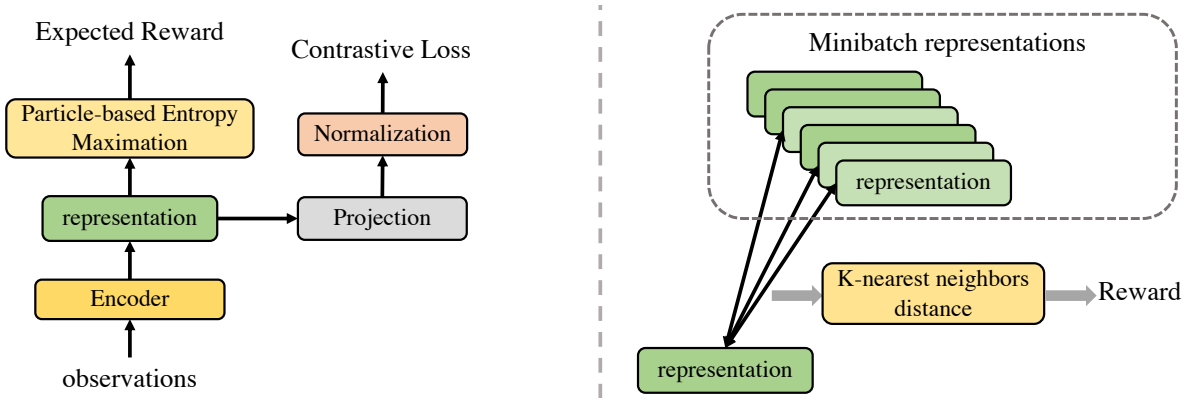


Figure 6.2 Diagram of the proposed method APT. On the left shows the objective of APT, which is to maximize the expected reward and minimize the contrastive loss. The contrastive loss learns an abstract representation from observations induced by the policy. We propose a particle-based entropy maximization based reward function such that we can deploy state-of-the-art RL methods to maximize entropy in an abstraction space of the induced by the policy. On the right shows the idea of our particle-based entropy, which measures the distance between each data point and its k nearest neighbors.

where $b(k)$ is a bias correction term that only depends on the hyperparameter k, and v_i^k is the volume of the hypersphere of radius $\|z_i - z_i^{(k)}\|$ between z_i and its k-th nearest neighbor $z_i^{(k)}$. $\|\cdot\|$ is the Euclidean distance.

$$\mathrm{v}_i^k = \frac{\|z_i - z_i^{(k)}\|^{n_{\mathcal{Z}}} \cdot \pi^{n_{\mathcal{Z}}/2}}{\Gamma\left(n_{\mathcal{Z}}/2 + 1\right)}, \tag{6.2}$$

where Γ is the gamma function. Intuitively, v_i^k reflects the sparsity around each particle and equation (6.1) is proportional to the average of the volumes around each particle.

By substituting equation (6.2) into equation (6.1), we can simplify the particle-based entropy estimation as a sum of the log of the distance between each particle and its k-th nearest neighbor.

$$H_{\text{particle}}(z) \propto \sum_{i=1}^{n} \log \|z_i - z_i^{(k)}\|^{n_{\mathcal{Z}}}. \tag{6.3}$$

Rather than using equation (6.3) as the entropy estimation, we find averaging the distance over all k nearest neighbors leads to a more robust and stable result, yielding our estimation of the entropy.

$$H_{\text{particle}}(z) \coloneqq \sum_{i=1}^{n} \log \left(c + \frac{1}{k} \sum_{z_i^{(j)} \in \mathrm{N}_k(z_i)} \|z_i - z_i^{(j)}\|^{n_{\mathcal{Z}}} \right), \tag{6.4}$$

where $\mathrm{N}_k(\cdot)$ denotes the k nearest neighbors around a particle, c is a constant for numerical stability (fixed to 1 in all our experiments).

We can view the particle-based entropy in equation (6.4) as an expected reward with the reward function being $r(z_i) = \log\left(c + \frac{1}{k}\sum_{z_i^{(j)} \in \mathrm{N}_k(z_i)} \|z_i - z_i^{(j)}\|^{nz}\right)$ for each particle z_i. This makes it possible to deploy RL algorithms to maximize entropy, concretely, for a batch of transitions $\{(s, a, s')\}$ sampled from the replay buffer. We consider the representation of each s' as a particle in the representation space and the reward function for each transition is given by

$$r(s, a, s') = \log\left(c + \frac{1}{k}\sum_{z^{(j)} \in \mathrm{N}_k(z=f_\theta(s))} \|f_\theta(s) - z^{(j)}\|^{nz}\right) \tag{6.5}$$

In order to keep the rewards on a consistent scale, we normalize the intrinsic reward by dividing it by a running estimate of the mean of the intrinsic reward. See Figure 6.2 for illustration of the formulation.

6.3.2 Learning Contrastive Representations

Our aforementioned entropy maximization is modular of the representation learning method we choose to use, the representation learning part can be swapped out for different methods if necessary. However, for entropy maximization to work, the representation needs to contain a compressed representation of the state. Recent work, CURL [153], ATC [280] and SPR [258], show contrastive learning (with data augmentation) helps learn meaningful representations in RL. We choose contrastive representation learning since it maximally distinguishes an observation s_{t_1} from alternative observations s_{t_2} according to certain distance metric in representation space, we hypothesize is helpful for learning meaningful representations for our nearest neighbors based entropy maximization. Our contrastive learning is based on the contrastive loss from SimCLR [56], chosen for its simplicity. We also use the same set of image augmentations as in DrQ [148] consisting of small random shifts and color jitter. Concretely, we randomly sample a batch of states (images) from the replay buffer $\{s_i\}_{i=1}^n$. For each state s_i, we apply random data augmentation and obtain two randomly augmented views of the same state, denoted as key $s_i^k = \mathrm{aug}(s_i)$ and query $s_i^v = \mathrm{aug}(s_i)$. The augmented observations are encoded into a small latent space using the encoder $z = f_\theta(\cdot)$ followed by a deterministic projection $h_\phi(\cdot)$ where a contrastive loss is applied. The goal of contrastive learning is to ensure that after the encoder and projection, s_i^k is relatively more close to s_i^v than any of the data points $\{s_j^k, s_j^v\}_{j=1, j\neq i}^n$.

$$\min_{\theta,\phi} -\frac{1}{2n}\sum_{i=1}^{n}\left[\log \frac{\exp(h_\phi(f_\theta(s_i^k))^T h_\phi(f_\theta(s_i^v)))}{\sum_{i=1}^n \mathbb{I}_{[j\neq i]}(\exp(h_\phi(f_\theta(s_i^k))^T h_\phi(f_\theta(s_j^k))) + \exp(h_\phi(f_\theta(s_i^k))^T h_\phi(f_\theta(s_j^v))))}\right].$$

Following DrQ, the representation encoder $f_\theta(\cdot)$ is implemented by the convolutional residual network followed by a fully-connected layer, a `LayerNorm` and a `Tanh` non-linearity. We

decrease the output dimension of the fully-connected layer after the convnet from 50 to 15. We find it helps to use spectral normalization [202] to normalize the weights and use ELU [64] as the non-linearity in between convolutional layers.

Table 6.1 positions our new approach with respect to existing ones. Figure 6.2 shows the resulting model. Training proceeds as in other algorithms maximizing extrinsic reward: by learning neural encoder f and computing intrinsic reward r and then trying to maximize this intrinsic return by training the policy. Algorithm 4 shows the pseudo-code of APT, we highlight the changes from DrQ to APT in color.

Algorithm 4: Training APT

Randomly Initialize f encoder
Randomly Initialize π and Q networks
for $e := 1, \infty$ **do**
  for $t := 1, T$ **do**
    Receive observation s_t from environment
    Take action $a_t \sim \pi(\cdot|s_t)$, receive observation s_{t+1} and ~~r_t~~ from environment
    $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, $ ~~r_t~~$, s'_t)$
    $\{(s_i, a_i, $ ~~r_i~~$, s'_i)\}_{i=1}^{N} \sim \mathcal{D}$ `// sample a mini batch`
    Train neural encoder f on mini batch `// representation learning`
    for *each* $i = 1..N$ **do**
      $a'_i \sim \pi(\cdot|s'_i)$
      $\hat{Q}_i = Q_{\theta'}(s'_i, a'_i)$
      Compute r_{APT} with equation (6.5) `// particle-based entropy reward`
      $y_i \leftarrow r_{\text{APT}} + \gamma \hat{Q}_i$
    end
    $loss_Q = \sum_i (Q(s_i, a_i) - y_i)^2$
    Gradient descent step on Q and π `// standard actor-critic`
  end
end

6.4 Related Work

State Space Entropy Maximization. Maximizing entropy of policy has been widely studied in RL, from inverse RL [341] to optimal control [294, 295, 245] and actor-critic [107]. State space entropy maximization has been recently used as an exploration method by estimating density of states and maximizing entropy [112]. In Hazan et al. [112] they present provably efficient exploration algorithms under certain conditions. VAE [144] based entropy estimation has been deployed in lower dimensional observation space [160]. However, due to the difficulty of estimating density in high dimensional space such as Atari games, such parametric exploration methods struggle to work in more challenging visual domains. In contrast, our work turns to particle based entropy maximization in a contrastive representation space. Maximizing particle-based entropy has been shown to improved data efficiency in

Table 6.1 Methods for pre-training RL in reward-free setting. Exploration: the method can explore efficiently. Visual: the method works well in visual RL. Off-policy: the method is compatible with off-policy RL optimization. * means only in state-based RL. c(s) is count-based bonus. $\psi(s,a)$: successor feature, $\phi(s)$: state representation.

Algorithm	Objective	Visual	Exploration	Off-policy	Pre-Trained model
MaxEnt [112]	$\max \mathrm{H}(s)$	✗	✓*	✗	$\pi(a\|s)$
CBB [28]	$\max \mathbb{E}_s[c(s)]$	✗	✓	✓	$\pi(a\|s)$
MEPOL [209]	$\max \mathrm{H}(s)$	✗	✓*	✗	$\pi(a\|s)$
VISR [109]	$\max -\mathrm{H}(z\|s)$	✓	✗	✓	$\psi(s,z), \phi(s)$
DIAYN [81]	$\max -\mathrm{H}(z\|s) + \mathrm{H}(a\|z,s)$	✗	✓*	✓	$\pi(a\|s,z)$
DADS [266]	$\max \mathrm{H}(s) - \mathrm{H}(s\|z)$	✗	✗	✓	$\pi(a\|s,z), q(s'\|s,z)$
EDL [43]	$\max \mathrm{H}(s) - \mathrm{H}(s\|z)$	✗	✓*	✓	$\pi(a\|s,z)$
APT	$\max \mathrm{H}(s)$	✓	✓	✓	$\pi(a\|s), Q(s,a)$

state-based RL as in MEPOL [209]. However, MEPOL's entropy estimation depends on importance sampling and the optimization based on on-policy RL algorithms, hindering further applications to challenging visual domains. MEPOL also assumes having access to the semantic information of the state, making it infeasible and not obvious how to modify it to work from pixels. In contrast, our method is compatible with deploying state-of-the-art off-policy RL and representation learning algorithms to maximize entropy. Nonparametric entropy maximization has been studied in goal conditioned RL [316]. Pitis et al. [231] proposes maximizing entropy of achieved goals and demonstrates significantly improved success rates in long horizon goal conditioned tasks. The work by Badia et al. [17] also considers k-nearest neighbor based count bonus to encourage exploration, yielding improved performance in Atari games. K-nearest neighbor based exploration is shown to improve exploration and data efficiency in model-based RL [285]. Concurrently, it has been shown to be an effective unsupervised pre-training objective for transferring learning in RL [45], their large scale experiments further demonstrate the effectiveness of unsupervised pre-training.

Data Efficient RL. To improve upon the sample efficiency of deep RL methods, various methods have been proposed: Kaiser et al. [140] introduce a model-based agent (SimPLe) and show that it compares favorably to standard RL algorithms when data is limited. Hessel et al. [116], Kielak [141], van Hasselt et al. [301] show combining existing RL algorithms (Rainbow) can boost data efficiency. Data augmentation has also been shown to be effective for improving data efficiency in vision-based RL [151, 148]. Temporal contrastive learning combined with model-based learning has been shown to boost data efficiency [258]. Combining contrastive loss with RL has been shown to improve data efficiency in CPC [114] despite only marginal gains. CURL [153] show substantial data-efficiency gains while follow-up results from Kostrikov et al. [148] suggest that most of the benefits come from its use of image augmentation. Contrastive loss has been shown to learn useful pretrained representations when training on expert demonstration [280], however in our work the agent has to explore the world itself and exploit collect experience.

Unsupervised Pre-Training RL. A number of recent works have sought to improve reinforcement learning via the addition of an unsupervised pretraining stage, in which the agent improves its representations prior to beginning learning on the target task. One common approach has been to allow the agent a period of fully-unsupervised interaction with the environment during which the agent is trained to learn a set of skills associated with different paths through the environment, as in DIAYN [81], Proto-RL [323], MUSIC [336], APS [178], and VISR [109]. Others have proposed to use self-supervised objectives to generate intrinsic rewards encouraging agents to visit new states, e.g., Pathak et al. [227] use the disagreement between an ensemble of latent-space dynamics models. However, our work is trained to maximize the entropy of the states induced by the policy. By visiting any state where the agent might be rewarded in a subsequent RL task, our work performs better or comparably well as other more complex and specialized state-of-the-art methods.

6.5 Results

We test APT in DeepMind Control Suite [DMControl; 289] and the Atari suite [29]. During the the long period of pre-training with environment rewards removed, we use DrQ to maximize the entropy maximization reward defined in equation (6.5). The pre-trained value function $Q(s, a)$ is fine-tuned to maximize task specific reward after being exposing to environment rewards during testing period. For our DeepMind control suite and Atari games experiments, we largely follow DrQ, except we perform two gradient steps per environment step instead of one. Our ablation studies confirm that these changes are not themselves responsible for our performance. Kornia [247] is used for efficient GPU-based data augmentations. Our model is implemented in Numpy [111] and PyTorch [224].

APT outperforms prior from scratch SOTA RL on DMControl. We evaluate the performance of different methods by computing the average success rate and episodic return at the end of training.

The agent is allowed a long unsupervised pre-training phase (5M steps), followed by a short test phase exposing to downstream reward, during which the pre-trained model is fine-tuned. We follow the evaluation setting of DrQ and test APT on a subset of DMControl suite, which includes training Walker, Cheetah, Hopper for various locomotion tasks. Models are pre-trained on Cheetah, Hopper, and Walker, and subsequently fine-tuned on respective downstream tasks. We additionally design more challenging sparse reward tasks where the robot is required to accomplish tasks guided only by sparse feedback signal. The reason we opted to design new sparse reward tasks is to have more diverse downstream tasks. As far as we know, there is only one Cartpole Swingup Sparse that is a CartPole based sparse reward task. Due to its 2D nature being quite limited, we eventually decided to design distinguishable downstream tasks based on a little bit more complex environment, e.g. Hopper Jump etc. The details of the tasks are included in the supplementary material.

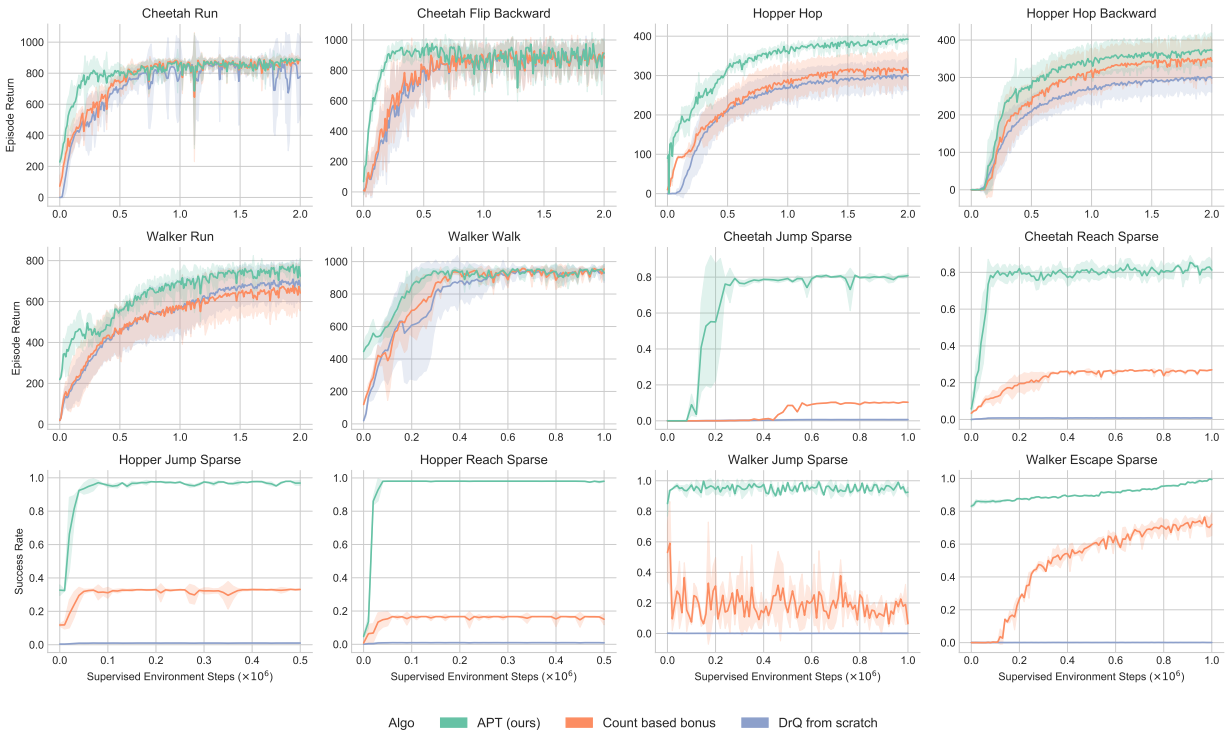


Figure 6.3 Results of different methods in environments from DMControl. All curves are the average of three runs with different seeds, and the shaded areas are standard errors of the mean.

The learning process of RL agents becomes highly inefficient in sparse supervision tasks when relying on standard exploration techniques. This issue can be alleviated by introducing intrinsic motivation, *i.e.*, denser reward signals that can be automatically computed, one approach that works well in high dimensional setting is count-based exploration [198, 219, 198].

The results are presented in Figure 6.3, APT significantly outperforms SOTA training from scratch (DrQ from scratch) and SOTA exploration method (count-based bonus) on every task. With only a few number of environment interactions, APT quickly adapt to downstream tasks and achieves higher return much more quicker than prior state-of-the-art RL algorithms. Notably, on the sparse reward tasks that are extremely difficult for training from scratch, APT yields significantly higher data efficiency and asymptotic performance.

APT outperforms from scratch SOTA RL in Atari. We test APT on the sample-efficient Atari setting [140, 301] which consists of the 26 easiest games in the Atari suite (as judged by above random performance for their algorithm).

We follow the evaluation setting in VISR, agents are allowed a long unsupervised training phase (250M steps) without access to rewards, followed by a short test phase with rewards.

The test phase contains 100K environment steps – equivalent to 400k frames, or just under two hours – compared to the typical standard of 500M environment steps, or roughly 39 days of experience. We normalize the episodic return with respect to expert human scores to account for different scales of scores in each game, as done in previous works. The human-normalized scores (HNS) of an agent on a game is calculated as $\frac{\text{agent score}-\text{random score}}{\text{human score}-\text{random score}}$ and aggregated across games by mean or median.

A full list of scores and aggregate metrics on the Atari 26 subset is presented in Table 6.2. The results on the full 57 Atari games suite is presented in supplementary material. For consistency with previous works, we report human and random scores from [116]. In the data-limited setting, APT achieves super-human performance on eight games and achieves scores higher than previous state-of-the-arts. In the full suite setting, APT achieves super-human performance on 15 games, compared to a maximum of 12 for any previous methods and achieves scores significantly higher than any previous methods.

Unsupervised pre-training on top of DrQ leads a significant increase in performance(a 54% increase in median score, a 73% increase in mean score, and 5 more games with human-level performance), surpassing DQN which trained on hundreds of millions of sampling steps.

Compared with SPR [258] which is a recent state-of-the-art model-based data-efficient algorithm, APT achieves comparable mean and median scores. The SPR is based on Rainbow which combines more advances than DrQ which is significantly simpler. While the representation of SPR is also learned by contrastive learning, it trains a model-based dynamic to predict its own latent state representations multiple steps into the future. This temporal representation learning, as illustrated in the SPR paper, contributes to its impressive results compared with standard contrastive representation learning. We believe that it is possible to combine temporal contrastive representation learning of SPR with the effective nonparametric entropy maximization of APT, which is an interesting future direction.

APT outperforms prior unsupervised RL. Despite there being many different proposed unsupervised RL methods, their successes are only demonstrated in simple state based environments. Prior works train the agent for a period of fully-unsupervised interaction with the environment, during which the agent is trained to learn a set of skills associated with different paths through the environment, as in DIAYN [81] and VIC [101], or to maximize the diversity of the states it encounters, as in MEPOL [209] and Hazan et al. [112]. Until recently, VISR [109] achieves improved results in Atari games using pixels as input based using a successor feature based approach. In order to compare with prior unsupervised RL methods, we choose DIAYN due to it being based on mutual information maximization and its reported high performance in state-based RL, and MEPOL due to it being based on entropy maximization. We implement them to take pixels as input in Atari games. Our implementation was checked against publicly available code and we made a best effort attempt to tune the algorithms in Atari games. We test two variants of DIAYN and MEPOL, using or not using contrastive representation learning as in APT. In order to ensure a fair comparison,

Table 6.2 Performance of different methods on the 26 Atari games considered by [140] after 100K environment steps. The results are recorded at the end of training and averaged over 10 random seeds for APT. APT outperforms prior methods on all aggregate metrics, and exceeds expert human performance on 7 out of 26 games while using a similar amount of experience. Prior work has reported different numbers for some of the baselines, particularly SimPLe and DQN. To be rigorous, we pick the best number for each game across the tables reported in van Hasselt et al. [301] and Kielak [141].

Game	Random	Human	SimPLe	DER	CURL	DrQ	SPR	VISR	APT (ours)
Alien	227.8	7127.7	616,9	739.9	558.2	771.2	801.5	364.4	**2614.8**
Amidar	5.8	1719.5	88.0	188.6	142.1	102.8	176.3	186.0	**211.5**
Assault	222.4	742.0	527.2	431.2	600.6	452.4	571.0	**12091.1**	891.5
Asterix	210.0	8503.3	1128.3	470.8	734.5	603.5	977.8	**6216.7**	185.5
Bank Heist	14.2	753.1	34.2	51.0	131.6	168.9	380.9	71.3	**416.7**
BattleZone	2360.0	37187.5	5184.4	10124.6	14870.0	12954.0	**16651.0**	7072.7	7065.1
Boxing	0.1	12.1	9.1	0.2	1.2	6.0	**35.8**	13.4	21.3
Breakout	1.7	30.5	16.4	1.9	4.9	16.1	17.1	**17.9**	10.9
ChopperCommand	811.0	7387.8	**1246.9**	861.8	1058.5	780.3	974.8	800.8	317.0
Crazy Climber	10780.5	23829.4	**62583.6**	16185.2	12146.5	20516.5	42923.6	49373.9	44128.0
Demon Attack	107805	35829.4	62583.6	16185.3	12146.5	20516.5	42923.6	**8994.9**	5071.8
Freeway	0.0	29.6	20.3	27.9	26.7	9.8	24.4	-12.1	**29.9**
Frostbite	65.2	4334.7	254.7	866.8	1181.3	331.1	**1821.5**	230.9	1796.1
Gopher	257.6	2412.5	771.0	349.5	669.3	636.3	715.2	498.6	**2590.4**
Hero	1027.0	30826.4	2656.6	6857.0	6279.3	3736.3	**7019.2**	663.5	6789.1
Jamesbond	29.0	302.8	125.3	301.6	471.0	236.0	365.4	**484.4**	356.1
Kangaroo	52.0	3035.0	323.1	779.3	872.5	940.6	**3276.4**	1761.9	412.0
Krull	1598.0	2665.5	**4539.9**	2851.5	4229.6	4018.1	2688.9	3142.5	2312.0
Kung Fu Master	258.5	22736.3	17257.2	14346.1	14307.8	9111.0	13192.7	16754.9	**17357.0**
Ms Pacman	307.3	6951.6	1480.0	1204.1	1465.5	960.5	1313.2	558.5	**2827.1**
Pong	-20.7	14.6	**12.8**	-19.3	-16.5	-8.5	-5.9	-26.2	-8.0
Private Eye	24.9	69571.3	58.3	97.8	218.4	-13.6	**124.0**	98.3	96.1
Qbert	163.9	13455.0	1288.8	1152.9	1042.4	854.4	669.1	666.3	**17671.2**
Road Runner	11.5	7845.0	5640.6	9600.0	5661.0	8895.1	**14220.5**	6146.7	4782.1
Seaquest	68.4	42054.7	683.3	354.1	384.5	301.2	583.1	706.6	**2116.7**
Up N Down	533.4	11693.2	3350.3	2877.4	2955.2	3180.8	**28138.5**	10037.6	8289.4
Mean HNS	0.000	1.000	44.3	28.5	38.1	35.7	**70.4**	64.31	69.55
Median HNS	0.000	1.000	14.4	16.1	17.5	26.8	41.5	12.36	**47.50**
# Superhuman	0	N/A	2	2	2	2	**7**	6	**7**

we test a variant of APT without contrastive representation learning.

The aggregated results are presented in Table 6.3, APT significantly outperforms prior state-based unsupervised RL algorithms DIAYN and MEPOL. Both baselines benefit from contrastive representation learning, but their scores are still significantly lower than APT's score, confirming that the effectiveness of the off-policy entropy maximization in APT. Compared with the state-of-the-art method in Atari VISR, APT achieves significantly higher median score despite having a lower mean score. From the scores breakdown presented in supplementary file, APT performs significantly better than VISR in hard exploration games, while VISR achieves higher scores in dense reward games. We attribute this to that maximizing state entropy leads to more exploratory behavior while successor features enables quicker adaptation for dense reward feedback. It is possible to combine VISR and APT to have the best of both worlds, which we leave as a future work.

Table 6.3 Evaluation in Atari games. The amount of RL interaction utilized is 100K. Mdn is the median of human-normalized scores, M is the mean and $> H$ is the number of games with human-level performance. CL denotes training representation encoder using contrastive learning and data augmentation. On each subset, we mark as bold the highest score.

	26 Game Subset			Full 57 Games		
Algorithm	Mdn	M	$> H$	Mdn	M	$> H$
CBB	1.23	21.94	3	–	–	–
MEPOL	0.34	17.94	2	–	–	–
DIAYN	1.34	25.39	2	2.95	23.90	6
CBB w/ CL	1.78	17.34	2	–	–	–
MEPOL w/ CL	1.05	21.78	3	–	–	–
DIAYN w/ CL	1.76	28.44	2	3.28	25.14	6
VISR	9.50	**128.07**	**7**	6.81	**102.31**	11
APT w/o CL	21.23	28.12	3	28.65	41.12	9
APT	**47.50**	69.55	**7**	**33.41**	47.73	**12**

Ablation study. We conduct several ablation studies to measure the contribution of each component in our method. We test two variants of APT that use the same number of gradient steps per environment step and use the same activation function as in DrQ. Another variant of APT is based on randomly selected neighbors to compute particle-based entropy. We also test a variant of APT that use a fixed randomly initialized encoder to study the impact of representation learning. Table 6.4 shows the performance of each variant of APT. Increasing gradient steps of updating value function from 1 to 2 and using `ELU` activation function yield higher scores. Using k-nearest neighbors is crucial to high scores, we believe the reason is randomly selected neighbors do not provide necessary incentive to explore. Using randomly initialized convolutional encoder downgrades performance significantly but still achieve higher score than DrQ, indicating our particle-based entropy maximization is robust and powerful.

Table 6.4 Scores on the 26 Atari games under consideration for variants of APT. Scores are averaged over 3 random seeds. All variants listed here use data augmentation.

Variant	Human-Normalized Score	
	median	mean
APT	**47.50**	**69.55**
APT w/o optim change	41.50	60.10
APT w/o arch change	45.71	67.82
APT w/ rand neighbor	20.80	24.97
APT w/ fixed encoder	33.24	41.08

Contrastive learning representation has been shown to have the "uniformity on the hypersphere"

property [312], this leads to the question that whether maximum entropy exploration in state space is important. To study this question, we have a variant of APT "Pos Reward APT" which receives a simple positive do not die signal but no particle-based entropy reward. We ran the experiments on MsPacman, we reduced the pretraining phase to 5M steps to reduce computation cost. The evaluation metrics are the number of ram states visited using [8] and the downstream zero shot performance on Atari game. APT visits nearly 27 times more unique ram states than "Pos Reward APT", showing that the entropy intrinsic reward is indispensable for exploration. In downstream task evaluation over 3 random seeds, "Pos Reward APT@0" achieves reward 363.7, "APT@0" achieves reward 687.1, showing that the "do not die" signal is insufficient for exploration or learning pretrained behaviors and representations.

Table 6.5 Scores on 5 Atari games under consideration for different variants of fine-tuning. Scores are averaged over 3 random seeds.

Mean Reward (3 seeds)	Alien	Freeway	Qbert	Private Eye	MsPacman
APT (pretrained head)	**2614.8**	**29.9**	17671.2	**96.1**	**2827.1**
APT (random head)	1755.0	15.2	**2138.3**	61.3	1724.9

We consider a variant of APT that re-initialize the head of pretrained actor-critic. We have run experiments in five different Atari games, as shown in Table 6.5, pretrained heads perform better than randomly initialized heads in 4 out of 5 games. The experiments demonstrate that finetuning from a pretrained actor-critic head accelerates learning. However, we believe that which one of the two is better depends on the alignment between downstream reward and intrinsic reward. It would be interesting to study how to better leverage downstream reward to finetune the pretrained model.

6.6 Discussion

Limitation: The fine-tuning strategy employed here (when combined with a value function) works best when the intrinsic and extrinsic rewards being of a similar scale. We believe the discrepancy between intrinsic reward scale and downstream reward scale possibly explain the suboptimal performance of APT in dense reward games. This is an interesting future direction to further improve APT, we hypothesize that reinitializing behaviors part (actor-critic heads) might be useful if the downstream reward scale is very different from pretraining reward scale. One of the principled ways could be adaptive normalization [302], it is an interesting future direction. One challenge of our method is the non-stationarity of the intrinsic reward, being non additive reward poses an interesting challenge for reinforcement learning methods. While our method outperforms training from scratch and prior works, we believe designing better optimization RL methods for maximizing our intrinsic reward can lead to more significant improvement.

Conclusion: A new unsupervised pre-training method for RL is introduced to address reward-free pre-training for visual RL, allowing the same task-agnostic pre-trained model to successfully tackle a broad set of RL tasks. Our major contribution is introducing a practical intrinsic reward derived from particle-based entropy maximization in abstract representation space. Empirical study on DMControl suite and Atari games show our method dramatically improves performance on tasks that are extremely difficult for training from scratch. Our method achieves the results of fully supervised canonical RL algorithms using a small fraction of total samples and outperforms data-efficient supervised RL methods.

For future work, there are a few ways in which our method can be improved. The long pre-training phase in our work is computationally intensive, since the exhaustive search and exploration is of high sample complexity. One way to remedy this is by combining our method with successful model-based RL and search approaches to reduce sample complexity. Furthermore, fine-tuning the whole pre-trained model can make it prone to catastrophic forgetting. As such, it is worth studying alternative methods to leverage the pre-trained models such as keeping the pretrained model unchanged and combine it with a randomly initialized model.

Acknowledgments

This research was supported by DARPA Data-Driven Discovery of Models (D3M) program. We would like to thank Misha Laskin, Olivia Watkins, Qiyang Li, Lerrel Pinto, Kimin Lee and other members at RLL and BAIR for insightful discussion and giving constructive comments. We would also like to thank anonymous reviewers for their helpful feedback for previous versions of our work.

6.7 General Implementation Details

For our Atari games and DeepMind Control Suite experiments, we largely follow DrQ [148], with the following exceptions. We use three layer convolutional neural network from [204] for policy network, and the Impala architecture for neural encoder with LSTM module removed. We use the ELU nonlinearity [64] in between layers of the encoder. The number of power iterations is 5 in spectral normalization.

The convolution neural network is followed by a full-connected layer normalized by `LayerNorm` [15] and a `tanh` nonlinearity applied to the output of fully-connected layer.

The data augmentation is a simple random shift which has been shown effective in visual domain RL in DrQ [148] and RAD [151]. Specifically, the images are padded each side by 4 pixels (by repeating boundary pixels) and then select a random 84×84 crop, yielding the original image. The replay buffer size is 100K. This procedure is repeated every time an image is sampled from the replay buffer. The learning rate of contrastive learning is 0.001,

the temperature is 0.1. The projection network is a two-layer MLP with hidden size of 128 and output size of 64. Batch size used in both RL and representation learning is 512. The pre-training phase consists of 5M environment steps on DMControl and 250M environment steps on Atari games. The evaluation is done for 125K environment steps at the end of training for 100K environment steps.

The implementation of APT can be found at https://github.com/rll-research/url_benchmark.

6.8 Atari Details

The corresponding hyperparameters used in Atari experiments are shown in Table 6.7 and Table 6.8.

6.9 DeepMind Control Suite Details

The action repeat hyperparameters are show in Table 6.6. The corresponding hyperparameters used in DMControl experiments are shown in Table 6.9 and Table 6.8.

Table 6.6 The action repeat hyper-parameter used for each environment.

Environment name	Action repeat
Cheetah	4
Walker	2
Hopper	2

6.10 Asymptotic Behavior of Intrinsic Reward

With the intrinsic reward defined in equation (6.5), we can derive that the intrinsic reward decreases to 0 as more of the state space is visited, which is a favorable property for pre-training.

Proposition 1. *Assume the MDP is episodic and its state space is finite $\mathcal{S} \subseteq \mathbb{R}^{n_S}$, the representation encoder $f_\theta : \mathbb{R}^{n_S} \to \mathbb{R}^{n_Z}$ is deterministic, and we have a buffer of observed states $(s_1, \ldots, s_T)$ with total sample size T. For an optimal policy that maximizes the intrinsic rewards defined as in equation (6.5) with $k \in \mathbb{N}$, we can derive the intrinsic reward is 0 in the limit of sample size T.*

$$\lim_{T \to \infty} r(s, a, s') = 0, \ \forall s \in \mathcal{S}.$$

Table 6.7 Hyper-parameters in the Atari suite experiments.

Parameter	Setting
Data augmentation	Random shifts and Intensity
Grey-scaling	True
Observation down-sampling	84×84
Frames stacked	4
Action repetitions	4
Reward clipping	$[-1, 1]$
Terminal on loss of life	True
Max frames per episode	108k
Update	Double Q
Dueling	True
Target network: update period	1
Discount factor	0.99
Minibatch size	32
RL optimizer	Adam
RL optimizer (pre-training): learning rate	0.0001
RL optimizer (fine-tuning): learning rate	0.001
RL optimizer: β_1	0.9
RL optimizer: β_2	0.999
RL optimizer: ϵ	0.00015
Max gradient norm	10
Training steps	100k
Evaluation steps	125k
Min replay size for sampling	1600
Memory size	Unbounded
Replay period every	1 step
Multi-step return length	10
Q network: channels	$32, 64, 64$
Q network: filter size	8×8, 4×4, 3×3
Q network: stride	$4, 2, 1$
Q network: hidden units	512
Non-linearity	`ReLU`
Exploration	ϵ-greedy
ϵ-decay	2500

Table 6.8 Hyper-parameters for Learning the Neural Encoder.

Parameter	Setting
Value of k	search in $\{3, 5, 10\}$
Temperature	0.1
Non-linearity	`ELU`
Network architecture	same as the Q network encoder (Atari) or the shared encoder (DMControl)
FC hidden size	1024
Output size	5

Table 6.9 Hyper-parameters in the DeepMind control suite experiments.

Parameter	Setting
Data augmentation	Random shifts
Frames stacked	3
Action repetitions	Table 6.6
Replay buffer capacity	100000
Random steps (fine-tuning phase)	1000
RL minibatch size	512
Discount γ	0.99
RL optimizer	Adam
RL learning rate	10^{-3}
Contrastive Learning Temperature	0.1
Shared encoder: channels	$32, 32, 32$
Shared encoder: filter size	3×3, 3×3, 3×3
Shared encoder: stride	$2, 2, 2, 1$
Actor update frequency	2
Actor log stddev bounds	$[-10, 2]$
Actor: hidden units	1024
Actor: layers	3
Critic Q-function: hidden units	1024
Critic target update frequency	2
Critic Q-function soft-update rate τ	0.01
Non-linearity	`ReLU`

While the assumption of finite state space may not be true for large complex environment like Atari games, the proposition 1 gives more insights on using this particular intrinsic reward for pre-training.

Proof. Since the intrinsic reward $r(s, a, s')$ defined in equation (6.5) depends on the k nearest neighbors in latent space and the encoder f_θ is deterministic, we just need to prove the visitation count $c(s)$ of s is larger than k as T goes infinity. We know the MDP is episodic, therefore as $T \to \infty$, all states communicate and $c(s) \to \infty$, thus we have $\lim_{T\to\infty} c(s) \geq k, \forall k \in \mathbb{N}, \forall s \in \mathcal{S}$. □

6.11 DeepMind Control Suite Sparse Environments

In addition to the existing tasks in DMControl, we tested different methods on three set customized sparse reward tasks: (1) *{HalfCheetah, Hopper, Walker} Jump Sparse*: the agent receives a positive reward 1 for jumping above a given height otherwise reward is 0. (2) *{HalfCheetah, Hopper, Walker} Reach Sparse*: the agent receives positive reward 1 for reaching a given target location otherwise reward is 0. (3) *Walker Turnover Sparse*: the initial position of Walker is turned upside down, and receives reward 1 for successfully turning itself over otherwise 0. In all the considered tasks, the episode ends when the goal is reached.

6.12 Scores on the full 57 Atari games

A comparison between APT and baselines on each individual Atari game is shown in Table 6.10. Prior work has reported different numbers for some of the baselines, particularly SimPLe and DQN. To be rigorous, we pick the best number for each game across the tables reported in van Hasselt et al. [301] and Kielak [141]. APT achieves super-human performance on 12 games, compared to a maximum of 11 for any previous methods and achieves scores significantly higher than any previous methods.

Table 6.10 Comparison of raw scores of each method on Atari games. On each subset, we mark as bold the highest score. For VISR, due to the lack of available source code, we made a best effort attempt to reproduce the algorithm.

Game	Random	Human	VISR	APT
Alien	227.8	7127.7	364.4	**2614.8**
Amidar	5.8	1719.5	186.0	**211.5**
Assault	222.4	742.0	**1209.1**	891.5
Asterix	210.0	8503.3	**6216.7**	185.5
Asteroids	7191	47388.7	**4443.3**	678.7
Atlantis	12850.0	29028.1	**140542.8**	40231.0
Bank Heist	14.2	753.1	71.3	**416.7**
Battle Zone	2360.0	37187.5	**7072.7**	7065.1
Beam Rider	363.9	16826.5	1741.9	**3487.2**
Berzerk	123.7	2630.4	490.0	**493.4**
Bowling	23.1	160.7	**21.2**	-56.5
Boxing	0.1	12.1	13.4	**21.3**
Breakout	1.7	30.5	**17.9**	10.9
Centipede	2090.9	12017.1	**7184.9**	6233.9
Chopper Command	811.0	7387.8	**800.8**	317.0
Crazy Climber	10780.5	23829.4	**49373.9**	44128.0
Defender	2874.5	18688.9	**15876.1**	5927.9
Demon Attack	107805	35829.4	**8994.9**	6871.8
Double Dunk	-18.6	-16.4	-22.6	**-17.2**
Enduro	0.0	860.5	-3.1	**-0.3**
Fishing Derby	-91.7	-38.7	-93.9	**-5.6**
Freeway	0.0	29.6	-12.1	**29.9**
Frostbite	65.2	4334.7	230.9	**1796.1**
Gopher	257.6	2412.5	498.6	**2190.4**
Gravitar	173.0	3351.4	328.1	**542.0**
Hero	1027.0	30826.4	663.5	**6789.1**
Ice Hockey	-11.2	0.9	**-18.1**	-30.1
Jamesbond	29.0	302.8	**484.4**	356.1
Kangaroo	52.0	3035.0	**1761.9**	412.0
Krull	1598.0	2665.5	**3142.5**	2312.0
Kung Fu Master	258.5	22736.3	16754.9	**17357.0**
Montezuma Revenge	0.0	4753.3	0.0	**0.2**
Ms Pacman	307.3	6951.6	558.5	**2527.1**
Name This Game	2292.3	8049.0	**2605.8**	1387.2
Phoenix	761.4	7242.6	**7162.2**	3874.2
Pitfall	-229.4	6463.7	-370.8	**-12.8**
Pong	-20.7	14.6	-26.2	**-8.0**
Private Eye	24.9	69571.3	**98.3**	96.1
Qbert	163.9	13455.0	666.3	**17671.2**
Riverraid	1338.5	17118.0	**5422.2**	4671.0
Road Runner	11.5	7845.0	**6146.7**	4782.1
Robotank	2.2	11.9	10.0	**13.7**
Seaquest	68.4	42054.7	706.6	**2116.7**
Skiing	-17098.1	-4336.9	**-19692.5**	-38434.1
Solaris	1236.3	12326.7	**1921.5**	841.8
Space Invaders	148.0	1668.7	**9741.0**	3687.2
Star Gunner	664.0	10250.0	**25827.5**	8717.0
Surround	-10.0	6.5	-15.5	**-2.5**
Tennis	-23.8	-8.3	0.7	**1.2**
Time Pilot	3568.0	5229.2	**4503.6**	2567.0
Tutankham	11.4	167.6	50.7	**124.6**
Up N Down	533.4	11693.2	**10037.6**	8289.4
Venture	0.0	1187.5	-1.7	**231.0**
Video Pinball	0.0	17667.9	**35120.3**	2817.1
Wizard Of Wor	563.5	4756.5	853.3	**1265.0**
Yars Revenge	3092.9	54576.9	**5543.5**	1871.5
Zaxxon	32.5	9173.3	897.5	**3231.0**
Mean Human-Norm'd	0.000	1.000	**68.42**	47.73
Median Human-Norm'd	0.000	1.000	9.41	**33.41**
#Superhuman	0	N/A	11	**12**

Chapter 7

Active Pretraining with Successor Features

7.1 Introduction

Deep unsupervised pretraining has achieved remarkable success in various frontier AI domains from natural language processing [74, 230, 39] to computer vision [113, 56]. The pre-trained models can quickly solve downstream tasks through few-shot fine-tuning [39, 57].

In reinforcement learning (RL), however, training from scratch to maximize extrinsic reward is still the dominant paradigm. Despite RL having made significant progress in playing video games [204, 251, 306, 16] and solving complex robotic control tasks [10, 6], RL algorithms have to be trained from scratch to maximize extrinsic return for every encountered task. This is in sharp contrast with how intelligent creatures quickly adapt to new tasks by leveraging previously acquired behaviors.

In order to bridge this gap, unsupervised pretraining RL has gained interest recently, from state-based [101, 81, 266, 209] to pixel-based RL [109, 179, 45]. In unsupervised pretraining RL, the agent is allowed to train for a long period without access to environment reward, and then got exposed to reward during testing. The goal of pretraining is to have data efficient adaptation for some downstream task defined in the form of rewards.

State-of-the-art unsupervised RL methods consider various ways of designing the so called intrinsic reward [26, 25, 101, 1], with the goal that maximizing this intrinsic return can encourage meaningful behavior in the absence of external rewards. There are two lines of work in this direction, we will discuss their advantages and limitations, and show that a novel combination yields an effective algorithm which brings the best of both world.

The first category is based on maximizing the mutual information between task variables ($p(z)$) and their behavior in terms of state visitation ($p(s)$) to encourage learning distinguishable task conditioned behaviors, which has been shown effective in state-based RL [101, 81] and

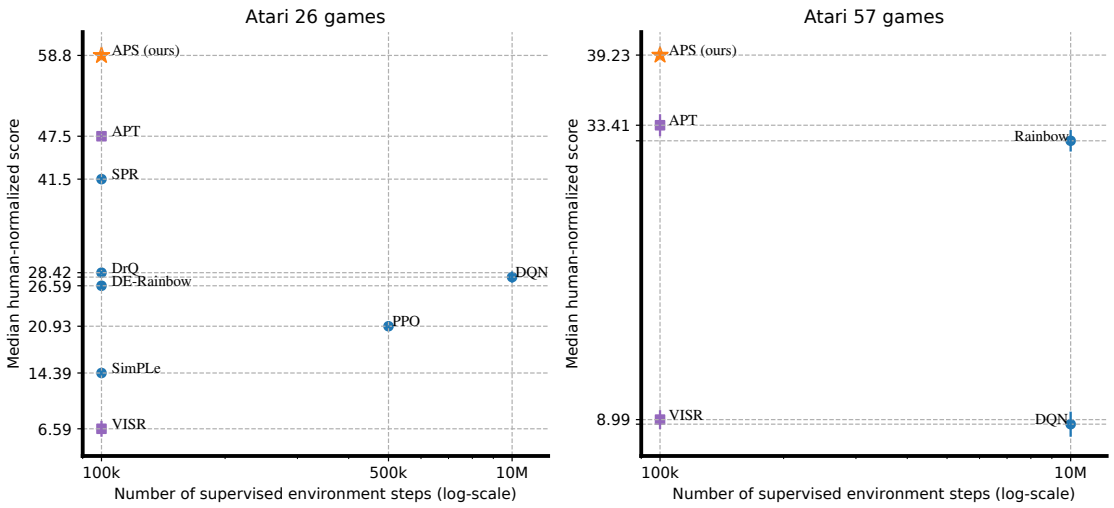


Figure 7.1 Median of human normalized score on the 26 Atari games considered by Kaiser et al. [140] (left) and the Atari 57 games considered in Mnih et al. [204](right). Fully supervised RL baselines are shown in circle. RL with unsupervised pretraining are shown in square. APS significantly outperforms all of the fully supervised and unsupervised pre-trained RL methods. Baselines: Rainbow [116], SimPLe [140], APT [179], Data-efficient Rainbow [141], DrQ [148], VISR [109], CURL [153], and SPR [258].

visual RL [109]. VISR proposed in Hansen et al. [109] is the prior state-of-the-art in this category. The objective of VISR is $\max I(s;z) = \max H(z) - H(s|z)$ where z is sampled from a fixed distribution. VISR proposes a successor features based variational approximation to maximize a variational lower bound of the intractable conditional entropy $-H(s|z)$. The advantage of VISR is that its successor features can quickly adapt to new tasks. Despite its effectiveness, the fundamental problem faced by VISR is lack of exploration.

Another category is based on maximizing the entropy of the states induced by the policy $\max H(s)$. Maximizing state entropy has been shown to work well in state-based domains [112, 209] and pixel-based domains [179]. It is also shown to be provably efficient under certain assumptions [112]. The prior state-of-the-art APT by Liu and Abbeel [179] show maximizing a particle-based entropy in a lower dimensional abstraction space can boost data efficiency and asymptotic performance. However, the issues with APT are that it is purely exploratory and task-agnostic and lacks of the notion of task variables, making it more difficult to adapt to new tasks compared with task-conditioning policies.

Our main contribution is to address the issues of APT and VISR by combining them together in a novel way. To do so, we consider the alternative direction of maximizing mutual information between states and task variables $I(s;z) = H(s) - H(s|z)$, the state entropy $H(s)$ encourages exploration while the conditional entropy encourages the agent to learn

task conditioned behaviors. Prior work that considered this objective had to either make the strong assumption that the distribution over states can be approximated with the stationary state-distribution of the policy [266] or rely on the challenging density modeling to derive a tractable lower bound [266, 43]. We show that the intractable conditional entropy, $-H(s|z)$ can be lower bounded and optimized by learning successor features. We use APT to maximize the state entropy $H(s)$ in an abstract representation space. Building upon this insight, we propose Active Pretraining with Successor Features (APS) since the agent is encouraged to actively explore and leverage the experience to learn behavior. By doing so, we experimentally find that they address the limitations of each other and significantly improve each other.

We evaluate our approach on the Atari benchmark [29] where we apply APS to DrQ [148] and test its performance after fine-tuning for 100K supervised environment steps. The results are shown in Figure 7.1. On the 26 Atari games considered by [140], our fine-tuning significantly boosts the data-efficiency of DrQ, achieving 106% relative improvement. On the full suite of Atari 57 games [204], fine-tuning APS pre-trained models significantly outperforms prior state-of-the-art, achieving human median score 3× higher than DQN trained with 10M supervised environment steps and outperforms previous methods combining unsupervised pretraining with task-specific finetuning.

7.2 Related Work

Our work falls under the category of mutual information maximization for unsupervised behavior learning. Unsupervised discovering of a set of task-agnostic behaviors by means of seeking to maximize an extrinsic reward has been explored in the the evolutionary computation community [162, 163]. This has long been studied as intrinsic motivation [25, 26], often with the goal of encouraging exploration [273, 220]. Entropy maximization in state space has been used to encourage exploration in state RL [112, 209, 259] and visual RL [179, 323]. Maximizing the mutual information between latent variable policies and their behavior in terms of state visitation has been used as an objective for discovering meaningful behaviors [123, 205, 101, 124, 81, 316]. Sharma et al. [266] consider a similar decomposition of mutual information, namely, $I(s; z) = H(s) - H(z|s)$, however, they assume $p(s|z) \approx p(s)$ to derive a different lower-bound of the marginal entropy. Different from Sharma et al. [266], Campos et al. [43] propose to first maximize $H(s)$ via maximum entropy estimation [112, 160] then learn behaviors, this method relies on a density model that provides an estimate of how many times an action has been taken in similar states. These methods are also only shown to work from explicit state-representations, and it is nonobvious how to modify them to work from pixels. The work by Badia et al. [17] also considers k-nearest neighbor based count bonus to encourage exploration, yielding improved performance on Atari games. This heuristically defined count-based bonus has been shown to be an effective unsupervised pretraining objective for RL [45]. Machado et al. [198] show the norm of learned successor features can be used to incentivize exploration as a reward bonus. Our work differs in that

Table 7.1 Comparing methods for pretraining RL in no reward setting. VISR [109], APT [179], MEPOL [209], DIYAN [81], DADS [266], EDL [43]. Exploration: the model can explore efficiently. Off-policy: the model is off-policy RL. Visual: the method works well in visual RL, e.g., Atari games. Task: the model conditions on latent task variables z. * means only in state-based RL.

Algorithm	Objective	Exploration	Visual	Task	Off-policy	Pre-Trained Model
APT	$\max \mathrm{H}(s)$	✓	✓	✗	✓	$\pi(a\|s), Q(s,a)$
VISR	$\max \mathrm{H}(z) - \mathrm{H}(z\|s)$	✗	✓	✓	✓	$\psi(s,z)$, $\phi(s)$
MEPOL	$\max \mathrm{H}(s)$	✓*	✗	✗	✗	$\pi(a\|s)$
DIAYN	$\max -\mathrm{H}(z\|s) + \mathrm{H}(a\|z,s)$	✗	✗	✓	✗	$\pi(a\|s,z)$
EDL	$\max \mathrm{H}(s) - \mathrm{H}(s\|z)$	✓*	✗	✓	✓	$\pi(a\|s,z), q(s'\|s,z)$
DADS	$\max \mathrm{H}(s) - \mathrm{H}(s\|z)$	✓	✗	✓	✗	$\pi(a\|s,z), q(s'\|s,z)$
APS	$\max \mathrm{H}(s) - \mathrm{H}(s\|z)$	✓	✓	✓	✓	$\psi(s,z)$, $\phi(s)$

$\psi(s)$: successor features, $\phi(s)$: state feature (*i.e.*, the representation of states).

we jointly maximize the entropy and learn successor features.

7.3 Preliminaries

Reinforcement learning considers the problem of finding an optimal policy for an agent that interacts with an uncertain environment and collects reward per action. The goal of the agent is to maximize its cumulative reward.

Formally, this problem can be viewed as a Markov decision process (MDP) defined by $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho_0, r, \gamma)$ where $\mathcal{S} \subseteq \mathbb{R}^{n_s}$ is a set of n_s-dimensional states, $\mathcal{A} \subseteq \mathbb{R}^{n_a}$ is a set of n_a-dimensional actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is the state transition probability distribution. $\rho_0 : \mathcal{S} \to [0,1]$ is the distribution over initial states, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0,1)$ is the discount factor. At environment states $s \in \mathcal{S}$, the agent take actions $a \in \mathcal{A}$, in the (unknown) environment dynamics defined by the transition probability $T(s'|s,a)$, and the reward function yields a reward immediately following the action a_t performed in state s_t. We define the discounted return $G(s_t, a_t) = \sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l})$ as the discounted sum of future rewards collected by the agent. In value-based reinforcement learning, the agent learns learns an estimate of the expected discounted return, a.k.a, state-action value function.

$$Q^\pi(s,a) = \mathbb{E}_{\substack{s_t=s \\ a_t=a}} \left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l}, s_{t+l+1}) \right].$$

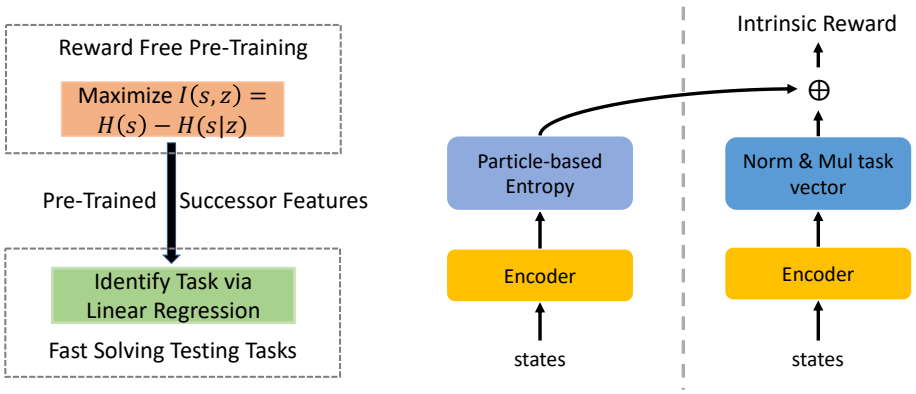


Figure 7.2 Diagram of the proposed method APS. On the left shows the concept of APS, during reward-free pretraining phase, reinforcement learning is deployed to maximize the mutual information between the states induced by policy and the task variables. During testing, the pre-trained state features can identify the downstream task by solving a linear regression problem , the pre-trained task conditioning successor features can then quickly adapt to and solve the task. On the right shows the components of APS. APS consists of maximizing state entropy in an abstract representation space (exploration, $\max H(s)$) and leveraging explored data to learn task conditioning behaviors (exploitation, $\max -H(s|z)$).

7.3.1 Successor Features

Successor features [72, 149, 22, 23] assume that there exist features $\phi(s, a, s') \in R^d$ such that the reward function which specifies a task of interest can be written as

$$r(s, a, s') = \phi(s, a, s')^T w,$$

where $w \in R^d$ is the task vector that specify how desirable each feature component is.

The key observation is that the state-action value function can be decomposed as a linear form [22]

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}_{\substack{s_t=s \\ a_t=a}} \left[\sum_{i=t}^{\infty} \gamma^{i-t} \phi(s_{i+1}, a_{i+1}, s'_{i+1}) \right]^T w \\ &\equiv \psi^\pi(s, a)^T w, \end{aligned}$$

where $\psi^\pi(s, a)$ are the successor features of π. Intuitively, $\psi(s, a)$ can be seen as a generalization of $Q(s, a)$ to multidimensional value function with reward $\phi(s, a, s')$

7.4 Method

We first introduce two techniques which our method builds upon in Section 7.4.1 and Section 7.4.2 and discuss their limitations. We provide preliminary evidence of the limitations

in Section 7.4.3. Then we propose APS in Section 7.4.4 to address their limitations.

7.4.1 Variational Intrinsic Successor Features (VISR)

The variational intrinsic successor features (VISR) maximizes the mutual information(I) between some policy-conditioning variable (z) and the states induced by the conditioned policy,

$$I(z; s) = H(z) - H(z|s),$$

where it is common to assume z is drawn from a fixed distribution for the purposes of training stability [81, 109].

This simplifies the objective to minimizing the conditional entropy of the conditioning variable, where s is sampled uniformly over the trajectories induced by π_θ.

$$\sum_{z,s} p(s, z) \log p(z|s) = \mathbb{E}_{s,z}[\log p(z|s)],$$

A variational lower bound is proposed to address the intractable objective,

$$J_{\text{VISR}}(\theta) = -\mathbb{E}_{s,z}[\log q(z|s)],$$

where $q(z|s)$ is a variational approximation. REINFORCE algorithm is used to learn the policy parameters by treating $\log q(z|s)$ as intrinsic reward. The variational parameters can be optimized by maximizing log likelihood of samples.

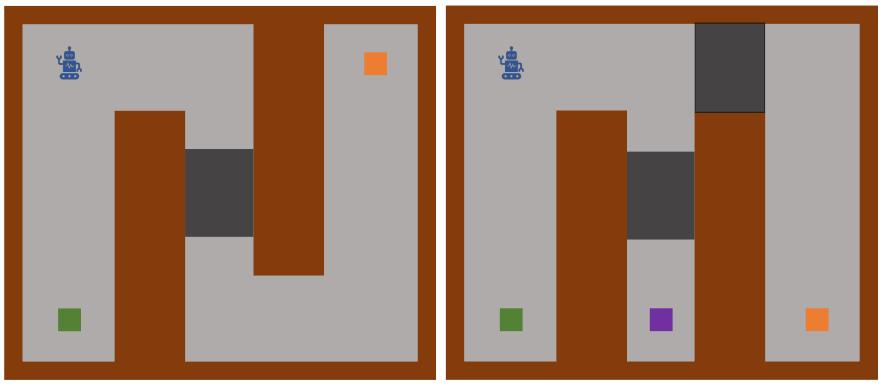


Figure 7.3 The passageway gridworld environments used in our experiments. On the left, the agent needs to fetch the key first by navigating to the green location to unlock the closed passageway (shown in black). Similarly, on the right, there is an additional key-passageway pair. The agent must fetch the key (shown in purple) to unlock the upper right passageway.

The key observation made by Hansen et al. [109] is restricting conditioning vectors z to correspond to task-vectors w of the successor features formulation $z \equiv w$. To satisfy this requirement, one can restrict the task vectors w and features $\phi(s)$ to be unit length and paremeterizing the discriminator $q(z|s)$ as the Von Mises-Fisher distribution with a scale parameter of 1.

$$r_{\text{VISR}}(s, a, s') = \log q(w|s) = \phi(s)^T w.$$

VISR has the rapid task inference mechanism provided by successor features with the ability of mutual information maximization methods to learn many diverse behaviors in an unsupervised way. Despite its effectiveness as demonstrated in Hansen et al. [109], VISR suffers from inefficient exploration. This issue limits the further applications of VISR in challenging tasks.

7.4.2 Unsupervised Active Pretraining (APT)

The objective of unsupervised active pretraining (APT) is to maximize the entropy of the states induced by the policy, which is computed in a lower dimensional abstract representation space.

$$J_{\text{APT}}(\theta) = H(h) = \sum_s p(h) \log p(h), \quad h = f(s),$$

where $f : R^{n_s} \to R^{n_h}$ is a mapping that maps observations s to lower dimensional representations h. In their work, Liu and Abbeel [179] learns the encoder by contrastive representation learning.

With the learned representation, APT shows the entropy of h can be approximated by a particle-based entropy estimation [275, 27], which is based on the distance between each particle $h_i = f(s_i)$ and its k-th nearest neighbor $h_i^\star$.

$$H(h) \approx H_{\text{APT}}(h) \propto \sum_{i=1}^{n} \log \|h_i - h_i^\star\|_{n_z}^{n_z}.$$

This estimator is asymptotically unbiased and consistent $\lim_{n\to\infty} H_{\text{APT}}(s) = H(s)$.

It helps stabilizing training and improving convergence in practice to average over all k nearest neighbors [179].

$$\hat{H}_{\text{APT}}(h) = \sum_{i=1}^{n} \log \left(1 + \frac{1}{k} \sum_{h_i^j \in \text{N}_k(h_i)} \|h_i - h_i^j\|_{n_h}^{n_h} \right),$$

where $\text{N}_k(\cdot)$ denotes the k nearest neighbors.

For a batch of transitions $\{(s, a, s')\}$ sampled from the replay buffer, each abstract representation $f(s')$ is treated as a particle and we associate each transition with a intrinsic reward given by

$$r_{\text{APT}}(s, a, s') = \log\left(1 + \frac{1}{k}\sum_{h^{(j)}\in \text{N}_k(h)} \|h - h^{(j)}\|_{n_z}^{n_z}\right)$$
$$\text{where } h = f_\theta(s'). \tag{7.1}$$

While APT achieves prior state-of-the-art performance in DeepMind control suite and Atari games, it does not conditions on latent variables (e.g. task) to capture important task information during pretraining, making it inefficient to quickly identity downstream task when exposed to task specific reward function.

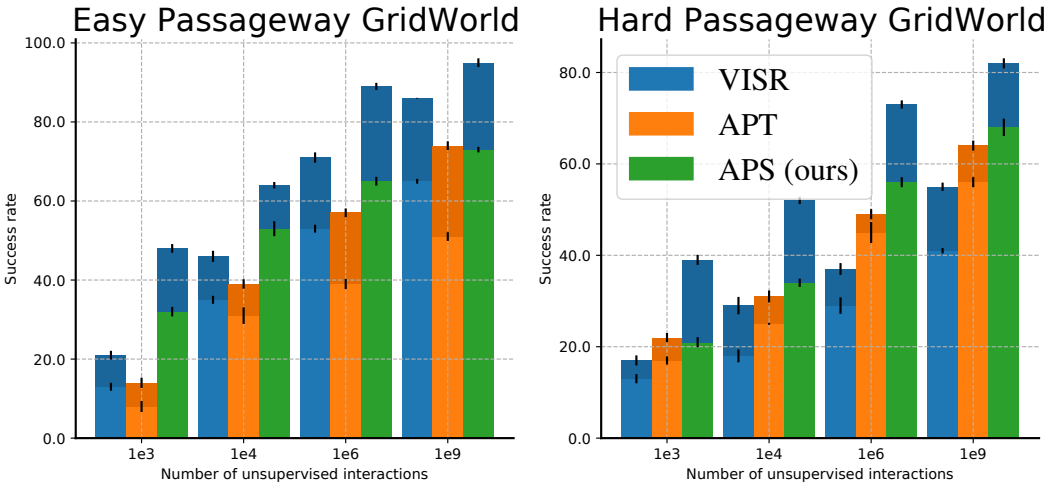


Figure 7.4 Performance of different methods on the gridworld environments in Figure 7.3. The results are recorded during testing phase after pretraining for a number of unsupervised interactions. The success rate are aggregated over 10 random seeds. The bottom of each bar is the zero-shot testing performance while the top is the fine-tuned performance.

7.4.3 Empirical Evidence of the Limitations of Existing Models

In this section we present two multi-step grid-world environments to illustrate the drawbacks of APT and VISR, and highlight the importance of both exploration and task inference. The environments, implemented with the pycolab game engine [279], are depicted shown in Figure 7.3, and are fully observable to the agent. At each episode, the agent starts from a randomly initialized location in the top left corner, with the task of navigating to the target location shown in orange. To do so, the agent has to first pick up a key(green, purple

area) that opens the closed passageway. The easy task shown in left of Figure 7.3 has one key and one corresponding passageway while the hard task has two key-passageway pairs. We evaluate the agent in terms of success rates. During evaluation, the agent receives an intermediate reward 1 for picking up key and 10 for completing the task. The hierarchical task presents a challenge to algorithms using only exploration bonus or successor features, as the exploratory policy is unlikely to quickly adapt to the task specific reward and the successor features is likely to never explore the space sufficiently.

Figure 7.4 shows the success rate of each method. APT performs worse than VISR at the easy level, possibly because successor features can quickly adapt to the downstream reward. On the other hand, APT significantly outperforms VISR at the hard level which requires a exploratory policy. Despite the simplicity, these two gridworld environments already highlight the weakness of each method. This observation confirms that existing formulations either fail due to inefficient exploration or slow adaption, and motivates our study of alternative methods for behavior discovery.

7.4.4 Active Pre-training with Successor Features

To address the issues of APT and VISR, we consider maximizing the mutual information between task variable (z) drawn from a fixed distribution and the states induced by the conditioned policy.

$$I(z; s) = H(s) - H(s|z).$$

The intuition is that the $H(s)$ encourages the agent to explore novel states while $H(s|z)$ encourages the agent to leverage the collected data to capture task information.

Directly optimizing $H(s)$ is intractable because the true distribution of state is unknown, as introduced in Section 7.4.2, APT [179] is an effective approach for maximizing $H(s)$ in high-dimensional state space. We use APT to perform entropy maximization.

$$r_{\text{APS}}^{\text{exploration}}(s, a, s') = \log\left(1 + \frac{1}{k}\sum_{h^{(j)}\in \text{N}_k(h)} \|h - h^{(j)}\|_{n_h}^{n_h}\right)$$
$$\text{where } h = f_\theta(s'). \tag{7.2}$$

As introduced in Section 7.4.1, VISR [109] is a variational based approach for maximizing $-H(z|s)$. However, maximizing $-H(z|s)$ is not directly applicable to our case where the goal is to maximize $-H(s|z)$.

This intractable conditional entropy can be lower-bounded by a variational approximation,

$$F = -H(s|z) \geq \mathbb{E}_{s,z}\left[\log q(s|z)\right].$$

Algorithm 5: Training APS

Randomly Initialize ϕ network // $L2$ normalized output
Randomly Initialize ψ network // $dim(output) = \#A \times dim(W)$
for $e := 1, \infty$ **do**
  sample w from $L2$ normalized $\mathcal{N}(0, I(dim(W)))$ // uniform ball
  $Q(\cdot, a|w) \leftarrow \psi(\cdot, a, w)^\top w, \forall a \in A$
  for $t := 1, T$ **do**
    Receive observation s_t from environment
    $a_t \leftarrow \epsilon$-greedy policy based on $Q(s_t, \cdot|w)$
    Take action a_t, receive observation s_{t+1} and reward ~~r_t~~ from environment
    $a' = \arg\max_a \psi(s_{t+1}, a, w)^\top w$
    Compute $r_{\text{APS}}(s_t, a, s_{t+1})$ with Equation (7.7) // intrinsic reward to max $I(s; z)$
    $y = r_{\text{APS}}(s_t, a, s_{t+1}) + \gamma\psi(s_{t+1}, a', w)^\top w$
    $loss_\psi = (\psi(s_t, a_t, w)^\top w - y_i)^2$
    $loss_\phi = -\phi(s_t)^\top w$ // minimize Von-Mises NLL
    Gradient descent step on ψ and ϕ // minibatch in practice
  end
end

This is because of the variational lower bound [20].

$$
\begin{aligned}
F &= \sum_{s,z} p(s,z) \log p(s|z) \\
&= \sum_{s,z} p(s,z) \log p(s|z) + \sum_{s,z} p(s,z) \log q(s|z) \\
&\quad - \sum_{s,z} p(s,z) \log q(s|z) \\
&= \sum_{s,z} p(s,z) \log q(s|z) + \sum_{z} p(z) D_{\text{KL}}(p(\cdot|z)||q(\cdot|z)) \\
&\geq \sum_{s,z} p(s,z) \log q(s|z) \\
&= \mathbb{E}_{s,z}[\log q(s|z)]
\end{aligned} \tag{7.3}
$$

Our key observation is that Von Mises-Fisher distribution is symmetric to its parametrization, by restricting $z \equiv w$ similarly to VISR, the reward can be written as

$$
r_{\text{APS}}^{\text{exploitation}}(s, a, s') = \log q(s|w) = \phi(s)^T w. \tag{7.4}
$$

We find it helps training by sharing the weights between encoders f and ϕ. The encoder is trained by minimizing the negative log likelihood of Von-Mises distribution $q(s|w)$ over the

data.

$$L = -\mathbb{E}_{s,w}\left[\log q(s|w)\right] = -\mathbb{E}_{s,w}\left[\phi(s_t)^\top w\right]. \tag{7.5}$$

Note that the proposed method is independent from the choices of representation learning for f, *e.g.*, one can use an inverse dynamic model [225, 40] to learn the neural encoder, which we leave for future work.

Put Equation (7.2) and Equation (7.4) together, our intrinsic reward can be written as

$$\begin{aligned} & r_{\text{APS}}(s,a,s') \\ & \quad = r_{\text{APS}}^{\text{exploitation}}(s,a,s') + r_{\text{APS}}^{\text{exploration}}(s,a,s') \qquad (7.6) \\ & \quad = \phi(s)^T w + \log\left(1 + \frac{1}{k}\sum_{h^{(j)}\in \text{N}_k(h)} \|h - h^{(j)}\|_{n_h}^{n_h}\right) \\ & \quad \text{where } h = \phi(s'), \qquad (7.7) \end{aligned}$$

The output layer of ϕ is L2 normalized, task vector w is randomly sampled from a uniform distribution over the unit circle.

Table 7.1 positions our new approach with respect to existing ones. Figure 7.2 shows the resulting model. Training proceeds as in other algorithms maximizing mutual information: by randomly sampling a task vector w and then trying to infer the state produced by the conditioned policy from the task vector. Algorithm 5 shows the pseudo-code of APS, we highlight the changes from VISR to APS in color.

7.4.5 Implementation Details

We largely follow Hansen et al. [109] for hyperparameters used in our Atari experiments, with the following three exceptions. We use the four layers convolutional network from Kostrikov et al. [148] as the encoder ϕ and f. We change the output dimension of the encoder from 50 to 5 in order to match the dimension used in VISR. While VISR incorporated LSTM [121] we excluded it for simplicity and accelerating research. We use ELU nonlinearities [64] in between convolutional layers. We do not use the distributed training setup in Hansen et al. [109], after every roll-out of 10 steps, the experiences are added to a replay buffer. This replay buffer is used to calculate all of the losses and change the weights of the network. The task vector w is also resampled every 10 steps. We use n-step Q-learning with $n = 10$.

Following Hansen et al. [109], we condition successor features on task vector, making $\psi(s,a,w)$ a UVFA [34, 250]. We use the Adam optimizer [143] with an learning rate 0.0001. We use discount factor $\gamma = .99$. Standard batch size of 32. ψ is coupled with a target network [204], with an update period of 100 updates.

Table 7.2 Performance of different methods on the 26 Atari games considered by [140] after 100K environment steps. The results are recorded at the end of training and averaged over 5 random seeds for APS. APS outperforms prior methods on all aggregate metrics, and exceeds expert human performance on 8 out of 26 games while using a similar amount of experience.

Game	Random	Human	SimPLe	DER	CURL	DrQ	SPR	VISR	APT	APS (ours)
Alien	227.8	7127.7	616,9	739.9	558.2	771.2	801.5	364.4	**2614.8**	934.9
Amidar	5.8	1719.5	88.0	188.6	142.1	102.8	176.3	186.0	**211.5**	178.4
Assault	222.4	742.0	527.2	431.2	600.6	452.4	571.0	**12091.1**	891.5	413.3
Asterix	210.0	8503.3	1128.3	470.8	734.5	603.5	977.8	**6216.7**	185.5	1159.7
Bank Heist	14.2	753.1	34.2	51.0	131.6	168.9	380.9	71.3	**416.7**	262.7
BattleZone	2360.0	37187.5	5184.4	10124.6	14870.0	12954.0	16651.0	7072.7	7065.1	**26920.1**
Boxing	0.1	12.1	9.1	0.2	1.2	6.0	35.8	13.4	21.3	**36.3**
Breakout	1.7	30.5	16.4	1.9	4.9	16.1	17.1	17.9	10.9	**19.1**
ChopperCommand	811.0	7387.8	1246.9	861.8	1058.5	780.3	974.8	800.8	317.0	**2517.0**
Crazy Climber	10780.5	23829.4	62583.6	16185.2	12146.5	20516.5	42923.6	49373.9	44128.0	**67328.1**
Demon Attack	107805	35829.4	62583.6	16185.3	12146.5	20516.5	42923.6	**8994.9**	5071.8	7989.0
Freeway	0.0	29.6	20.3	27.9	26.7	9.8	24.4	-12.1	**29.9**	27.1
Frostbite	65.2	4334.7	254.7	866.8	1181.3	331.1	**1821.5**	230.9	1796.1	496.5
Gopher	257.6	2412.5	771.0	349.5	669.3	636.3	715.2	498.6	**2590.4**	2386.5
Hero	1027.0	30826.4	2656.6	6857.0	6279.3	3736.3	7019.2	663.5	6789.1	**12189.3**
Jamesbond	29.0	302.8	125.3	301.6	471.0	236.0	365.4	484.4	356.1	**622.3**
Kangaroo	52.0	3035.0	323.1	779.3	872.5	940.6	3276.4	1761.9	412.0	**5280.1**
Krull	1598.0	2665.5	**4539.9**	2851.5	4229.6	4018.1	2688.9	3142.5	2312.0	4496.0
Kung Fu Master	258.5	22736.3	17257.2	14346.1	14307.8	9111.0	13192.7	16754.9	17357.0	**22412.0**
Ms Pacman	307.3	6951.6	1480.0	1204.1	1465.5	960.5	1313.2	558.5	**2827.1**	2092.3
Pong	-20.7	14.6	**12.8**	-19.3	-16.5	-8.5	-5.9	-26.2	-8.0	12.5
Private Eye	24.9	69571.3	58.3	97.8	218.4	-13.6	**124.0**	98.3	96.1	117.9
Qbert	163.9	13455.0	1288.8	1152.9	1042.4	854.4	669.1	666.3	17671.2	**19271.4**
Road Runner	11.5	7845.0	5640.6	9600.0	5661.0	8895.1	**14220.5**	6146.7	4782.1	5919.0
Seaquest	68.4	42054.7	683.3	354.1	384.5	301.2	583.1	706.6	2116.7	**4209.7**
Up N Down	533.4	11693.2	3350.3	2877.4	2955.2	3180.8	**28138.5**	10037.6	8289.4	4911.9
Mean Human-Norm'd	0.000	1.000	44.3	28.5	38.1	35.7	70.4	64.31	69.55	**99.04**
Median Human-Norm'd	0.000	1.000	14.4	16.1	17.5	26.8	41.5	12.36	47.50	**58.80**
# Superhuman	0	N/A	2	2	2	2	7	6	7	**8**

7.5 Results

We test APS on the full suite of 57 Atari games [29] and the sample-efficient Atari setting [140, 301] which consists of the 26 easiest games in the Atari suite (as judged by above random performance for their algorithm).

We follow the evaluation setting in VISR [109] and APT [179], agents are allowed a long unsupervised training phase (250M steps) without access to rewards, followed by a short test phase with rewards. The test phase contains 100K environment steps – equivalent to 400k frames, or just under two hours – compared to the typical standard of 500M environment steps, or roughly 39 days of experience. We normalize the episodic return with respect to expert human scores to account for different scales of scores in each game, as done in previous works. The human-normalized performance of an agent on a game is calculated as $\frac{\text{agent score} - \text{random score}}{\text{human score} - \text{random score}}$ and aggregated across games by mean or median.

When testing the pre-trained successor features ψ, we need to find task vector w from the rewards. To do so, we rollout 10 episodes (or 40K steps, whichever comes first) with the trained APS, each conditioned on a task vector chosen uniformly on a 5-dimensional sphere. From these initial episodes, we combine the data across all episodes and solve the linear regression problem. Then we fine-tune the pre-trained model for 60K steps with the inferred task vector, and the average returns are compared.

A full list of scores and aggregate metrics on the Atari 26 subset is presented in Table 7.2. The results on the full 57 Atari games suite is presented in Supplementary Material. For consistency with previous works, we report human and random scores from [116].

In the data-limited setting, APS achieves super-human performance on eight games and achieves scores higher than previous state-of-the-arts.

In the full suite setting, APS achieves super-human performance on 15 games, compared to a maximum of 12 for any previous methods and achieves scores significantly higher than any previous methods.

7.6 Analysis

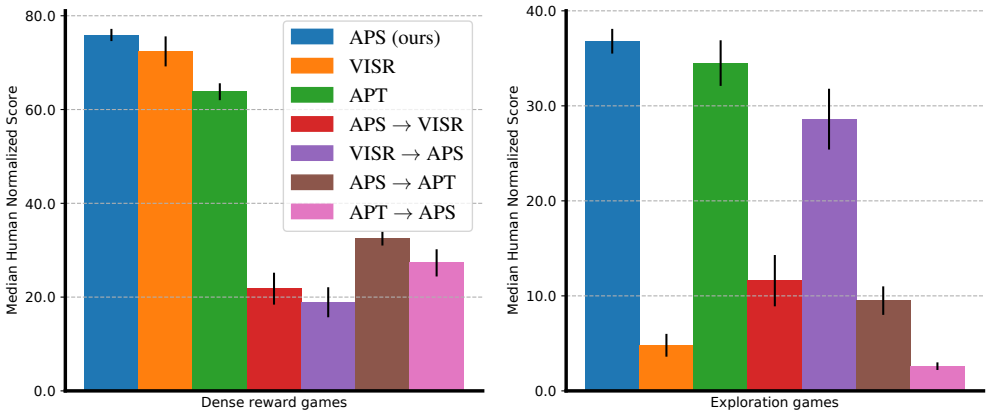


Figure 7.5 Scores of different methods and their variants on the 26 Atari games considered by Kaiser et al. [140]. $X \to Y$ denotes training method Y using the data collected by method X at the same time.

Contribution of Exploration and Exploitation In order to measure the contributions of components in our method, we aim to answer the following two questions in this ablation study. Compared with APT ($\max H(s)$), is the improvement solely coming from better fast task solving induced by $\max -H(s|z)$ and the exploration is the same? Compared with VISR ($\max H(z) - H(z|s)$), is the improvement solely coming from better exploration due to $\max H(s) - H(s|z)$ and the task solving ability is the same?

We separate Atari 26 subset into two categories. Dense reward games in which exploration is simple and exploration games which require exploration. In addition to train the model as before, we simultaneously train another model using the same data, *e.g.* APS $\rightarrow$ APT denotes when training APS simultaneously training APT using the same data as APS. As shown in Figure 7.5, on dense reward games, APS $\rightarrow$ APT performs better than APT $\rightarrow$ APS. On exploration games, APS $\rightarrow$ APT significantly outperforms APT $\rightarrow$ APS. Similarly APS $\rightarrow$ VISR performs better than the other way around. Together, the results indicate that entropy maximization and variational successor features improves each other in a nontrivial way, and both are important to the performance gain of APS.

Table 7.3 Scores on the 26 Atari games for variants of APS, VISR, and APT. Scores of considered variants are averaged over 3 random seeds.

Variant	Human-Normalized Score	
	mean	median
APS	**99.04**	**58.80**
APS w/o fine-tune	81.41	49.18
VISR (controlled, w/ fine-tune)	68.95	31.87
APT (controlled, w/o fine-tune)	58.23	19.85
APS w/o shared encoder	87.59	51.45

Fine-Tuning Helps Improve Performance We remove fine-tuning from APS that is we evaluate its zero-shot performance, the same as in Hansen et al. [109]. We also employ APS's fine-tuning scheme to VISR, namely 250M (without access to rewards, followed by a short task identify phase (40K steps) and a fine-tune phase (60K steps). The results shown in Table 7.3 demonstrate that fine-tuning can boost performance. APS w/o fine-tune outperforms all controlled baselines, including VISR w/ fine-tune.

Shared Encoder Can Boost Data-Efficiency We investigate the effect of using ϕ as the encoder f. To do so, we consider a variant of APS that learns the encoder f as in APT by contrastive representation learning. The performance of this variant is denoted as APS w/o shared encoder shown in Table 7.3. Sharing encoder can boost data efficiency, we attribute the effectiveness to ϕ better captures the relevant information which is helpful for computing intrinsic reward. We leave the investigation of using other representation learning methods as future work.

7.7 Conclusion

In this paper, we propose a new unsupervised pretraining method for RL. It addresses the limitations of prior mutual information maximization-based and entropy maximization-based methods and combines the best of both worlds. Empirically, APS achieves state-of-the-art

performance on the Atari benchmark, demonstrating significant improvements over prior work.

Our work demonstrates the benefit of leveraging state entropy maximization data for task-conditioned skill discovery. We are excited about the improved performance by decomposing mutual information as $H(s) - H(s|z)$ and optimizing them by particle-based entropy and variational successor features. In the future, it is worth studying how to combine approaches designed for maximizing the alternative direction $-H(z|s)$ with the particle-based entropy maximization.

7.8 Acknowledgment

We thank members of Berkeley Artificial Intelligence Research (BAIR) Lab for many insightful discussions. This work was supported by Berkeley Deep Drive, the Open Philanthropy Project, and Intel.

7.9 Experiment Details

The corresponding hyperparameters used in Atari experiments are shown in Table 7.4 and Table 7.5. We follow Kostrikov et al. [148] to use data augmentation techniques that consist of a simple random shift which has been shown effective in visual domain RL. Specifically, the images are padded each side by 4 pixels (by repeating boundary pixels) and then select a random 84×84 crop, yielding the original image. This procedure is repeated every time an image is sampled from the replay buffer.

We also use the same generalized policy improvement(GPI) [22, 24] as in VISR with the number of polices 10. GPI is also used in VISR to ensure a fair comparison. Per common practice, we average performance of our agent over 5 random seeds. The evaluation is done for 125K environment steps at the end of training for 100K environment steps. We follow Hansen et al. [109] to use one MLP for each dimension of the successor feature.

The ablated variant APS w/o shared encoder follows APT [179] but the output dimension of the neural encoder f is decreased to 5 in order to match the default APS. The projection network in contrastive learning is a two-layer MLP with hidden size of 128 and output size of 64. We also use the same temperature and other hyperparamters as APT for the ablation study.

7.10 Scores Breakdown on 57 Atari games

A comparison between APS and baselines on each individual game of the 57 Atari game suite is shown in Table 7.6. APS achieves super-human performance on 15 games, compared to a

Table 7.4 Hyper-parameters for RL.

Parameter	Setting
Terminal on loss of life	True
Reward clipping (fine-tuning phase)	$[-1, 1]$
Data augmentation	Random shifts and Intensity
Grey-scaling	True
Observation down-sampling	84×84
Frames stacked	4
Action repetitions	4
Max frames per episode	108k
Update	Double Q
Target network: update period	100
Discount factor	0.99
Minibatch size	32
ψ, ϕ optimizer	Adam
ψ, ϕ optimizer (pre-training phase): learning rate	0.0001
ψ, ϕ optimizer (fine-tuning phase): learning rate	0.001
ψ, ϕ optimizer: β_1	0.9
ψ, ϕ optimizer: β_2	0.999
ψ, ϕ optimizer: ϵ	0.00015
Max gradient norm	10
Training steps (fine-tuning phase)	60K
Task identity steps (fine-tuning phase)	40K
Training steps (pre-training phase)	5M
Evaluation steps	125K
Min replay size for sampling	1600
Memory size	Unbounded
Replay period every	1 step
Multi-step return length	10
ψ network: channels	$32, 64, 64$
ψ network: filter size	8×8, 4×4, 3×3
ψ network: stride	$4, 2, 1$
ψ network: hidden units	512
ψ Non-linearity	`ReLU`
Exploration	ϵ-greedy
ϵ-decay	2500

Table 7.5 Hyper-parameters for Learning ϕ.

Parameter	Setting
Value of k	search in $\{3, 5, 10\}$
ϕ network: channels	$32, 64, 64$
ϕ network: filter size	8×8, 4×4, 3×3
ϕ network: stride	$4, 2, 1$
ϕ network: hidden units	512
ϕ network Non-linearity	`ELU`
FC hidden size	1024
Output size	5

maximum of 12 for any previous methods and achieves scores significantly higher than any previous methods.

Table 7.6 Comparison of raw scores of each method on Atari games. Results are averaged over five random seeds. @N represents the amount of RL interaction utilized at fine-tuning phase.

Game	Random	Human	VISR	APT	APS (ours)
Alien	227.8	7127.7	364.4	**2614.8**	934.9
Amidar	5.8	1719.5	186.0	**211.5**	188.4
Assault	222.4	742.0	**1209.1**	891.5	413.3
Asterix	210.0	8503.3	**6216.7**	185.5	1159.5
Asteroids	7191	47388.7	**4443.3**	678.7	1519.7
Atlantis	12850.0	29028.1	140542.8	40231.0	**18920.0**
Bank Heist	14.2	753.1	71.3	**416.7**	262.7
Battle Zone	2360.0	37187.5	7072.7	7065.1	**26920.1**
Beam Rider	363.9	16826.5	1741.9	3487.2	**4981.2**
Berzerk	123.7	2630.4	490.0	**493.4**	387.4
Bowling	23.1	160.7	21.2	-56.5	**56.5**
Boxing	0.1	12.1	13.4	21.3	**36.3**
Breakout	1.7	30.5	17.9	10.9	**19.1**
Centipede	2090.9	12017.1	**7184.9**	6233.9	3915.7
Chopper Command	811.0	7387.8	800.8	317.0	**2517.0**
Crazy Climber	10780.5	23829.4	49373.9	44128.0	**67328.1**
Defender	2874.5	18688.9	15876.1	5927.9	**19921.5**
Demon Attack	107805	35829.4	**8994.9**	6871.8	7989.0
Double Dunk	-18.6	-16.4	-22.6	-17.2	**-8.0**
Enduro	0.0	860.5	-3.1	-0.3	**216.8**
Fishing Derby	-91.7	-38.7	-93.9	-5.6	**-2.1**
Freeway	0.0	29.6	-12.1	**29.9**	27.1
Frostbite	65.2	4334.7	230.9	**1796.1**	496.1
Gopher	257.6	2412.5	498.6	**2190.4**	2590.4
Gravitar	173.0	3351.4	328.1	**542.0**	487.0
Hero	1027.0	30826.4	663.5	6789.1	**12189.3**
Ice Hockey	-11.2	0.9	-18.1	-30.1	**-11.3**
Jamesbond	29.0	302.8	484.4	356.1	**622.3**
Kangaroo	52.0	3035.0	1761.9	412.0	**5280.1**
Krull	1598.0	2665.5	3142.5	2312.0	**4496.0**
Kung Fu Master	258.5	22736.3	16754.9	17357.0	13112.1
Montezuma Revenge	0.0	4753.3	0.0	147.0	**211.0**
Ms Pacman	307.3	6951.6	558.5	**2527.1**	2092.3
Name This Game	2292.3	8049.0	2605.8	1387.2	**6898.8**
Phoenix	761.4	7242.6	**7162.2**	3874.2	6871.8
Pitfall	-229.4	6463.7	-370.8	-12.8	**-6.2**
Pong	-20.7	14.6	-26.2	-8.0	**12.5**
Private Eye	24.9	69571.3	98.3	96.1	**117.9**
Qbert	163.9	13455.0	666.3	17671.2	**19271.4**
Riverraid	1338.5	17118.0	5422.2	4671.0	**10521.3**
Road Runner	11.5	7845.0	**6146.7**	4782.1	5919.0
Robotank	2.2	11.9	10.0	**13.7**	12.6
Seaquest	68.4	42054.7	706.6	2116.7	**4209.7**
Skiing	-17098.1	-4336.9	**-19692.5**	-38434.1	-9102.1
Solaris	1236.3	12326.7	**1921.5**	841.8	1095.4
Space Invaders	148.0	1668.7	**9741.0**	3687.2	3693.8
Star Gunner	664.0	10250.0	25827.5	8717.0	**42970.0**
Surround	-10.0	6.5	-15.5	**-2.5**	-5.8
Tennis	-23.8	-8.3	0.7	1.2	**8.7**
Time Pilot	3568.0	5229.2	4503.6	2567.0	**4586.5**
Tutankham	11.4	167.6	50.7	**124.6**	45.6
Up N Down	533.4	11693.2	**10037.6**	8289.4	4911.9
Venture	0.0	1187.5	-1.7	**231.0**	136.0
Video Pinball	0.0	17667.9	**35120.3**	2817.1	154414.1
Wizard Of Wor	563.5	4756.5	853.3	1265.0	**1732.1**
Yars Revenge	3092.9	54576.9	5543.5	1871.5	**6539.5**
Zaxxon	32.5	9173.3	897.5	3231.0	**5819.2**
Mean Human-Norm'd	0.000	1.000	68.42	47.78	**103.04**
Median Human-Norm'd	0.000	1.000	9.41	33.41	**39.23**
#Superhuman	0	N/A	11	12	**15**

Chapter 8

Contrastive Intrinsic Control

8.1 Introduction

Deep Reinforcement Learning (RL) is a powerful approach toward solving complex control tasks in the presence of extrinsic rewards. Successful applications include playing video games from pixels [204], mastering the game of Go [271, 272], robotic locomotion [255, 256, 229] and dexterous manipulation [242, 215, 216] policies. While effective, the above advances produced agents that are unable to generalize to new downstream tasks beyond the one they were trained to solve. Humans and animals on the other hand are able to acquire skills with minimal supervision and apply them to solve a variety of downstream tasks. In this work, we seek to train agents that acquire skills without supervision with generalization capabilities by efficiently adapting these skills to downstream tasks.

Over the last few years, unsupervised RL has emerged as a promising framework for developing RL agents that can generalize to new tasks. In the unsupervised RL setting, agents are first pre-trained with self-supervised intrinsic rewards and then finetuned to downstream tasks with extrinsic rewards. Unsupervised RL algorithms broadly fall into three categories - knowledge-based, data-based, and competence-based methods[1]. Knowledge-based methods maximize the error or uncertainty of a predictive model [225, 226, 41]. Data-based methods maximize the entropy of the agent's visitation [180, 324]. Competence-based methods learn skills that generate diverse behaviors [82, 102]. This work falls into the latter category of competence-based exploration methods.

Unlike knowledge-based and data-based algorithms, competence-based algorithms simultaneously address both the exploration challenge as well as distilling the generated experience in the form of reusable skills. This makes them particularly appealing, since the resulting skill-based policies (or skills themselves) can be finetuned to efficiently solve downstream tasks. While there are many self-supervised objectives that can be utilized, our work falls

[1]These categories for exploration algorithms were introduced by Srinivas and Abbeel [277] and inspired by Oudeyer et al. [221].

$$I(\tau; z) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z)$$

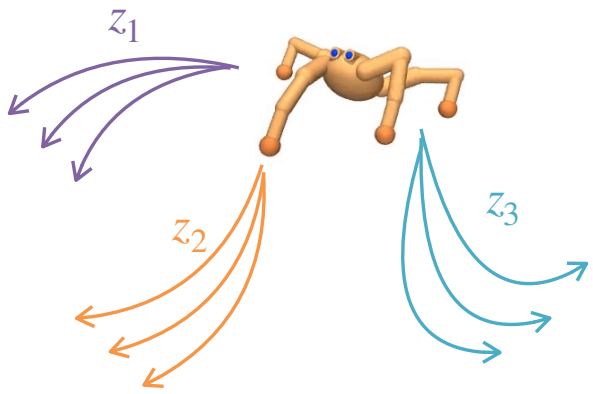


Figure 8.1 This work deals with unsupervised skill discovery through mutual information maximization. We introduce Contrastive Intrinsic Control (CIC) – a new unsupervised RL algorithm that explores and adapts more efficiently than prior methods.

into a family of methods that learns skills by maximizing the mutual information between visited states and latent skill vectors. Many earlier works have investigated optimizing such objectives [82, 102, 150, 265]. However, competence-based methods have been empirically challenging to train and have under-performed when compared to knowledge and data-based methods [155].

In this work, we take a closer look at the challenges of pre-training agents with competence-based algorithms. We introduce Contrastive Intrinsic Control (CIC) – an exploration algorithm that uses a new estimator for the mutual information objective. CIC combines particle estimation for state entropy [274, 180] and noise contrastive estimation [106] for the conditional entropy which enables it to both generate diverse behaviors *(exploration)* and discriminate high-dimensional continuous skills *(exploitation)*. To the best of our knowledge, CIC is the first exploration algorithm to utilize noise contrastive estimation to discriminate between state transitions and latent skill vectors. Empirically, we show that CIC adapts to downstream tasks more efficiently than prior exploration approaches on the Unsupervised Reinforcement Learning Benchmark (URLB). CIC achieves 79% higher returns on downstream tasks than prior competence-based algorithms and 18% higher returns than the next-best exploration algorithm overall.

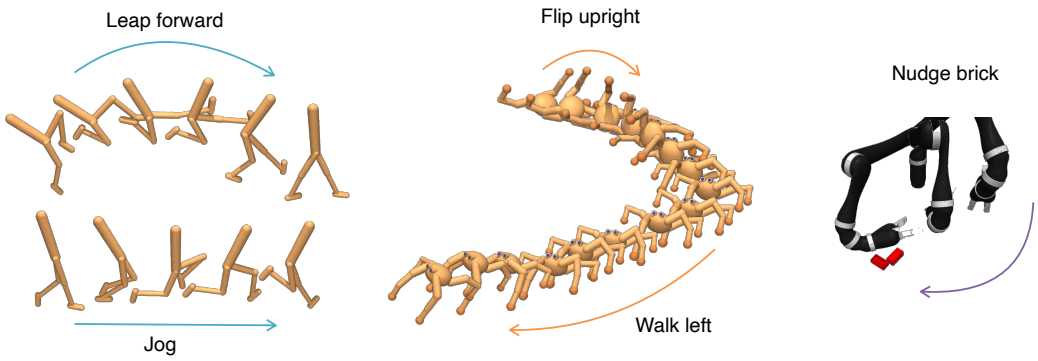


Figure 8.2 Qualitative visualizations of unsupervised skills discovered in Walker, Quadruped, and Jaco arm environments. The Walker learns to balance and move, the Quadruped learns to flip upright and walk, and the 6 DOF robotic arm learns how to move without locking. Unlike prior competence-based methods for continuous control which evaluate on OpenAI Gym (e.g. Eysenbach et al. [82]), which reset the environment when the agent loses balance, CIC is able to learn skills in fixed episode length environments which are much harder to explore (see Appendix 8.18).

8.2 Background and Notation

Markov Decision Process: We operate under the assumption that our system is described by a Markov Decision Process (MDP) [284]. An MDP consiss of the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ which has states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition dynamics $p(s'|s,a) \sim \mathcal{P}$, a reward function r, and a discount factor γ. In an MDP, at each timestep t, an agent observes the current state s, selects an action from a policy $a \sim \pi(\cdot|s)$, and then observes the reward and next state once it acts in the environment: $r, s' \sim \text{env.step}(a)$. Note that usually r refers to an extrinsic reward. However, in this work we will first be pre-training an agent with intrinsic rewards r^{int} and finetuning on extrinsic rewards r^{ext}.

For convenience we also introduce the variable $\tau(s)$ which refers to any function of the states s. For instance τ can be a single state, a pair of states, or a sequence depending on the algorithm. Our method uses $\tau = (s, s')$ to encourage diverse state transitions while other methods have different specifications for τ. Importantly, τ does not denote a state-action trajectory, but is rather shorthand for any function of the states encountered by the agent. In addition to the standard MDP notation, we will also be learning skills $z \in \mathcal{Z}$ and our policy will be skill-conditioned $a \sim \pi(\cdot|s,z)$.

Unsupervised Skill Discovery through Mutual Information Maximization: Most competence-based approaches to exploration maximize the mutual information between states and skills. Our work and a large body of prior research [82, 265, 102, 2, 161, 181] aims to

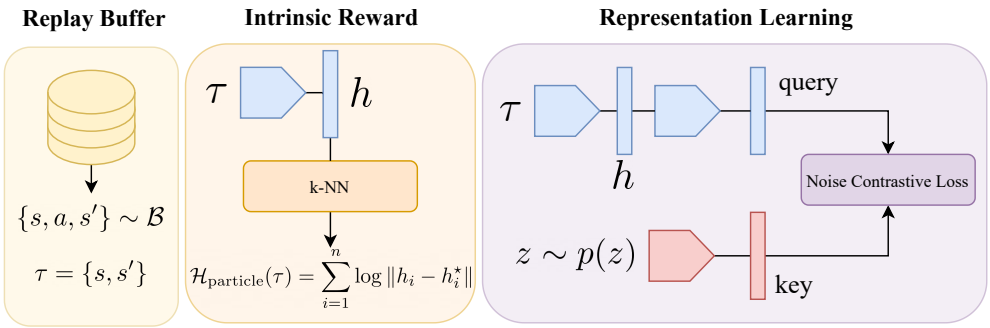


Figure 8.3 Architecture illustrating the practical implementation of CIC . During a gradient update step, random $\tau = (s, s')$ tuples are sampled from the replay buffer, then a particle estimator is used to compute the entropy and a noise contrastive loss to compute the conditional entropy. The contrastive loss is backpropagated through the entire architecture. The entropy and contrastive terms are then scaled and added to form the intrinsic reward. The RL agent is optimized with a DDPG [175].

maximize a mutual information objective with the following general form:

$$I(\tau; z) = \mathcal{H}(z) - \mathcal{H}(z|\tau) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z) \tag{8.1}$$

Competence-based algorithms use different choices for τ and can condition on additional information such as actions or starting states. For a full summary of competence-based algorithms and their objectives see Table 8.2 in Appendix 8.13.

Lower Bound Estimates of Mutual Information: The mutual information $I(s; z)$ is intractable to compute directly. Since we wish to maximize $I(s; z)$, we can approximate this objective by instead maximizing a lower bound estimate. Most known mutual information maximization algorithms use the variational lower bound introduced in Barber and Agakov [20]:

$$I(\tau; z) = \mathcal{H}(z) - \mathcal{H}(z|\tau) \geq \mathcal{H}(z) + \mathbb{E}[\log q(z|\tau)] \tag{8.2}$$

The variational lower bound can be applied to both decompositions of the mutual information. The design decisions of a competence-based algorithm therefore come down to (i) which decomposition of $I(\tau; z)$ to use, (ii) whether to use discrete or continuous skills, (iii) how to estimate $H(z)$ or $H(\tau)$, and finally (iv) how to estimate $H(z|\tau)$ or $H(\tau|z)$.

8.3 Motivation

Results from the recent Unsupervised Reinforcement Learning Benchmark (URLB) [155] show that competence-based approaches underperform relative to knowledge-based and

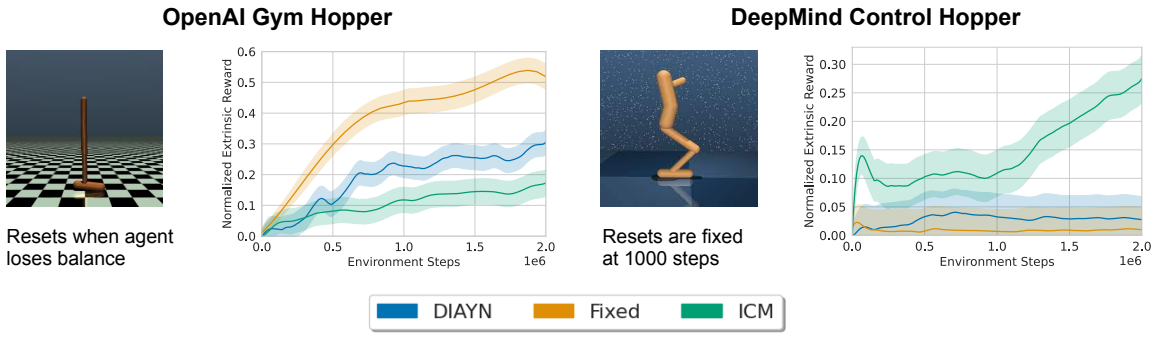


Figure 8.4 To empirically demonstrate issues inherent to competence-based exploration methods, we run DIAYN [82] and compare it to ICM [225] and a *Fixed* baseline where the agent receives an intrinsic reward of 1.0 for each timestep and no extrinsic reward on both OpenAI Gym *(episode resets when agent loses balance)* and DeepMind Control (DMC) *(episode is fixed for 1k steps)* Hopper environments. Since Gym and DMC rewards are on different scales, we normalize rewards based on the maximum reward achieved by any algorithm (1k for Gym, 3 for DMC). While DIAYN is able to achieve higher extrinsic rewards than ICM on Gym, the Fixed intrinsic reward baseline performs best. However, on DMC the Fixed and DIAYN agents achieve near-zero reward while ICM does not. This is consistent with findings of prior work that DIAYN is able to learn diverse behaviors in Gym [82] as well as the observation that DIAYN performs poorly on DMC environments [155]

data-based baselines on DeepMind Control (DMC). We argue that the underlying issue with current competence-based algorithms when deployed on harder exploration environments like DMC has to do with the currently used estimators for $I(\tau; z)$ rather than the objective itself. To produce structured skills that lead to diverse behaviors, $I(\tau; z)$ estimators must (i) explicitly encourage diverse behaviors and (ii) have the capacity to discriminate between high-dimensional continuous skills. Current approaches do not satisfy both criteria.

Competence-base algorithms do not ensure diverse behaviors: Most of the best known competence-based approaches [82, 102, 2, 161], optimize the first decomposition of the mutual information $\mathcal{H}(z) - \mathcal{H}(z|\tau)$. The issue with this decomposition is that while it ensures diversity of skill vectors it does not ensure diverse behavior from the policy, meaning $\max \mathcal{H}(z)$ does not imply $\max \mathcal{H}(\tau)$. Of course, if $H(z) - \mathcal{H}(z|\tau)$ is maximized and the skill dimension is sufficiently large, then $\mathcal{H}(\tau)$ will also be maximized implicitly. Yet in practice, to learn an accurate discriminator $q(z|\tau)$, the above methods assume skill spaces that are much smaller than the state space (see Table 8.2), and thus behavioral diversity may not be guaranteed. In contrast, the decomposition $I(\tau; z) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z)$ ensures diverse behaviors through the entropy term $\mathcal{H}(\tau)$. Methods that utilize this decomposition include Liu and Abbeel [181], Sharma et al. [265].

Why it is important to utilize high-dimensional skills: Once a policy is capable of generating diverse behaviors, it is important that the discriminator can distill these behaviors into

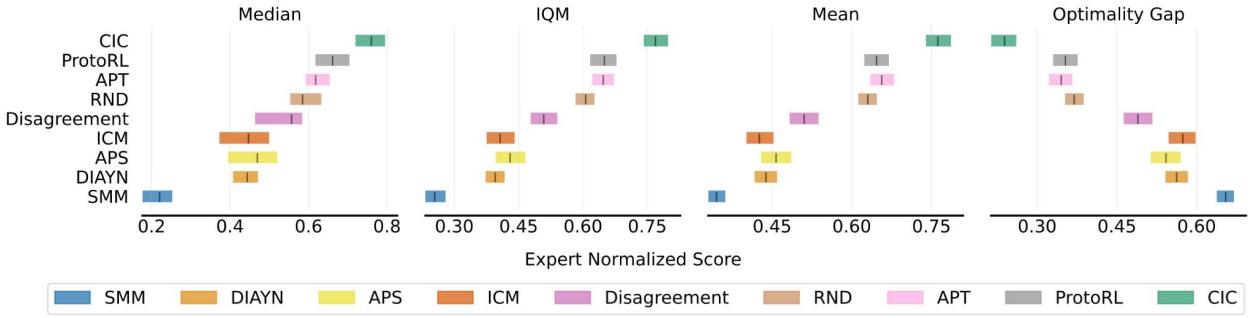


Figure 8.5 We report the aggregate statistics using stratified bootstrap intervals [3] for 12 downstream tasks on URLB with 10 seeds, so each statistic for each algorithm has 120 seeds in total. We find that overall, CIC achieves leading performance on URLB in terms of the IQM, mean, and OG statistics. As recommended by Agarwal et al. [3], we use the IQM as our primary performance measure. In terms of IQM, CIC improves upon the next best skill discovery algorithm (APS) by 79% and the next best algorithm overall (ProtoRL) by 18%.

distinct skills. If the set of behaviors outnumbers the set of skills, this will result in degenerate skills – when one skill maps to multiple different behaviors. It is therefore important that the discriminator can accommodate continuous skills of sufficiently high dimension. Empirically, the discriminators used in prior work utilize only low-dimensional continuous skill vectors. DIAYN [82] utilized 16 dimensional skills, DADS [265] utilizes continuous skills of dimension $2-5$, while APS [181], an algorithm that utilizes successor features [21, 110] for the discriminator, is only capable of learning continuous skills with dimension 10. We show how small skill spaces can lead to ineffective exploration in a simple gridworld setting in Appendix 8.16.

On the importance of benchmarks for evaluation: While prior competence-based approaches such as DIAYN [82] were evaluated on OpenAI Gym [37], Gym environment episodes terminate when the agent loses balance thereby leaking some aspects of extrinsic signal to the exploration agent. On the other hand, DMC episodes have fixed length. We show in Fig 8.4 that this small difference in environments results in large performance differences. Specifically, we find that DIAYN is able to learn diverse skills in Gym but not in DMC, which is consistent with both observations from DIAYN and URLB papers. Due to fixed episode lengths, DMC tasks are harder for reward-free exploration since agents must learn to balance without supervision.

8.4 Method

8.4.1 Contrastive Intrinsic Control

From Section 8.3 we are motivated to find a lower bound for $I(\tau; z)$ with a discriminator that is capable of supporting high-dimensional continuous skills[2]. Additionally, we wish to increase the diversity of behaviors so that the discriminator can continue learning new skills throughout training. To improve the discriminator, we propose to utilize noise contrastive estimation (NCE) [106] between state-transitions and latent skills as a lower bound for $I(\tau; z)$.[3] It has been shown previously that such estimators provide a valid lower bound for mutual information [214]. However, to the best of our knowledge, this is the first work to investigate contrastive representation learning for intrinsic control.

Representation Learning: Specifically, we propose to learn embeddings with the following representation learning objective, which is effectively CPC between state-transitions and latent skills:

$$I(\tau; z) \geq \mathbb{E}[f(\tau, z) - \log \frac{1}{N} \sum_{j=1}^{N} \exp(f(\tau_j, z))]. \tag{8.3}$$

where $f(\tau, z)$ is any real valued function. For convenience, we define the discriminator $\log q(\tau|z)$ as

$$\log q(\tau|z) \coloneqq f(\tau, z) - \log \frac{1}{N} \sum_{j=1}^{N} \exp(f(\tau_j, z)). \tag{8.4}$$

For our practical algorithm, we parameterize this function as

$$f(\tau, z) = g_{\psi_1}(\tau)^\top g_{\psi_2}(z) / \|g_{\psi_1}(\tau)\| \|g_{\psi_2}(z)\| T, \tag{8.5}$$

where $\tau = (s, s')$ is a transition tuple, g_{ψ_k} are neural encoders, and T is a temperature parameter. This inner product is similar to the one used in SimCLR [55].

The representation learning loss backpropagates gradients from the NCE loss which maximizes similarity between state-transitions and corresponding skills.

$$\begin{aligned} F_{NCE}(\tau) = & \frac{g_{\psi_1}(\tau_i)^\top g_{\psi_2}(z_i)}{\|g_{\psi_1}(\tau_i)\| \|g_{\psi_2}(z_i)\| T} \\ & - \log \frac{1}{N} \sum_{j=1}^{N} \exp \left(\frac{g_{\psi_1}(\tau_j)^\top g_{\psi_2}(z_i)}{\|g_{\psi_1}(\tau_j)\| \|g_{\psi_2}(z_i)\| T} \right) \end{aligned} \tag{8.6}$$

[2] In high-dimensional state-action spaces the number of distinct behaviors can be quite large.

[3] Note that τ is not a trajectory but some function of states.

We provide pseudocode for the CIC representation learning loss below:

```
"""
PyTorch-like pseudocode for the CIC loss
"""

def cic_loss(s, s_next, z, temp):
    """
    - states: s, s_next (B, D)
    - skills: z (B, D)
    """

    tau = concat(s, s_next, dim=1)

    query = query_net(z)
    key = key_net(tau)

    query = normalize(query, dim=1)
    key = normalize(key, dim=1)

    """
    positives are on diagonal
    negatives are off diagonal
    """

    logits = matmul(query, key.T) / temp
    labels = arange(logits.shape[0])

    loss = cross_entropy(logits, labels)

    return loss
```

Listing 8.1: Pseudocode for the CIC loss

Intrinsic reward: Although we have a representation learning objective, we still need to specify the intrinsic reward for the algorithm for which there can be multiple choices. Prior works consider specifying an intrinsic reward that is proportional to state-transition entropy [180], the discriminator [82], a similarity score between states and skills [315], and the uncertainty of the discriminator [281]. We investigate each of these choices and find that an intrinsic reward that maximizes state-transition entropy coupled with representation learning via the CPC loss defined in Sec. 8.4.1 is the simplest variant that also performs well (see Table 8.1).

For the intrinsic reward, we use a particle estimate [274, 27] as in Liu and Abbeel [180] of the state-transition entropy. Similar to Liu and Abbeel [180], Yarats et al. [324] we estimate the entropy up to a proportionality constant, because we want the agent to maximize entropy rather than estimate its exact value.

The APT particle entropy estimate is proportional to the distance between the current visited state transition and previously seen neighboring points.

$$\mathcal{H}_{particle}(\tau) \propto \frac{1}{N_k} \sum_{h_i^\star \in N_k}^{N_k} \log \|h_i - h_i^\star\| \tag{8.7}$$

where h_i is an embedding of τ_i shown in Fig. 8.3, h_i^* is a kNN embedding, N_k is the number of kNNs, and $N-1$ is the number of negatives. The total number of elements in the summation is N because it includes one positive.

Explore and Exploit: With these design choices the two components of the CIC algorithm can be interpreted as *exploration* with intrinsic rewards and *exploitation* using representation learning to distill behaviors into skills. The marginal entropy maximizes the diversity of state-transition embeddings while the contrastive discriminator $\log q(\tau|z)$ encourages exploitation by ensuring that skills z lead to predictable states τ. Together the two terms incentivize the discovery of diverse yet predictable behaviors from the RL agent. While CIC shares a similar intrinsic reward structure to APT [180], we show that the new representation learning loss from the CIC estimator results in substantial performance gains in Sec 8.6.

8.5 Practical Implementation

Our practical implementation consists of two main components: the RL optimization algorithm and the CIC architecture. For fairness and clarity of comparison, we use the same RL optimization algorithm for our method and all baselines in this work. Since the baselines implemented in URLB [155] use a DDPG[4] [175] as their backbone, we opt for the same DDPG architecture to optimize our method as well (see Appendix 8.11).

CIC Architecture: We use a particle estimator as in Liu and Abbeel [180] to estimate $\mathcal{H}(\tau)$. To compute the variational density $q(\tau|z)$, we first sample skills from uniform noise $z \sim p(z)$ where $p(z)$ is the uniform distribution over the $[0, 1]$ interval. We then use two MLP encoders to embed $g_{\psi_1}(\tau)$ and $g_{\psi_2}(z)$, and optimize the parameters ψ_1, ψ_2 with the CPC loss similar to SimCLR [55] since $f(\tau, z) = g_{\psi_1}(\tau)^T g_{\psi_2}(z)$. We fix the hyperparameters across all domains and downstream tasks.

Adapting to downstream tasks: To adapt to downstream tasks we follow the same procedure for competence-based method adaptation as in URLB [155]. During the first 4k environment interactions we populate the DDPG replay buffer with samples and use the extrinsic rewards collected during this period to finetune the skill vector z. While it's common to finetune skills with Cross Entropy Adaptation (CMA), given our limited budget of 4k samples (only 4 episodes) we find that a simple grid sweep of skills over the interval $[0, 1]$ produces the best results. After this, we fix the skill z and finetune the DDPG actor-critic parameters against

[4]It was recently was shown that a DDPG achieves state-of-the-art performance [322] on DeepMind Control [288] and is more stable than SAC [108] on this benchmark.

the extrinsic reward for the remaining 96k steps. Note that competence-based methods in URLB also finetune their skills during the first 4k finetuning steps ensuring a fair comparison between the methods.

8.6 Experimental Setup

Environments We evaluate our approach on tasks from URLB, which consists of twelve downstream tasks across three challenging continuous control domains for exploration algorithms – walker, quadruped, and Jaco arm. Walker requires a biped constrained to a 2D vertical plane to perform locomotion tasks while balancing. Quadruped is more challenging due to a higher-dimensional state-action space and requires a quadruped to in a 3D environment to learn locomotion skills. Jaco arm is a 6-DOF robotic arm with a three-finger gripper to move and manipulate objects without locking. All three environments are challenging in the absence of an extrinsic reward.

Baselines: We compare CIC to baselines across all three exploration categories. Knowledge-based basedlines include ICM [225], Disagreement [226], and RND [41]. Data-based baselines incude APT [180] and ProtoRL [324]. Competence-based baselines include DIAYN [82], SMM [161], and APS [181]. The closest baselines to CIC are APT, which is similar to CIC but without state-skill CPC representation learning (no discriminator), and APS which uses the same decomposition of mutual information as CIC and also uses a particle entropy estimate for $\mathcal{H}(\tau)$. The main difference between APS and CIC is that APS uses successor features while CIC uses a contrastive estimator for the discriminator. For further details regarding baselines we refer the reader to Appendix 8.12.

Evaluation: We follow an identical evaluation to the 2M pre-training setup in URLB. First, we pre-train each RL agent with the intrinsic rewards for 2M steps. Then, we finetune each agent to the downstream task with extrinsic rewards for 100k steps. All baselines were run for 10 seeds per downstream task for each algorithm using the code and hyperparameters provided by URLB [155]. Built on top of URLB, CIC is also run for 10 seeds per task. A total of $1080 = 9$ algorithms $\times$ 12 tasks $\times$ 10 seeds experiments were run for the main results. Importantly, all baselines and CIC use a DDPG agent as their backbone.

To ensure that our evaluation statistics are unbiased we use stratified bootstrap confidence intervals to report aggregate statistics across M runs with N seeds as described in *Rliable* [3] to report statistics for our main results in Fig. 8.5. Our primary success metric is the interquartile mean (IQM) and the Optimality Gap (OG). IQM discards the top and bottom 25% of runs and then computes the mean. It is less susceptible to outliers than the mean and was shown to be the most reliable statistic for reporting results for RL experiments in Agarwal et al. [3]. OG measures how far a policy is from optimal (expert) performance. To define expert performance we use the convention in URLB, which is the score achieved by a randomly initialized DDPG after 2M steps of finetuning (20x more steps than our finetuning

budget).

8.7 Results

We investigate empirical answers to the following research questions: (Q1) How does CIC adaptation efficiency compare to prior competence-based algorithms and exploration algorithms more broadly? (Q2) Which intrinsic reward instantiation of CIC performs best? (Q3) How do the two terms in the CIC objective affect algorithm performance? (Q4) How does skill selection affect the quality of the pre-trained policy? (Q5) Which architecture details matter most?

Adaptation efficiency of CIC and exploration baslines: Expert normalized scores of CIC and exploration algorithms from URLB are shown in Fig. 8.3. We find that CIC substantially outperforms prior competence-based algorithms (DIAYN, SMM, APS) achieving a 79% higher IQM than the next best competence-based method (APS) and, more broadly, achieving a 18% higher IQM than the next best overall baseline (ProtoRL). In further ablations, we find that the contributing factors to CIC's performance are its ability to accommodate substantially larger continuous skill spaces than prior competence-based methods.

Intrinsic reward specification: The intrinsic reward for competence-based algorithms can be instantiated in many different ways. Here, we analyze intrinsic reward for CIC with the form $r_{int} = H(\tau) + D(\tau, z)$, where D is some function of (τ, z). Prior works, select D to be (i) the discriminator [181], (ii) a cosine similarity between embeddings [315], (iii) uncertainty of the discriminator [281], and (iv) just the entropy $D(\tau, z) = 0$ [180]. We run CIC with each of these variants on the walker and quadruped tasks and measure the final mean performance across the downstream tasks (see Tab. 8.1). The results show that the entropy-only intrinsic reward performs best. For this reason the intrinsic reward and representation learning aspects of CIC are decoupled. We hypothesize that the reason why a simple entropy-only intrinsic reward works well is that state-skill CPC representation learning clusters similar behaviors together. Since redundant behaviors are clustered, maximizing the entropy of state-transition embeddings produces increasingly diverse behaviors.

The importance of representation learning: To what extent does representation learning with state-skill CPC (see Eq. 8.3) affect the agent's exploration capability? To answer this question we train the CIC agent with the entropy intrinsic reward with and without the representation learning auxiliary loss for 2M steps. The zero-shot reward plotted in Fig. 8.6 indicates that without representation learning the policy collapses. With representation learning, the agent is able to discover diverse skills evidenced by the non-zero reward. This result suggests that state-skill CPC representation learning is a critical part of CIC.

Qualitative analysis of CIC behaviors: Qualitatively, we find that CIC is able to learn locomotion behaviors in DMC without extrinsic information such as early termination as in OpenAI Gym. While most skills are higher entropy and thus more chaotic, we show in Fig 8.2

	disc.	similarity	uncertainty	entropy
walker	0.80	0.79	0.78	0.82
quad.	0.44	0.63	0.75	0.74
mean	0.62	0.71	0.77	0.78

Table 8.1 Analyzing four different intrinsic reward specifications for CIC, we find that entropy-based intrinsic reward performs best, suggesting that the CIC discriminator is primarily useful for representation learning. These are normalized scores averaged over 3 seeds across 8 downstream tasks (24 runs per data point).

that structured behaviors can be isolated by fixing a particular skill vector. For example, in the walker and quadruped domains - balancing, walking, and flipping skills can be isolated. For more qualitative investigations we refer the reader to Appendix 8.17.

8.8 Conclusion

We have introduced a new competence-based algorithm – Contrastive Intrinsic Control (CIC) – which enables more effective exploration than prior unsupervised skill discovery algorithms by explicitly encouraging diverse behavior while distilling predictable behaviors into skills with a contrastive discriminator. We showed that CIC is the first competence-based approach to achieve leading performance on URLB. We hope that this encourages further research in developing RL agents capable of generalization.

8.9 Acknowledgements

We would like to thank Ademi Adeniji, Xinyang Geng, Fangchen Liu for helpful discussions. We would also like to thank Phil Bachman for useful feedback. This work was partially supported by Berkeley DeepDrive, NSF AI4OPT AI Institute for Advances in Optimization under NSF 2112533, and the Office of Naval Research grant N00014-21-1-2769.

8.10 Competence-based Exploration Algorithms

The competence-based algorithms considered in this work aim to maximize $I(\tau; s)$. The algorithms differ by ho they decompose mutual information, whether they explicitly maximize behavioral entropy, their skill space (discrete or continuous) and their intrinsic reward structure. We provide a list of common competence-based algorithms in Table 8.2.

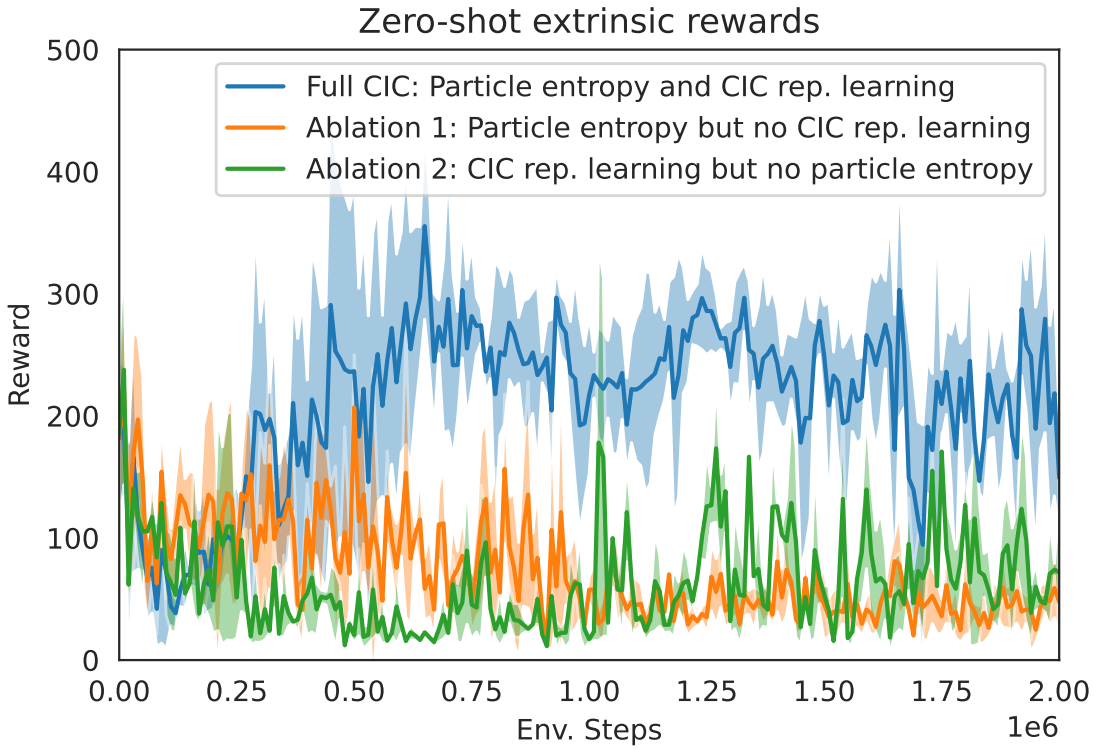


Figure 8.6 Mean zero-shot extrinsic rewards for Quadruped stand over 3 seeds with and without state-skill representation learning. Without representation learning, the algorithm collapses. Similarly, with CIC representation learning but no entropy term (in which case we use the discriminator as the intrinsic reward) the policy also collapses. Note that there is no finetuning happening here. We're showing the task-specific extrinsic reward during reward-free pre-training as a way to sense-check the exploration policy.

8.11 Deep Deterministic Policy Gradient (DDPG)

A DDPG is an actor-critic RL algorithm that performs off-policy gradient updates and learns a Q function $Q_\phi(s, a)$ and an actor $\pi_\theta(a|s)$. The critic is trained by satisfying the Bellman equation.

$$\mathcal{L}_Q(\phi, \mathcal{D}) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}} \left[\left(Q_\phi(s_t, a_t) - r_t - \gamma Q_{\bar{\phi}}(s_{t+1}, \pi_\theta(s_{t+1})) \right)^2 \right]. \tag{8.8}$$

Here, $\bar{\phi}$ is the Polyak average of the parameters ϕ. As the critic minimizes the Bellman error, the actor maximizes the action-value function.

$$\mathcal{L}_\pi(\theta, \mathcal{D}) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[Q_\phi(s_t, \pi_\theta(s_t)) \right]. \tag{8.9}$$

Table 8.2 Prior Competence-based Unsupervised Skill Discovery Algorithms

Algorithm	Intrinsic Reward	Decomposition	Explicit max $\mathcal{H}(\tau)$	Skill Dim.	Skill Space
SSN4HRL [86]	$\log q_\psi(z\|s_t)$	$H(z) - H(z\|\tau)$	No	6	discrete
VIC [102]	$\log q_\psi(z\|s_H))$	$H(z) - H(z\|\tau)$	No	60	discrete
VALOR [2]	$\log q_\psi(z\|s_{1:H})$	$H(z) - H(z\|\tau)$	No	64	discrete
DIAYN [82]	$\log q_\psi(z\|s_t)$	$H(z) - H(z\|\tau)$	No	128	discrete
DADS [265]	$q_\psi(s'\|z,s) - \sum_i \log q(s'\|z_i,s)$	$H(\tau) - H(\tau\|z)$	Yes	5	continuous
VISR [110]	$\log q_\psi(z\|s_t)$	$H(z) - H(z\|\tau)$	No	10	continuous
APS [181]	$F_{\text{Successor}}(s\|z) + \mathcal{H}_{\text{particle}}(s)$	$\mathcal{H}(\tau) - \mathcal{H}(\tau\|z)$	Yes	10	continuous

Table 8.3 A list of competence-based algorithms. We describe the intrinsic reward optimized by each method and the decomposition of the mutual information utilized by the method. We also note whether the method explicitly maximizes state transition entropy. Finally, we note the maximal dimension used in each work and whether the skills are discrete or continuous. All methods prior to CIC only support small skill spaces, either because they are discrete or continuous but low-dimensional.

8.12 Baselines

For baselines, we choose the existing set of benchmarked unsupervised RL algorithms on URLB. We provide a quick summary of each method. For more detailed descriptions of each baseline we refer the reader to URLB [155]

Competence-based Baselines: CIC is a competence-based exploration algorithm. For baselines, we compare it to DIAYN [82], SMM [161], and APS [181]. Each of these algorithms is described in Table 8.2. Notably, APS is a recent state-of-the-art competence-based method that is the most closely related algorithm to the CIC algorithm.

Knowledge-based Baselines: For knowledge-based baselines, we compare to ICM [225], Disagreement [226], and RND [41]. ICM and RND train a dynamics model and random network prediction model and define the intrinsic reward to be proportional to the prediction error. Disagreement trains an ensemble of dynamics models and defines the intrinsic reward to be proportional to the uncertainty of an ensemble.

Data-based Baselines: For data-based baselines we compare to APT [180] and ProtoRL [324]. Both methods use a particle estimator to estimate the state visitation entropy. ProtoRL also performs discrete contrastive clustering as in Caron et al. [46] as an auxiliary task and uses the resulting clusters to compute the particle entropy. While ProtoRL is more effective than APT when learning from pixels, on state-based URLB APT is competitive with ProtoRL.

Our method CIC is effectively a skill-conditioned APT agent with a contrastive discriminator.

8.13 Relation to Prior Skill Discovery Methods

The most closely relatd prior algorithms to CIC are APT [180] and APS [181]. Both CIC and APS use the $\mathcal{H}(\tau) - \mathcal{H}(\tau|z)$ decomposition of the mutual information and both used a particle estimator [274] to compute the state entropy as in Liu and Abbeel [180]. The main difference between CIC and APS is the discriminator. APS uses successor features as in Hansen et al. [110] for its discriminator while CIC uses a noise contrastive estimator. Unlike successor features, which empirically only accommodate low-dimensional continuous skill spaces (see Table 8.2), the noise contrastive discriminator is able to leverage higher continuous dimensional skill vectors. Like APT, CIC has an intrinsic reward that maximizes $\mathcal{H}(\tau)$. However, CIC also does contrastive skill learning to shape the embedding space and outputs a skill-conditioned policy.

The CIC discriminator is similar to the one used in DISCERN [315], a goal-conditioned unsupervised RL algorithm. Both methods use a contrastive discriminator by sampling negatives and computing an inner product between queries and keys. The main differences are (i) that DISCERN maximizes $I(\tau; g)$ where g are image goal embeddings while CIC maximizes $I(\tau; z)$ where z are abstract skill vectors; (ii) DISCERN uses the DIAYN-style decomposition $I(\tau; g) = H(g) - H(g|\tau)$ while CIC decomposes through $H(\tau) - H(\tau|z)$, and (iii) DISCERN discards the $H(g)$ term by sampling goals uniformly while CIC explicitly maximizes $\mathcal{H}(\tau)$. While DISCERN and CIC share similarities, DISCERN operates over image goals while CIC operates over abstract skill vectors so the two methods are not directly comparable.

Finally, another similar algorithm to CIC is DADS [265] which also decomposes through $H(\tau) - H(\tau|z)$. While CIC uses a contrastive density estimate for the discriminator, DADS uses a maximum likelihood estimator similar to DIAYN. DADS maximizes $I(s'|s, z)$ and estimates entropy $\mathcal{H}(s'|s)$ by marginalizing over z such that $\mathcal{H}(s'|s) = -\log \sum_i q(s'|s, z_i)$ while CIC uses a particle estimator.

8.14 Hyper-parameters

Baseline hyperparameters are taken from URLB [155], which were selected by performing a grid sweep over tasks and picking the best performing set of hyperparameters. Except for the skill dimension, hyperparameters for CIC are borrowed from URLB.

Table 8.4 Hyper-parameters used for CIC .

DDPG hyper-parameter	Value
Replay buffer capacity	10^6
Action repeat	1
Seed frames	4000
n-step returns	3
Mini-batch size	1024
Seed frames	4000
Discount (γ)	0.99
Optimizer	Adam
Learning rate	10^{-4}
Agent update frequency	2
Critic target EMA rate (τ_Q)	0.01
Features dim.	1024
Hidden dim.	1024
Exploration stddev clip	0.3
Exploration stddev value	0.2
Number pre-training frames	2×10^6
Number fine-turning frames	1×10^5
CIC hyper-parameter	**Value**
Skill dim	64 continuous
Prior	Uniform [0,1]
Skill sampling frequency (steps)	50
State net arch. $g_{\psi_1}(s)$	$\dim(\mathcal{O}) \to 1024 \to 1024 \to 64$ ReLU MLP
Skill net arch. $g_{\psi_2}(z)$	$64 \to 1024 \to 1024 \to 64$ ReLU MLP
Prediction net arch.	$64 \to 1024 \to 1024 \to 64$ ReLU MLP

8.15 Raw Numerical Results

We provide a list of raw numerical results for finetuning CIC and baselines in Tables 8.5 and 8.6. All baselines were run using the code provided by URLB [155] for 10 seeds per downstream task.

Statistic	ICM	Dis.	RND	APT	Proto	DIAYN	APS	SMM	CIC	% CIC > APS	% CIC > Proto
Median ↑	0.45	0.56	0.58	0.62	0.66	0.44	0.47	0.22	0.76	+61%	+15%
IQM ↑	0.41	0.51	0.61	0.65	0.65	0.40	0.43	0.25	0.77	+79%	+18%
Mean ↑	0.43	0.51	0.63	0.66	0.65	0.44	0.46	0.35	0.76	+65%	+17%
OG ↓	0.57	0.49	0.37	0.35	0.35	0.56	0.54	0.65	0.24	-44%	-68%

Table 8.5 Statics for downstream task normalized scores for CIC and baselines from URLB [155]. CIC improves over both the prior leading competence-based method APS [181] and overall next-best exploration algorithm ProtoRL [324] across all readout statistics. Each data point is a statistic computed using 10 seeds and 12 downstream tasks (120 experiments per data point). The statistics are computed using RLiable [3].

Pre-trainining for 2×10^6 environment steps											
Domain	Task	Expert	DDPG	CIC	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	799	538±27	631 ± 34	417±16	346±13	474±39	544±14	456±12	450±24	319±17	465±20
	Run	796	325±25	486 ± 25	247±21	208±15	406±30	392±26	306±13	426±26	158±8	134±16
	Stand	984	899±23	959 ± 2	859±23	746±34	911±5	942±6	917±27	924±12	695±46	721±44
	Walk	971	748±47	885 ± 28	627±42	549±37	704±30	773±70	792±41	770±44	498±27	527±79
Quadruped	Jump	888	236±48	595 ± 42	178±35	389±62	637±12	648±18	617±44	96±7	660±43	463±51
	Run	888	157±31	505 ± 47	110±18	337±30	459±6	492±14	373±33	96±6	433±29	281±17
	Stand	920	392±73	761 ± 54	312±68	512±89	766±43	872±23	716±56	123±11	851±43	542±53
	Walk	866	229±57	723 ± 43	126±27	293±37	536±39	770±47	412±54	80±6	576±81	436±79
Jaco	Reach bottom left	193	72±22	138 ± 9	111±11	124±7	110±5	103±8	129±8	45±7	39±6	76±8
	Reach bottom right	203	117±18	145 ± 7	97±9	115±10	117±7	100±6	132±8	46±11	38±5	88±11
	Reach top left	191	116±22	153 ± 7	82±14	106±12	99±6	73±12	123±9	36±3	19±4	68±6
	Reach top right	223	94±18	163 ± 4	103±11	139±7	100±6	90±10	159±7	47±6	28±6	76±10

Table 8.6 Performance of CIC and baselines on state-based URLB after first pre-training for 2×10^6 steps and then finetuning with extrinsic rewards for 1×10^5. All baselines were run for 10 seeds per downstream task for each algorithm using the code provided by URLB [155]. A total of $1080 = 9$ algorithms $\times$ 12 tasks $\times$ 10 seeds experiments were run.

8.16 Toy Example to Illustrate the Need for Larger Skill Spaces

We illustrate the need for larger skill spaces with a gridworld example. Suppose we have an agent in a 10×10 sized gridworld and that we have four discrete skills at our disposal. Now let $\tau = s$ and consider how we may achieve maximal $I(\tau; z)$ in this setting. If we decompose $I(\tau; z) = \mathcal{H}(z) - \mathcal{H}(z|\tau)$ then we can achieve maximal $\mathcal{H}(z)$ by sampling the four skills uniformly $z \sim p(z)$. We can achieve $\mathcal{H}(z|\tau) = 0$ by mapping each skill to a distinct neighboring state of the agent. Thus, our mutual information is maximized but as a result the agent only explores four out of the hundrend available states in the gridworld.

Now suppose we consider the second decomposition $I(\tau; z) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z)$. Since the agent is maximizing $\mathcal{H}(\tau)$ it is likely to visit a diverse set of states at first. However, as soon as it learns an accurate discriminator we will have $\mathcal{H}(\tau|z)$ and again the skills can be mapped to

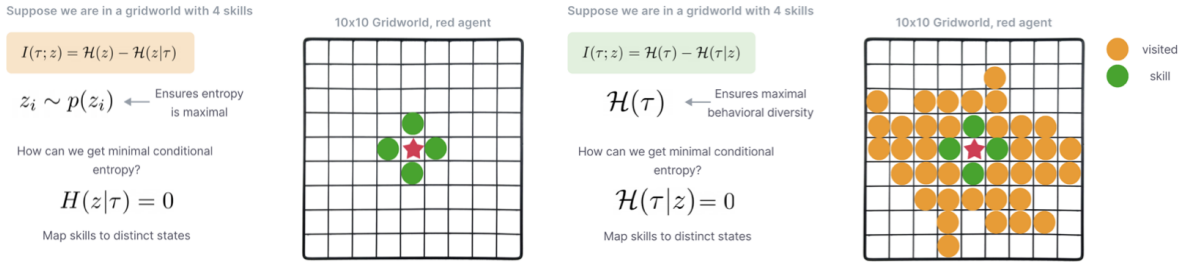


Figure 8.7 A gridworld example motivating the need for large skill spaces. In this environment, we place an agent in a 10×10 gridworld and provide the agent access to four discrete skills. We show that the mutual information objective can be maximized by mapping these four skills to the nearest neighboring states resulting in low behavioral diversity and exploring only four of the hundred available states.

neighboring states to achieve minimal conditional entropy. As a result, the skill conditioned policy will only be able to reach four out of the hundrend possible states in this gridworld. This argument is shown visually in Fig. 8.7.

Skill spaces that are too large can also be an issue. Consider if we had 100 skills at our disposal in the same gridworld. Then the agent could minimize the conditional entropy by mapping each skill to a unique state which would result in the agent memorizing the environment by finding a one-to-one mapping between states and skills. While this is a potential issue it has not been encountered in practice yet since current competence-based methods support small skill spaces relative to the observation space of the environment.

8.17 Qualitative Analysis of Skills

We provide two additional qualitative analyses of behaviors learned with the CIC algorithm. First, we take a simple pointmass setting and set the skill dimension to 1 in order to ablate the skills learned by the CIC agent in a simple setting. We sweep over different values of z and plot the behavioral flow vector field (direction in which point mass moves) in Fig.8.8. We find that the pointmass learns skills that produce continuous motion and that the direction of the motion changes as a function of the skill value. Near the origin the pointmass learns skills that span all directions, while near the edges the point mass learns to avoid wall collisions. Qualitatively, many behaviors are periodic.

Qualitatively, we find that methods like DIAYN that only support low dimensional skill vectors and do not explicitly incentivize diverse behaviors in their objective produce policies that map skills to a small set of static behaviors. These behaviors shown in Fig. 8.9 are non-trivial but also have low behavioral diversity and are not particularly useful for solving

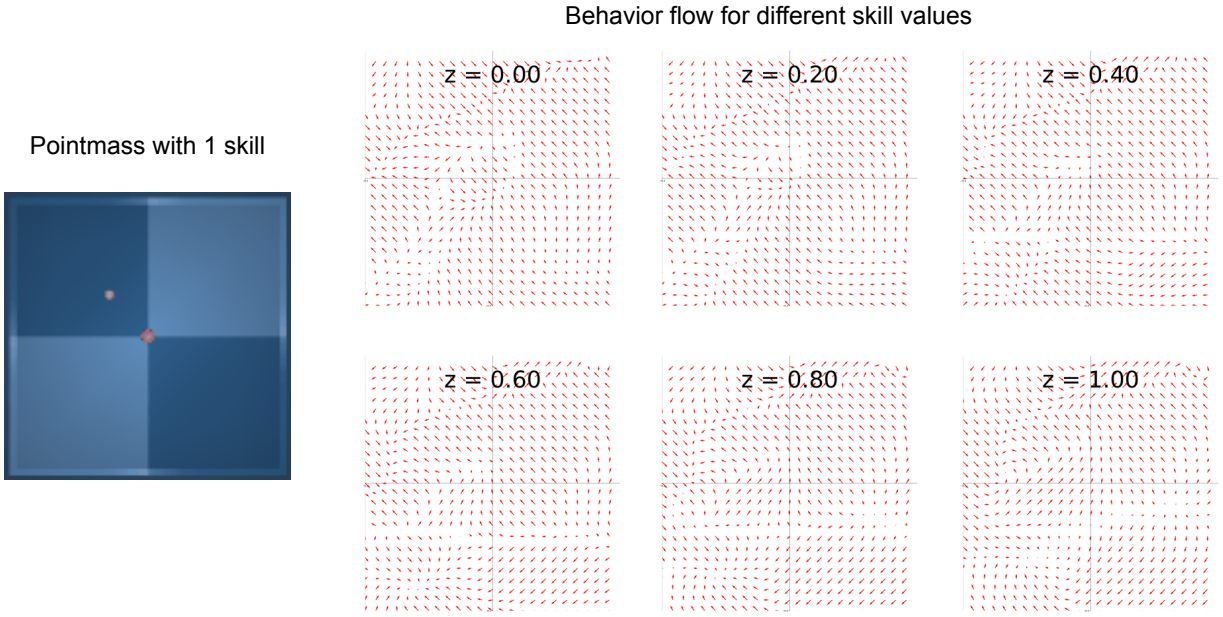


Figure 8.8 Learning curves for finetuning pre-trained agents for 100k steps. Task performance is aggregated for each domain, such that each curve represents the mean normalized scores over $4 \times 10 = 40$ seeds. The shaded regions represent the standard error. CIC surpasses the performance of the prior state-of-the-art on Walker and Jaco tasks while tying on Quadruped. CIC is the only algorithm that performs consistently well across all three domains.

the downstream task. This observation is consistent with Zahavy et al. [330] where the authors found that DIAYN maps to static "yoga" poses in DeepMind Control. In contrast, behaviors produce by CIC are dynamic resulting flipping, jumping, and locomotive behaviors that can then be adapted to efficiently solve downstream tasks.

8.18 OpenAI Gym vs. DeepMind control: How Early Termination Leaks Extrinsic Signal

Prior work on unsupervised skill discovery for continuous control [82, 265] was evaluated on OpenAI Gym [37] and showed diverse exploration on Gym environments. However, Gym environment episodes terminate early when the agent loses balance, thereby leaking information about the extrinsic task (e.g. balancing or moving). However, DeepMind Control (DMC) episodes have a fixed length of 1k steps. In DMC, exploration is therefore harder since the agent needs to learn to balance without any extrinsic signal.

To evaluate whether the difference in the two environments has impact on competence-based exploration, we run DIAYN on the hopper environments from both Gym and DMC. We

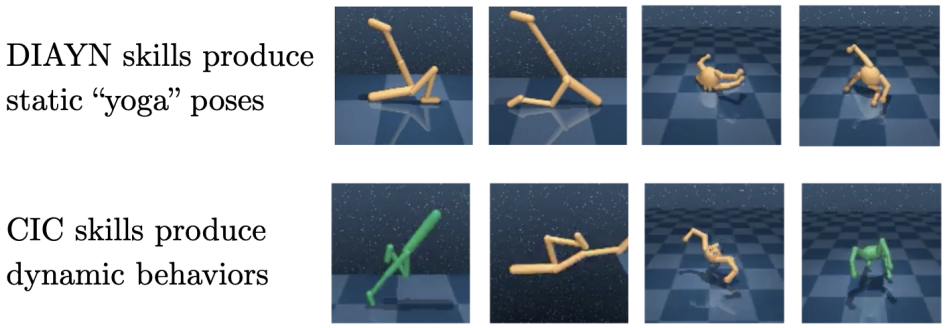


Figure 8.9 Qualitative visualization of DIAYN and CIC pre-training on the Walker and Quadruped domains from URLB. Confirming findings in prior work [330], we also find that DIAYN policies produce static but non-trivial behaviors mapping to “yoga" poses while CIC produces diverse and dynamic behaviors such as walking, flipping, and standing. Though it’s hard to see from these images, all the DIAYN skills get stuck in frozen poses while the CIC skills are producing dynamic behavior with constant motion.

compare to ICM, a popular exploration baseline, and a Fixed baseline where the agent receives an intrinsic reward of 1 for each timestep and no algorithms receive extrinsic rewards. We then measure the extrinsic reward, which loosely corresponds to the diversity of behaviors learned. Our results in Fig. 8.4 show that indeed DIAYN is able to learn diverse behaviors in Gym but not in DMC while ICM is able to learn diverse behaviors in both environments. Interestingly, the Fixed baseline achieves the highest reward on the Gym environment by learning to stand and balance. These results further motivate us to evaluate on URLB which is built on top of DMC.

8.19 CIC vs Other Types of Contrastive Learning for RL

Contrastive learning in CIC is different than prior vision-based contrastive learning in RL such as CURL [152], since we are not performing contrastive learning over augmented images but rather over state transitions and skills. The contrastive objective in CIC is used for unsupervised learning of behaviors while in CURL it is used for unsupervised learning of visual features.

We provide pseudocode for the CIC loss below:

```
def discriminator_loss(states, next_states, skills, temp):
    """
    - states and skills are sampled from replay buffer
    - skills were sampled from uniform dist [0,1] during agent rollout
    - states / next_states: dim (B, D_state)
    - skills: dim (B, D_skill)
```

```
    """

    transitions = concat(states, next_states, dim=1)

    query = skill_net(skills) # (B, D_hidden) -> (B, D_hidden)
    key = transition_net(transitions) # (B, 2*D_state) -> (B, D_hidden)

    query = normalize(query, dim=1)
    key = normalize(key, dim=1)

    logits = matmul(query, key.T) / temp # (B, B)
    labels = arange(logits.shape[0])

    # positives are on diagonal, negatives are off diagonal
    # for each skill, negatives are sampled from transitions
    # while skills are fixed
    loss = cross_entropy(logits, labels)

    return loss
```

Listing 8.2: CIC discriminator loss

This is substantially different from prior contrastive learning works in RL such as CURL [152], which perform contrastive learning over images.

```
def curl_loss(obs, W, temp):
    """
    - observation images are sampled from replay buffer
    - obs: dim (B, C, H, W)
    - W: projection matrix (D_hidden, D_hidden)
    """

    query = aug(obs)
    key = aug(obs)

    query = cnn_net(query) # (B, D_hidden)
    key = cnn_net(key) # (B, D_hidden)

    logits = matmul(matmul(query, W), key.T) / temp # (B, B)
    labels = arange(logits.shape[0])

    # positives are on diagonal
    # negatives are off diagonal
    loss = cross_entropy(logits, labels)

    return loss
```

Listing 8.3: CURL contrastive loss

8.20 On estimates of Mutual Information

In this work we have presented CIC - a new competence-based algorithm that achieves leading performance on URLB compared to prior unsupervised RL methods.

One might wonder whether estimating the exact mutual information (MI) or maximizing the tightest lower bound thereof is really the goal for unsupervised RL. In unsupervised representation learning, state-of-the-art methods like CPC and SimCLR maximize the lower bound of MI based on Noise Contrastive Estimation (NCE). However, as proven in CPC [214] and illustrated in Poole et al. [233] NCE is upper bounded by $\log N$, meaning that the bound is loose when the MI is larger than $\log N$. Nevertheless, these methods have been repeatedly shown to excel in practice. In Tschannen et al. [298] the authors show that the effectiveness of NCE results from the inductive bias in both the choice of feature extractor architectures and the parameterization of the employed MI estimators.

We have a similar belief for unsupervised RL - that with the right parameterization and inductive bias, the MI objective will facilitate behavior learning in unsupervised RL. This is why CIC lower bounds MI with (i) the particle based entropy estimator to ensure explicit exploration and (ii) a contrastive conditional entropy estimator to leverage the power of contrastive learning to discriminate skills. As demonstrated in our experiments, CIC outperforms prior methods, showing the effectiveness of optimizing an intrinsic reward with the CIC MI estimator.

8.21 Limitations

While CIC achieves leading results on URLB, we would also like to address its limitations. First, in this paper we only consider MDPs (and not partially observed MDPs) where the full state is observable. We focus on MDPs because generating diverse behaviors in environments with large state spaces has been the primary bottleneck for competence-based exploration. Combining CIC with visual representation learning to scale this method to pixel-based inputs is a promising future direction for research not considered in this work. Another limitation is that our adaptation strategy to downstream tasks requires finetuning. Since we learn skills, it would be interesting to investigate alternate ways of adapting that would enable zero-shot generalization such as learning generalized reward functions during pre-training.

Chapter 9

Exploration for Diverse AI Supervision

9.1 Introduction

Training large transformers [304] using next token prediction has led to substantial AI advancements, as evidenced by the groundbreaking results they have produced [253, 217]. While this generative AI approach has yielded remarkable AI results, it heavily relies on human supervision. For instance, state-of-the-art AI models including ChatGPT [253] along with a range of other models [61, 97**?** , *inter alia*], rely on fine-tuning through human demonstrations, demanding significant human involvement and domain expertise. This reliance on extensive human supervision presents a substantial challenge since human supervision requires domain expertise, is time consuming, and is tedious. Moreover, humans can struggle to provide reliable supervision in highly specialized domains. For instance, ChatGPT possesses a greater depth of knowledge than the average human, which makes it difficult to rely on humans to provide supervision for ChatGPT. Moreover, while our most advanced AI systems have made significant strides, they still necessitate thorough, human-guided processes to enhance their ability to answer factual or mathematical queries [174]. Yet, when it comes to more intricate and mission-critical tasks, such as navigating complex tax or law regulations, these challenges will demand even more specialized expertise and effort.

Prior works attempt to explore alternatives to human supervision, by using AI supervision instead. For example in mathematical reasoning, these studies propose sampling self generated solutions for human curated questions from large language models and employ techniques like rejection sampling, along with other techniques, to curate training data for the model [65, 213, 18, 127, 331, 327, *inter alia*]. While learning from such sampled content proves effective, a significant challenge persists: the sampled contents often lack the necessary diversity, resulting in a rapid saturation of the learning process [327, 331]. Moreover, the sampling approach has been confined to solutions exclusively, relying on human-curated questions, thus imposing constraints on the diversity of generated data.

To tackle these limitations, we propose a novel approach for using AI models to autonomously

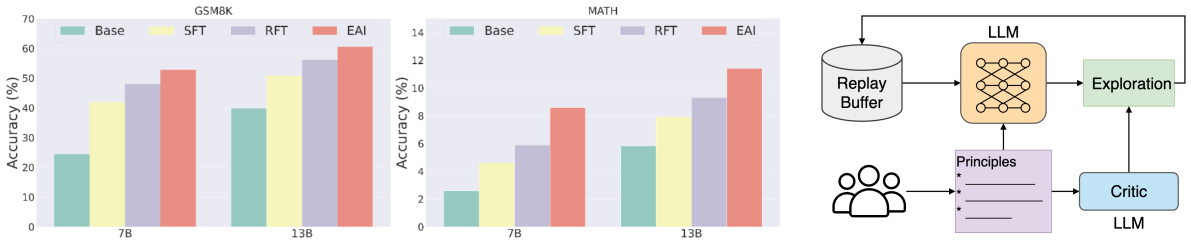


Figure 9.1 Exploratory AI improves mathematical reasoning via exploration. Left and middle: Test accuracy on mathematical reasoning benchmark GSM8K. Baselines include Vicuna, supervised finetuning Vicuna on training set (denoted as SFT), and supervised finetuning Vicuna on rejection sampled model generated diverse solutions on training set (denoted as RFT). Our Exploratory AI (EAI) substantially outperforms all baselines. Right: Our approach EAI generates diverse data for learning by exploring with the guidance of principles and critiques.

generate *diverse* data for learning purposes. This concept draws inspiration the APT algorithm [179] designed for unsupervised RL pretraining [276, 154, 225]. RL pretraining studies exploring in a reward-free environment to develop skills for quickly maximize various downstream rewards. APT allows training RL agent to learn skills by autonomously explore reward free environment based on evaluating novelty of encountered states using particle based entropy estimation [27, 275]. Adapting APT to large language models presents several challenges, including computational complexity and the difficulty of learning reward functions and exploration policies [92, 65]. Rather than relying on traditional RL techniques, we harness the unique capabilities of large language models, such as their ability to learn from context and follow instructions. In essence, we use them to perform the roles of both a reward function and an exploration policy. Our approach, which we term Exploratory AI (EAI), involves two key components: an actor and a critic. The actor is responsible for generating novel content in natural language, while the critic evaluates this generated content and provides critiques to guide the actor's exploration. By evaluating the novelty of the generated contents, our method allows for effective exploration in the rich space of natural language. EAI can generate diverse data independently of human intervention. This makes it more scalable and automated, positioning it as a preferable alternative to methods like supervised finetuning or rejection sampling that depend on data curated by humans. Furthermore, EAI provides an interpretable window into the behavior and knowledge of the model. It sheds light on how well the model possesses knowledge and its reasoning behind generating novel questions. One can look at generations and their corresponding evaluations which provide valuable insights about how model generates and evaluates.

We evaluate our approach on mathematical reasoning benchmarks GSM8K [65] and MATH [115], EAI substantially improves performance on challenging reasoning tasks, outperforming both human supervision and AI supervision baselines. In contrast to human supervision, our approach is autonomous and more scalable. When compared to prior state-of-the-art AI supervision baselines including RFT [327] and WizardMath [194], our method provides a

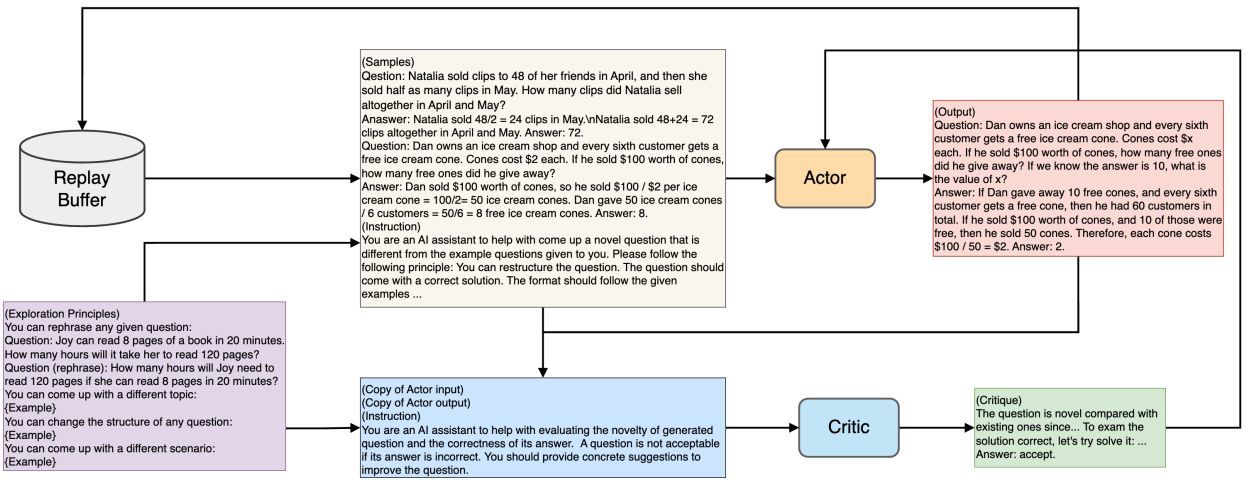


Figure 9.2 Generating diverse data in the Exploratory AI Framework. In the diagram, we demonstrate how the actor generates diverse content by conditioning on samples from the replay buffer and exploration principles. These principles include rephrasing question, coming up a novel topic, restructuring question, and coming up a new scenario, we provide examples associated with the principles to guide exploration. The actor's input and its generated output undergo evaluation by the critic. The critic assesses the novelty of the generated data; when the evaluation is favorable, the data is stored in the replay buffer. In cases where the evaluation does not meet the criteria, the critic provides critiques to guide the actor. The replay buffer can be initialized with a pre-existing human-created dataset (*e.g.*, GSM8K training set) or can remain empty for starting from scratch with zero-shot exploration.

straightforward yet highly effective solution for the generation of high-quality and diverse data.

Our contributions are two-fold: (a) In contrast to the predominant reliance on human supervision, our novel approach, EAI, leverages the capabilities of large language models to autonomously generate diverse high-quality training data. It achieves this by harnessing these models for self-guided exploration, inspired by unsupervised reinforcement learning pretraining. (b) We conduct an extensive series of experiments to systematically assess the effectiveness of EAI. Our results show that EAI substantially outperform prior human supervision and AI supervision state-of-the-arts, and significantly improve model performance.

9.2 Exploratory AI for Diverse AI Supervision

We present our approach for harnessing AI models to create AI supervision, in order to address the reliance on extensive human supervision.

Our method employs a dynamic interplay between generation and evaluation. This concept draws inspiration from unsupervised RL pretraining (URL) [154] and particularly the APT algorithm [179]. RL pretraining studies exploring in a reward-free environment to develop skills for quickly maximizing various downstream rewards. APT allows training RL agent to learn skills by autonomously exploring a reward free environment based on evaluating novelty of encountered states using particle based entropy estimation [27, 275].

Adapting APT directly to large language models presents several challenges, including grappling with computational complexity and the difficulty of learning reward functions and exploration policies [92, 65]. In response, we propose Exploratory AI (EAI), a novel approach that circumvents the need for direct reinforcement learning (RL) by harnessing the power of large language models for exploration. Our method explore the natural language space by employing these language models to assess the novelty of generated content and guide the exploration process. Our approach consists of two key components: an "actor" responsible for generating novel content and a "critic" that evaluates the actor's outputs and provides feedback to guide further content generation.

Concretely, we instruct the actor to generate content that diverges from samples from the replay buffer. The replay buffer can be initialized with a pre-existing human-created dataset (e.g., GSM8K training set) or can remain empty for zero-shot exploration. Similar to APT, we found having pre-existing samples accelerates learning and encourages the actor to have more long term exploratory behaviors. We then instruct the critic to assess the actor's outputs and provides critiques. This feedback loop guides the actor in refining and enhancing its content. This iterative process continues until it reaches a predefined maximum number of iterations, and the resulting outputs are stored in a dataset. The data can then be used for finetuning AI models.

Actor prompt

You are an AI assistant to help with come up a novel question that is different from the example questions given to you. The question should come with a correct solution. Please follow the given principle in generating the question. **{principle}**

Critic prompt

You are an AI assistant to help with evaluating the novelty of generated question and the correctness of its answer. A question is not acceptable if its answer is incorrect. You should provide concrete suggestions to improve the question. Explain your reasoning step by step and output final evaluation on novelty and correctness at the end. Follow the given principle on evaluating the novelty. **{principle}**

We equip both the actor and critic with a curated set of guiding principles to facilitate the generation and evaluation of diverse questions. These principles include rephrasing question, coming up a novel topic, restructuring question, and coming up a new scenario, we provide examples associated with the principles to guide exploration.

Principles for exploration

You can rephrase any given question:
Question: Joy can read 8 pages of a book in 20 minutes. How many hours will it take her to read 120 pages?
Question (rephrase): How many hours will Joy need to read 120 pages if she can read 8 pages in 20 minutes?
You can come up with a different topic:
Question: Jack is stranded on a desert island. He wants some salt to season his fish. He collects 2 liters of seawater in an old bucket. If the water is 20% salt, how many ml of salt will Jack get when all the water evaporates?
Question (topic): Samantha is designing a circular garden in her backyard. The garden has a diameter of 8 meters. She wants to build a path around the garden that is 1 meter wide. What is the area of the path, in square meters, that Samantha will need to pave with stones or concrete?
You can change the structure of any question:
Question: Dan owns an ice cream shop and every sixth customer gets a free ice cream cone. Cones cost $2 each. If he sold $100 worth of cones, how many free ones did he give away?
Question (restructured): Dan owns an ice cream shop and every sixth customer gets a free ice cream cone. Cones cost $x each. If he sold $100 worth of cones, how many free ones did he give away? If we know the answer is 10, what is the value of x?
You can come up with a different scenario:
Question: Ed has 2 dogs, 3 cats and twice as many fish as cats and dogs combined. How many pets does Ed have in total?
Question (scenario): Sarah owns 4 bicycles, 2 skateboards, and three times as many pairs of rollerblades as bicycles and skateboards combined. How many wheeled sports equipment items does Sarah have in total?

While it's theoretically possible to provide all these principles to the model, in this study, we opt to a more controlled approach. To balance the quantity of generated data for each principle, we uniformly sample one principle at a time and input it to both the actor and critic. The actor is instructed to follow the principle (*e.g.*, restructuring the question) during question generation. Similarly, the critic's role is to evaluate the diversity, considering the selected principle. It's worth noting that the critic's principle is worded slightly differently from the exploration principle; for a detailed list, please refer to Appendix 9.8. Our method is shown in Figure 9.2 and the algorithm is shown in Algorithm 6.

Exploratory AI has several attractive properties as an approach for facilitating AI supervision in language models:

1. EAI can generate diverse AI supervision for learning, independently of human input, making it more scalable compared with supervised finetuning or rejection sampling based on human curated data.

2. EAI provides an interpretable window into the behavior and knowledge of the model. It sheds light on how well the model possesses knowledge and its reasoning behind generating novel questions. One can look at generations and their corresponding evaluations which provide valuable insights about how model generates and evaluates.

3. EAI's versatility allows for a fusion of the best aspects of supervised finetuning and

Algorithm 6: Exploratory AI for diverse AI supervision.

Required: One (or two) large language models M for actor and critic.
Replay Buffer B, empty or optionally initialized with pre-existing data.
Initialize
for $i = 1$ **to** max iterations **do**
  Randomly sample data points from B
  Use preassigned principle or sample one principle.
  for $i = j$ **to** max rounds **do**
    Prompt the actor with the principle to generate content (a question and its answer) that in the same domain but diverge from the sampled inputs (questions and answers) sampled from B
    Prompt the critic with the principle to evaluate the diversity of generated question and correctness of answer, and decide whether to accept
    if Accepted **then**
      Save generated question and answer to B
      break
    else
      Continue to prompt actor with the critique as additional input
    end if
  end for
end for

prompting. Users can prompt the model to focus on certain topics or aspects by directing actor and critic with different prompting strategies.

4. EAI demonstrates its effectiveness by excelling in mathematical reasoning tasks, as we will demonstrate in our experiments. Moreover, its capabilities are not limited to mathematics; it holds promise for a broad spectrum of language-related tasks in principle.

In empirical experiments, we will evaluate the utility of EAI for mathematical reasoning and analysis its effectiveness.

9.3 Setting

We evaluate our method on the mathematical reasoning tasks, and achieve better results that EAI largely improve results and significantly outperforms prior state-of-the-arts.

Benchmarks. We evaluate our method on the mathematical reasoning tasks GSM8K and MATH. GSM8K exams model's mathematical reasoning capabilities, we finetune model on the training split, and evaluate model on the test split. The GSM8k dataset includes around 7,500 training and 1,319 test math problems for high school-level students, involving basic

arithmetic operations. Problems typically require 2 to 8 steps for a solution. The MATH dataset comprises 7,500 training and 5,000 challenging test problems from prestigious math competitions (AMC 10, AMC 12, AIME) covering various academic domains, including prealgebra, algebra, number theory, counting and probability, geometry, intermediate algebra, and precalculus.

Baselines. We compare our approach with (a) Base model including Vicuna 7B, 13B, and 30B [61]. Vicuna is LLaMA2 finetuned on user conversations shared online (ShareGPT). We use Vicuna as base model for all baselines and our method; (b) Supervised finetuning (SFT) on training set of the original GSM8K or MATH, in which a language model is finetuned on human written exemplars of questions–answers pairs. SFT has been widely used in prior works for improving language models mathematical reasoning [165, 297, 217, *inter alia*] and following user intention [97**?** , *inter alia*]. We also compare with WizardMath [194] which does SFT on ChatGPT annotated questions and solutions, as well as MAmmoTH [329] which uses GPT4 annotated solutions; (c) Rejection sampling finetuning (RFT) [327] which applies supervised finetuning on rejection sampled model generated data. We provide baseline scores for SFT and RFT from both their original papers and our implementations using Vicuna, ensuring a fair and comprehensive comparison; (d) Proprietary models including GPT-4 [217], ChatGPT [253], and Claude2 [12]. All baselines are evaluated by prompting them to output step by step reasoning followed by final answers [318].

Generated data size. We sample roughly the same amount of data for each principle outlined in Section 9.2. To optimize computational cost, we have set the number of interaction rounds in Algorithm **??** to a maximum of two. Our preliminary experiments revealed that this two-round interaction is typically sufficient for the actor to produce high-quality and diverse data. For each of the four principles – 'rephrase question', 'introduce a new topic', 'restructure the question', and 'introduce a new scenario' – we generate approximately 25,000 samples for GSM8K and approximately 15,000 samples for MATH. The generation on 8 A100 80GB GPUs take from 40 to 200 hours depending on the model size and the specific principles applied.

9.4 Results

Benchmarks on math reasoning. The results presented in the Table 9.1 demonstrate the notable effectiveness of the method EAI in the context of pass@1 performance on the GSM8k and MATH datasets. A key highlight is the absolute improvement of EAI over previous state-of-the-art methods, emphasized in bold red numbers. Specifically, EAI exhibits superior performance in two distinct scenarios: firstly, when ChatGPT is used to supervise LLaMA2, and secondly, when Vicuna supervises itself. The table provides a comprehensive comparison across various models and methods, including GPT-4, ChatGPT, Claude 2, LLaMA2 (with and without SFT and RFT), and Vicuna. The performance of EAI, especially in the LLaMA2 and Vicuna settings, shows marked improvements in both the GSM8K and

Table 9.1 Results of pass@1 (%) on GSM8k and MATH. In this study, to ensure equitable and cohesive evaluations, we report the scores of all models under the same settings of greedy decoding. Bold numbers in red are the absolute improvement of EAI over prior state-of-the-arts. Notably, EAI outperforms state-of-the-arts both when using ChatGPT for exploration to supervise LLaMA2 and when using Vicuna to supervise itself.

Finetune Model	Method	AI supervision	Data amount	Params	GSM8K	MATH
-	GPT-4	-	-	-	92.0	42.5
	ChatGPT	-	-	-	80.8	34.1
	Claude 2	-	-	-	88.0	32.5
LLaMA2	LLaMA2	-	-	7B	14.6	2.5
				13B	28.7	3.9
	SFT	-	7.5K	7B	41.6	-
				13B	50.0	-
	RFT	LLaMA2	47K	7B	47.5	5.6
				13B	54.8	9.6
	WizardMath	ChatGPT	96K	7B	54.9	10.7
				13B	63.9	14.0
	MAmmoTH	GPT4	260K	7B	51.7	31.2
				13B	61.7	36.0
	EAI	GPT4	96K	7B	**56.6** (+1.7)	**11.6** (+0.9)
				13B	**65.2** (+1.3)	**15.1** (+1.1)
Vicuna	Vicuna	-	-	7B	24.4	2.6
				13B	39.8	5.8
	SFT	-	7.5K	7B	42.0	4.6
				13B	50.8	7.9
	RFT	Vicuna	48K	7B	48.1	5.9
				13B	56.3	9.3
	EAI	Vicuna	48K	7B	**52.9** (+4.8)	**8.6** (+2.7)
				13B	**60.5** (+4.2)	**11.4** (+2.1)

MATH datasets. For instance, in the LLaMA2 model, EAI achieves a significant gain in both datasets, irrespective of the number of parameters (7B or 13B). This trend is consistent in the Vicuna model as well, where EAI again shows superior performance compared to the baseline Vicuna model. These results underscore the efficacy of EAI in leveraging AI supervision to enhance model performance. The gains are prominent across model sizes, indicating EAI's scalability and effectiveness in handling complex AI models. This result on the GSM8k and MATH datasets provides compelling evidence of the effectiveness of generating diverse AI supervision by EAI to enhance complex problem-solving tasks.

Comparison of diversity. We evaluate EAI in terms of the diversity of generated data. We compare RFT and EAI in terms of the submodularity diversity gain [32, 208]. This metric serves as an indicator of the extent to which the generated data contribute to the overall diversity of the dataset. A higher diversity gain suggests that the newly generated

questions exhibit greater dissimilarity from the existing dataset. We measure the gain over GSM8K training set by varying the amount of generated content. We use OpenAI GPT embedding `text-embedding-ada-002` to encode the data. The results of diversity gain and t-SNE are depicted in Figure 9.3 demonstrate that EAI consistently outperforms RFT in terms of diversity, thereby providing a more diverse set of generated data.

Effect of sampled inputs. The Table 9.2 presents the results of an experiment examining the impact of varying the number of samples on GSM8K and MATH. As the number of samples increases from 0 to 8, we observe a steady incremental improvement on both GSM8K and MATH. On GSM8K, the performance rises from 50.1 to 52.9. On MATH, the effect is more pronounced. These results suggest that increasing the number of samples has a positive effect on both GSM8K and MATH, highlighting the significance of larger sample size for in context exploration.

Table 9.2 Effect of different number of samples from replay buffer.

Number	0	1	4	8
GSM8K	50.1	50.8	51.9	**52.9**
MATH	6.6	7.1	7.5	**8.6**

Scaling with generated data. We assess the performance of EAI in terms of sample efficiency on the GSM8K dataset. Our primary focus lies in understanding how the results evolve in response to varying amounts of generated data. Sample efficiency holds paramount importance, given that autoregressive data generation is inefficient. Enhanced sample efficiency broadens the practical utility of our approach in real-world applications. The results depicted in Figure 9.4 clearly illustrate a significant advantage for EAI over the previous state-of-the-art RFT. Notably, as more data is employed, RFT exhibits improved performance, but its sample efficiency lags behind EAI by a substantial margin. At just 16K data points, EAI outperforms RFT's performance at 48K data points, achieving more than a 3x higher level of sample efficiency.

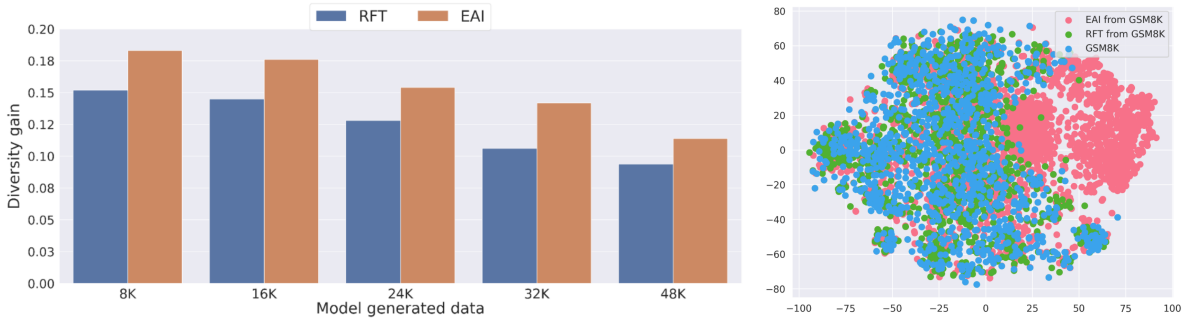


Figure 9.3 (Left): Comparison of diversity gains achieved by adding generated data to the GSM8K training set. EAI outperforms other baselines in terms of diversity. (Right): t-SNE comparison of human-curated GSM8K, RFT, and EAI-generated outputs, depicting embeddings of questions.

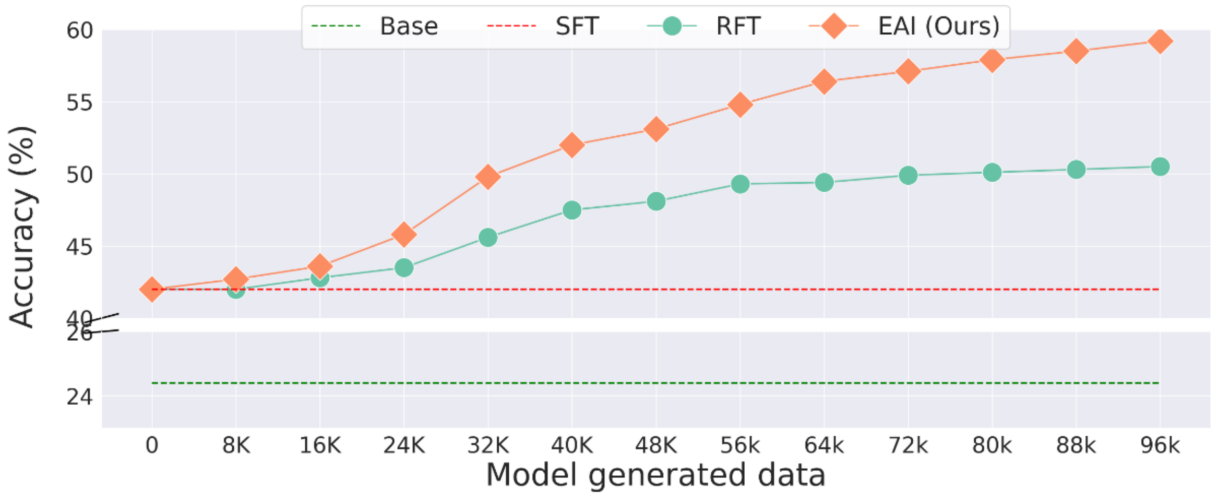


Figure 9.4 Data scaling on GSM8K. Shown are GSM8K accuracy with different amount of generated data. EAI generates high quality data for learning and scales well with data.

Table 9.3 Effect of different exploration principles on GSM8K and MATH.

rephrase	new topic	restructure	new scenario	GSM8K	MATH
✓	✓	✓	✓	**52.9**	**8.6**
✗	✓	✓	✓	48.8	7.1
✓	✗	✓	✓	49.7	7.8
✓	✓	✗	✓	48.9	6.9
✓	✓	✓	✗	49.5	7.5
✓	✗	✗	✗	48.1	6.3
✗	✓	✗	✗	47.6	6.0
✗	✗	✓	✗	48.5	6.2
✗	✗	✗	✓	47.8	6.3

Evaluating the effect of exploration principles. The results of varying exploration principles, as shown in Table 9.3, reveal some interesting insights. When all principles are in place (✓ for rephrase, new topic, restructure, and new scenario), the model performs at its best on GSM8K and MATH. This suggests that using all principles simultaneously leads to the most favorable outcomes. Among the principles, the most critical ones appear to be "rephrase" and "restructure", as seen when one of them is removed (✗). Without "rephrase" the performance drops on both datasets, emphasizing that the ability to rephrase and generate diverse content is crucial. Similarly, the omission of "restructure" leads to a significant drop in MATH scores, highlighting the significance of introducing novel question-structuring approaches for solving more challenging problems.

Evaluating the effect of critic. We remove the critic from the framework and evaluate its performance. To ensure a fair comparison, we follow the same procedure as with the default EAI and generate an equal amount of data. We evaluate on GSM8K, MATH, MBPP, and HumanEval. All methods are based on LLaMA2 [297]. We compare EAI and EAI w/o critic with Self-instruct [314] and RFT [327]. Self-instruct involves a loop of generation and filtering which resembles EAI's actor-critic, except that EAI comes with principles guided in context exploration. Comparing EAI w/o critic with Self-instruct shows the effectiveness of the principles guided in context exploration component in EAI framework. Comparing EAI with EAI w/o critic can show the importance of the critic. RFT relies on sampling-based exploration; comparing EAI w/o critic with it shows the effectiveness of in-context exploration and comparing EAI with it further reveals the importance of the critic. The results in Table 9.4 show that EAI w/o critic significantly outperforms all baselines, showing the effectiveness of principles guided in context exploration. Adding the critic back to the EAI framework further substantially improves the results, achieving significantly better results than Self-instruct, RFT, and EAI w/o critic. A qualitative analysis provided in Appendix 9.7 reveals how the critic aids in guiding exploration. Both quantitative and qualitative results show that the critic is important for achieving the best exploration; removing the critic leads to substantially lower results.

Table 9.4 Evaluation of the effectiveness of critic.

	LLaMA2	Self-Instruct	RFT	EAI w/o critic	**EAI**
GSM8K (%)	14.6	43.4	47.5	50.5	**52.9**
MATH (%)	2.5	3.9	5.6	7.1	**8.6**
MBPP (%)	26.1	33.7	41.8	42.5	**44.6**
HumanEval (%)	11.9	12.8	13.9	14.5	**15.8**

Scaling with human annotation size. Figure 9.5 illustrates the results obtained when utilizing varying amounts of human annotation data from the GSM8K training set. We employ three different approaches in our experiments: SFT which directly finetunes the base model, Vicuna-7B, on the provided data. RFT which leverages the provided data to perform rejection sampling from the model. EAI which utilizes the provided data to initialize a replay buffer and explore new content for training. The results consistently demonstrate that EAI significantly outperforms all the baseline methods across various levels of human annotation data, underscoring its efficacy in generating high-quality training data. Remarkably, our experiments reveal that EAI performs admirably even in the absence of any human annotations, hinting at the potential to entirely eliminate the need for human intervention in the process.

Benchmark on code generation. Having evaluated the effectiveness of EAI in improving mathematical reasoning, we further evaluate its application in a different domain: code

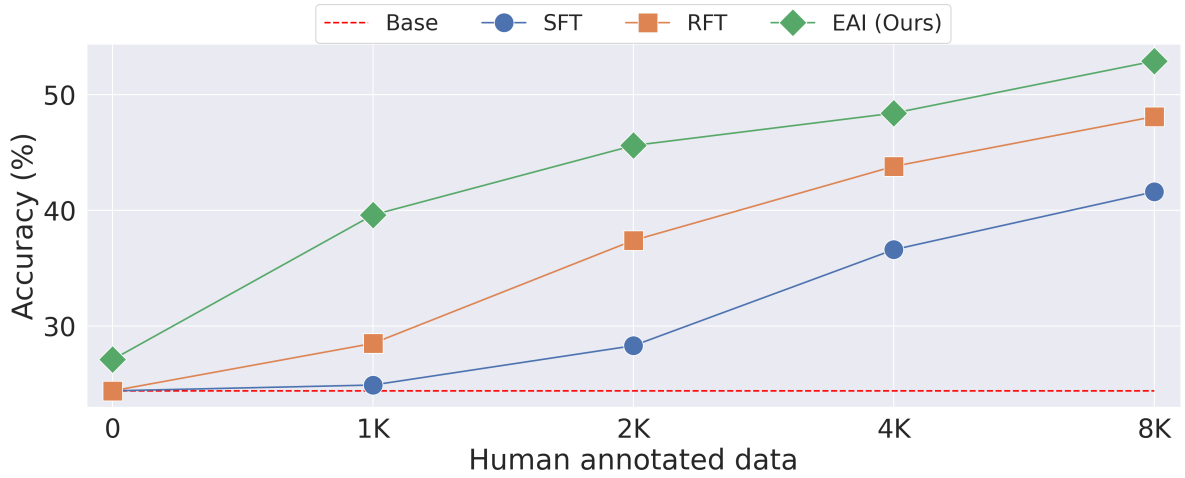


Figure 9.5 Performance on GSM8K with different amount of human annotated data. EAI performs well even without human annotation and scales well with more human provided annotations.

Table 9.5 Evaluations on code generation tasks. LLaMA2 and CodeLLaMA are pretrained models. SFT, RFT, EAI are trained using MBPP training split. All methods are evaluated using MBPP test split, and HumanEval dataset. Red numbers are absolute increase compared with best performing baselines.

	LLaMA2	LLaMA2+SFT	LLaMA2+RFT	LLaMA2+**EAI**
MBPP (%)	26.1	38.5	41.8	**44.6** (+2.8)
HumanEval	11.9	13.2	13.9	**16.2** (+2.3)
	CodeLLaMA	CodeLLaMA+SFT	CodeLLaMA+RFT	CodeLLaMA+**EAI**
MBPP (%)	52.5	54.3	55.6	**58.9** (+3.3)
HumanEval	31.1	33.5	36.1	**39.5** (+3.4)

generation. Unlike the math reasoning task, where the model generates the reasoning path and final answer, in code generation the model needs to output a Python program in response to a given question, such as 'Write a function to convert degrees to radians.' Specifically, we apply EAI to the MBPP task [13], following the same procedure as before. This task comprises around 1,000 crowd-sourced Python programming problems designed to be solvable by entry-level programmers. It covers programming fundamentals, standard library functionality, and more. Each problem includes a task description, a code solution, and three automated test cases. We collected 18,000 examples using EAI and RFT based on the MBPP training split, respectively. We then compared EAI with supervised finetuning as well as RFT on the training split. Subsequently, we evaluated the models on the MBPP test split and conducted

zero-shot evaluation on HumanEval [51]. For evaluation, we used LLaMA2 [297], which is pretrained on text, and CodeLLaMA [248], which is further pretrained on code, representing different model choices. Table 9.5 shows the results of 0-shot HumanEval and 3-shot MBPP. EAI substantially outperforms the baselines and significantly improves both LLaMA2 and CodeLLaMA. This confirms the effectiveness of EAI in exploration and in improving code generation performance.

9.5 Related Work

Transformers [304] trained using next token prediction have gave rise to many state-of-the-art AI systems [253, 217]. The remarkable AI results achieved with this generative AI approach heavily hinge upon the availability of diverse and high-quality data. For instance, state-of-the-art AI models including ChatGPT [253] and GPT4 [217] along with a range of other open source models such as Vicuna, Koala, and Dolly [**?** 97, 61, *inter alia*], require extensive finetuning through human demonstrations. This process involves human conversations with ChatGPT or written demonstrations, demanding significant human involvement and domain expertise. Previous research has explored various avenues to enhance performance and sample efficiency, as well as alternative sources of supervision. To align with human preferences, there has been active research into developing simple algorithms for learning from human preferences [186, 328, 76, 297, *inter alia*]. In contrast to human demonstrations or feedback, another line of work explores the utilization of environmental feedback, such as unit test errors [157, 59, 268], which has demonstrated improved results in coding tasks, or using using LLMs to provide AI supervision based exploration techniques for applications in solving games [78, 307] and demonstrate improved results. Furthermore, some prior research leveraged proprietary APIs to indirectly obtain high-quality human data, enhancing model capabilities in areas like instruction following [314, 320, 287, 97, 61] and mathematical reasoning [194, 207, 329, *inter alia*]. Another line of research explores the use of models to supervise themselves [283, 199, 127, 18, 327], yielding improved results in reasoning tasks and alignment with human preferences. Our work focuses on generating diverse and high-quality data using AI models and we demonstrate applying our proposed approach to enhance open-source models by having them self-generate learning data. Our approach's exploration technique is related to unsupervised RL based exploration [276, 178, 179, 44, 225, 209, 81, 222, 241, *inter alia*], however, our method does not require training RL agent directly and relies on LLMs to explore. Additionally, some works have delved into more detailed forms of human supervision [174], demonstrating that LLMs benefit more from step-by-step process-based supervision than sparse outcome-based supervision. Our research centers on the data dimension, with a specific emphasis on harnessing AI models to generate diverse high-quality AI supervision. To this end, we introduce an actor-critic based approach based with human provided principles for automating the exploration process.

9.6 Conclusion

In this work we propose an approach to automatically generate diverse, high-quality data from AI models. Our approach Exploratory AI consists of prompting an actor model to generate diverse contents that are different from existing contents, and using a critic model for evaluating the novelty of generated data and providing critiques to guide the exploration process. Experimental evaluations confirms the effectiveness of EAI, demonstrating its capacity to generate diverse content and substantially enhance model performance on GSM8K and MATH datasets.

In terms of future prospects, our approach of generating diverse content with AI models opens up interesting possibilities, such as extending EAI to evaluate novelty across the entire data buffer, employing either a brute force approach (evaluating all data through a critic) or employing embedding similarity search techniques. Moreover, there's potential in extending EAI to incorporate multiple actor and critic models, or in experimenting with various strategies to enhance exploration. It would also be interesting to apply our method to proprietary APIs to source even more diverse and higher-quality data.

9.7 Case Study of EAI

We conduct a qualitative study how actor and critic accomplish exploration. On GSM8K, we manually inspect 100 examples randomly sampled from EAI explored dataset to better understand how does the exploration work. We observed that the critic excels at assessing the novelty of generated questions, providing precise feedback to enhance them. The actor effectively utilizes the critic's feedback to refine the generated questions. Of the 100 samples reviewed, in 24 instances, the critic initially found the questions lacking in novelty and provided suggestions for improvement. Out of these, 19 were subsequently refined and accepted after revisions. Below is a representative example of this interaction where the critic offers advice to significantly differentiate the generated question. Here, the critic recommends incorporating the mathematical concept of volumes and presenting an alternative scenario

> `[Critic guides exploration]`
> `(First round)`
> `Actor's generation:` Question: Jake is planning to buy a new skateboard which costs $200. Jake only has a quarter of the money he requires. His brother decided to give him $20 for it, and his uncle three times as much as his brother. How much more does Jake need to raise to purchase the skateboard? (answer omitted)
>
> `Critic's critique:` The new question you've provided is somewhat similar in structure to the existing questions, as it involves basic arithmetic operations (multiplication, division, addition, and subtraction). While it introduces a different context with hats and balls, the underlying math principles are similar to those used in the existing questions. To make the new question substantially different, you could consider introducing more complex mathematical concepts such as volumes or using a different scenario altogether. ✗
>
> `(Second round)`
>
> `Actor's generation:` An artist is creating a sculpture using various geometric shapes. The sculpture consists of a cube, a sphere, a cylinder, and a pyramid. The cube has a side length of 6 units, the sphere has a radius of 4 units, the cylinder has a height of 8 units and a base radius of 3 units, and the pyramid has a base edge length of 5 units and a height of 7 units. Calculate the total volume of the artist's sculpture. A: omitted
>
> `Critic's critique:` The new question is substantially different from the existing questions. It involves a slightly more complex problem-solving process. It is good to be added to math questions. The decision is accept. ✓

We also conducted a study to determine how effective the critic is at evaluating the correctness of answers. Of the 100 critiques assessed, the critic accurately identified incorrectness in 13 actor generations and offered critiques to correct 10 of them. However, there were 7 instances where the critic failed to recognize the errors. The remaining actor generations were accurate. It's worth noting that, due to the uniform usage of the four exploration principles,

out of the 100 actor generations, 50 were either paraphrased questions or questions with new scenarios, these 50 were less prone to errors as they didn't necessitate significant changes to the mathematical structure. In fact, only 2 solutions among them were incorrect. On the other hand, those that involved restructuring or tackled new topics were more error-prone: 7 from restructuring and 4 from new topics were incorrect. Intriguingly, the critic corrected 6 out of the 7 restructured ones, but only 1 out of the 4 new topics. This might suggest that questions on new topics are more open-ended and might need multiple rounds of interactions for accurate evaluation. Further refinement in this verification process will be the focus of our future work.

Our analysis revealed that the critic excels in evaluation rather than generation. This observation is exemplified in the quoted example, it can be seen that even when the model cannot directly answer the question, it can effectively assess the correctness of the actor's response. This discovery is intriguing, as the actor's response effectively serves as a 'hint'. This observation aligns with prior works that found models are better at evaluating answer (or improving an answer) than generating it from scratch in various settings [59, 199, 186, 65, *inter alia*].

> `[Critic cannot solve question directly]`
> `Input:` A magician has 5 hats, each with a different number of balls inside. The first hat has 3 balls, the second hat has twice as many balls as the first, the third hat has half as many balls as the second, the fourth hat has 5 balls, and the fifth hat has 4 times as many balls as the fourth. How many balls are in the magician's hats altogether?
>
> `Output:` There are 3 + 2 * 3 = 7 balls in the first two hats. There are 5 - 2 = 3 balls in the third hat. There are 4 * 3 = 12 balls in the fourth and fifth hats. In total, there are 7 + 3 + 12 = 22 balls in the magician's hats. Answer: 22. ✗

> `[Critic can evaluate actor's answer]`
> `Input:` Please review whether the answer for the following question is correct. Q: A magician has 5 hats, each with a different number of balls inside. The first hat has 3 balls, the second hat has twice as many balls as the first, the third hat has half as many balls as the second, the fourth hat has 5 balls, and the fifth hat has 4 times as many balls as the fourth. How many balls are in the magician's hats altogether? A: There are 3 + 2 x 3/2 = 3 + 3 = 6 balls in the first two hats. There are 5 - 3/2 = 5 - 1.5 = 3.5 balls in the third hat. There are 4 x 5 = 20 balls in the fourth hat. In total, there are 6 + 3.5 + 20 = 29.5 balls. Since you can't have a fraction of a ball, you would need to round up to the nearest whole number of balls. Therefore, there are 29 balls in the magician's hats altogether.
>
> `Output:` The magician has 5 hats, each with a different number of balls inside. The first hat has 3 balls, the second hat has twice as many balls as the first, so it has 3 * 2 = 6 balls. The third hat has half as many balls as the second, so it has 6 / 2 = 3 balls. The fourth hat has 5 balls. The fifth hat has 4 times as many balls as the fourth, so

it has 5 * 4 = 20 balls. In total, there are 3 + 6 + 3 + 5 + 20 = 37 balls. Since you can't have a fraction of a ball, you would need to round up to the nearest whole number of balls. Therefore, there are 37 balls in the magician's hats altogether. The answer provided is incorrect. ✓

9.8 Prompt

The critic's principle is based on the exploration principle but is worded slightly differently from the actor's exploration principle, in order to tell critic what kind of new questions are diverse and acceptable. The critic will use the principle to evaluate diversity of questions and evaluate the correctness of answer.

Principles for critique

A different topic is acceptable:
Question: Jack is stranded on a desert island. He wants some salt to season his fish. He collects 2 liters of seawater in an old bucket. If the water is 20% salt, how many ml of salt will Jack get when all the water evaporates?
Question (topic): Samantha is designing a circular garden in her backyard. The garden has a diameter of 8 meters. She wants to build a path around the garden that is 1 meter wide. What is the area of the path, in square meters, that Samantha will need to pave with stones or concrete?
A question with different structure is acceptable:
Question: Dan owns an ice cream shop and every sixth customer gets a free ice cream cone. Cones cost $2 each. If he sold $100 worth of cones, how many free ones did he give away?
Question (restructured): Dan owns an ice cream shop and every sixth customer gets a free ice cream cone. Cones cost $x each. If he sold $100 worth of cones, how many free ones did he give away? If we know the answer is 10, what is the value of x?
Rephrased question is acceptable:
Question: Joy can read 8 pages of a book in 20 minutes. How many hours will it take her to read 120 pages?
Question (rephrase): How many hours will Joy need to read 120 pages if she can read 8 pages in 20 minutes?
A different scenario is acceptable:
Question: Ed has 2 dogs, 3 cats and twice as many fish as cats and dogs combined. How many pets does Ed have in total?
Question (scenario): Sarah owns 4 bicycles, 2 skateboards, and three times as many pairs of rollerblades as bicycles and skateboards combined. How many wheeled sports equipment items does Sarah have in total?

9.9 Experiment Details

We use a temperature of 0.7 for the actor during exploration as in prior work [65], and we sample 10 actor generations for every batch of samples from the replay buffer. We

use a temperature of 0.0 for the critic since we found that it performs best. Following prior work [327], we filter out reasoning paths with incorrect answers or calculations—based on Python evaluation—for the 'paraphrasing' and 'new scenarios' exploration categories. However, we do not apply this filter to the 'restructuring' or 'new topics' exploration categories, as we do not have ground truth answers for these two categories. The evaluations for all baselines and our approach are conducted with deterministic sampling following prior work and report maj1@1 (accuracy) across all experiments. We follow prior work by conducting evaluations using deterministic sampling for both our approach and the baseline methods. We report maj1@1 accuracy across all experimental setups. All models are trained with the same hyperparameters: global batch size = 128, learning rate = 2e-5, epochs = 3, sequence length = 2048. The training is done with 8x A100 80GB GPUs.

Chapter 10

Conclusion and Future Work

In this dissertation, our goal is to develop a machine capable of learning everything – that includes learning from any existing data and discovering to go beyond existing data.

We show a series of research in Chapter 2, Chapter 3, Chapter 4, and Chapter 5 that fully resolve the challenges of limited context size, enabling AI to learn from any existing data.

We show our research on discovering in Chapter 6, Chapter 7, Chapter 8, and Chapter 9 that enables AI to learn without human specifying domain knowledge, paving the road for discovering and going beyond existing data.

Looking ahead, I plan to focus on advances in efficient scaling, reasoning, and discovering in general domains.

10.1 Powerful reasoning and efficient scaling

Despite advancements allow handling of nearly infinite context sizes, akin to a Turing machine's infinitely large tape, current AI systems exhibit significant limitations in their ability to utilize this context for effective step-by-step reasoning. This shortfall is crucial, as it restricts the practical applicability of AI across various domains. Unlike the theoretical capabilities of a Turing machine, which can perform extended computations over its tape, AI systems struggle to translate the extensive data they can access into complex, logical sequences of thought or action. Regarding scaling, the need for an efficient scaling paradigm cannot be overstated – this is because the current scaling trend demands exponential amount of compute and data for achieving tiny linear amount of improvement; a new paradigm is required for achieving much more efficient reasoning.

10.2 Generating data and learning in general domains

Current AI systems rely heavily on large, high-quality datasets, which are costly, may miss certain scenarios, and can become outdated quickly. To address these challenges, there's increasing focus on creating autonomous learning frameworks that allow AI to generate its own data through simulations or interactions with the environment. I aim to build on my previous research on discovering in gameplay to advance discovering and learning in more general domains. This could not only lead to a self-improving loop but also enable the discovery of new knowledge that humans wouldn't be able to achieve.

Bibliography

[1] Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

[2] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[3] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice, 2021.

[4] Arize AI. Needle in a haystack - pressure testing llms. https://github.com/Arize-ai/LLMTest_NeedleInAHaystack, 2023.

[5] Emanuele Aiello, Lili Yu, Yixin Nie, Armen Aghajanyan, and Barlas Oguz. Jointly training large autoregressive multimodal models. *arXiv preprint arXiv:2309.15564*, 2023.

[6] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[7] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

[8] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R. Devon Hjelm. Unsupervised state representation learning in atari. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8766–8779, 2019.

[9] Kapathy Andrej. GitHub - karpathy/nanoGPT: The simplest, fastest repository for training/finetuning medium-sized GPTs. — github.com. https://github.com/karpathy/nanoGPT, 2023. [Accessed 16-May-2023].

[10] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5048–5058, 2017.

[11] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? a large-scale empirical study. *arXiv preprint arXiv: Arxiv-2006.05990*, 2020.

[12] Anthropic. Introducing claude, 2023. URL https://www.anthropic.com/index/introducing-claude.

[13] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[14] Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, et al. Openflamingo: An open-source framework for training large autoregressive vision-language models. *arXiv preprint arXiv:2308.01390*, 2023.

[15] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[16] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 507–517. PMLR, 2020.

[17] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[18] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[19] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1728–1738, 2021.

[20] David Barber and Felix V. Agakov. The im algorithm: A variational approach to information maximization. In *Advances in neural information processing systems*, 2003.

[21] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1606.05312*, 2016.

[22] André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, David Silver, and Hado van Hasselt. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4055–4065, 2017.

[23] Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 501–510. PMLR, PMLR, 2018.

[24] André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087, 2020.

[25] Andrew G Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 17–47. Springer, 2013.

[26] Andrew G Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*, pages 112–19. Piscataway, NJ, 2004.

[27] J Beirlant. Nonparametric entropy estimation: An overview. *International Journal of the Mathematical Statistics Sciences*, 6:17–39, 1997.

[28] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[29] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[30] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. *arXiv preprint arXiv:2102.08602*, 2021.

[31] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[32] Jeff Bilmes. Submodularity in machine learning and artificial intelligence. *arXiv preprint arXiv:2202.00132*, 2022.

[33] Christian Bischof. *Parallel computing: Architectures, algorithms, and applications*, volume 15. IOS Press, 2008.

[34] Diana Borsa, André Barreto, John Quan, Daniel J. Mankowitz, Hado van Hasselt, Rémi Munos, David Silver, and Tom Schaul. Universal successor features approximators. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[35] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[36] William Brandon, Aniruddha Nrusimha, Kevin Qian, Zachary Ankner, Tian Jin, Zhiye Song, and Jonathan Ragan-Kelley. Striped attention: Faster ring attention for causal transformers. *arXiv preprint arXiv:2311.09431*, 2023.

[37] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[38] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

[39] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[40] Yuri Burda, Harrison Edwards, Deepak Pathak, Amos J. Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[41] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*. OpenReview.net, 2019.

[42] Minwoo Byeon, Beomhee Park, Haecheon Kim, Sungjun Lee, Woonhyuk Baek, and Saehoon Kim. Coyo-700m: Image-text pair dataset. https://github.com/kakaobrain/coyo-dataset, 2022.

[43] Victor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giró-i-Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1317–1327. PMLR, 2020.

[44] Víctor Campos, Pablo Sprechmann, Steven Hansen, Andre Barreto, Steven Kapturowski, Alex Vitvitskyi, Adria Puigdomenech Badia, and Charles Blundell. Beyond fine-tuning: Transferring behavior in reinforcement learning. *arXiv preprint arXiv:2102.13515*, 2021.

[45] Víctor Campos, Pablo Sprechmann, Steven Stenberg Hansen, Andre Barreto, Charles Blundell, Alex Vitvitskyi, Steven Kapturowski, and Adria Puigdomenech Badia. Coverage as a principle for discovering transferable behavior in reinforcement learning, 2021.

[46] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, 2020.

[47] Micah Carroll, Orr Paradise, Jessy Lin, Raluca Georgescu, Mingfei Sun, David Bignell, Stephanie Milani, Katja Hofmann, Matthew Hausknecht, Anca Dragan, et al. Unimask: Unified inference in sequential decision problems. *arXiv preprint arXiv:2211.10869*, 2022.

[48] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *CoRR*, abs/2106.01345, 2021. URL https://arxiv.org/abs/2106.01345.

[49] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[50] Lin Chen, Jisong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. Sharegpt4v: Improving large multi-modal models with better captions. *arXiv preprint arXiv:2311.12793*, 2023.

[51] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[52] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 34:5834–5847, 2021.

[53] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

[54] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[55] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 2020.

[56] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 2020.

[57] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[58] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022.

[59] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*, 2023.

[60] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

[61] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna.lmsys.org*, 2023.

[62] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[63] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[64] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[65] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[66] Zichen Jeff Cui, Yibin Wang, Nur Muhammad, Lerrel Pinto, et al. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

[67] W Dai, J Li, D Li, AMH Tiong, J Zhao, W Wang, B Li, P Fung, and S Hoi. Instructblip: towards general-purpose vision-language models with instruction tuning. arxiv. *Preprint posted online on June*, 15:2023, 2023.

[68] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[69] Anthony Danalis, Ki-Yong Kim, Lori Pollock, and Martin Swany. Transformations to parallel codes for communication-computation overlap. In *SC'05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, pages 58–58. IEEE, 2005.

[70] Anthony Danalis, Lori Pollock, Martin Swany, and John Cavazos. Mpi-aware compiler optimizations for improving communication-computation overlap. In *Proceedings of the 23rd international conference on Supercomputing*, pages 316–325, 2009.

[71] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[72] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[73] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

[74] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, pages 4171–4186, 2018. doi: 10.18653/v1/N19-1423.

[75] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

[76] Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

[77] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[78] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. *arXiv preprint arXiv:2302.06692*, 2023.

[79] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[80] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[81] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[82] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

[83] Facebook. Fully Sharded Data Parallel: faster AI training with fewer GPUs — engineering.fb.com. https://engineering.fb.com/2021/07/15/open-source/fsdp/, 2023. [Accessed 16-May-2023].

[84] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

[85] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

[86] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *International Conference on Learning Representations*. OpenReview.net, 2018.

[87] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[88] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

[89] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1582–1591. PMLR, 2018.

[90] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. In *European Conference on Computer Vision*, pages 89–106. Springer, 2022.

[91] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[92] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.

[93] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, et al. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv preprint arXiv:2304.15010*, 2023.

[94] Xinyang Geng and Hao Liu. Openllama: An open reproduction of llama. *URL: https://github. com/openlm-research/open_ llama*, 2023.

[95] Xinyang Geng and Hao Liu. Openllama: An open reproduction of llama, may 2023. *URL https://github. com/openlm-research/open_ llama*, 2023.

[96] Xinyang Geng, Hao Liu, Lisa Lee, Dale Schuurams, Sergey Levine, and Pieter Abbeel. Multimodal masked autoencoders learn transferable representations. *arXiv preprint arXiv:2205.14204*, 2022.

[97] Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. Koala: A dialogue model for academic research. *Blog post, April*, 1, 2023.

[98] Andrew Gibiansky. Bringing hpc techniques to deep learning. *Baidu Research, Tech. Rep.*, 2017.

[99] gkamradt. Needle in a haystack - pressure testing llms. https://github.com/gkamradt/LLMTest_NeedleInAHaystack/tree/main, 2023. [Online; accessed 7-Feb-2024].

[100] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus, 2019. URL http://Skylion007.github.io/OpenWebTextCorpus.

[101] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[102] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. In *International Conference on Learning Representations*, 2017.

[103] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33:1474–1487, 2020.

[104] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[105] Pierre-Louis Guhur, Shizhe Chen, Ricardo Garcia, Makarand Tapaswi, Ivan Laptev, and Cordelia Schmid. Instruction-driven history-aware policies for robotic manipulations. *arXiv preprint arXiv:2209.04899*, 2022.

[106] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/gutmann10a.html.

[107] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR, 2018.

[108] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[109] Steven Hansen, Will Dabney, André Barreto, David Warde-Farley, Tom Van de Wiele, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[110] Steven Hansen, Will Dabney, André Barreto, David Warde-Farley, Tom Van de Wiele, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. In *International Conference on Learning Representations*, 2020.

[111] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[112] Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2681–2691. PMLR, 2019.

[113] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735. IEEE, 2020. doi: 10.1109/CVPR42600.2020.00975.

[114] Olivier J. Hénaff. Data-efficient image recognition with contrastive predictive coding. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4182–4192. PMLR, 2020.

[115] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[116] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3215–3222. AAAI Press, 2018.

[117] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[118] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[119] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[120] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022.

[121] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[122] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv: Arxiv-2203.15556*, 2022.

[123] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Curiosity-driven exploration in deep reinforcement learning via bayesian neural networks. 2016.

[124] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[125] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[126] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In *International Conference on Machine Learning*, pages 9099–9117. PMLR, 2022.

[127] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022.

[128] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training

of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

[129] Joshua Hursey and Richard L Graham. Building a fault tolerant mpi application: A ring communication example. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pages 1549–1556. IEEE, 2011.

[130] Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*, 2023.

[131] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021.

[132] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34: 1273–1286, 2021.

[133] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4): 620, 1957.

[134] Jiantao Jiao, Weihao Gao, and Yanjun Han. The nearest neighbor information estimator is adaptively near minimax rate-optimal. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3160–3171, 2018.

[135] Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Chia-Yuan Chang, and Xia Hu. Growlength: Accelerating llms pretraining by progressively growing training length. *arXiv preprint arXiv:2310.00576*, 2023.

[136] Yang Jin, Kun Xu, Liwei Chen, Chao Liao, Jianchao Tan, Bin Chen, Chenyi Lei, An Liu, Chengru Song, Xiaoqiang Lei, et al. Unified language-vision pretraining with dynamic discrete visual tokenization. *arXiv preprint arXiv:2309.04669*, 2023.

[137] Ryan Julian, Benjamin Swanson, Gaurav S Sukhatme, Sergey Levine, Chelsea Finn, and Karol Hausman. Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning. *arXiv e-prints*, pages arXiv–2004, 2020.

[138] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873): 583–589, 2021.

[139] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993.

[140] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[141] Kacper Kielak. Do recent advancements in model-based deep reinforcement learning really improve data efficiency? *arXiv preprint arXiv:2003.10181*, 2020.

[142] Changyeon Kim, Younggyo Seo, Hao Liu, Lisa Lee, Jinwoo Shin, Honglak Lee, and Kimin Lee. Guide your agent with adaptive multimodal rewards. *Advances in Neural Information Processing Systems*, 36, 2024.

[143] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6980.

[144] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[145] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[146] Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055*, 2022.

[147] Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. *arXiv preprint arXiv:2205.05198*, 2022.

[148] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[149] Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016.

[150] Taehwan Kwon. Variational intrinsic control revisited. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=P0p33rgyoE.

[151] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. arXiv:2004.14990, 2020.

[152] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, 2020.

[153] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5639–5650. PMLR, 2020.

[154] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. Urlb: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv:2110.15191*, 2021.

[155] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. Urlb: Unsupervised reinforcement learning benchmark, 2021. URL https://openreview.net/forum?id=lwrPkQP_is.

[156] Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

[157] Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328, 2022.

[158] Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

[159] Kuang-Huei Lee, Ofir Nachum, Mengjiao Yang, Lisa Lee, Daniel Freeman, Winnie Xu, Sergio Guadarrama, Ian Fischer, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *arXiv preprint arXiv:2205.15241*, 2022.

[160] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[161] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric P. Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *CoRR*, abs/1906.05274, 2019.

[162] Joel Lehman and Kenneth O Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[163] Joel Lehman and Kenneth O Stanley. Novelty search and the problem with objectives. In *Genetic programming theory and practice IX*, pages 37–56. Springer, 2011.

[164] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[165] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.

[166] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Jingkang Yang, and Ziwei Liu. Otter: A multi-modal model with in-context instruction tuning. *arXiv preprint arXiv:2305.03726*, 2023.

[167] Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E. Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can open-source llms truly promise on context length?, June 2023. URL https://lmsys.org/blog/2023-06-29-longchat.

[168] Dacheng Li, Rulin Shao, Anze Xie, Eric P Xing, Joseph E Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. Lightseq: Sequence level parallelism for distributed training of long context transformers. *arXiv preprint arXiv:2310.03294*, 2023.

[169] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022.

[170] KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023.

[171] Shenggui Li, Jiarui Fang, Zhengda Bian, Hongxin Liu, Yuliang Liu, Haichen Huang, Boxiang Wang, and Yang You. Colossal-ai: A unified deep learning system for large-scale parallel training. *arXiv preprint arXiv:2110.14883*, 2021.

[172] Shenggui Li, Fuzhao Xue, Chaitanya Baranwal, Yongbin Li, and Yang You. Sequence parallelism: Long sequence training from system perspective. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2391–2404, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.134. URL https://aclanthology.org/2023.acl-long.134.

[173] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[174] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[175] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[176] Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023.

[177] Fangchen Liu, Hao Liu, Aditya Grover, and Pieter Abbeel. Masked autoencoding for scalable and generalizable decision making. *arXiv preprint arXiv:2211.12740*, 2022.

[178] Hao Liu and Pieter Abbeel. Aps: Active pre-training with successor features. In *International Conference on Machine Learning*, pages 6736–6747. PMLR, 2021.

[179] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. In *Advances in Neural Information Processing Systems*, 2021.

[180] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *arXiv preprint arXiv:2103.04551*, 2021.

[181] Hao Liu and Pieter Abbeel. APS: active pretraining with successor features. In *International Conference on Machine Learning*, 2021.

[182] Hao Liu and Pieter Abbeel. Emergent agentic transformer from chain of hindsight experience. *International Conference on Machine Learning*, 2023.

[183] Hao Liu and Pieter Abbeel. Blockwise parallel transformer for large context models. *Advances in neural information processing systems*, 2023.

[184] Hao Liu, Lisa Lee, Kimin Lee, and Pieter Abbeel. Instruction-following agents with jointly pre-trained vision-language models. *arXiv preprint arXiv:2210.13431*, 2022.

[185] Hao Liu, Tom Zahavy, Volodymyr Mnih, and Satinder Singh. Palm up: Playing in the latent manifold for unsupervised pretraining. *Advances in Neural Information Processing Systems*, 35:35880–35893, 2022.

[186] Hao Liu, Carlo Sferrazza, and Pieter Abbeel. Chain of hindsight aligns language models with feedback. *arXiv preprint arXiv:2302.02676*, 2023.

[187] Hao Liu, Wilson Yan, and Pieter Abbeel. Language quantized autoencoders: Towards unsupervised text-image alignment. *Advances in neural information processing systems*, 2023.

[188] Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. World model on million-length video and language with ringattention. *arXiv preprint arXiv:2402.08268*, 2024.

[189] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. *International Conference on Learning Representations(ICLR)*, 2024.

[190] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023.

[191] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[192] Xiaoran Liu, Hang Yan, Shuo Zhang, Chenxin An, Xipeng Qiu, and Dahua Lin. Scaling laws of rope-based extrapolation. *arXiv preprint arXiv:2310.05209*, 2023.

[193] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[194] Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

[195] Ruipu Luo, Ziwang Zhao, Min Yang, Junwei Dong, Minghui Qiu, Pengcheng Lu, Tao Wang, and Zhongyu Wei. Valley: Video assistant with large language model enhanced ability. *arXiv preprint arXiv:2306.07207*, 2023.

[196] Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. Mega: moving average equipped gated attention. *arXiv preprint arXiv:2209.10655*, 2022.

[197] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*, 2023.

[198] Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5125–5133. AAAI Press, 2020.

[199] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

[200] Luckeciano C Melo. Transformers are meta-reinforcement learners. In *International Conference on Machine Learning*, pages 15340–15359. PMLR, 2022.

[201] Maxim Milakov and Natalia Gimelshein. Online normalizer calculation for softmax. *arXiv preprint arXiv:1805.02867*, 2018.

[202] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[203] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[204] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529, 2015.

[205] Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2125–2133, 2015.

[206] MosaicML. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. URL https://www.mosaicml.com/blog/mpt-7b.

[207] Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.

[208] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023. URL http://probml.github.io/book2.

[209] Mirco Mutti, Lorenzo Pratissoli, and Marcello Restelli. A policy gradient method for task-agnostic exploration. *arXiv preprint arXiv:2007.04640*, 2020.

[210] Sharan Narang, Hyung Won Chung, Yi Tay, William Fedus, Thibault Fevry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong Lan, et al. Do transformer modifications transfer across implementations and applications? *arXiv preprint arXiv:2102.11972*, 2021.

[211] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. Pipedream: Generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 1–15, 2019.

[212] Deepak Narayanan, Amar Phanishayee, Kaiyu Shi, Xie Chen, and Matei Zaharia. Memory-efficient pipeline-parallel dnn training. In *International Conference on Machine Learning*, pages 7937–7947. PMLR, 2021.

[213] Ansong Ni, Jeevana Priya Inala, Chenglong Wang, Alex Polozov, Christopher Meek, Dragomir Radev, and Jianfeng Gao. Learning math reasoning from self-sampled correct and partially-correct solutions. In *The Eleventh International Conference on Learning Representations*, 2022.

[214] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[215] OpenAI. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018.

[216] OpenAI. Solving rubik's cube with a robot hand. *ArXiv*, abs/1910.07113, 2019.

[217] OpenAI. Gpt-4 technical report, 2023.

[218] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. *arXiv preprint arXiv:2303.06349*, 2023.

[219] Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2721–2730. PMLR, 2017.

[220] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[221] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

[222] Seohong Park, Kimin Lee, Youngwoon Lee, and Pieter Abbeel. Controllability-aware unsupervised skill discovery. *arXiv preprint arXiv:2302.05103*, 2023.

[223] Keiran Paster, Sheila A McIlraith, and Jimmy Ba. Planning from pixels using inverse dynamics models. *arXiv preprint arXiv:2012.02419*, 2020.

[224] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.

[225] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2778–2787. PMLR, PMLR, 2017.

[226] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International Conference on Machine Learning*, 2019.

[227] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5062–5071. PMLR, 2019.

[228] Suraj Patil, William Berman, Robin Rombach, and Patrick von Platen. amused: An open muse reproduction. *arXiv preprint arXiv:2401.01808*, 2024.

[229] Xue Bin Peng, P. Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37:143:1–143:14, 2018.

[230] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202.

[231] Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 7750–7761. PMLR, 2020.

[232] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866*, 2023.

[233] Ben Poole, Sherjil Ozair, Aäron van den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5171–5180. PMLR, 2019. URL http://proceedings.mlr.press/v97/poole19a.html.

[234] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[235] Markus N Rabe and Charles Staats. Self-attention does not need o(n2) memory. *arXiv preprint arXiv:2112.05682*, 2021.

[236] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[237] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[238] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[239] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

[240] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[241] Sai Rajeswar, Pietro Mazzaglia, Tim Verbelen, Alexandre Piché, Bart Dhoedt, Aaron Courville, and Alexandre Lacoste. Mastering the unsupervised reinforcement learning benchmark from pixels. 2023.

[242] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[243] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *International Conference on Machine Learning*, pages 8844–8856. PMLR, 2021.

[244] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506, 2020.

[245] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*, 2012.

[246] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[247] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3674–3683, 2020.

[248] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

[249] Kiersten M Ruff and Rohit V Pappu. Alphafold and implications for intrinsically disordered proteins. *Journal of Molecular Biology*, 433(20):167208, 2021.

[250] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1312–1320. JMLR.org, 2015.

[251] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

[252] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

[253] J. Schulman, B. Zoph, C. Kim, J. Hilton, J. Menick, J. Weng, J. F. C. Uribe, L. Fedus, L. Metz, M. Pokorny, R. G. Lopes, S. Zhao, A. Vijayvergiya, E. Sigler, A. Perelman, C. Voss, M. Heaton, J. Parish, D. Cummings, R. Nayak, V. Balcom, D. Schnurr, T. Kaftan, C. Hallacy, N. Turley, N. Deutsch, and V. Goel. Chatgpt: Optimizing language models for dialogue. *OpenAI Blog*, 2022. URL https://openai.com/blog/chatgpt.

[254] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1889–1897. JMLR.org, 2015.

[255] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

[256] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[257] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *Advances in Neural Information Processing Systems*, 2017.

[258] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021.

[259] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration. In *International Conference on Machine Learning*, 2021.

[260] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. *arXiv preprint arXiv:2206.14244*, 2022.

[261] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

[262] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *arXiv preprint arXiv: Arxiv-2206.11251*, 2022.

[263] Dhruv Shah, Blazej Osinski, Brian Ichter, and Sergey Levine. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. *arXiv preprint arXiv:2207.04429*, 2022.

[264] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[265] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020.

[266] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[267] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[268] Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.

[269] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[270] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[271] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[272] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Artfhur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[273] Özgür Simsek and Andrew G. Barto. An intrinsic reward mechanism for efficient exploration. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 833–840. ACM, 2006. doi: 10.1145/1143844.1143949.

[274] Harshinder Singh, Neeraj Misra, Vladimir Hnizdo, Adam Fedorowicz, and Eugene Demchuk. Nearest neighbor estimates of entropy. *American Journal of Mathematical and Management Sciences*, 23(3-4):301–321, 2003.

[275] Harshinder Singh, Neeraj Misra, Vladimir Hnizdo, Adam Fedorowicz, and Eugene Demchuk. Nearest neighbor estimates of entropy. *American journal of mathematical and management sciences*, 23(3-4):301–321, 2003.

[276] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[277] Aravind Srinivas and Pieter Abbeel. Unsupervised learning for reinforcement learning, 2021. URL https://icml.cc/media/icml-2021/Slides/10843_QHaHBNU.pdf.

[278] Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

[279] Tom Stepleton. The pycolab game engine. https://github.com/deepmind/pycolab, 2017.

[280] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, 2021.

[281] DJ Strouse, Kate Baumli, David Warde-Farley, Vlad Mnih, and Steven Hansen. Learning more skills through optimistic exploration. *CoRR*, abs/2107.14226, 2021. URL https://arxiv.org/abs/2107.14226.

[282] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[283] Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision. *arXiv preprint arXiv:2305.03047*, 2023.

[284] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[285] Ruo Yu Tao, Vincent François-Lavet, and Joelle Pineau. Novelty search in representational space for sample efficient exploration. *arXiv preprint arXiv:2009.13579*, 2020.

[286] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

[287] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

[288] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[289] Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, and Nicolas Heess. dm_control: Software and tasks for continuous control. *arXiv preprint arXiv:2006.12983*, 2020.

[290] Yi Tay, Mostafa Dehghani, Samira Abnar, Hyung Won Chung, William Fedus, Jinfeng Rao, Sharan Narang, Vinh Q Tran, Dani Yogatama, and Donald Metzler. Scaling laws vs model architectures: How does inductive bias influence scaling? *arXiv preprint arXiv:2207.10551*, 2022.

[291] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28, 2022.

[292] Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

[293] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[294] Emanuel Todorov. General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*, pages 4286–4292. IEEE, 2008.

[295] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 1049–1056. ACM, 2009. doi: 10.1145/1553374.1553508.

[296] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar,

et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[297] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[298] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkxoh24FPH.

[299] Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. Focused transformer: Contrastive training for context scaling. *arXiv preprint arXiv:2307.03170*, 2023.

[300] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, pages 2094–2100. AAAI Press, 2016.

[301] Hado van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? *arXiv preprint arXiv:1906.05243*, pages 14322–14333, 2019.

[302] Hado P van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. *Advances in Neural Information Processing Systems*, 29:4287–4295, 2016.

[303] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[304] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017.

[305] Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual description. *arXiv preprint arXiv:2210.02399*, 2022.

[306] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[307] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[308] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[309] Junxiong Wang, Jing Nathan Yan, Albert Gu, and Alexander M Rush. Pretraining without attention. *arXiv preprint arXiv:2212.10544*, 2022.

[310] Shibo Wang, Jinliang Wei, Amit Sabne, Andy Davis, Berkin Ilbeyi, Blake Hechtman, Dehao Chen, Karthik Srinivasa Murthy, Marcello Maggioni, Qiao Zhang, et al. Overlap communication with dependent computation via decomposition in large deep learning models. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pages 93–106, 2022.

[311] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[312] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

[313] Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, et al. Internvid: A large-scale video-text dataset for multimodal understanding and generation. *arXiv preprint arXiv:2307.06942*, 2023.

[314] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[315] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards, 2018.

[316] David Warde-Farley, Tom Van de Wiele, Tejas D. Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[317] Christopher Watkins. Learning from delayed rewards. 01 1989.

[318] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[319] Philipp Wu, Arjun Majumdar, Kevin Stone, Yixin Lin, Igor Mordatch, Pieter Abbeel, and Aravind Rajeswaran. Masked trajectory models for prediction, representation, and control. *arXiv preprint arXiv:2305.02968*, 2023.

[320] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

[321] Yuanzhong Xu, HyoukJoong Lee, Dehao Chen, Blake Hechtman, Yanping Huang, Rahul Joshi, Maxim Krikun, Dmitry Lepikhin, Andy Ly, Marcello Maggioni, et al. Gspmd: general and scalable parallelization for ml computation graphs. *arXiv preprint arXiv:2105.04663*, 2021.

[322] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021.

[323] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. *arXiv preprint arXiv:2102.11271*, 2021.

[324] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, 2021.

[325] Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

[326] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5, 2022.

[327] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

[328] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.

[329] Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

[330] Tom Zahavy, Andre Barreto, Daniel J Mankowitz, Shaobo Hou, Brendan O'Donoghue, Iurii Kemaev, and Satinder Baveja Singh. Discovering a set of policies for the worst case reward, 2021.

[331] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35: 15476–15488, 2022.

[332] Shuangfei Zhai, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and Josh Susskind. An attention free transformer. *arXiv preprint arXiv:2105.14103*, 2021.

[333] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023.

[334] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023.

[335] Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts. *arXiv preprint arXiv:2110.01786*, 2021.

[336] Rui Zhao, Yang Gao, Pieter Abbeel, Volker Tresp, and Wei Xu. Mutual information state intrinsic control. *arXiv preprint arXiv:2103.08107*, 2021.

[337] Kaizhi Zheng, Xiaotong Chen, Odest Chadwicke Jenkins, and Xin Wang. Vlmbench: A compositional benchmark for vision-and-language manipulation. *Advances in Neural Information Processing Systems*, 35:665–678, 2022.

[338] Lianmin Zheng, Zhuohan Li, Hao Zhang, Yonghao Zhuang, Zhifeng Chen, Yanping Huang, Yida Wang, Yuanzhong Xu, Danyang Zhuo, Eric P Xing, et al. Alpa: Automating inter-and {Intra-Operator} parallelism for distributed deep learning. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 559–578, 2022.

[339] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.

[340] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[341] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[342] Simiao Zuo, Qingru Zhang, Chen Liang, Pengcheng He, Tuo Zhao, and Weizhu Chen. Moebert: from bert to mixture-of-experts via importance-guided adaptation. *arXiv preprint arXiv:2204.07675*, 2022.