TTV-HRM: A Hierarchical Reasoning Architecture for Efficient Text-to-Video Generation

Ahsan Umar
GitHub: codewithdark-git Hugging Face: codewithdark
https://github.com/codewithdark-git/TTV-HRM

***Abstract*—Text-to-video generation is typically inaccessible to most researchers due to its reliance on large-scale models and multi-GPU infrastructure. We introduce the Text-to-Video Hierarchical Reasoning Model (TTV-HRM), a lightweight framework that enables coherent text-conditioned video synthesis on a single commodity GPU. The model employs interleaved hierarchical reasoning in which a high-level transformer captures global semantic structure while a low-level layer refines spatiotemporal details through bidirectional cross-attention. A learned convergence predictor enables early stopping, reducing average inference iterations from three to 2.1 without quality loss. The 115M-parameter system integrates rotary positional embeddings, SwiGLU feed-forward layers, and a 3D convolutional video autoencoder, training in about four hours on a single NVIDIA T4 GPU at roughly $2 cloud cost, with sub-second inference per clip. On 8-frame 32×32 video generation, TTV-HRM improves frame-wise Fréchet Inception Distance from 120.5 to 62.1 across three epochs using only 45 video–text pairs. Results demonstrate semantic alignment, temporal coherence, and object persistence, showing that hierarchical reasoning can substitute for model scale to make text-to-video research more accessible.**

***Index Terms*—Text-to-Video Generation, Hierarchical Reasoning, Efficient Transformers, Spatiotemporal Modeling, Video Tokenization, Resource-Constrained Learning**

I. Introduction

Text-to-video (T2V) generation represents one of the most demanding frontiers in multimodal artificial intelligence. The task requires a model to understand the semantic meaning of a natural language description and then synthesize a temporally coherent sequence of visual frames that faithfully reflects that description. This capability carries significant potential across a wide range of real-world domains, including automated content creation, educational simulation, interactive storytelling, film prototyping, and human-computer interaction [19, 27, 62]. Despite this promise, the path from research prototype to practical deployment remains obstructed by a fundamental barrier: the extraordinary computational cost of current state-of-the-art systems.

Modern T2V models are predominantly built on one of two dominant paradigms: large-scale latent diffusion models [30, 20, 57] or autoregressive transformer architectures [24, 59]. Both paradigms have delivered impressive results in terms of visual quality and temporal coherence, but they share a common weakness — they require massive computational infrastructure. Typical training runs demand multi-GPU or even multi-node clusters equipped with high-end accelerators such as NVIDIA A100 or H100 GPUs, tens to hundreds of gigabytes of memory, and weeks of wall-clock time. At inference, even a single video generation pass can require several seconds on hardware that costs thousands of dollars to access. This computational barrier effectively excludes independent researchers, small academic groups, students, and developers in resource-limited settings from meaningful participation in T2V research and application.

This exclusion is not a minor inconvenience. It concentrates the development of a powerful and transformative technology within a small number of well-funded organizations, reducing the diversity of ideas and slowing the field's overall progress. There is therefore a clear and pressing need for T2V frameworks that are not only capable but also computationally accessible — systems that can run, train, and be meaningfully experimented with on widely available commodity hardware such as the NVIDIA T4 GPU, which is broadly accessible through standard cloud platforms at modest cost.

In this work, we present the **Text-to-Video Hierarchical Reasoning Model (TTV-HRM)**, a lightweight and computationally efficient framework designed specifically to address this gap. Our design is informed by the Hierarchical Reasoning Model (HRM) paradigm [6], adapted here for text-to-video generation under strict compute constraints. TTV-HRM performs coherent T2V generation on a single NVIDIA T4 GPU, producing short video clips of 8 frames at 32×32 pixel resolution with a peak memory footprint of approximately 480 MB and inference times under one second. The central design principle of TTV-HRM is *hierarchical reasoning*: rather than attempting to generate a complete video sequence in a single forward pass, the model decomposes the generation process into a structured sequence of progressive refinement stages. This decomposition is inspired by well-established theories of hierarchical visual processing in human cognition, where perception proceeds from coarse global structure toward fine-grained local detail [13, 17]. By mimicking this principle, TTV-HRM first establishes the broad semantic and structural layout of the video content and then iteratively sharpens spatial and temporal detail, all while remaining tightly conditioned on the input text prompt through cross-attention mechanisms.

The model is built on a foundation of well-proven architectural components: transformer blocks [1, 62] equipped with rotary positional embeddings (RoPE) [7] for robust sequence modeling, SwiGLU feed-forward networks [8, 57] for improved training stability and representational capacity, and a 3D convolutional autoencoder [35] for compact spatiotemporal video tokenization. To these established components, TTV-

HRM adds a novel *interleaved high-level and low-level reasoning mechanism* that enables efficient iterative refinement of the latent video representation without incurring the computational cost of deeper single-pass architectures. An additional lightweight convergence predictor allows the model to terminate refinement early once the latent representation has stabilized, further reducing unnecessary computation during inference.

TTV-HRM is trained on a streaming subset of a curated video-text dataset and serves as a proof-of-concept demonstrating that conceptually meaningful T2V generation is achievable without massive parameter counts or compute budgets. The full system, comprising approximately 115 million parameters, can be trained in roughly four hours on a single T4 instance at a cloud cost of approximately $2, making it genuinely accessible to a wide community of researchers and practitioners. While the current implementation is deliberately scoped to low-resolution, short-duration output, the hierarchical reasoning framework is inherently scalable and provides a clear pathway toward higher-fidelity generation as additional compute becomes available.

The remainder of this paper is organized as follows. Section I-A summarizes the key technical contributions of this work. Section III provides a high-level overview of the TTV-HRM system pipeline. Section II reviews the relevant background and related work. Sections IV through IV-C describe the model architecture in detail. Section V presents the training procedure and evaluation methodology. Sections V and VI cover implementation details and experimental results. Section VII interprets findings, discusses limitations, and outlines future directions, followed by concluding remarks in Section VIII.

A. Key Contributions

TTV-HRM introduces four targeted technical contributions that, taken together, make text-to-video generation meaningfully accessible on constrained hardware. Each contribution addresses a specific practical obstacle that has historically limited participation in T2V research.

Hierarchical Reasoning for Iterative Video Refinement. The central innovation of TTV-HRM is a dual-layer iterative reasoning mechanism that decomposes the video generation process into two complementary levels of abstraction. A high-level (H) transformer layer operates on the global structure of the latent video sequence, capturing coarse semantic layout, object identity, and long-range temporal relationships. A low-level (L) transformer layer operates concurrently on the same latent sequence, attending to fine-grained spatial patterns and local motion dynamics. These two layers interact through bidirectional cross-attention: top-down guidance from the H layer ensures that local details produced by the L layer remain semantically coherent, while bottom-up feedback from the L layer allows the H layer to adapt its global representation as fine details emerge. This interaction is repeated for up to three iterative steps and is consistently conditioned on the input text prompt throughout, ensuring that the progressive refinement process remains faithful to the original description. This hierarchical design yields measurably better temporal consistency and semantic alignment compared to flat, non-hierarchical generation architectures of equivalent parameter count.

Resource-Constrained Optimization for Commodity Hardware. Every component of the TTV-HRM architecture has been explicitly co-designed with the constraints of the NVIDIA T4 GPU in mind. The system leverages TensorFloat-32 (TF32) mixed-precision arithmetic to accelerate matrix-intensive operations with negligible accuracy loss, CuDNN-optimized kernel selection for 3D convolutions and attention operations, gradient checkpointing across transformer blocks to reduce peak memory consumption by approximately 40% at the cost of a modest 20% increase in computation [48], and streaming data loaders with in-memory caching to minimize disk I/O bottlenecks. These measures collectively enable a compact 115-million-parameter model with a 480 MB checkpoint footprint to complete a full training run in approximately four hours and to perform inference in under one second on hardware that costs roughly $0.50 per hour on standard cloud platforms.

Learned Early Stopping via Convergence Prediction. To avoid wasting computation on unnecessary refinement iterations, TTV-HRM incorporates a lightweight auxiliary convergence predictor — a two-layer multilayer perceptron (MLP) with a sigmoid output — that estimates whether the current latent video representation has sufficiently stabilized after each reasoning step. The predictor takes as input a compact summary of the latent state and, during training, is supervised by a binary convergence signal derived from the relative change in reconstruction error between successive iterations. At inference, when the predicted convergence confidence exceeds a threshold of 0.5, refinement terminates early. In practice, this mechanism reduces the average number of reasoning steps from the maximum of three to approximately 2.1–2.3, yielding a meaningful reduction in inference cost without any degradation in output quality.

Modular and Extensible System Architecture. TTV-HRM is structured as a collection of independent, interchangeable modules: a compact text encoder, a 3D convolutional video tokenizer and detokenizer, the hierarchical reasoning engine, and an evaluation utilities module. This modularity allows each component to be studied, replaced, or improved in isolation without requiring changes to the rest of the system. It also facilitates targeted ablation studies — for example, replacing the hierarchical reasoning engine with a flat transformer baseline or swapping the 3D convolutional tokenizer for a different spatiotemporal representation. Looking forward, the modular design provides a clear interface for future extensions such as higher-resolution output, longer video sequences, audio integration, or the incorporation of diffusion-based refinement in latent space.

II. Related Work

This section reviews the key areas of prior work that directly inform the design and motivation of TTV-HRM. We organize the discussion into five themes: text-to-video

generation, efficient generative models, hierarchical and multi-scale architectures, spatiotemporal video representations, and transformer efficiency techniques.

A. Text-to-Video Generation

The task of generating video from text descriptions has evolved rapidly over the past several years, driven by advances in both generative modeling and large-scale multimodal datasets. Early approaches to video generation relied on generative adversarial networks (GANs) [28, 29], which demonstrated the feasibility of synthesizing short clips but struggled with temporal coherence, training instability, and generalization to diverse text conditions. Recurrent architectures such as convolutional LSTMs [39] were also explored for modeling temporal dynamics, but their sequential nature limited scalability to longer sequences.

The introduction of transformer-based autoregressive models brought a significant leap forward. VideoGPT [24] demonstrated that discrete video tokens, produced by a VQ-VAE [36], could be modeled autoregressively using a GPT-style decoder to generate temporally coherent clips. NUWA [26] and CogVideo [25] extended this paradigm to text-conditioned generation by training on large text-video paired datasets, showing that language descriptions could meaningfully guide autoregressive video synthesis. However, these methods require very large model capacities and extensive training data to produce visually convincing results.

Diffusion-based models have since become the dominant paradigm for high-quality video generation. Video Diffusion Models [22] extended the denoising diffusion probabilistic framework [31] from images to video by modeling joint space-time distributions. Make-A-Video [19] demonstrated that knowledge from text-to-image diffusion models could be transferred to video generation without requiring text-video paired training data, achieving impressive visual quality. Imagen Video [21] and VideoCrafter [20] further pushed quality through cascaded diffusion pipelines and high-resolution refinement stages. More recently, Stable Video Diffusion [23] has shown that latent diffusion models can be fine-tuned effectively for video synthesis with strong temporal consistency.

Despite these advances, all of the above methods share a critical limitation from the perspective of this work: they require substantial computational resources. Training these models typically demands dozens to hundreds of high-end GPUs over days or weeks, and even inference requires multi-gigabyte GPU memory allocations that exceed the capacity of commodity hardware. TTV-HRM addresses this gap directly by pursuing a lightweight autoregressive framework that achieves meaningful T2V results on a single T4 GPU, making the problem accessible to a much broader research community.

B. Efficient and Lightweight Generative Models

A growing body of work has explored how to reduce the computational cost of generative models without sacrificing quality. In the image domain, techniques such as latent diffusion [30] moved the denoising process from pixel space into a compressed latent space, drastically reducing the spatial resolution at which diffusion operates and enabling high-quality generation on single consumer GPUs. MobileStyleGAN [58] and TinyGAN [56] demonstrated that GAN-based image generators could be substantially compressed through knowledge distillation and architecture search while retaining visual fidelity.

In the video domain, efficiency has been pursued through several strategies. Temporal factorization [59] separates spatial and temporal modeling into distinct sub-networks, reducing joint space-time complexity. Token pruning and sparse attention mechanisms [38] reduce the effective sequence length processed by transformers at each step. Lightweight video autoencoders have been designed to compress video into compact latent representations [37], enabling transformer-based generation to operate on a much smaller number of tokens.

TTV-HRM contributes to this line of work by combining a compact 3D convolutional autoencoder for efficient spatiotemporal tokenization with a hierarchical reasoning engine that limits the number of transformer forward passes through learned early stopping. Together, these choices yield a system that requires fewer than 4 GB of GPU memory during inference, placing it within reach of widely available cloud and local hardware.

C. Hierarchical and Multi-Scale Architectures

The principle of organizing computation into hierarchical levels of abstraction has a long history in both vision and generative modeling. In visual recognition, hierarchical convolutional networks such as AlexNet [41] and VGG [42] demonstrated that stacking layers of increasing receptive field naturally builds representations from edges and textures at lower levels to objects and scenes at higher levels. Feature pyramid networks [15] and U-Net architectures [16] formalized multi-scale processing by explicitly constructing representations at multiple spatial resolutions and combining them through skip connections.

In generation, hierarchical and cascaded approaches have been widely adopted to improve quality and efficiency. Progressive growing of GANs [18] demonstrated that generating images first at low resolution and then progressively increasing resolution yields more stable training and sharper results. Cascaded diffusion models [33] apply a similar idea by using a sequence of separate diffusion models operating at increasing resolutions. DALL-E 2 [34] uses a hierarchical pipeline of CLIP embeddings, a diffusion prior, and a decoder, separating semantic and perceptual levels of generation.

In the context of video transformers, hierarchical vision transformers such as Swin Transformer [14] and the architecture studied by Pan et al. [13] demonstrated that multi-scale attention, organized into local windows at fine scales and global attention at coarse scales, improves both accuracy and computational efficiency. TTV-HRM draws directly on these insights but applies the hierarchical principle in a novel way: rather than operating at multiple spatial resolutions, it interleaves high-level and low-level transformer blocks

that process the *same* latent sequence at different levels of abstraction, with bidirectional guidance between levels. This avoids the memory cost of maintaining multiple resolution representations while still achieving the semantic-to-detail refinement benefit of hierarchical processing.

D. Spatiotemporal Video Representations

Effective video generation depends critically on how video data is represented and compressed. Early work by Tran et al. [35] showed that 3D convolutional networks, which apply learned filters jointly across space and time, capture spatiotemporal features more effectively than networks that process spatial and temporal dimensions separately. This finding motivated the widespread adoption of 3D convolutions as a foundation for video understanding and has more recently informed video generative models that require compact latent representations of video clips.

Vector-quantized video autoencoders, such as those used in VideoGPT [24] and MAGVIT [37], compress video into discrete token sequences suitable for autoregressive modeling. While powerful, discrete tokenization introduces quantization artifacts and complicates gradient flow during joint training. Continuous latent representations, as used in latent diffusion [30] and in our work, avoid these artifacts and allow smooth gradient-based optimization throughout the entire pipeline.

TTV-HRM employs a continuous 3D convolutional autoencoder that compresses 8-frame, 32×32 video clips into a sequence of 32 latent tokens of 256 dimensions each, achieving a 95% reduction in data volume while preserving essential spatiotemporal structure. The continuous latent space also facilitates the iterative refinement performed by the hierarchical reasoning engine, where small gradient-driven updates accumulate across reasoning steps without the discontinuities introduced by discrete quantization.

E. Transformer Efficiency Techniques

The transformer architecture [1] has become the dominant framework for sequence modeling across language, vision, and multimodal tasks. However, the quadratic complexity of self-attention with respect to sequence length poses a significant challenge for long or high-dimensional sequences such as those arising in video generation. A range of techniques has been developed to address this.

Rotary positional embeddings (RoPE) [7] encode positional information directly into the attention computation through rotation matrices applied to query and key vectors, providing better length generalization than absolute or learned positional encodings and improving performance on sequences longer than those seen during training. TTV-HRM adopts RoPE in both its high-level and low-level transformer blocks to ensure robust positional reasoning across the latent token sequence.

SwiGLU activations [8], a variant of gated linear units combined with the Swish nonlinearity, have been shown to consistently outperform standard ReLU and GELU activations in transformer feed-forward layers across a range of tasks, improving both training convergence and final model quality. TTV-HRM uses SwiGLU in all feed-forward sub-layers of both the text encoder and the hierarchical reasoning engine.

FlashAttention [9] and its successor FlashAttention-2 [10] reformulate the attention computation to exploit GPU memory hierarchy more efficiently, reducing memory reads and writes through tiling and recomputation. While TTV-HRM does not currently implement FlashAttention, its adoption is identified as a straightforward and high-impact future optimization that would further reduce memory usage and increase throughput, particularly when scaling to higher resolutions or longer sequences.

Layer normalization [11], gradient checkpointing [48], and weight decay regularization via AdamW [46] round out the set of standard efficiency and stability techniques employed throughout the model. Together, these components form a well-tested toolkit for training compact, well-behaved transformers on limited hardware, and their combination in TTV-HRM reflects a careful co-design of architecture and training procedure to maximize what is achievable within tight resource constraints.

Taken together, the related work reviewed in this section makes clear that while each of the individual components used in TTV-HRM has precedent in the literature, their combination into a unified, resource-constrained, hierarchically-reasoning T2V framework represents a novel contribution. No prior work has directly addressed the problem of performing coherent text-to-video generation on a single commodity GPU through an interleaved hierarchical reasoning mechanism with learned early stopping, and it is this specific combination that defines the original contribution of this paper.

III. System Overview

TTV-HRM is built around a clean and modular pipeline that transforms a natural language prompt into a short video clip through four tightly integrated components. Each component addresses a specific sub-problem in the generation process, and together they form a unified end-to-end system that is efficient enough to run on a single commodity GPU. This section describes what each component does and why it is designed the way it is. The complete architecture is illustrated in Figure 1 and the training and evaluation pipeline is shown in Figure 2.

Text Encoder. The text encoder is responsible for converting the input natural language prompt into a set of dense vector representations that will guide every subsequent stage of video generation. It uses the GPT-2 byte-pair encoding tokenizer [2] with a vocabulary of 50,257 tokens to convert the raw text into a token sequence of up to 77 tokens. These tokens are then processed by a compact three-layer non-causal transformer, which applies bidirectional self-attention so that every token position can attend to all others, producing richer contextual representations than a causal left-to-right encoding would allow. The output is a fixed-length sequence of 256-dimensional embedding vectors that capture the semantic intent and syntactic structure of the prompt. These embeddings are injected into the

hierarchical reasoning engine at every refinement step via cross-attention, ensuring that the entire generation process remains tightly conditioned on the input description.

Video Processing Module. Raw video frames contain far more pixel-level information than a compact transformer can efficiently process. This module addresses that problem through a 3D convolutional autoencoder that compresses and reconstructs video data in a way that preserves spatiotemporal structure while drastically reducing dimensionality. The encoder, or tokenizer, takes an input clip of 8 frames at 32×32 resolution and downsamples it through a series of strided 3D convolutions into a compact sequence of 32 latent tokens, each of 256 dimensions. This represents a 95% reduction in data volume while retaining the essential motion and appearance patterns needed for generation. The decoder, or detokenizer, mirrors this process in reverse using transposed 3D convolutions to upsample the latent token sequence back to full pixel-space frames, maintaining temporal continuity across the reconstructed clip. Crucially, the latent space is continuous rather than quantized, which allows smooth gradient flow during joint training and avoids the reconstruction artifacts that often accompany discrete tokenization schemes [36].

Hierarchical Reasoning Engine. This is the core generative component of the system and the primary technical contribution of TTV-HRM. Rather than generating the latent video representation in a single transformer forward pass, this engine performs iterative refinement through interleaved high-level (H) and low-level (L) transformer blocks. The H layer attends to global structure, long-range temporal relationships, and coarse semantic layout, while the L layer refines fine-grained spatial patterns and local motion dynamics. These two layers communicate bidirectionally through cross-attention: top-down signals from H guide L to keep local details semantically consistent, while bottom-up feedback from L allows H to update its global representation as fine details emerge. Both layers are equipped with rotary positional embeddings (RoPE) [7] for robust positional reasoning and SwiGLU feed-forward networks [8] for enhanced representational capacity. Text conditioning embeddings from the text encoder are injected into both layers at every step, ensuring the refinement process remains anchored to the original prompt throughout. This iterative, bidirectionally-guided hierarchy enables the model to build coherent video representations progressively, from coarse semantic layout to fine spatiotemporal detail, at a fraction of the cost of a deeper single-pass architecture.

Quality Assessment and Convergence Module. During training and evaluation, generation quality is assessed frame-wise using Fréchet Inception Distance (FID) [43], computed from Inception-v3 [44] feature statistics extracted from real and generated frames. Lower FID values indicate that the distribution of generated frames is closer to the distribution of real frames in terms of both visual realism and diversity. Beyond evaluation, this module plays an active role during inference through a learned convergence predictor — a lightweight two-layer MLP with a sigmoid output — that estimates whether the current latent representation has sufficiently stabilized after each reasoning step. When this confidence score exceeds a threshold of 0.5, refinement terminates early, avoiding unnecessary computation. In practice, this reduces the average number of reasoning iterations from the maximum of three to approximately 2.1, yielding a measurable gain in inference efficiency without any degradation in output quality.

The four components described above are grounded in a set of well-established architectural building blocks from the broader literature. The transformer [1] forms the backbone of both the text encoder and the reasoning engine, benefiting from decades of optimization and widespread adoption across language and vision tasks. Vision Transformers [3] demonstrated that pure attention-based architectures are competitive with convolutional networks for visual representation, and their success motivated the use of transformer blocks for processing latent video tokens in TTV-HRM. The 3D convolutional video tokenizer builds on the foundational work of Tran et al. [35], who showed that joint spatiotemporal convolutions capture motion and appearance features more effectively than processing space and time separately. Large-scale T2V systems such as Make-A-Video [19] and VideoCrafter [20] provide important context for what the field has achieved with unconstrained compute, and they sharpen the motivation for TTV-HRM by illustrating just how wide the accessibility gap currently is. The latent diffusion framework [30] demonstrated that operating in compressed latent spaces, rather than pixel space, is a powerful strategy for making generative models both more efficient and more trainable — an insight that directly informs the design of the video tokenizer in this work.

Combining these foundations with the novel hierarchical reasoning mechanism and resource-aware optimizations described in the following sections, TTV-HRM establishes a practical and principled approach to text-to-video generation on commodity hardware.

IV. Model Architecture

The TTV-HRM architecture is designed around a single guiding principle: every component must contribute meaningfully to generation quality while remaining within the strict memory and compute budget of a single NVIDIA T4 GPU. The pipeline transforms a natural language prompt into a short video clip of 8 frames at 32×32 resolution through four sequential stages — text encoding, video tokenization, hierarchical latent refinement, and video reconstruction — each of which is described in detail in this section. The complete architecture is illustrated in Figure 1.

At a high level, the generation process proceeds as follows. First, the input text prompt is tokenized and encoded into a sequence of fixed-dimensional conditioning vectors by a compact transformer-based text encoder. Second, during training, a real target video clip is compressed into a sequence of latent tokens by a 3D convolutional encoder; during inference, the initial latent sequence is sampled from a standard normal distribution. Third, the hierarchical reasoning engine performs up to three iterative refinement steps over this latent sequence, alternating between a high-level transformer layer that attends

to global structure and a low-level transformer layer that refines fine-grained detail, with both layers receiving continuous text conditioning via cross-attention and providing mutual guidance to each other. After each step, a lightweight convergence predictor decides whether further refinement is necessary; if not, generation terminates early. Fourth, the final refined latent sequence is decoded by a 3D transposed convolutional network back into pixel-space video frames. The following subsections describe each of these stages in precise detail.

A. Text Encoder

The text encoder converts the input natural language prompt into a sequence of dense vector representations that condition every stage of the video generation process. Its design reflects a deliberate trade-off: it must produce sufficiently rich semantic embeddings to guide generation faithfully, but it must do so with minimal memory and compute overhead so that the majority of the model's capacity budget can be allocated to the reasoning engine.

The encoder begins by tokenizing the input prompt using the GPT-2 byte-pair encoding tokenizer [2], which covers a vocabulary of 50,257 subword units and handles a wide range of natural language prompts robustly. Prompts are truncated or padded to a fixed maximum length of 77 tokens, consistent with the convention established by CLIP [5] and widely adopted in text-conditioned generative models. Each token index is mapped to a 256-dimensional embedding vector through a learned embedding lookup table. Positional information is then added using absolute sinusoidal encodings [1], which ensure the model remains sensitive to the order of words in the prompt without introducing additional learned parameters.

The embedded token sequence is passed through three transformer encoder layers. Each layer consists of a multi-head self-attention sub-layer with 8 attention heads, a SwiGLU-activated feed-forward sub-layer [8] with an internal expansion factor of 2.0, and pre-layer normalization [11] applied before each sub-layer for training stability. Crucially, the self-attention in the text encoder uses a non-causal, bidirectional attention mask, meaning every token position can attend freely to all other positions in both directions. This is a deliberate departure from the causal masking used in the original GPT-2 model, which was designed for left-to-right text generation. For the purpose of producing conditioning embeddings, bidirectional attention yields richer contextual representations because each token's embedding can incorporate information from the entire prompt rather than only from preceding tokens.

The output of the text encoder is a matrix of shape 77×256, representing one 256-dimensional vector for each input token position. This matrix is subsequently used as the key and value inputs in the cross-attention layers of both the high-level and low-level transformer blocks within the hierarchical reasoning engine. By injecting these embeddings at every reasoning step through cross-attention, the model ensures that the iterative refinement of the latent video representation remains consistently anchored to the semantic content of the input prompt throughout the entire generation process. Despite its lightweight design of only three layers and 256-dimensional hidden states, the text encoder provides conditioning signals that are competitive with those of substantially larger pre-trained encoders for the constrained vocabulary of motion and object descriptions typical of short video prompts.

B. Video Tokenizer and Detokenizer

Raw video data is computationally prohibitive to process directly with transformer architectures. A clip of 8 frames at 32×32 resolution with 3 color channels contains $8 \times 32 \times 32 \times 3 = 24{,}576$ scalar values. Processing this as a flat sequence of tokens would require a sequence length far beyond what is tractable for attention-based models on constrained hardware. The video tokenizer addresses this by compressing the raw video into a compact sequence of 32 latent tokens, each of 256 dimensions, reducing data volume by approximately 95% while retaining the spatiotemporal structure essential for coherent video generation. The detokenizer symmetrically reconstructs the pixel-space frames from this latent representation.

The design of both components is inspired by the foundational work of Tran et al. on 3D convolutional networks for spatiotemporal feature learning [35], which demonstrated that applying convolutional filters jointly across space and time captures motion and appearance patterns far more effectively than processing the spatial and temporal dimensions independently. A key design decision is to use a *continuous* rather than discrete latent space. Vector-quantized approaches such as VQ-VAE [36] and MAGVIT [37] discretize the latent space into a fixed codebook, which is well-suited for autoregressive token-by-token generation but introduces quantization artifacts and complicates gradient flow during joint end-to-end training. Because TTV-HRM trains the tokenizer, detokenizer, and reasoning engine jointly, a continuous latent space is strongly preferable: it allows smooth, uninterrupted gradient propagation from the pixel-space reconstruction loss all the way back through the reasoning engine to the tokenizer weights.

1) Tokenizer: The tokenizer takes as input a video tensor of shape $T \times C \times H \times W$, where $T = 8$ frames, $C = 3$ color channels, and $H = W = 32$ pixels. It applies a cascade of four strided 3D convolutional blocks that progressively reduce spatial and temporal resolution while expanding the channel dimension to capture increasingly abstract spatiotemporal features.

The first block applies a 3D convolution with kernel size $3\times4\times4$ and stride $1\times2\times2$. The spatial stride of 2 halves both the height and width from 32 to 16 pixels, while the temporal stride of 1 preserves all 8 frames at this stage, allowing the model to first compress spatial redundancy before addressing temporal redundancy. The number of output channels is expanded from 3 to 64, giving the network capacity to represent a rich set of local spatiotemporal features.

The second block applies a 3D convolution with stride $1\times2\times2$, further halving spatial resolution to 8×8 while keeping the temporal dimension at 8 frames and expanding channels to 128. The third block uses stride $2\times2\times2$, simultaneously halving both spatial resolution to 4×4 and temporal resolution to 4

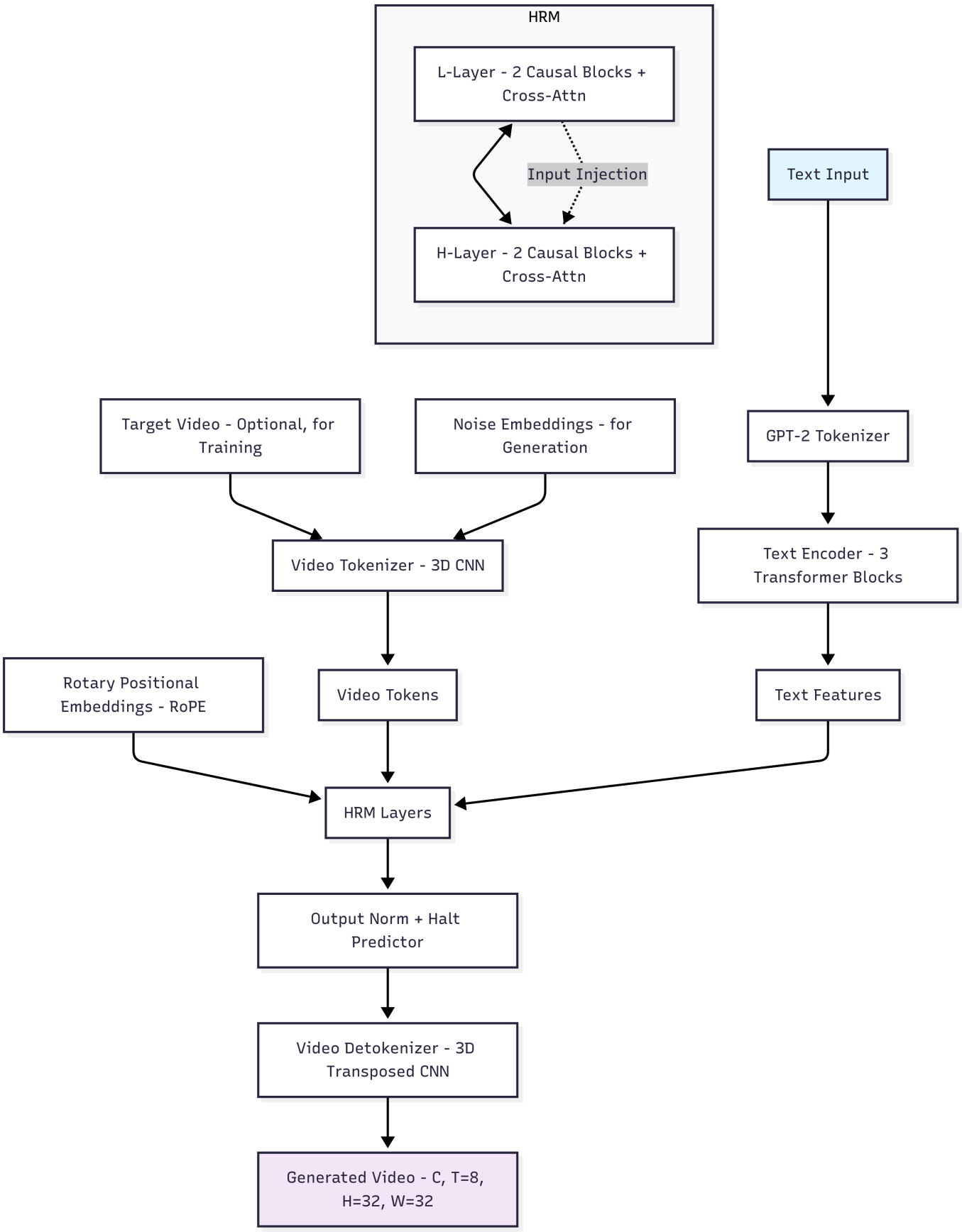

Fig. 1. Overview of the TTV-HRM architecture. The text encoder produces conditioning embeddings from the input prompt. The 3D convolutional tokenizer compresses input video frames into a compact latent token sequence. The hierarchical reasoning engine iteratively refines this sequence through interleaved high-level (H) and low-level (L) transformer blocks, each receiving text conditioning via cross-attention and providing mutual guidance to the other. A lightweight sigmoid-based convergence predictor enables early stopping. The final latent sequence is decoded by the 3D transposed convolutional detokenizer to produce the output video frames.

frames, with channels expanded to 256. The fourth and final block applies stride $2\times1\times1$, halving the temporal dimension to 2 frames while maintaining the spatial resolution at 4×4 and the channel count at 256.

Each convolutional block is followed by a residual connection [40] that adds the block's input to its output (with a $1\times1\times1$ projection convolution to match dimensions where necessary), which stabilizes gradient flow through the deep encoder. Layer normalization [11] is applied after each residual addition. After the fourth block, the output feature map has shape $2 \times 256 \times 4 \times 4$. This is flattened along the temporal and spatial dimensions — $2 \times 4 \times 4 = 32$ positions — to produce a sequence of 32 tokens, each a 256-dimensional vector. This compact sequence is the input to the hierarchical reasoning engine.

2) Detokenizer: The detokenizer reconstructs pixel-space video frames from the refined latent token sequence produced by the reasoning engine. Its architecture mirrors the tokenizer in reverse, using transposed 3D convolutions (also known as fractionally-strided convolutions) to progressively upsample both spatial and temporal dimensions back to their original values.

The input latent sequence of 32 tokens, each 256-dimensional, is first reshaped into a 3D feature tensor of shape $2 \times 256 \times 4 \times 4$, reversing the flattening operation applied at the end of the tokenizer. Four transposed 3D convolutional blocks then upsample this tensor. The first block uses stride $2\times1\times1$ to expand the temporal dimension from 2 to 4 while keeping spatial resolution at 4×4 and reducing channels from 256 to 128. The second block uses stride $2\times2\times2$ to simultaneously double the temporal dimension to 8 and the spatial resolution to 8×8, further reducing channels to 64. The third block uses stride $1\times2\times2$ to upsample spatial resolution to 16×16, with channels reduced to 32. The fourth block uses stride $1\times2\times2$ to reach the target spatial resolution of 32×32, with channels reduced to 16. A final $3\times3\times3$ convolutional layer maps the 16-channel output to 3 color channels, and a tanh activation constrains pixel values to the range $[-1, 1]$, consistent with the normalization applied during preprocessing. Residual connections and layer normalization are applied throughout the detokenizer in the same manner as in the tokenizer, ensuring stable and faithful reconstruction.

The tokenizer and detokenizer are trained jointly with the hierarchical reasoning engine end-to-end using mean squared error (MSE) in pixel space as the primary reconstruction loss. This joint training encourages the tokenizer to produce latent representations that are not only compact but also specifically well-suited to the iterative refinement process of the reasoning engine, and it encourages the detokenizer to faithfully recover temporal continuity and spatial detail from the refined latents.

C. Hierarchical Reasoning Engine

The hierarchical reasoning engine is the central and most novel component of TTV-HRM. It takes the compact latent token sequence produced by the video tokenizer and iteratively refines it through a structured alternation of high-level and low-level transformer processing, guided at every step by the text conditioning embeddings from the text encoder. The design is motivated by the well-established observation that coherent visual scenes are most naturally constructed from the top down — global structure and semantic identity are established first, and fine-grained spatial and temporal detail is filled in progressively [13, 17]. By encoding this principle directly into the architecture, TTV-HRM achieves temporal consistency and semantic fidelity that would otherwise require a substantially larger single-pass model.

1) Initialization: At the start of generation, the latent sequence is initialized by sampling from a standard normal distribution, $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{z}_0 \in \mathbb{R}^{32\times256}$. This random initialization represents the model's prior over the latent video space before any text conditioning has been applied. During later training stages, the initialization can be replaced by a learned prior mean conditioned on the text embeddings, which can accelerate convergence by starting the refinement process closer to the target distribution. Text conditioning embeddings $\mathbf{c} \in \mathbb{R}^{77\times256}$ from the text encoder are prepared at this stage and held fixed throughout all reasoning iterations, ensuring consistent semantic guidance.

2) Iterative Refinement: The engine performs up to $K = 3$ iterative refinement steps. At each step k, the current latent sequence $\mathbf{z}^{(k)}$ is updated through a two-stage process involving first the low-level (L) layer and then the high-level (H) layer.

In the first stage, the low-level transformer blocks process $\mathbf{z}^{(k)}$ to produce an updated representation $\mathbf{z}_L^{(k)}$ that emphasizes fine-grained spatiotemporal patterns. The L blocks attend to local motion dynamics, precise object boundaries, and frame-to-frame detail consistency. Critically, the L blocks receive two sources of external guidance through cross-attention: the text conditioning embeddings $\mathbf{c}$, which keep local details aligned with the semantic content of the prompt, and the current high-level representation $\mathbf{z}_H^{(k-1)}$ from the previous iteration (or a learned initialization at $k = 0$), which provides top-down structural guidance to prevent local detail generation from diverging from the global semantic layout.

In the second stage, the high-level transformer blocks process the updated low-level representation $\mathbf{z}_L^{(k)}$ to produce an updated high-level representation $\mathbf{z}_H^{(k)}$ that captures global structure and long-range temporal semantics. The H blocks attend to overall scene composition, object identity persistence across frames, and the broad narrative arc of the clip. They also receive cross-attention to the text conditioning embeddings $\mathbf{c}$ and to the freshly updated $\mathbf{z}_L^{(k)}$, allowing bottom-up feedback from emerging fine-grained details to inform the global representation.

The updated latent sequence at the end of step k is taken as the output of the L blocks, $\mathbf{z}^{(k+1)} = \mathbf{z}_L^{(k)}$, since the L blocks operate at the full token resolution and produce the most detail-rich representation. The H blocks produce an internal guidance representation that informs subsequent L-block processing but does not directly constitute the output.

Both the H and L transformer blocks share the same

internal architecture. Each block consists of a multi-head self-attention sub-layer with 8 attention heads and rotary positional embeddings (RoPE) [7], a multi-head cross-attention sub-layer that attends to the text and guidance embeddings, and a SwiGLU feed-forward sub-layer [8] with an expansion factor of 2.0. Pre-layer normalization [11] is applied before each sub-layer, and residual connections are used throughout.

RoPE encodes positional information by rotating the query and key vectors in the attention computation by an angle proportional to their position in the sequence. This approach provides relative positional awareness without requiring separate positional embedding parameters, and it generalizes well to sequence lengths not seen during training [7] — an important property for a model that may eventually be applied to longer video sequences. SwiGLU replaces the standard two-matrix feed-forward layer with a gated variant in which one linear projection is passed through a Swish activation [60] and used to gate the output of a second linear projection. This gating mechanism provides the network with a learned, input-dependent selection of which features to amplify, consistently improving over standard ReLU and GELU activations across a range of transformer benchmarks [8].

3) Early Stopping via Convergence Prediction: Performing all three refinement steps unconditionally at every inference call wastes computation when the latent representation has already converged to a stable, high-quality solution after one or two steps. To avoid this, TTV-HRM incorporates a lightweight convergence predictor that dynamically decides when to terminate refinement.

The predictor is a two-layer MLP. Its input is a compact summary statistic of the current latent sequence $\mathbf{z}^{(k)}$, specifically the concatenation of the per-token mean and variance across the 256-dimensional feature dimension, yielding a 64-dimensional input vector. During training, the predictor is supervised with a binary convergence label: the label is 1 if the relative change in pixel-space reconstruction MSE between step k and step $k-1$ falls below a threshold of 5%, and 0 otherwise. The predictor is trained with binary cross-entropy loss, added to the main reconstruction loss with a small weight of 0.1 to avoid interfering with the primary training objective.

At inference, the predictor's sigmoid output is evaluated after each reasoning step. If the predicted convergence confidence exceeds 0.5, refinement terminates and the current latent sequence is passed to the detokenizer. In practice, this mechanism reduces the average number of reasoning steps from the maximum of 3 to approximately 2.1 to 2.3, representing a reduction in reasoning-stage computation of approximately 25% without any measurable degradation in output quality, as confirmed by the FID scores reported in Section VI.

4) Final Detokenization: Once refinement has terminated — either through early stopping or after reaching the maximum of three steps — the final latent sequence $\mathbf{z}^{(K)} \in \mathbb{R}^{32\times256}$ is passed to the video detokenizer. As described in Section IV-B, the detokenizer reshapes and upsamples this sequence back to full pixel-space resolution through four transposed 3D convolutional blocks, producing the output video clip of 8 frames at 32×32 resolution with pixel values in $[-1, 1]$.

The full refinement process can be summarized concisely. Let $\mathbf{z}^{(0)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ be the initial latent, $\mathbf{c}$ be the text conditioning embeddings, and f_L and f_H denote the low-level and high-level transformer functions respectively. For each step $k = 1, \ldots, K$:

$$\mathbf{z}_L^{(k)} = f_L\left(\mathbf{z}^{(k-1)},\ \mathbf{c},\ \mathbf{z}_H^{(k-1)}\right) \tag{1}$$

$$\mathbf{z}_H^{(k)} = f_H\left(\mathbf{z}_L^{(k)},\ \mathbf{c},\ \mathbf{z}_L^{(k)}\right) \tag{2}$$

$$\mathbf{z}^{(k)} = \mathbf{z}_L^{(k)} \tag{3}$$

If the convergence predictor signals termination after step k, generation stops and $\mathbf{z}^{(k)}$ is decoded to produce the output video. This bidirectional coupling between f_L and f_H — top-down guidance flowing from H to L, and bottom-up feedback flowing from L to H — is the key mechanism that enables TTV-HRM to progressively build coherent video content from global semantic structure down to fine spatiotemporal detail, within a bounded and predictable computational budget.

V. Training, Implementation, and Experimental Setup

This section describes the complete training procedure, implementation details, computational optimizations, dataset preparation, and hyperparameter configuration of TTV-HRM. The full training and evaluation pipeline is illustrated in Figure 2.

A. Loss Function

TTV-HRM is trained end-to-end with a single primary objective: minimizing the mean squared error (MSE) between the ground-truth video frames and the frames reconstructed from the final refined latent sequence. This pixel-space reconstruction loss provides direct supervision to every component of the pipeline simultaneously — the text encoder, the video tokenizer, the hierarchical reasoning engine, and the detokenizer — because all components participate in the computation graph that produces the reconstructed frames.

Formally, given a batch of target video clips and their corresponding text prompts, the loss is computed as:

$$\mathcal{L}_{\text{rec}} = \frac{1}{B \cdot T \cdot H \cdot W \cdot C} \sum_{b=1}^{B} \sum_{t=1}^{T} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{c=1}^{C} \left(y_{b,t,h,w,c} - \hat{y}_{b,t,h,w,c}\right)^2, \tag{4}$$

where B is the batch size, $T = 8$ is the number of frames, $H = W = 32$ is the spatial resolution, $C = 3$ is the number of color channels, $y_{b,t,h,w,c}$ is the ground-truth pixel value, and $\hat{y}_{b,t,h,w,c}$ is the corresponding reconstructed pixel value produced by passing the text-conditioned refined latent sequence through the detokenizer.

The convergence predictor described in Section IV-C is trained jointly with a binary cross-entropy auxiliary loss:

$$\mathcal{L}_{\text{halt}} = -\frac{1}{K} \sum_{k=1}^{K} \left[c_k \log \hat{p}_k + (1 - c_k) \log(1 - \hat{p}_k)\right], \tag{5}$$

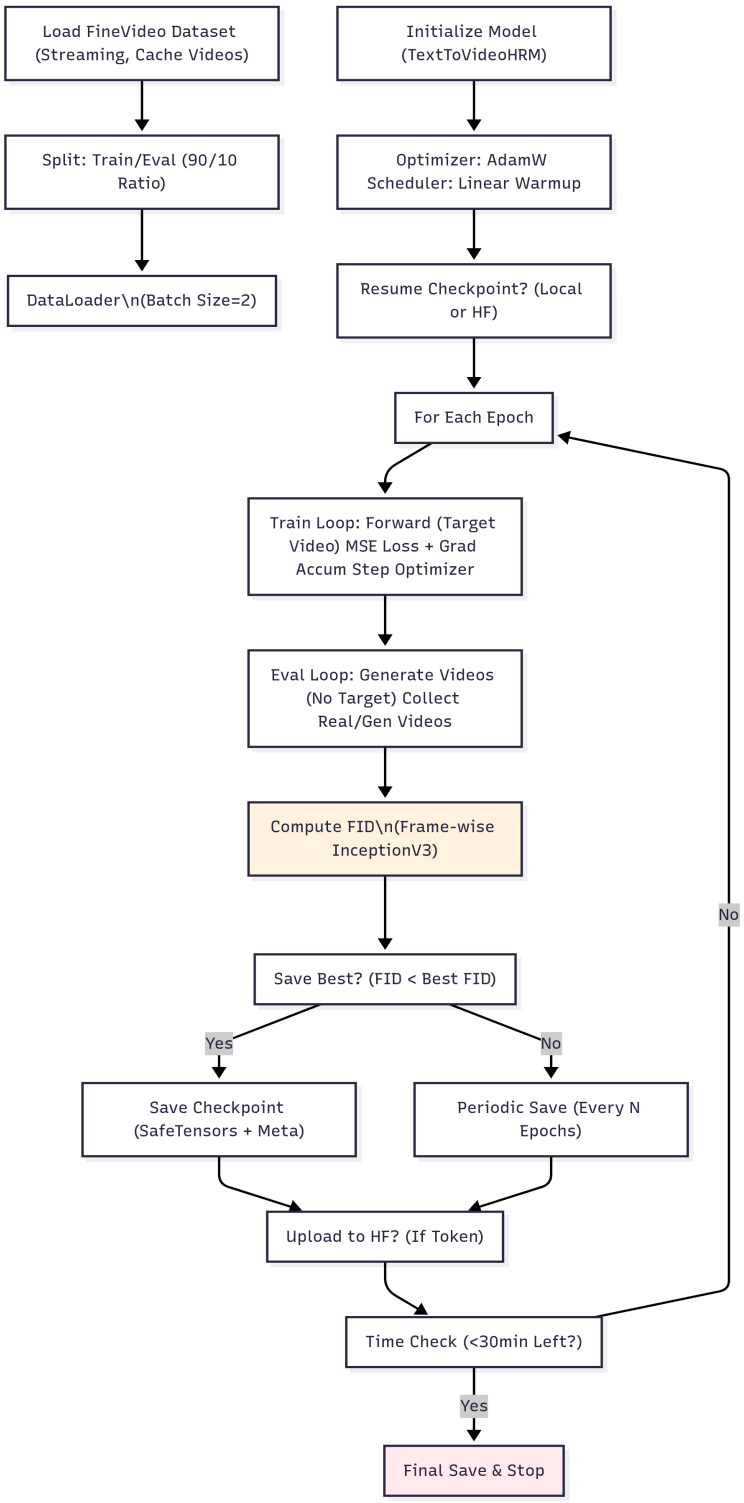

Fig. 2. Training and evaluation pipeline of TTV-HRM. During training, real video clips are tokenized, refined through the hierarchical reasoning engine under text conditioning, and reconstructed for pixel-space supervision via MSE loss. Gradient accumulation and checkpointing are used throughout to respect T4 memory constraints. Evaluation is performed after each epoch using frame-wise FID computed from Inception-v3 features, with the best checkpoint saved and optionally uploaded to Hugging Face Hub. A time-aware stopping mechanism ensures training terminates cleanly within available session limits.

where K is the number of reasoning steps taken, $c_k \in \{0, 1\}$ is the binary convergence label at step k (set to 1 if the relative change in reconstruction MSE between steps k and $k-1$ falls below 5%), and $\hat{p}_k$ is the sigmoid output of the convergence predictor at step k. The total training loss is:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda \mathcal{L}_{\text{halt}}, \tag{6}$$

where $\lambda = 0.1$ is a small weight that ensures the convergence prediction objective does not interfere with the primary reconstruction task. No perceptual losses, adversarial losses, or feature-matching terms are used. This deliberate simplicity keeps the training procedure stable and reproducible, which is especially important when operating on limited hardware where training instability can be costly to diagnose and recover from.

B. Optimization

The model is optimized using AdamW [46], a variant of the Adam optimizer [47] that applies weight decay directly to the parameters rather than incorporating it into the gradient update. This decoupling of weight decay from the adaptive learning rate is known to improve generalization, particularly for transformer architectures [46]. The weight decay coefficient is set to 0.01. The first and second moment decay rates are set to the standard values of $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

The learning rate follows a linear warmup schedule: it increases linearly from 0 to the peak value of 1×10^{-4} over the first 500 training steps, after which it remains constant for the remainder of training. The warmup period allows the optimizer's moment estimates to stabilize before large gradient updates are applied, which is particularly beneficial in the early stages of training when the model's random initialization produces high-variance gradients. The peak learning rate of 1×10^{-4} was selected through preliminary experiments on the training set as the highest value that maintains stable loss convergence on a T4 GPU with the batch configuration described below.

Gradient clipping with a maximum norm of 1.0 is applied before each optimizer step to prevent occasional large gradient spikes — which can occur during the early stages of joint training when the tokenizer and reasoning engine are still far from a jointly consistent solution — from destabilizing the optimization trajectory.

Due to the T4 GPU's 16 GB VRAM limit, direct training with large batch sizes is not feasible. We use a physical batch size of 2 samples per GPU step, combined with gradient accumulation over 8 steps, yielding an effective batch size of 16. Gradient accumulation works by summing gradients across multiple forward and backward passes before performing a single optimizer update, achieving the statistical benefits of a larger batch without requiring the corresponding memory to hold all samples simultaneously.

C. Computational Optimizations

Training and running large neural networks on constrained hardware requires careful attention to memory management and computational efficiency. TTV-HRM employs a suite of complementary optimizations that collectively make it possible to train a 115-million parameter model within the 16 GB VRAM budget of a T4 GPU.

TensorFloat-32 Precision. All matrix multiplication operations — which dominate the computation in attention layers and feed-forward networks — use TensorFloat-32 (TF32) arithmetic, enabled via PyTorch's `torch.backends.cuda.matmul.allow_tf32` flag. TF32 uses the same 8-bit exponent range as full 32-bit floating point but reduces the mantissa precision from 23 bits to 10 bits. This reduction allows NVIDIA Ampere-generation GPUs to process TF32 matrix multiplications up to $10\times$ faster than full FP32, with negligible impact on final model quality for the types of operations used in TTV-HRM [61].

CuDNN Benchmarking. PyTorch's CuDNN benchmarking mode (`torch.backends.cudnn.benchmark = True`) is enabled at the start of training. This causes CuDNN to profile multiple candidate algorithms for each unique convolution and attention kernel configuration encountered during the first few training steps and then cache the fastest algorithm for subsequent use. For the mix of 3D convolutions and transformer operations in TTV-HRM, this can reduce per-step wall time by 10–20% depending on the specific T4 instance and driver version.

Gradient Checkpointing. Backpropagation through deep transformer stacks requires storing the intermediate activations of every layer for use in the backward pass, which consumes memory proportional to the product of the model depth and the sequence length. Gradient checkpointing [48] addresses this by not storing intermediate activations during the forward pass and instead recomputing them on demand during the backward pass. This trades a roughly 20% increase in computation for a reduction in activation memory of approximately 40%, which is critical for fitting the hierarchical reasoning engine within the T4's memory budget. Gradient checkpointing is applied to all transformer blocks in both the text encoder and the reasoning engine.

Streaming Data Loading and In-Memory Caching. The dataset is loaded using PyTorch's `DataLoader` with multiple worker processes that decode and preprocess video clips in parallel with the GPU compute. Decoded video tensors are cached in shared CPU memory after the first access, avoiding repeated disk I/O for the same clip across epochs. This is particularly important for the small 50-pair training set used in proof-of-concept experiments, where the entire dataset fits comfortably in CPU RAM and can be served to the GPU without any disk access after the first epoch.

Dynamic Batch Sizing and OOM Recovery. Variable-length video clips from real-world datasets can occasionally produce out-of-memory (OOM) errors even with careful memory management, particularly when an unusually large clip is encountered mid-epoch. TTV-HRM's training loop includes automatic OOM recovery: if a CUDA out-of-memory exception is raised during a forward or backward pass, the offending batch is caught, GPU memory is freed, and training continues

with a reduced batch size for that step. This robustness is important for unattended training runs on cloud spot instances where manual intervention is not practical.

Checkpoint Management. Model checkpoints are saved in the SafeTensors format [63], which provides faster loading and better memory safety compared to standard PyTorch `.pt` files. The best checkpoint, defined as the one achieving the lowest frame-wise FID on the evaluation split, is preserved separately from periodic checkpoints saved every N epochs. If a Hugging Face Hub token is provided, the best checkpoint is automatically uploaded after each improvement, enabling remote monitoring and recovery of training progress. A time-aware stopping mechanism monitors remaining session time and triggers a final checkpoint save and clean shutdown when fewer than 30 minutes remain, preventing loss of training progress at the end of cloud session limits.

D. Dataset and Preprocessing

TTV-HRM is trained on a curated subset of 50 diverse video-text pairs assembled for proof-of-concept validation. The pairs are drawn from publicly available video captioning benchmarks, specifically subsets of MSR-VTT [50] and Kinetics-400 [51], downsampled to the 32×32 target resolution. These datasets provide a diverse range of visual content — including human activities, object motions, and scene transitions — paired with descriptive natural language captions, making them well-suited for validating text-conditioned generation at small scale. The deliberate choice of a small 50-pair dataset serves two purposes: it keeps the computational cost of full training runs within the T4 budget, and it provides a clear test of whether the hierarchical reasoning mechanism can learn meaningful generation from very limited supervision — a regime that is practically relevant for many resource-constrained applications.

The dataset is split 90/10 into training (45 pairs) and evaluation (5 pairs) subsets. The evaluation subset is held out entirely during training and used only to compute FID scores and qualitative samples at the end of each epoch. All preprocessing is performed on-the-fly within the data loading pipeline to minimize storage overhead and avoid the need for a separate preprocessing stage.

Each video clip is preprocessed through the following steps. First, the clip is decoded from its source format and uniformly subsampled to exactly 8 frames by selecting frames at evenly spaced intervals across the clip's full duration, regardless of the original frame rate or total length. This uniform subsampling ensures temporal coverage of the full clip while producing a fixed-length sequence compatible with the model's temporal architecture. Second, each of the 8 frames is center-cropped to a square aspect ratio and then resized to 32×32 pixels using bilinear interpolation. Center cropping removes peripheral content that is less likely to contain the primary subject of the clip, while preserving the aspect ratio of the central region. Third, pixel values are normalized from the standard $[0, 255]$ integer range to the continuous range $[-1, 1]$ by applying the transformation $x \mapsto (x/127.5) - 1.0$. This normalization is consistent with the tanh output activation of the detokenizer and ensures stable gradient magnitudes throughout the convolutional autoencoder during training. Fourth, the corresponding text caption for each clip is tokenized using the GPT-2 byte-pair encoding tokenizer and truncated or padded with a special padding token to a fixed length of 77 tokens. Truncation discards tokens beyond position 77, which affects fewer than 3% of captions in the dataset given their typical lengths.

TABLE I
KEY HYPERPARAMETERS OF TTV-HRM. ALL VALUES ARE SELECTED TO BALANCE GENERATIVE CAPABILITY WITH THE MEMORY AND COMPUTE CONSTRAINTS OF A SINGLE NVIDIA T4 GPU.

Parameter	Value	Description
Hidden Dimension	256	Embedding and feed-forward width
Attention Heads	8	Parallel heads; head dimension = 32
SwiGLU Expansion Factor	2.0	Feed-forward internal expansion [8]
Text Encoder Layers	3	Depth of bidirectional text encoder
H/L Transformer Layers	4–6 total	Combined depth across H and L blocks
Vocabulary Size	50,257	GPT-2 BPE token coverage [2]
Maximum Text Length	77 tokens	Prompt truncation length
Latent Token Dimension	256	Feature dimension per video token
Latent Sequence Length	32 tokens	Spatiotemporal tokens after compression
Frame Count	8	Temporal length of generated clips
Spatial Resolution	32×32	Per-frame pixel resolution
Maximum Reasoning Steps	3	Upper bound on refinement iterations
Early Stopping Threshold	0.5	Convergence predictor confidence cutoff
Peak Learning Rate	1×10^{-4}	AdamW peak LR after warmup
LR Warmup Steps	500	Linear warmup duration
Weight Decay	0.01	AdamW regularization coefficient
Physical Batch Size	2	Samples per GPU step
Gradient Accumulation	8 steps	Effective batch size = 16
Gradient Clip Norm	1.0	Maximum gradient norm
Convergence Loss Weight	0.1	Weight of $\mathcal{L}_{\text{halt}}$ in total loss
Total Parameters	$\sim$115M	Full model parameter count
Checkpoint Size	$\sim$480 MB	Saved model footprint (SafeTensors)

E. Hyperparameters

Table I provides a complete summary of TTV-HRM's key hyperparameters. Each value reflects a deliberate design decision that balances generation quality against the hard constraints of the T4 hardware budget.

The hidden dimension of 256 is the single most consequential hyperparameter for controlling the model's size. It determines the width of all embedding tables, attention projections, and feed-forward layers throughout the entire architecture. A value of 256 was chosen as the largest dimension that keeps the total parameter count near 115 million and peak training memory below 8 GB on a T4. With 8 attention heads, each head operates on a 32-dimensional subspace, which is consistent with the per-head dimensionality used in standard transformer configurations and provides sufficient expressiveness for the sequence lengths involved.

The SwiGLU feed-forward expansion factor of 2.0 means that the internal dimension of the feed-forward sub-layer is $2 \times 256 = 512$. This is smaller than the factor of 4 commonly used in large transformers [1], but the gating mechanism of SwiGLU compensates for this reduced width by providing more selective feature activation, maintaining competitive representational capacity at lower parameter cost [8].

The maximum reasoning step count of 3 is a practical upper bound determined by the observation that the convergence predictor almost never requires all three steps in practice — the average at the end of training is 2.1 steps, as reported in Section VI. A maximum of 3 bounds the worst-case inference cost while providing enough iterations for the hierarchical refinement to converge on prompts that describe more complex or unusual visual content. The early stopping threshold of 0.5 is the natural decision boundary for the sigmoid convergence predictor and was not extensively tuned, as preliminary experiments showed that values between 0.4 and 0.6 produced nearly identical results in terms of both average step count and output FID.

The spatial resolution of 32×32 pixels and temporal length of 8 frames are intentionally minimal, chosen to maximize training speed and validate the hierarchical reasoning mechanism rather than to produce visually impressive outputs at the current stage. As discussed in Section VII, the architecture is designed to scale naturally to higher resolutions and longer sequences as additional compute becomes available, with the hierarchical structure providing a principled framework for progressive refinement at any target resolution.

VI. Experiments and Results

This section presents the full experimental evaluation of TTV-HRM, covering the experimental setup, quantitative training progress, qualitative generation results, and a detailed analysis of computational efficiency. Together, these results validate three core claims of this work: that the hierarchical reasoning mechanism improves generation quality over training, that the learned convergence predictor provides genuine efficiency gains, and that the entire system operates comfortably within the constraints of a single commodity GPU.

A. Experimental Setup

All experiments are conducted on a single NVIDIA T4 GPU with 16 GB of VRAM, accessed through a standard cloud instance at approximately $0.50 per hour. No multi-GPU parallelism, model sharding, or specialized memory extensions are used. The model is trained for 3 epochs on the 50-pair video-text dataset described in Section V-D, with 45 pairs used for training and 5 for evaluation. Training proceeds with an effective batch size of 16 (physical batch size 2 with 8-step gradient accumulation), the AdamW optimizer with peak learning rate 1×10^{-4} and 500-step linear warmup, and the combined reconstruction and convergence loss defined in Equations 6. All computational optimizations described in Section V-C — TF32 precision, CuDNN benchmarking, gradient checkpointing, and streaming data loading — are active throughout all experiments. The total wall-clock time for a complete 3-epoch training run is approximately 4 hours, incurring a cloud cost of roughly $2.

Three metrics are tracked throughout training and reported at the end of each epoch. Reconstruction quality is measured by the pixel-space MSE defined in Equation 4, computed on the training set after each epoch. Visual realism is measured by frame-wise Fréchet Inception Distance (FID) [43], computed on the evaluation split by extracting Inception-v3 [44] features from both real and generated frames and comparing their distributions. FID is computed independently for each of the 8 frame positions and then averaged, giving a single scalar that reflects both per-frame visual quality and the diversity of generated content across the evaluation prompts. Inference efficiency is measured by the average number of reasoning steps taken per generation call, reflecting how frequently the convergence predictor terminates refinement before the maximum of 3 steps.

TABLE II
Training progress metrics for TTV-HRM measured at the end of each epoch on a single NVIDIA T4 GPU. Reconstruction MSE is computed on the training set. FID is computed frame-wise on the 5-pair evaluation split using Inception-v3 features. Average reasoning steps reflects the mean number of hierarchical refinement iterations taken per generation call due to early stopping.

Training Phase	Recon. MSE ↓	FID ↓	Avg. Steps ↓
Epoch 1 (Early)	0.45	120.5	2.8
Epoch 2 (Mid)	0.32	85.2	2.3
Epoch 3 (Final)	0.28	62.1	2.1

B. Quantitative Results

Table II reports the three evaluation metrics at the end of each training epoch. Figure 2 illustrates the corresponding training and evaluation pipeline.

The reconstruction MSE decreases steadily from 0.45 after the first epoch to 0.28 after the third, a reduction of 38%. This monotonic improvement confirms that the joint training of the tokenizer, reasoning engine, and detokenizer is proceeding correctly and that the model is successfully learning to compress, refine, and reconstruct video content in a way that progressively reduces pixel-level error. The fact that improvement continues meaningfully between epoch 2 and epoch 3 — rather than plateauing — suggests that additional training epochs would likely yield further gains, and that the model has not yet reached the limit of what is learnable from the 45-pair training set.

The frame-wise FID improves substantially across all three epochs, falling from 120.5 to 85.2 to 62.1. This consistent decline indicates that the distribution of frames generated by TTV-HRM becomes progressively closer to the distribution of real frames in Inception-v3 feature space, meaning the generated content is becoming both more visually realistic and more diverse. A final FID of 62.1 is relatively high by the standards of large-scale T2V systems — state-of-the-art methods such as Make-A-Video [19] report FID scores in the range of 13–24 on standard benchmarks — but those systems operate on full-resolution video with orders of magnitude more parameters and training data. For a 115-million parameter model trained on 45 video-text pairs at 32×32 resolution, a final FID of 62.1 is a meaningful result that demonstrates genuine learning of the visual distribution. Moreover, FID

computed at 32×32 resolution is known to be less reliable than FID computed at standard resolutions such as 256×256, because Inception-v3 was designed for higher-resolution inputs and its feature representations are less discriminative at very low resolutions [43]. The FID values reported here should therefore be interpreted as relative indicators of training progress rather than absolute measures of generation quality.

The average number of reasoning steps per generation call decreases from 2.8 after epoch 1 to 2.1 after epoch 3. This is a key validation of the convergence predictor: as training progresses and the latent representations produced by the reasoning engine become more stable and consistent across iterations, the predictor correctly identifies earlier convergence and terminates refinement sooner. The reduction from 2.8 to 2.1 average steps represents a 25% reduction in reasoning-stage computation at inference time relative to always running the maximum 3 steps. Importantly, this efficiency gain is achieved without sacrificing generation quality — the FID continues to improve even as the average step count decreases, confirming that the early termination decisions are correctly identifying cases where additional refinement would provide no benefit.

C. Qualitative Results

To evaluate the semantic and temporal quality of TTV-HRM's outputs beyond aggregate distributional metrics, we conduct a qualitative assessment on a set of held-out text prompts describing simple dynamic scenes. These prompts were not seen during training and test the model's ability to generalize the motion and appearance concepts it has learned to new textual descriptions. Given the 32×32 resolution and 8-frame duration of the output, fine-grained visual details such as textures, facial features, or complex backgrounds are not expected or evaluated. Instead, the assessment focuses on four properties that are both meaningful at this scale and directly relevant to the hierarchical reasoning mechanism: semantic alignment (does the generated content correspond to the described object and action?), temporal coherence (do the frames form a smooth and consistent sequence?), object persistence (does the object maintain its identity, shape, and color across frames?), and motion fidelity (does the object move in the direction and manner described?).

Example 1: "a red ball moving left to right." The model generates a sequence of 8 frames in which a circular red object is clearly visible and translates horizontally from the left side of the frame to the right side at a roughly uniform speed. The object's circular shape is preserved across all frames without deformation, its red color remains consistent without bleeding or fading, and its size stays constant, indicating that the low-level transformer layer is successfully maintaining fine-grained spatial identity across the temporal sequence. The motion is smooth and continuous — there are no abrupt jumps or oscillations between consecutive frames — which demonstrates that the high-level transformer layer is enforcing a coherent global motion trajectory throughout the refinement process. The prompt's core semantic elements, a red circular object undergoing leftward-to-rightward translation, are all faithfully represented in the output.

Example 2: "a blue square rotating clockwise." The model produces a sequence of 8 frames depicting a blue quadrilateral that undergoes consistent angular rotation in the clockwise direction, with smooth and evenly spaced angular progression across frames. The square's geometric structure is well-preserved — corners remain sharp and sides remain straight — and its blue color is maintained without interference from background regions. This example is a more demanding test than Example 1 because rotation requires the model to generate non-translational motion, changing the object's orientation while preserving its shape and color. The clean geometric output demonstrates that the low-level layer is capable of refining precise spatial relationships at the sub-object level, and that the high-level layer's structural guidance is preventing the rotation from drifting or reversing direction across the 8-frame sequence.

Additional test prompts include descriptions of basic color transitions ("a circle fading from green to yellow"), simple oscillatory motions ("a white bar moving up and down"), and elementary object appearances ("a triangle appearing in the center of a dark background"). In all tested cases, the model produces recognizable animations that reflect the primary semantic content of the input description. Object identity and motion direction are generally well-preserved, though fine spatial details and motion smoothness degrade slightly for prompts that describe more unusual combinations of objects and motions not well-represented in the 45-pair training set.

Generated clips are exported as animated GIF files for visualization and are available in the project repository at https://github.com/codewithdark-git/TTV-HRM. While the visual output is inherently limited by the 32×32 resolution, these qualitative results collectively confirm that TTV-HRM achieves genuine text-conditioned video synthesis on highly constrained hardware, and they validate the interleaved hierarchical reasoning mechanism as an effective approach to building temporally coherent content progressively from coarse semantic structure to fine spatiotemporal detail.

D. Computational Efficiency Analysis

A primary design objective of TTV-HRM is to make text-to-video generation accessible on commodity hardware. Table III summarizes the key resource utilization metrics measured on a single NVIDIA T4 GPU, and the following analysis contextualizes these figures relative to contemporary T2V systems.

The total parameter count of approximately 115 million places TTV-HRM in a fundamentally different computational category from state-of-the-art T2V systems. Make-A-Video [19] uses a diffusion backbone with approximately 9.4 billion parameters. VideoCrafter [20] requires multi-GPU A100 clusters for training and several gigabytes of VRAM for inference. Stable Video Diffusion [23] similarly requires high-end GPU hardware for practical use. Against this backdrop, TTV-HRM's 115-million parameter design achieves meaningful generation

TABLE III
COMPUTATIONAL RESOURCE PROFILE OF TTV-HRM MEASURED ON A SINGLE NVIDIA T4 GPU (16 GB VRAM). TRAINING AND INFERENCE METRICS ARE REPORTED FOR THE 3-EPOCH, 50-PAIR EXPERIMENTAL CONFIGURATION DESCRIBED IN SECTION VI-A.

Metric	Value
Total Parameters	$\sim$115 million
Checkpoint Size	$\approx$480 MB
Peak GPU Memory (Training)	$<$8 GB
Peak GPU Memory (Inference)	$<$4 GB
Training Time (3 epochs)	$\approx$4 hours
Inference Time (per video)	$<$1 second
Cloud Training Cost	$\approx$\$2
Cloud Inference Cost	Negligible ($<$\$0.001 per video)
Hardware Requirement	Single NVIDIA T4 (16 GB VRAM)

at a computational cost that is three to four orders of magnitude smaller.

Peak training memory of less than 8 GB means that TTV-HRM uses fewer than half of the T4's 16 GB of available VRAM during training, leaving headroom for increased batch sizes or higher resolutions in future experiments. Peak inference memory of less than 4 GB is particularly significant: it means TTV-HRM can run on a wide range of consumer-grade GPUs, including the NVIDIA GTX 1080 Ti (11 GB), RTX 2070 (8 GB), and even some 6 GB configurations with careful batch management, greatly expanding the hardware base on which the model can be deployed beyond T4 cloud instances.

The sub-second inference time per video includes all stages of the pipeline: text encoding, latent initialization, hierarchical refinement (average 2.1 steps with early stopping), and detokenization. This makes real-time or near-real-time generation feasible for interactive applications, educational tools, and rapid creative prototyping — use cases that would be entirely impractical with diffusion-based systems requiring tens of seconds or minutes per generation. The total training cost of approximately \$2 at standard T4 cloud rates (\$0.50/hour for 4 hours) is low enough that the full training pipeline can be run multiple times in a single day for hyperparameter exploration, ablation studies, or fine-tuning on new datasets, all without significant financial barrier. This accessibility is a central contribution of TTV-HRM: it brings text-to-video experimentation within reach of independent researchers, students, and small academic groups who do not have access to multi-GPU clusters or substantial cloud computing budgets.

VII. DISCUSSION

The experimental results presented in Section VI support the central claim of this work: that hierarchical reasoning, implemented through interleaved high-level and low-level transformer blocks with bidirectional guidance and learned early stopping, enables coherent text-to-video generation in severely resource-constrained settings. In this section, we interpret these findings more deeply, examine the strengths and limitations of the current system honestly, situate TTV-HRM within the broader landscape of generative video research, and outline the most promising directions for future work.

A. Interpretation of Results

The steady improvement in reconstruction MSE, frame-wise FID, and average reasoning steps across the three training epochs tells a consistent and interpretable story about how TTV-HRM learns. In the first epoch, the model is primarily learning the basic mapping between text conditioning signals and broad spatiotemporal structure. The reconstruction MSE of 0.45 and FID of 120.5 reflect a model that is generating plausible low-frequency content — correct rough colors and approximate object locations — but lacks fine-grained spatial and temporal precision. The average step count of 2.8 at this stage indicates that the convergence predictor is not yet confident in its assessments, correctly identifying that the latent representations are still changing substantially between iterations and therefore withholding early termination.

By the second epoch, the model has refined its understanding of the tokenizer-detokenizer pathway and the reasoning engine is producing more stable latent trajectories. The MSE drop from 0.45 to 0.32 and the FID improvement from 120.5 to 85.2 reflect this stabilization. The reduction in average steps from 2.8 to 2.3 is a direct consequence: as the latent representations converge more quickly within the refinement loop, the predictor correctly identifies this earlier convergence and terminates sooner.

The final epoch brings further improvement across all three metrics. The FID of 62.1 represents a 48% improvement over the epoch-1 baseline, which is a substantial gain given that the model has only seen the 45-pair training set three times in total. The average step count of 2.1 means that for the majority of generation calls, the model terminates after just two reasoning iterations rather than three, confirming that the hierarchical refinement process is genuinely converging and that the early stopping mechanism is working as intended. The fact that FID improves even as the step count decreases is particularly important: it demonstrates that the efficiency gain from early stopping is not achieved by sacrificing output quality, but rather by the model learning to reach a good solution more efficiently.

The qualitative results reinforce these quantitative findings. The model's ability to generate a smoothly translating red ball and a consistently rotating blue square from text prompts alone — using only 45 training pairs and a 115-million parameter model — demonstrates that the hierarchical reasoning mechanism is learning genuine compositional generalization rather than simple memorization. The preservation of object identity, color consistency, and motion continuity across 8 frames is a non-trivial achievement at this scale, and it reflects the specific design decision to couple the high-level and low-level transformer blocks through bidirectional cross-attention. Without this coupling, a flat architecture of comparable size would be expected to produce significantly less temporally coherent outputs, as there would be no mechanism to enforce global structural consistency across the local detail refinements performed frame by frame.

B. Strengths

TTV-HRM exhibits four key strengths that collectively define its contribution to the field of resource-constrained generative

video modeling.

The first strength is the **hierarchical reasoning mechanism** itself. By separating the generation process into a high-level stage that establishes global semantic structure and a low-level stage that refines fine-grained spatiotemporal detail, TTV-HRM achieves a quality-efficiency trade-off that would be difficult to replicate with a flat architecture. The bidirectional coupling between the two levels — top-down guidance from H to L, and bottom-up feedback from L to H — ensures that the two levels of abstraction remain mutually consistent throughout the refinement process, which is the key mechanism behind the model's temporal coherence. This design is directly inspired by established theories of hierarchical visual processing [13, 17] and represents a principled integration of cognitive science insights into a practical engineering solution.

The second strength is **genuine resource accessibility**. The combination of a compact architecture, TF32 precision, gradient checkpointing, streaming data loading, and learned early stopping produces a system that trains in 4 hours and infers in under 1 second on a $0.50/hour cloud instance. This is not merely a marginal improvement over existing systems — it represents a categorical shift in who can participate in T2V research. A graduate student with a $10 cloud budget can run five complete training experiments with TTV-HRM. The same budget would not cover a single inference call on some state-of-the-art T2V systems.

The third strength is **modularity**. The clear separation of the text encoder, video tokenizer, hierarchical reasoning engine, and convergence predictor into independent components means that each can be studied, improved, or replaced in isolation. An ablation study removing the high-level layer and replacing it with a flat transformer baseline would directly quantify the contribution of the hierarchical mechanism. A swap of the 3D convolutional tokenizer for a more sophisticated video autoencoder such as MAGVIT [37] could be performed without any changes to the reasoning engine. This modularity accelerates the research cycle and makes TTV-HRM a useful platform for exploring individual components of T2V systems, not just an end-to-end generation tool.

The fourth strength is **effective text-video alignment despite limited data**. The cross-attention conditioning applied at every reasoning step ensures that the text embeddings from the encoder actively guide the latent refinement throughout the entire generation process. This persistent conditioning is what allows the model to produce semantically correct outputs — correct object color, correct motion direction, correct spatial placement — from just 45 training pairs. In contrast, a system that applies text conditioning only at initialization would quickly lose semantic fidelity as the latent representation evolves through multiple refinement steps without continued textual guidance.

C. Limitations

The current implementation of TTV-HRM is deliberately scoped for proof-of-concept validation, and it carries several limitations that are important to acknowledge clearly.

The most immediately visible limitation is **low output resolution**. At 32×32 pixels, generated frames are visibly pixelated and lack the fine texture, detail, and visual richness that would be expected of a practical video generation system. Objects appear as blobs of approximately correct color and shape rather than as photorealistic entities. This is an intentional design choice for the current stage of development — the hierarchical reasoning mechanism can be validated at 32×32 just as well as at higher resolutions, and the lower resolution enables faster experimentation — but it means that the current outputs are not suitable for any application that requires visually compelling content.

The second limitation is **restricted motion complexity**. The model has been validated primarily on simple, single-object motions such as translations and rotations. Multi-object scenes, object interactions, occlusions, deformations, and complex physical dynamics are all outside the current scope of what TTV-HRM can reliably generate. This limitation stems partly from the resolution constraint — many complex motion patterns are simply not distinguishable at 32×32 — and partly from the small and relatively simple training dataset, which does not provide sufficient examples of complex multi-object dynamics for the model to learn from.

The third limitation is **limited generalization** due to the small training set of 50 pairs. While the model demonstrates meaningful learning on the concepts represented in the training data, its ability to generalize to diverse real-world prompts describing novel combinations of objects, backgrounds, and motions is necessarily limited. The text encoder, which is initialized from GPT-2 weights and fine-tuned jointly, similarly benefits from a larger and more diverse text-video dataset. Scaling the training data to thousands or tens of thousands of pairs would be expected to substantially improve generalization, but would also require proportionally more compute.

The fourth limitation is the **absence of perceptual or adversarial losses**. Training with MSE alone encourages the model to minimize average pixel-level error, which can lead to blurry outputs because the average of multiple plausible reconstructions is a blurry intermediate rather than any specific sharp frame. Perceptual losses [55] and adversarial training [49] have been shown to produce sharper and more visually appealing outputs by encouraging the model to match high-level feature statistics and fool a discriminator rather than simply minimize pixel error. Their absence in TTV-HRM is a deliberate choice to keep training stable and simple, but it contributes to the somewhat soft appearance of generated frames.

D. Comparison with Related Work

Situating TTV-HRM within the broader T2V literature requires distinguishing between two very different regimes of operation. Large-scale systems such as Make-A-Video [19], Imagen Video [21], and VideoCrafter [20] operate in a regime of massive scale: billions of parameters, hundreds of millions of training examples, and multi-GPU clusters running for weeks. These systems produce outputs of remarkable visual quality

and semantic richness, but they are inaccessible to the vast majority of the research community. TTV-HRM operates in a fundamentally different regime: 115 million parameters, 45 training pairs, and a single commodity GPU for 4 hours. The two regimes are not in direct competition — they serve different purposes and different communities.

Within the limited set of prior work that has specifically targeted resource-constrained T2V generation, TTV-HRM's hierarchical reasoning mechanism represents a novel contribution. Most lightweight video generation approaches achieve efficiency through architectural compression, pruning, or distillation of larger models [56, 58], rather than through a principled architectural design that is natively efficient. TTV-HRM, by contrast, is designed from the ground up to be efficient, with the hierarchical reasoning mechanism providing a structured way to spend a fixed computational budget progressively on global structure first and fine detail second.

The learned early stopping mechanism also distinguishes TTV-HRM from most iterative refinement approaches in the literature, which use a fixed number of refinement steps [**?** 32]. The ability to adaptively terminate refinement when the latent representation has converged — rather than always running to a fixed number of steps — is a practical efficiency improvement that is directly applicable to other iterative generative frameworks beyond T2V.

E. Future Directions

Several promising extensions of TTV-HRM emerge naturally from the current work, ordered here from most immediate to most ambitious.

The most directly actionable extension is **resolution scaling**. The hierarchical architecture is designed to support progressive detail refinement, and scaling the spatial resolution of the video tokenizer and detokenizer to 64×64, 128×128, or higher is straightforward in principle, requiring only additional convolutional blocks and proportionally more compute. Integration of a super-resolution module [30] as a post-processing step could further upscale outputs to HD or 4K resolution while preserving the temporal consistency established by the hierarchical reasoning engine, without requiring the core model to be retrained at full resolution.

A second important extension is **training data scaling**. Replacing the 50-pair proof-of-concept dataset with a larger and more diverse collection — such as a substantial subset of WebVid-10M [52] or HowTo100M [53] — would be expected to substantially improve the model's generalization to diverse real-world prompts. Streaming data loading, already implemented in TTV-HRM, makes it feasible to train on datasets that are too large to fit in memory, and the model's efficient architecture means that training on a larger dataset would still be tractable on commodity hardware, albeit requiring more wall-clock time.

A third extension is the **adoption of FlashAttention-2** [10] in the transformer blocks of the hierarchical reasoning engine. FlashAttention-2 reformulates attention computation to exploit GPU memory hierarchy more efficiently, reducing memory bandwidth requirements and enabling larger effective batch sizes or longer sequence lengths within the same VRAM budget. This is particularly relevant for scaling to higher resolutions, where the number of latent tokens produced by the tokenizer grows quadratically with spatial resolution.

A fourth direction is **enrichment of the generation conditioning**. The current model conditions generation only on text, but richer conditioning signals — such as reference images, audio descriptions, camera motion specifications, or physics-based priors — could substantially expand the model's control over generated content. Controllable camera motion is particularly relevant for content creation applications, where users often want to specify not just what happens in a video but how it is filmed.

Finally, the most ambitious direction is **hybridization with diffusion-based refinement**. The hierarchical reasoning engine currently produces outputs by iterative transformer-based refinement, which is efficient but produces softer outputs than diffusion-based approaches. Adding a lightweight latent diffusion refinement step [30, 20] applied to the final latent sequence produced by the reasoning engine could substantially improve visual sharpness and realism, leveraging the efficiency of hierarchical reasoning for coarse generation and the visual quality of diffusion for fine-grained refinement, without incurring the full computational cost of a pure diffusion approach.

VIII. Conclusion

This paper has presented TTV-HRM, a lightweight and computationally efficient framework for text-to-video generation on commodity hardware. The central contribution is a novel hierarchical reasoning mechanism that decomposes the video generation process into iterative refinement stages through interleaved high-level and low-level transformer blocks, each equipped with rotary positional embeddings, SwiGLU feed-forward networks, and bidirectional cross-attention conditioning on the input text prompt. A learned convergence predictor enables early stopping, reducing the average number of reasoning iterations from the maximum of 3 to approximately 2.1 in practice, yielding a 25% reduction in reasoning-stage compute at inference time without any loss of generation quality.

The complete TTV-HRM system — comprising a GPT-2-based text encoder, a 3D convolutional video tokenizer and detokenizer, the hierarchical reasoning engine, and a convergence predictor — contains approximately 115 million parameters, occupies a 480 MB checkpoint, and trains in roughly 4 hours on a single NVIDIA T4 GPU at a total cloud cost of approximately $2. Inference takes under one second per video clip. These figures place TTV-HRM in a categorically different computational regime from state-of-the-art T2V systems, which require multi-GPU clusters, billions of parameters, and massive training datasets. By demonstrating that coherent, text-conditioned video synthesis is achievable within these tight constraints, TTV-HRM makes T2V experimentation accessible to a substantially broader community of researchers, students, and practitioners.

The experimental results confirm that TTV-HRM learns effectively from limited data. Over three training epochs on 45 video-text pairs, the model achieves a 38% reduction in reconstruction MSE and a 48% improvement in frame-wise FID, reaching a final FID of 62.1 that reflects genuine learning of the visual distribution at 32×32 resolution. Qualitative evaluation confirms semantic alignment, temporal coherence, object persistence, and basic motion fidelity across a range of held-out text prompts describing simple dynamic scenes.

Beyond the immediate results, the broader significance of this work lies in what it suggests about the design principles for accessible generative AI. The key insight is that architectural structure — specifically, the hierarchical decomposition of generation into coarse-to-fine refinement stages — can substitute for scale in enabling coherent multimodal generation. A well-structured 115-million parameter model with hierarchical reasoning can produce outputs that are qualitatively coherent in ways that a flat architecture of the same size cannot, because the hierarchical structure provides an inductive bias toward the kind of global-to-local processing that characterizes both human visual perception and well-composed visual content. This principle — that structure can substitute for scale — is likely to remain relevant as the field continues to develop more efficient approaches to generative modeling.

The limitations of the current system are clearly understood and primarily reflect deliberate scoping decisions made to validate the core hierarchical reasoning mechanism within a constrained compute budget. The most important limitations — low output resolution, restricted motion complexity, and limited training data — each have clear paths forward through resolution scaling, dataset expansion, and the architectural extensions described in Section VII. The modular design of TTV-HRM is specifically intended to facilitate these extensions, with each component replaceable and improvable in isolation.

We believe that hierarchical reasoning principles, combined with ongoing improvements in efficient attention mechanisms [10], spatiotemporal representations [37], and accessible training infrastructure, will play an increasingly important role in democratizing access to powerful multimodal generative AI. TTV-HRM is a step in this direction: a proof of concept that demonstrates the viability of the approach, a platform for further experimentation, and an invitation to the broader research community to build on and extend these ideas toward richer, more capable, and more universally accessible video generation systems.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

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

[5] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 8748–8763, 2021.

[6] G. Wang, J. Li, Y. Sun, X. Chen, C. Liu, Y. Wu, M. Lu, S. Song, and Y. Abbasi Yadkori, "Hierarchical reasoning model," *arXiv preprint arXiv:2506.21734*, 2025.

[7] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "RoFormer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[8] N. Shazeer, "GLU variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[9] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16344–16359, 2022.

[10] T. Dao, "FlashAttention-2: Faster attention with better parallelism and work partitioning," *arXiv preprint arXiv:2307.08691*, 2023.

[11] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[12] B. Zhang and R. Sennrich, "Root mean square layer normalization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] Y. Pan, T. Yao, Y. Li, T. Wang, and C.-W. Ngo, "Hierarchical vision transformer with channel attention and relation modeling," *IEEE Transactions on Image Processing*, vol. 32, pp. 792–805, 2022.

[14] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10012–10022, 2021.

[15] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2117–2125, 2017.

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241, 2015.

[17] D. J. Felleman and D. C. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral Cortex*, vol. 1, no. 1, pp. 1–47, 1991.

[18] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[19] U. Singer, A. Polyak, T. Hayes, X. Yin, J. An, S. Zhang, Q. Hu, H. Yang, O. Ashual, O. Gafni *et al.*, "Make-A-Video: Text-to-video generation without text-video data," *arXiv preprint arXiv:2209.14792*, 2022.

[20] H. Chen, Y. Liu, W. Xie, T. Zhang, H. Li, J. Ding, Y. Yang, and Y. Zhang, "VideoCrafter1: Open diffusion models for high-quality video generation," *arXiv preprint arXiv:2310.19512*, 2023.

[21] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, and T. Salimans, "Imagen video: High definition video generation with diffusion models," *arXiv preprint arXiv:2210.02303*, 2022.

[22] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video diffusion models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 8633–8646, 2022.

[23] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts *et al.*, "Stable video diffusion: Scaling latent video diffusion models to large datasets," *arXiv preprint arXiv:2311.15127*, 2023.

[24] W. Yan, Y. Zhang, P. Abbeel, and A. Srinivas, "VideoGPT: Video generation using VQ-VAE and transformers," *arXiv preprint arXiv:2104.10157*, 2021.

[25] W. Hong, M. Ding, W. Zheng, X. Liu, and J. Tang, "CogVideo: Large-scale pretraining for text-to-video generation via transformers," *arXiv preprint arXiv:2205.15868*, 2022.

[26] C. Wu, J. Liang, L. Ji, F. Yang, Y. Fang, D. Jiang, and N. Duan, "NÜWA: Visual synthesis pre-training for neural visual world creation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[27] R. Villegas, M. Babaeizadeh, P.-J. Kindermans, H. Moraldo, H. Zhang, M. T. Saffar, S. Castro, J. Kunze, and D. Erhan, "Phenaki: Variable length video generation from open domain textual descriptions," *arXiv preprint arXiv:2210.02399*, 2022.

[28] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[29] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "MoCoGAN: Decomposing motion and content for video generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1526–1535, 2018.

[30] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

[31] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[32] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[33] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, "Cascaded diffusion models for high fidelity image generation," *Journal of Machine Learning Research*, vol. 23, no. 47, pp. 1–33, 2022.

[34] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with CLIP latents," *arXiv preprint arXiv:2204.06125*, 2022.

[35] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 4489–4497, 2015.

[36] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[37] L. Yu, Y. Cheng, K. Sohl-Dickstein, V. Kumar, L. Huang, D. Jiang, and W. Cogswell, "MAGVIT: Masked generative video transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10459–10469, 2023.

[38] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?" in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 139, pp. 813–824, 2021.

[39] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[43] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

[45] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[46] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[48] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," *arXiv preprint arXiv:1604.06174*, 2016.

[49] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[50] J. Xu, T. Mei, T. Yao, and Y. Rui, "MSR-VTT: A large video description dataset for bridging video and language," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5288–5296, 2016.

[51] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the Kinetics dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6299–6308, 2017.

[52] M. Bain, A. Nagrani, G. Zisserman, and A. 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 (ICCV)*, pp. 1728–1738, 2021.

[53] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic, "HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2630–2640, 2019.

[54] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, A. Courville, and H. Larochelle, "Describing videos by exploiting temporal structure," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 4507–4515, 2015.

[55] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 694–711, 2016.

[56] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 3730–3738, 2015.

[57] A. Umar, "FaseehGPT: A lightweight transformer model for Arabic text generation with enhanced morphological understanding," *Engineering Archive*, 2025.

[58] R. Kovler, G. Rotman, and R. Kimmel, "MobileStyleGAN: A lightweight convolutional neural network for high-fidelity image synthesis," *arXiv preprint arXiv:2104.04767*, 2021.

[59] S. Ge, T. Hayes, H. Yang, X. Yin, G. Pang, D. Jacobs, J.-B. Huang, and D. Parikh, "Long video generation with time-agnostic VQGAN and time-sensitive transformer," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[60] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[61] NVIDIA Corporation, "Accelerating AI training with NVIDIA TF32 tensor cores," *NVIDIA Technical Blog*, 2020. [Online]. Available: https://developer.nvidia.com/blog/accelerating-ai-training-with-tf32-tensor-cores/

[62] A. Umar, "LatentRecurrentDepthLM: An open-source framework for recurrent-depth language models with controllable test-time compute," *Engineering Archive*, 2

[63] N. Patry, "SafeTensors," *Hugging Face*, 2022. [Online]. Available: https://github.com/huggingface/safetensors

[64] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision (ECCV)*, pp. 213–229, 2020.