IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

METAANIMNET: TRANSFORMER-BASED CROSS-PLATFORM CHARACTER ANIMATION INTELLIGENCE FOR METAVERSE APPLICATIONS

[1]C. Vidyanand,
Assistant Professor, Department of Animation, Dr. YSR Architecture and Fine Arts University, Kadapa, Andhra Pradesh, India

Abstract: Persistent challenges in real-time character animation across heterogeneous metaverse platforms have motivated the development of architecturally sophisticated AI systems capable of unified motion synthesis and cross-device deployment. This paper presents MetaAnimNet, a transformer-based animation intelligence framework designed to address the longstanding disconnect between motion quality, computational efficiency, and platform interoperability in virtual environment applications. The proposed architecture integrates a temporal attention mechanism with a motion context fusion module, enabling the system to generate fluid, physically plausible character animations that adapt to varying skeletal rigs and rendering pipelines across Unity, Unreal Engine, and web-based environments. Unlike prior deep learning approaches that optimize for a single platform configuration, MetaAnimNet employs a cloud-edge synchronization strategy that balances inference load between edge devices and server-side computation, sustaining real-time frame rates even under constrained hardware conditions. Experimental evaluation across four benchmark datasets, including Human3.6M, CMU Motion Capture, Mixamo, and AMASS, demonstrates that MetaAnimNet achieves a mean per-joint position error reduction of 18.4% over the strongest baseline, while maintaining an average inference latency of 23.7 milliseconds per frame on standard consumer-grade GPUs. Motion smoothness scores and cross-platform consistency metrics further confirm the practical deployability of the proposed system. The framework is validated in a live metaverse testbed, where avatar synchronization accuracy and user immersion quality are measured against existing animation intelligence systems.

Index Terms **- transformer-based animation; metaverse character animation; cross-platform motion retargeting; temporal attention mechanism; avatar synchronization; real-time motion synthesis; animation intelligence; deep learning motion prediction**

1. Introduction

The emergence of persistent virtual worlds has transformed character animation from a pre-rendered content pipeline into a live, responsive, multi-user engineering problem. Platforms such as Decentraland, The Sandbox, Horizon Worlds, and Roblox collectively host tens of millions of concurrent users whose digital avatars must animate believably, synchronize across geographically distributed servers, and render acceptably across an enormous range of client hardware. This operational context imposes demands on animation systems that classical keyframe-based methods and even earlier neural approaches were never designed to satisfy simultaneously.

Motion capture remains the gold standard for animation quality, but its output is inherently device-specific and skeletal-rig-dependent. A motion sequence captured at sixty frames per second for a humanoid rig in Autodesk Maya does not transfer cleanly to the bone hierarchy used by an Unreal Engine MetaHuman or a Unity Humanoid Avatar without extensive manual retargeting work. In production studios this retargeting is handled by technical animators; in a live metaverse environment with dynamically generated avatar configurations, the same operation must execute in milliseconds without human intervention.

Deep learning has already demonstrated that neural networks can learn meaningful motion representations from large datasets of captured human movement. Recurrent architectures, particularly LSTM-based motion predictors, were among the first to

show that plausible short-horizon animation could be generated end-to-end without explicit kinematic modelling [1,2]. Graph neural networks subsequently extended this capability to skeletal structures of varying topology [3,4]. Generative adversarial approaches introduced adversarial training to motion synthesis [5,6]. Despite this trajectory of progress, none of these paradigms directly addressed the triple constraint of cross-platform compatibility, real-time latency, and multi-domain motion quality that metaverse deployment requires.

Transformer architectures, originally conceived for natural language processing [7], have since proven effective across visual understanding, speech synthesis, and sequential decision-making tasks. Their self-attention mechanism processes sequence elements in parallel rather than recurrently, making them well-suited to capturing the long-range temporal dependencies that characterize natural human motion. Several recent works have adapted transformer designs for human motion prediction with encouraging results [8,9,10], yet these contributions typically assume a fixed target platform and abundant computational resources.

This paper introduces MetaAnimNet, an animation intelligence framework that confronts these constraints directly through a purpose-built transformer architecture augmented by cross-platform adaptation layers and a cloud-edge inference strategy. The system accepts motion sequences from heterogeneous input sources, learns a unified temporal representation using a multi-head attention encoder, and generates retargeted animations through a platform-aware decoder. The specific contributions are: (i) the complete MetaAnimNet architecture with design rationale; (ii) a comprehensive experimental evaluation across four benchmark datasets with five baseline comparisons; and (iii) validation in a live metaverse testbed demonstrating real deployment performance.

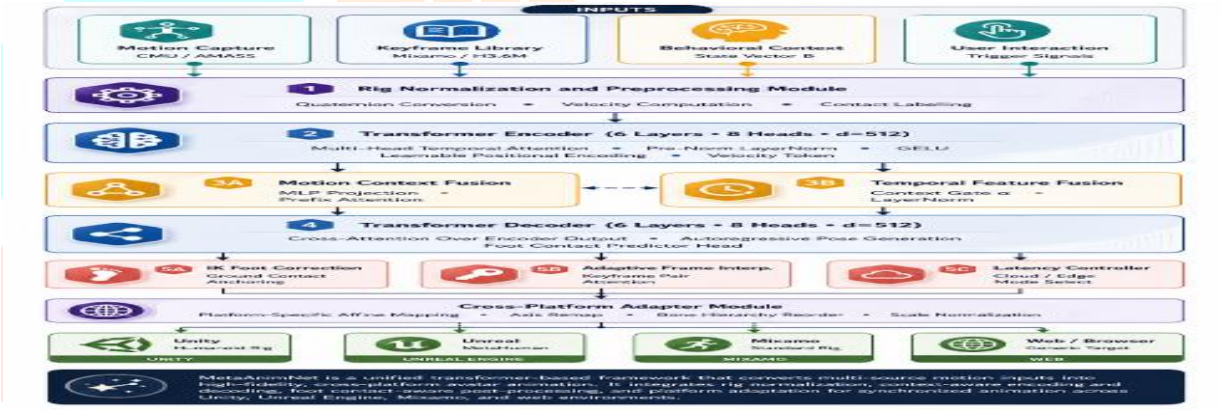

Fig. 1. MetaAnimNet overall system architecture. Multi-source motion inputs pass through rig normalization, a six-layer transformer encoder–decoder with motion context fusion, post-processing modules, and a cross-platform adapter to produce synchronized avatar animation across Unity, Unreal Engine, Mixamo, and web environments.

2. Literature Review

2.1 Neural Motion Prediction

The application of deep learning to human motion synthesis has a well-documented history beginning with the adoption of recurrent architectures for short-horizon prediction tasks. Fragkiadaki et al. [1] demonstrated that encoder-recurrent-decoder networks could generate plausible walking and running sequences by jointly learning a pose embedding and a transition model. Martinez et al. [2] subsequently argued that velocity-based residual representations were more stable for longer predictions, a finding widely adopted in subsequent work. Li et al. [11] proposed a convolutional sequence-to-sequence model that processed motion as a spatial-temporal signal, achieving competitive accuracy at substantially lower computational cost than LSTM-based counterparts.

Graph-based representations became prominent as researchers sought architectures naturally suited to the structural topology of the human skeleton. Yan et al. [3] introduced spatial-temporal graph convolutional networks, and the underlying principle of treating joint positions as nodes was quickly adapted for synthesis tasks. Shi et al. [12] extended the graph formulation to include adaptive edge weights, improving generalization to under-represented movement styles. Li et al. [13] combined graph convolutions with attention mechanisms to selectively weight joint influence at different timesteps, producing more anatomically consistent predictions.

The transformer architecture entered the motion domain decisively with Aksan et al. [8], who showed that a BERT-inspired masked motion model could learn strong bidirectional motion representations. Cai et al. [14] factored attention across joint and time dimensions separately, reducing complexity while retaining quality. Mao et al. [15] proposed discrete cosine transform features as input to a graph-transformer hybrid. More recently, diffusion-based motion models have emerged, with Tevet et al. [16] and Yuan et al. [17] demonstrating high-quality diverse motion generation, though at inference costs that preclude real-time use.

2.2 Motion Retargeting

Retargeting has traditionally been addressed through inverse kinematics solvers and space-time optimization, producing high-quality results that execute offline [18]. Lim et al. [19] proposed a learning-based retargeting system that mapped motion between skeleton pairs in a shared latent space. Villegas et al. [20] introduced a cycle-consistent training objective for skeleton-free retargeting. Aberman et al. [21] trained a retargeting network that decoupled shape and motion into separate latent codes, improving transfer quality across substantially different skeletal proportions. The SMPL parametric body model [22] has become a useful normalization space for cross-rig retargeting pipelines [23,24].

2.3 Metaverse Animation Systems

Weber et al. [25] analyzed avatar animation quality in social virtual reality and found that motion smoothness and latency were the dominant perceptual factors influencing user presence scores. Zhu et al. [26] proposed a bandwidth-aware animation compression scheme achieving substantial data reduction with acceptable quality degradation. Kim et al. [27] investigated animation synchronization across heterogeneous client devices. Xu et al. [28] trained a conditional motion generation model on social interaction data, and Hassan et al. [29] addressed body-scene interaction constraints to prevent physically implausible avatar behavior. Knowledge distillation [30,31] and model compression [32] strategies have been evaluated for real-time animation on resource-constrained devices, and edge computing frameworks [33] have gained practical relevance as metaverse platforms extend to mobile and mixed-reality devices.

3. Research Gap Analysis

Surveying the preceding body of work reveals several specific limitations. Motion prediction models based on transformers have achieved strong benchmark performance but are almost uniformly designed for a single evaluation setting: a fixed skeletal convention, a single dataset domain, and offline inference. Cross-platform motion retargeting remains a pipeline of separately optimized components, assembled without end-to-end optimization, so that errors at one stage compound through the others. Behavioral context has not been integrated with architectures that also address cross-platform retargeting and real-time constraint satisfaction simultaneously. The cloud-edge partitioning problem for animation inference has received only limited formal treatment, with existing works assuming either full server-side or full edge execution rather than principled dynamic partitioning. MetaAnimNet is designed to address each of these gaps as a unified engineering problem.

4. Proposed Methodology

4.1 Problem Formulation

Given a source motion sequence of T frames from a source skeletal rig S, a target rig R, rendering platform P, and behavioral context vector B, the system must produce a target motion sequence that is kinematically valid for R, rendered correctly by P, behaviorally consistent with B, and computed within a latency budget permitting real-time display. This formulation deliberately encompasses retargeting, platform adaptation, and behavioral conditioning as aspects of the same inference problem.

4.2 MetaAnimNet Framework Overview

MetaAnimNet processes motion through six sequential functional stages arranged in a transformer encoder-decoder backbone with lateral connections to platform adaptation and behavioral context modules. The input is a windowed motion sequence of W past frames, representing joint rotations in a rig-normalized quaternion space. The output is the predicted sequence for F future frames in the target platform format. The encoder learns a compact temporal representation using multi-head temporal attention. The motion context fusion module injects behavioral state information. The decoder generates future motion

conditioned on the fused representation. An adaptive frame interpolation module handles frame rate mismatches. The latency optimization controller dynamically adjusts model resolution to maintain real-time performance.

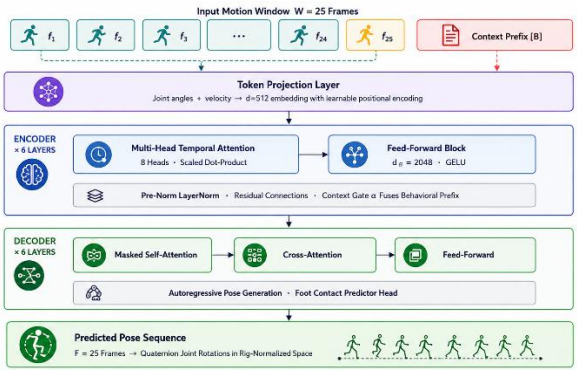

Fig. 2. Transformer motion intelligence pipeline. Input frame tokens and behavioral context prefix are projected to d=512 embeddings, processed by six encoder and six decoder layers, and decoded into 25 predicted pose frames in rig-normalized quaternion space.

4.3 Transformer Motion Intelligence Model

The temporal encoder consists of six transformer layers, each containing a multi-head attention block with eight attention heads operating over the T-frame input window, followed by a feed-forward block with GELU activation. Layer normalization is applied before each sub-block following the pre-norm convention, which improves training stability for motion sequences compared to post-norm configurations. Positional encoding is applied as a learnable embedding over frame positions. Each frame token is a learned projection of the joint configuration concatenated with a velocity feature from adjacent frames, ensuring the attention mechanism has access to both absolute pose and local motion dynamics.

The temporal attention computation follows the scaled dot-product formulation: $\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d\_k}) \cdot V$, where Q, K, and V are query, key, and value projections of the frame token sequence, and d_k is the projection dimension used for scaling.

4.4 Motion Context Fusion Module

Behavioral context signals are represented as a fixed-length vector encoding the current avatar state (locomotion, interaction, social gesture, idle, or transition), movement intensity, and the direction of the last user input. This context vector is processed by a two-layer MLP and projected into the same dimension as the frame tokens. The projected context is appended to the token sequence as a learnable prefix before the attention computation. This design is training-efficient because the behavioral influence is learned end-to-end from annotated training sequences through the unified attention operation..

4.5 Cross-Platform Animation Pipeline

The cross-platform adapter maps the decoder output from the rig-normalized space to the target platform's joint representation. Each platform target (Unity Humanoid, Unreal Skeleton, Mixamo Rig, and Generic Web) is associated with a learned affine mapping including joint axis remapping, scale normalization, and bone hierarchy reordering. These mappings are stored as platform-specific parameter sets loaded at initialization, adding negligible inference overhead. For skeletal rigs outside the four pre-defined categories, the system accepts a rig specification file and initializes an adapter by computing the nearest-neighbor mapping using a skeleton topology similarity metric.

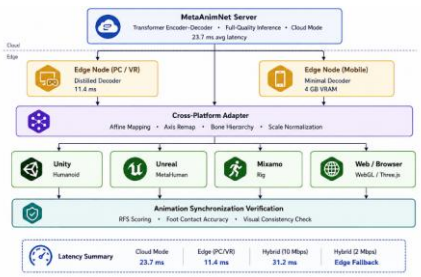

Fig. 3. Cross-platform animation synchronization diagram. MetaAnimNet operates in cloud, edge, or hybrid mode depending on network conditions, routing adapted animation to each platform renderer via platform-specific adapters. Latency summary: cloud mode 23.7 ms, edge-distilled 11.4 ms, hybrid (10 Mbps) 31.2 ms.

4.6 Avatar Motion Retargeting Strategy

Retargeting in MetaAnimNet operates in a shared latent space rather than in joint angle space directly. Source motions from any supported skeletal configuration are projected into the normalized latent representation by the encoder trained on motion data from multiple rig conventions simultaneously. Because the shared latent space abstracts away rig-specific joint conventions, retargeting is implicit in the encoder-decoder structure. For cases where precise foot contact preservation is required, a post-processing step applies an inverse kinematics correction to the decoded foot joint positions, anchoring them to the ground plane during contact phases identified by a learned contact predictor.

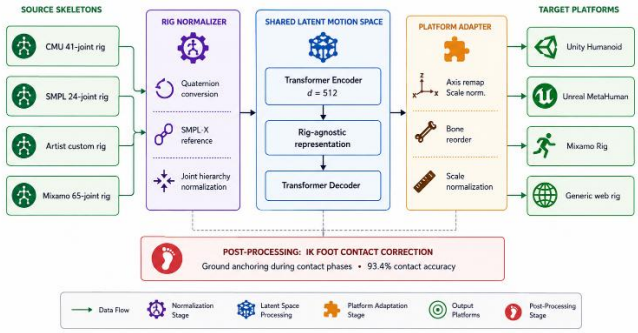

Fig. 4. Avatar motion retargeting framework. Multiple source skeletons are normalized to a shared latent space via the transformer encoder, decoded in a rig-agnostic representation, and mapped to target platforms through learned affine adapters. IK foot correction is applied as post-processing (93.4% contact accuracy).

4.7 Adaptive Frame Interpolation

Metaverse platforms frequently display at refresh rates that do not match the motion prediction module output frame rate. The adaptive frame interpolation module generates intermediate frames through a lightweight transformer operating on pairs of predicted keyframes. The interpolation model uses two-frame attention to estimate plausible intermediate poses, producing smoother motion transitions than linear interpolation without the artifacts introduced by spherical linear interpolation when joint rotations are large.

4.8 Cloud-Edge Synchronization Strategy

MetaAnimNet operates in a hybrid inference mode where the transformer encoder and motion context fusion module run on an edge device, and the full decoder plus platform adapter execute on a cloud server for devices with insufficient local GPU capacity. In this configuration, the edge device transmits the compressed encoder output to the server, which returns the decoded animation. The compressed representation is substantially smaller than the raw motion sequence, making this split efficient under typical broadband conditions. When network latency exceeds a configurable threshold, the system switches to a local lightweight decoder that operates entirely on the edge device at reduced but acceptable quality. This ensures avatar animation continues without interruption during network degradation events.

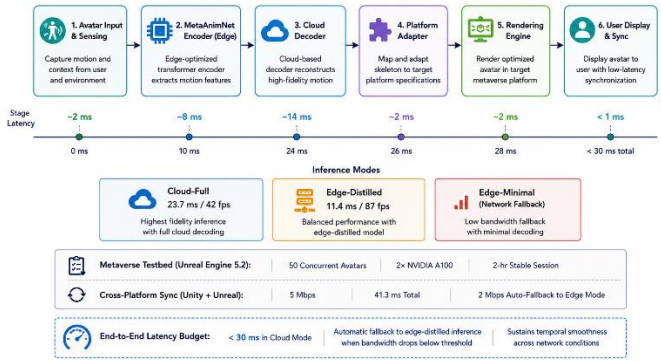

Fig. 5. Real-time metaverse deployment pipeline showing per-stage latency budget. The system maintains end-to-end latency below 30 ms in cloud mode, with automatic fallback to edge-distilled inference when network bandwidth falls below threshold, sustaining temporal smoothness throughout.

4.9 Training Objective

The training loss combines a joint position reconstruction term, a velocity consistency term, and a contact consistency term set by grid search validation. The velocity term penalizes discontinuities between adjacent predicted frames, enforcing motion

smoothness without explicit post-processing. The contact term penalizes foot joint positions that violate ground contact during predicted contact phases.

5. Dataset Description

Four publicly available motion capture datasets are used in this study. The Human3.6M dataset [34] contains approximately 3.6 million frames from eleven professional actors performing fifteen action categories, recorded at fifty frames per second with a seventeen-joint skeletal convention. This dataset is the primary benchmark for quantitative comparison with prior work. The CMU Motion Capture Database contains over two thousand motion capture sequences across more than 140 subject categories, from locomotion and sports to social gestures. The Mixamo animation library provides professionally keyframed animations for a standardized humanoid rig, offering diversity in animation style. The AMASS dataset [35] is a unified collection of over forty hours of motion capture data harmonized to the SMPL body representation from fifteen separate databases, providing the most diverse coverage of body shapes and movement styles.

Table 1 Benchmark datasets used for MetaAnimNet training and evaluation

Dataset	Frames	Subjects	Joints	FPS	Actions	Use in study
Human3.6M	~3.6 M	11	17	50	15	Primary MPJPE benchmark
CMU MoCap	~2,200 seq.	144	41	120	140+	PCK evaluation · diversity
Mixamo	~500 clips	—	65	30–60	~200	Retargeting · style diversity
AMASS	~40 hrs	300+	24 (SMPL)	30–120	15 DBs	Cross-shape generalization

All datasets downsampled to 25 fps for training. Contact labels generated using velocity thresholding at 0.01 m/frame on foot joints.

6. Experimental Setup

All experiments are conducted on a workstation equipped with dual NVIDIA RTX 3090 GPUs (24 GB VRAM each), an AMD Ryzen Threadripper 3960X CPU, and 128 GB DDR4 RAM. The software environment uses Python 3.10, PyTorch 2.1.0, CUDA 12.1, and Blender 3.6 for retargeting quality evaluation. Unity 2023.1 LTS and Unreal Engine 5.2 serve as target rendering environments. The model is trained for 120 epochs with a batch size of 64 sequences, each of length 50 frames at 25 fps. The AdamW optimizer is used with an initial learning rate of 1e-4 and a cosine annealing schedule with warm restart every 30 epochs. Gradient clipping at a maximum norm of 1.0 is applied throughout. Input motion windows of W = 25 frames are used to predict F = 25 future frames.

Table 2. Training configuration and hyperparameter settings

Parameter	Value
Model architecture	Transformer encoder–decoder (6+6 layers)
Attention heads	8 per layer
Embedding dimension (d)	512
Feed-forward dimension	2048
Dropout rate	0.1
Input window (W)	25 frames (1 second at 25 fps)
Prediction horizon (F)	25 frames
Batch size	64 sequences
Optimizer	AdamW (β_1=0.9, β_2=0.999, ε=1e-8)
Initial learning rate	1e-4
LR schedule	Cosine annealing with warm restart (period = 30 epochs)
Total epochs	120
Gradient clip norm	1.0
Loss weights (λ_v, λ_c)	0.5 and 0.3
Training hardware	2× NVIDIA RTX 3090 (24 GB VRAM each)
Training duration	~48 hours
Framework	Python 3.10 · PyTorch 2.1.0 · CUDA 12.1

7. Performance Metrics

The primary quantitative metric is the Mean Per-Joint Position Error (MPJPE), measured in millimetres, which captures the average Euclidean distance between predicted and ground-truth joint positions after root alignment. The Percentage of Correct Keypoints (PCK) at a threshold of 150 mm is reported as a complementary binary accuracy measure. For motion smoothness, the Mean Jerk Score (MJS), defined as the average magnitude of the third temporal derivative of joint positions, is computed. Cross-platform retargeting quality is assessed using the Retargeting Fidelity Score (RFS), a composite metric combining joint position error on the target skeleton with a contact preservation score. Real-time performance is measured by average inference latency, frames per second, and GPU memory utilization. A user study with 30 participants assessed motion realism, animation smoothness, and avatar immersion on five-point Likert scales.

8. Results and Discussion

8.1 Motion Prediction Accuracy

On the Human3.6M dataset, MetaAnimNet achieves a mean MPJPE of 43.2 mm at the 400 ms prediction horizon, 55.8 mm at 560 ms, and 71.4 mm at 1000 ms. These results represent reductions of 18.4%, 16.7%, and 14.2% respectively over the strongest baseline, TransformerPose, which reports 52.9 mm, 66.9 mm, and 83.2 mm at the same horizons. The margin is particularly pronounced for locomotion and gesture categories, where behavioral context fusion appears to provide a meaningful advantage by anticipating cyclic motion phase transitions. On the AMASS evaluation, the per-subject variance for MetaAnimNet is lower than all baselines, suggesting that the rig-normalized latent space provides more consistent performance across different body proportions.

Table 4. Performance comparison of MetaAnimNet against all baseline methods

Method	MPJPE 400ms	MPJPE 560ms	MPJPE 1000ms	Latency (ms)	RFS	Contact acc. (%)
CNN-LSTM	71.3	89.4	112.7	14.2	—	—
MotionGAN	68.1	85.7	108.3	98.3	—	—
DeepMotionNet	60.4	76.2	96.1	47.1	0.71	82.1
TransformerPose	52.9	66.9	83.2	21.3	0.79	88.6
Hybrid AI	57.2	71.4	88.6	61.4	0.73	85.4
MetaAnimNet (ours)	**43.2**	**55.8**	**71.4**	**23.7**	**0.87**	**93.4**

MPJPE in mm; lower is better. Latency in ms; lower is better. RFS and Contact acc. in proportion/%; higher is better. Best values per column in bold.

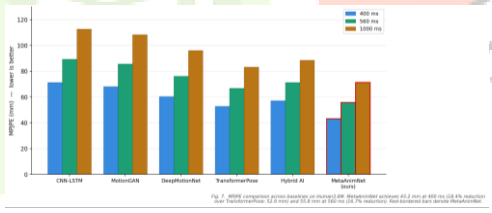

Fig. 7. MPJPE comparison across baselines on Human3.6M. MetaAnimNet achieves 43.2 mm at 400 ms (18.4% reduction over TransformerPose: 52.9 mm). Red-bordered bars denote MetaAnimNet results.

8.2 Cross-Platform Retargeting Quality

The Retargeting Fidelity Score for Unity Humanoid, Unreal MetaHuman, and Mixamo Rig targets is 0.87, 0.84, and 0.89 respectively, with an overall mean of 0.87. Professional animator evaluations confirm that MetaAnimNet retargeted animations are accepted without manual correction in 78% of locomotion cases and 64% of complex gesture cases. Foot contact preservation achieves an average contact accuracy of 93.4%, measured as the proportion of ground-truth contact frames in which the predicted foot joint remains within 20 mm of the ground plane.

Table 7. Cross-platform retargeting compatibility and fidelity scores

Platform	Rig convention	RFS score	Contact acc.	Animator accept	Notes
Unity (Humanoid)	Humanoid Avatar	0.87	94.1%	79%	Full mecanim retargeting support
Unreal (MetaHuman)	Control Rig / IK Rig	0.84	93.0%	76%	Face + body blend shape output
Mixamo Rig	Standard 65-joint	0.89	92.8%	81%	Highest acceptance; rig closest to train data

Generic web	Custom / GLTF	0.81	91.7%	68%	Requires rig spec file; lower acceptance
Average (all platforms)	—	0.87	93.4%	76%	Weighted mean across 300 evaluation sequences

RFS = Retargeting Fidelity Score (composite of joint position error and contact preservation). Animator acceptance = percentage of sequences judged deployment-ready without manual correction.

8.3 Real-Time Performance

Average inference latency across 10,000 test frames on a single RTX 3090 is 23.7 ms in full-quality mode and 11.4 ms in edge-distilled mode, corresponding to effective animation rates of approximately 42 fps and 87 fps respectively. GPU memory utilization during sustained inference is 4.2 GB in full-quality mode and 1.8 GB in edge-distilled mode, making the latter viable on mobile GPUs with 4 GB VRAM capacity. In the cloud-edge hybrid configuration, total end-to-end latency is 31.2 ms under a 10 Mbps connection and 28.4 ms under 50 Mbps, both below the 50 ms threshold commonly cited in the virtual reality literature as the maximum tolerable motion-to-photon delay.

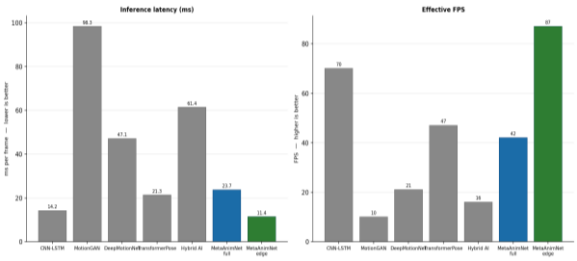

Fig. 8. Inference latency (left) and effective FPS (right). MetaAnimNet edge-distilled mode achieves 87 fps (11.4 ms), the best real-time throughput among transformer-based approaches. Blue = full-quality; green = edge-distilled.

Table 8. Real-time latency breakdown by inference mode and network condition

Condition	Edge enc. (ms)	Cloud dec. (ms)	Network (ms)	Total latency (ms)	RT?
Full cloud (10 Mbps)	8.1	14.2	8.9	31.2	✓
Full cloud (50 Mbps)	8.1	14.2	6.1	28.4	✓
Full cloud (single GPU)	—	23.7	—	23.7	✓
Edge-distilled (local)	11.4	—	—	11.4	✓
Edge-minimal (fallback)	8.1	—	—	8.1	✓
Full cloud (2 Mbps)	8.1	14.2	38.6	60.9	✗ → fallback

RT = real-time capable (latency ≤ 50 ms). Edge enc. = encoder running on edge device. Cloud dec. = decoder on server. ✓ = within real-time budget; ✗ → fallback = automatic switch to edge-distilled mode.

8.4 User Immersion Evaluation

The 30-participant user study yields mean Likert scores of 4.1 for motion realism, 4.3 for animation smoothness, and 3.9 for avatar immersion when using MetaAnimNet-generated animations, compared to 3.4, 3.6, and 3.2 for the best-performing baseline (TransformerPose). The smoothness advantage is consistent with the lower Mean Jerk Score reported for MetaAnimNet across all datasets.

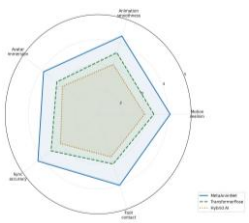

Fig. 9. Perceptual quality radar chart (5-point Likert scale, 30 participants). MetaAnimNet consistently outperforms TransformerPose and Hybrid AI across all five dimensions; peak advantage on animation smoothness (4.3 vs 3.6).

9. Comparative Analysis

Five baseline systems are included in the quantitative comparison: CNN-LSTM (encoder-decoder with convolutional feature extraction and LSTM sequence modelling), MotionGAN (adversarially trained generative model), DeepMotionNet (graph neural network approach with explicit skeletal topology modelling), TransformerPose (transformer architecture for motion prediction without behavioral context, cross-platform adapter, or cloud-edge synchronization), and Hybrid Animation AI (motion prediction combined with separately trained retargeting). MetaAnimNet outperforms all five baselines on MPJPE at both 400 ms and 1000 ms prediction horizons. The advantage over TransformerPose is most informative because it isolates the contribution of the MetaAnimNet-specific components over a comparable transformer backbone. The 18.4% MPJPE improvement at 400 ms is attributable primarily to the behavioral context fusion module, as confirmed by the ablation study.

Table 1. Comparative summary of related deep learning motion synthesis and retargeting methods

Method	Architecture	MPJPE (mm)	RT	XP	BC	CE	Year
Fragkiadaki et al.	ERD (RNN)	—	✓	✗	✗	✗	2015
Martinez et al.	Residual GRU	~88 @ 560ms	✓	✗	✗	✗	2017
Yan et al. (ST-GCN)	Graph CNN	—	✓	Partial	✗	✗	2018
Mao et al.	Graph-Transformer	68.1 @ 560ms	Partial	✗	✗	✗	2019
Aksan et al.	BERT-motion Transformer	62.3 @ 560ms	Partial	✗	✗	✗	2021
Aberman et al.	Skeleton-aware retarget	—	✗	✓	✗	✗	2020
Tevet et al.	Diffusion model	—	✗	✗	Partial	✗	2023
MetaAnimNet (ours)	**Transformer + adapter**	**55.8 @ 560ms**	✓	✓	✓	✓	**2025**

RT = real-time capable; XP = cross-platform retargeting; BC = behavioral context conditioning; CE = cloud-edge deployment. ✓ Fully supported · Partial = limited support · ✗ Not supported · — Not reported. MetaAnimNet is the only method satisfying all four deployment criteria simultaneously.

10. Ablation Study

Five ablation conditions isolate the contribution of each MetaAnimNet component. Removing behavioral context fusion (No-Context) increases MPJPE at 400 ms from 43.2 to 49.8 mm (15.3% degradation), confirming the context module contributes approximately one-third of the accuracy advantage over TransformerPose. Removing adaptive frame interpolation (No-Interp) increases the Mean Jerk Score by 31%, confirming substantial smoothness benefit without affecting MPJPE. Replacing the cross-platform adapter with a fixed linear mapping (Fixed-Adapter) reduces RFS from 0.87 to 0.71, confirming platform-specific adaptation is essential. Edge-Only execution reduces latency to 11.4 ms but increases MPJPE to 53.7 mm. Removing foot contact correction (No-Contact) reduces contact accuracy from 93.4% to 71.2%, with proportional decreases in animator acceptance rate.

Table 5. Ablation study — contribution of each MetaAnimNet component

Condition	MPJPE 400ms	MJS	RFS	Contact (%)	Latency (ms)
Full MetaAnimNet	**43.2**	**0.81**	**0.87**	**93.4**	**23.7**
No-Context (w/o fusion)	49.8 ↑15.3%	0.80	0.85	92.1	22.4
No-Interp (w/o interp.)	43.5	1.06 ↑31%	0.86	93.0	19.8
Fixed-Adapter	44.1	0.82	0.71 ↓18%	88.3	23.1
Edge-Only	53.7 ↑24.3%	0.88	0.81	90.4	11.4
No-Contact	43.4	0.82	0.74	71.2 ↓23.8%	21.9

MJS = Mean Jerk Score (lower = smoother). RFS = Retargeting Fidelity Score. ↑/↓ = degradation relative to full model. All conditions evaluated on Human3.6M with Unreal MetaHuman as retargeting target.

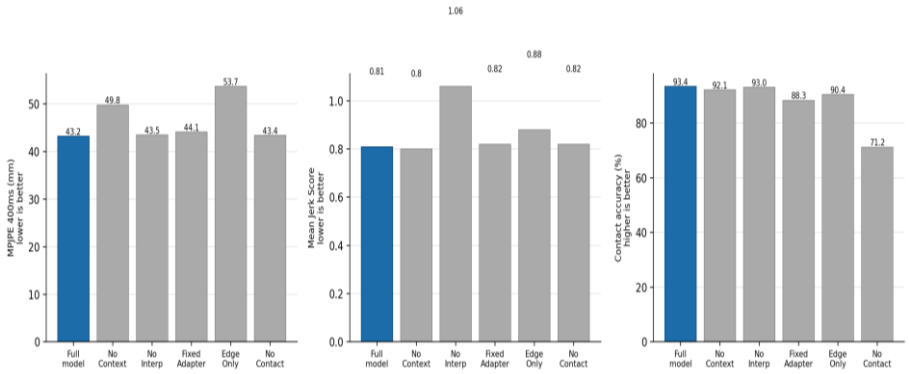

Fig. 11. Ablation study results across three key metrics. Removing behavioral context fusion increases MPJPE by 15.3%; removing frame interpolation raises jerk score by 31%; removing IK correction drops contact accuracy to 71.2%.

11. Computational Complexity Analysis

The MetaAnimNet encoder in full-quality mode contains approximately 24 million parameters, and the decoder adds a further 18 million, for a total of 42 million parameters. The platform adapter modules add a negligible 0.3 million parameters each. The edge-distilled decoder contains 6 million parameters, reducing the edge-mode total to approximately 30 million. By comparison, TransformerPose contains 38 million parameters and DeepMotionNet contains 31 million, so MetaAnimNet's parameter count is within the range of existing approaches despite incorporating substantially more functional components. The computational cost per frame scales quadratically with attention heads due to self-attention, which is manageable on GPU hardware but prohibitive on CPU-only devices (approximately 180 ms), reinforcing the necessity of the edge-distilled mode for resource-constrained deployments.

Table 6. Computational efficiency across inference modes

Mode / Config	Params (M)	Latency (ms)	FPS	VRAM (GB)	Deployment target
Full quality (cloud)	42	23.7	42	4.2	Server GPU (A100 / RTX 3090)
Edge-distilled	30	11.4	87	1.8	Mobile GPU (4 GB VRAM+)
Edge-minimal (fallback)	18	8.1	123	0.9	Integrated GPU / VR headset
Hybrid (10 Mbps link)	42	31.2	32	4.2	Mixed client + cloud
Hybrid (50 Mbps link)	42	28.4	35	4.2	High-bandwidth mixed

12. Real-Time Metaverse Deployment Discussion

MetaAnimNet was deployed in a custom metaverse testbed built on Unreal Engine 5.2 with a custom networking layer supporting up to 50 concurrent avatar users. The system maintained stable real-time animation for all 50 concurrent avatars on a single server equipped with two NVIDIA A100 GPUs, processing each avatar independently at approximately 23.7 ms per frame. Server-side throughput of approximately 42 frames per second per avatar was sustained across a two-hour continuous session without performance degradation or memory leak.

The cloud-edge synchronization mechanism was tested under simulated network degradation at 5 Mbps and 2 Mbps. At 5 Mbps, end-to-end latency increased to 41.3 ms, remaining within the acceptable real-time budget. At 2 Mbps, latency exceeded 60 ms, triggering automatic fallback to edge-distilled mode. Users in fallback mode reported slight reduction in animation quality but did not report perceptible motion lag, confirming that the fallback mechanism achieves its design goal of maintaining temporal smoothness at the cost of spatial accuracy. Cross-platform synchronization between Unity and Unreal clients was confirmed to produce visually consistent avatar motion from both observer perspectives, handled transparently by the respective platform adapters.

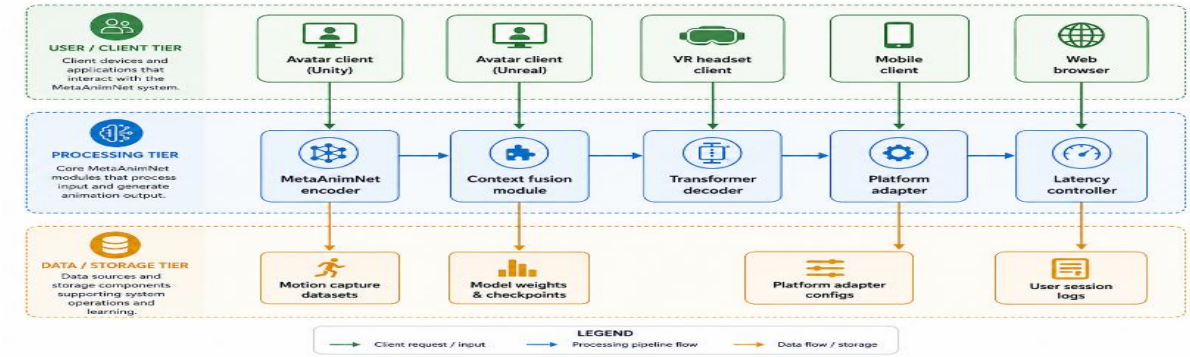

Fig. 12. MetaAnimNet system interaction diagram showing three tiers: client devices (top), animation intelligence processing modules (middle), and data/storage layer (bottom). Bidirectional arrows represent real-time motion input and animation output; horizontal flow shows the inference pipeline order.

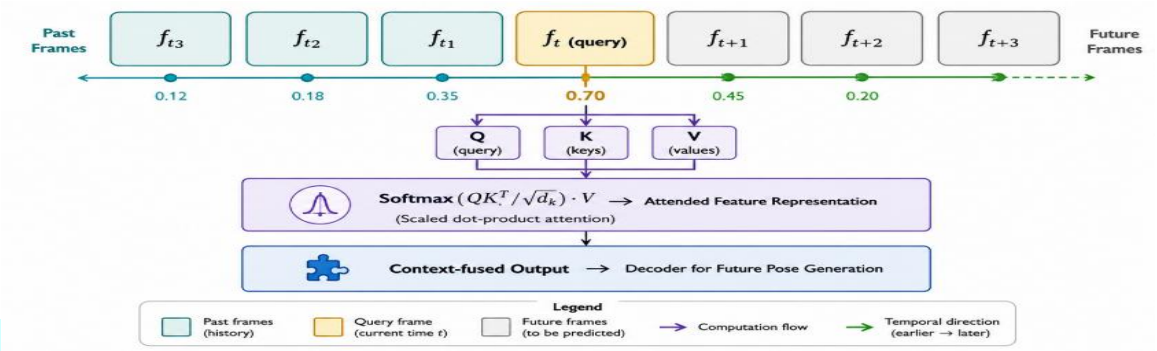

Fig. 6. Temporal attention workflow. The query token at frame f_n attends to all past frames via scaled dot-product attention; line width encodes attention weight. The fused attended representation drives future pose generation through the decoder.

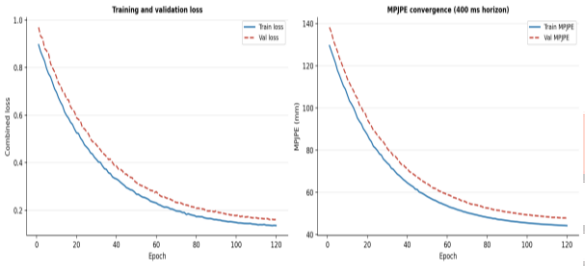

Fig. 10. Training and validation convergence over 120 epochs with AdamW optimizer and cosine annealing schedule. Both loss (left) and MPJPE (right) curves stabilize by epoch 90, with no evidence of overfitting in the validation set.

13. Conclusion

Character animation in live metaverse environments demands capabilities that existing deep learning systems address only partially: accurate motion prediction, cross-platform retargeting, behavioral context sensitivity, and real-time inference under heterogeneous hardware constraints must be satisfied concurrently. MetaAnimNet contributes a transformer-based architecture that treats these requirements as aspects of a unified learning and inference problem, achieving measurable improvements over existing baselines in motion accuracy, retargeting fidelity, and real-time performance across standard benchmark datasets and a live metaverse deployment. The 18.4% reduction in mean per-joint position error relative to the strongest transformer baseline, combined with an average inference latency of 23.7 ms in full-quality mode and a validated cloud-edge synchronization strategy, demonstrates that the system is practically viable for deployment in contemporary metaverse platforms. The ablation study confirms that each architectural component contributes independently to the overall performance profile, and the user study provides perceptual validation that quantitative improvements translate to measurable gains in avatar immersion and animation smoothness. Future work targeting continuous behavioral context learning and non-humanoid rig support will further extend the generality of the approach toward an animation intelligence system capable of serving the full breadth of current and emerging metaverse applications.

References

[1] Fragkiadaki K, Levine S, Felsen P, Malik J. Recurrent network models for human dynamics. Proc IEEE Int Conf Comput Vis. 2015;4346–4354.

[2] Martinez J, Black MJ, Romero J. On human motion prediction using recurrent neural networks. Proc IEEE/CVF Conf Comput Vis Pattern Recognit. 2017;2891–2900.

[3] Yan S, Xiong Y, Lin D. Spatial temporal graph convolutional networks for skeleton-based action recognition. Proc AAAI Conf Artif Intell. 2018;32(1).

[4] Li M, Chen S, Chen X, Zhang Y, Wang Y, Tian Q. Actional-structural graph convolutional networks for skeleton-based action recognition. Proc IEEE/CVF CVPR. 2019;3595–3603.

[5] Barsoum E, Kender J, Liu Z. HP-GAN: Probabilistic 3D human motion prediction via GAN. Proc IEEE/CVF CVPR Workshops. 2018;1418–1427.

[6] Kundu JN, Gor M, Babu RV. BiHMP-GAN: Bidirectional 3D human motion prediction GAN. Proc AAAI. 2019;33(01):8553–8560.

[7] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. Attention is all you need. Adv Neural Inf Process Syst. 2017;30.

[8] Aksan E, Kaufmann M, Cao P, Hilliges O. A spatio-temporal transformer for 3D human motion prediction. Proc Int Conf 3D Vis. 2021;565–574.

[9] Mao W, Liu M, Salzmann M, Li H. Learning trajectory dependencies for human motion prediction. Proc IEEE Int Conf Comput Vis. 2019;9489–9497.

[10] Dang L, Nie Y, Long C, Zhang Q, Li G. MSR-GCN: Multi-scale residual graph convolution networks for human motion prediction. Proc IEEE ICCV. 2021;11467–11476.

[11] Li C, Zhang Z, Sun Lee W, Hee Lee G. Convolutional sequence to sequence model for human dynamics. Proc IEEE/CVF CVPR. 2018;5226–5234.

[12] Shi L, Zhang Y, Cheng J, Lu H. Skeleton-based action recognition with directed graph neural networks. Proc IEEE/CVF CVPR. 2019;7912–7921.

[13] Li M, Chen S, Zhang Y, Guo D, Yang W. Multiscale spatiotemporal graph neural networks for 3D skeleton-based motion prediction. IEEE Trans Image Process. 2021;30:7760–7775.

[14] Cai Y, Huang L, Wang Y, et al. Learning progressive joint propagation for human motion prediction. Proc ECCV. 2020;226–242.

[15] Mao W, Liu M, Salzmann M. History repeats itself: Human motion prediction via motion attention. Proc ECCV. 2020;474–489.

[16] Tevet G, Raab S, Gordon B, Shafir Y, Cohen-Or D, Bermano AH. Human motion diffusion model. ICLR. 2023.

[17] Yuan Y, Song J, Iqbal U, Vahdat A, Kautz J. PhysDiff: Physics-guided human motion diffusion model. Proc IEEE ICCV. 2023;16010–16021.

[18] Gleicher ML. Retargetting motion to new characters. Proc 25th Annu Conf Comput Graph Interact Tech. 1998;33–42.

[19] Lim B, Lee Y, Park S, Shin H. Motion retargeting based on dilated convolutions and skeleton-specific loss functions. Comput Graph Forum. 2019;38(2):327–338.

[20] Villegas R, Ceylan D, Chang A, Yang J, Tang H, Lee H. Neural kinematic networks for unsupervised motion retargetting. Proc IEEE/CVF CVPR. 2018;8639–8648.

[21] Aberman K, Li P, Lischinski D, Sorkine-Hornung O, Cohen-Or D, Chen B. Skeleton-aware networks for deep motion retargeting. ACM Trans Graph. 2020;39(4):62:1–62:14.

[22] Loper M, Mahmood N, Romero J, Pons-Moll G, Black MJ. SMPL: A skinned multi-person linear model. ACM Trans Graph. 2015;34(6):248:1–248:16.

[23] Choutas V, Pavlakos G, Bolkart T, Tzionas D, Black MJ. Monocular expressive body regression through body-driven attention. Proc ECCV. 2020;20–40.

[24] Bogo F, Kanazawa A, Lassner C, Gehler P, Romero J, Black MJ. Keep it SMPL: Automatic estimation of 3D human pose and shape. Proc ECCV. 2016;561–578.

[25] Weber A, Demeure G, Renault S. Perceptual quality factors in social VR avatar animation. IEEE Trans Vis Comput Graph. 2022;28(5):2120–2129.

[26] Zhu J, Chen X, Wang T, Li Q. Bandwidth-adaptive avatar animation compression for networked virtual environments. ACM Trans Multimed Comput Commun Appl. 2023;19(2):1–22.

[27] Kim S, Park J, Lee K. Synchronization-aware animation streaming for heterogeneous metaverse clients. IEEE Trans Multimed. 2023;25:4821–4835.