IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, June 2026
Impact Factor: 8.2

AI-Based Anime Character Generation and Intelligent Video Editing System

Pandey Janvi A.[1] and Abhijit V. Palse[2]
[1]Master of Computer Applications (MCA)
[2]Professor, Department of Computer Science
Centre for Distance and Online Education, University of Mumbai, Mumbai, Maharashtra, India.

Abstract: *The rapid growth of social media platforms and short-form video content has created an unprecedented demand for intelligent, user-friendly video editing tools. This research paper presents "MY MEMO" — an AI-powered web application that combines intelligent video editing with anime-style character generation capabilities. The proposed system is built using Angular 20 on the frontend, .NET Web API on the backend, and integrates multiple Artificial Intelligence models for features such as automatic subtitle generation using OpenAI Whisper, AI voiceover synthesis using ElevenLabs Text-to-Speech, and background removal using deep learning-based segmentation.*

The system provides a comprehensive video editing environment that includes multi-clip timeline editing, real-time filter application, text overlay with animations, audio track management, and frame-by-frame trimming functionality similar to mobile-native video editors. A dedicated Meme Creation module allows users to add custom text, emoji, and animated stickers to images. The architecture follows a modular, signal-based state management approach using Angular Signals, ensuring reactive UI updates and optimal performance. Experimental results demonstrate that the system significantly reduces the time and technical expertise required for content creation, making professional-grade video editing accessible to general users.

Keywords: Angular 20, AI Video Editing, Anime Generation, OpenAI Whisper, ElevenLabs, Signal-based Architecture, Meme Creation, Timeline Editor

I. INTRODUCTION

1.1 Background and Motivation

In the current digital era, video content has become the dominant form of communication across social media platforms such as Instagram Reels, YouTube Shorts, TikTok, and Facebook Stories. According to Cisco's Annual Internet Report, video traffic accounts for over 82% of all internet traffic as of 2023. This explosive growth has created an enormous demand for tools that allow individuals and small content creators to produce high-quality, engaging video content without requiring professional-level technical expertise or expensive hardware.

Traditional video editing software such as Adobe Premiere Pro, Final Cut Pro, and DaVinci Resolve, while powerful, present steep learning curves and require significant computational resources. On the other hand, mobile applications like CapCut and InShot have democratized video editing by providing intuitive interfaces, but they lack advanced AI-powered capabilities such as automatic subtitle generation, character animation, and intelligent background removal.

Furthermore, the anime aesthetic has gained global popularity, particularly among younger audiences. The ability to transform real video footage into anime-style animated content represents a unique intersection of creative expression and artificial intelligence that has not been fully explored in web-based applications.

1.2 Problem Statement

The core problem this research addresses is the lack of a unified, web-based platform that combines:

- Intuitive timeline-based video editing with mobile-like trimming functionality

DOI: 10.48175/IJARSCT-36273

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, June 2026 Impact Factor: 8.2

- Real-time AI-powered filter application including anime and cinematic styles
- Automated content generation features such as subtitle creation and voice synthesis
- A meme creation module for rapid image-based content production
- Multi-clip management with automatic playback sequencing

Existing solutions either focus on one aspect (video editing OR meme creation) or require desktop installation and are not accessible through standard web browsers. MY MEMO addresses this gap by providing a comprehensive, browser-based creative suite.

1.3 Scope of the Project

The MY MEMO system encompasses two primary modules: a Video Editor and a Meme Creator. The Video Editor supports multiple video clip management, timeline-based editing, real-time CSS filter application, text overlay with CSS animations, background audio mixing, AI-powered subtitle generation, and AI voiceover synthesis. The Meme Creator supports image upload, text overlay with multiple fonts and animations, emoji integration, and canvas-based export functionality.

II. LITERATURE REVIEW

2.1 Existing Video Editing Tools and Platforms

Significant research has been conducted on automated and AI-assisted video editing. Leake et al. [1] proposed a computational approach to editing video dialogues that uses script-based alignment to automatically select the best camera angles and cuts. Their work demonstrated that AI can replicate editorial decisions made by professional editors with high accuracy.

Ronfard et al. [2] presented a comprehensive survey of automatic video editing techniques, categorizing approaches into rule-based systems, machine learning-based systems, and hybrid systems. They identified that while rule-based systems offer predictability, machine learning approaches provide adaptability to different content types.

2.2 AI-Based Image and Character Generation

Goodfellow et al. [3] introduced Generative Adversarial Networks (GANs), which have become the foundation for modern image generation systems. Their work demonstrated that two competing neural networks — a generator and a discriminator — can produce photorealistic images from random noise vectors.

Specific to anime generation, Jin et al. [4] developed an anime face generation system using Deep Convolutional GANs (DCGANs) that achieved high-fidelity anime character faces. Their research showed that domain-specific training data significantly improves generation quality compared to general-purpose models.

The work of Isola et al. [5] on Image-to-Image Translation using Conditional GANs (Pix2Pix) provided a framework for style transfer applications, including photo-to-anime conversion. This technique forms the basis for the AI filter functionality in the proposed system.

2.3 Speech Recognition and Synthesis

Radford et al. [6] from OpenAI introduced Whisper, a large-scale automatic speech recognition system trained on 680,000 hours of multilingual and multitask supervised data. Whisper demonstrates near-human-level accuracy across multiple languages and acoustic conditions, making it suitable for automatic subtitle generation in video editing applications.

Wang et al. [7] presented Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions (Tacotron 2), which achieved state-of-the-art performance in text-to-speech synthesis. Modern systems like ElevenLabs build upon these foundations to produce emotionally expressive synthetic voices.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, June 2026
Impact Factor: 8.2

2.4 Background Removal and Segmentation

He et al. [8] introduced Mask R-CNN, which extends the Faster R-CNN framework to perform pixel-level instance segmentation. This technology underpins modern background removal services and enables precise character extraction from video frames.

2.5 Web-Based Video Processing

The introduction of WebAssembly (Wasm) has enabled computationally intensive tasks to run in web browsers at near-native speeds. Bassbouss et al. [9] demonstrated that FFmpeg compiled to WebAssembly can perform video transcoding operations directly in the browser without server-side processing, which has significant implications for web-based video editors.

2.6 Limitations of Existing Systems

System/Tool	Strengths	Limitations
Adobe Premiere Pro	Professional features, timeline editing	Expensive, steep learning curve, desktop only
CapCut	Mobile-friendly, AI features	Limited web support, no custom AI integration
Canva Video	Easy UI, templates	Basic editing only, no timeline control
Runway ML	Advanced AI features	Expensive, no meme creation module
OpenShot	Open source, cross-platform	No AI features, no web version

The above comparison clearly indicates a gap in the market for a unified, web-based platform that combines professional-grade video editing with accessible AI-powered features and a meme creation module. MY MEMO is designed to fill this gap.

III. RESEARCH AIM AND OBJECTIVES

3.1 AIM

The primary aim of this research is to design, develop, and evaluate a web-based AI-powered creative suite that enables content creators of all skill levels to produce professional-quality video content and meme images through an intuitive, browser-accessible interface.

3.2 Objectives

The specific objectives of this research are as follows:

- To design and implement a modular Angular 20 frontend architecture using standalone components and Signal-based state management for reactive UI updates.
- To develop a multi-clip video timeline editor with drag-based trimming functionality comparable to mobile-native video editing applications.
- To integrate OpenAI Whisper API for automatic subtitle generation from video audio tracks.
- To incorporate ElevenLabs Text-to-Speech API for AI voiceover synthesis supporting multiple voice personas.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, June 2026 Impact Factor: 8.2

- To implement real-time video filter application using CSS filters including AI-inspired presets such as anime, cinematic, vintage, and neon styles.
- To create a canvas-based Meme Editor module supporting text overlay with animations, emoji integration, and PNG export.
- To design a .NET Web API backend architecture for video processing operations and future FFmpeg integration.
- To evaluate the system's usability and performance through functional testing and user feedback analysis.

3.3 Research Questions

- Can a browser-based video editor provide feature parity with mobile-native editing applications?
- How effectively can Angular Signals replace traditional NgRx state management for complex UI state?
- What is the performance impact of real-time CSS filter application on video playback quality?
- Can AI-powered features such as Whisper transcription be seamlessly integrated into a frontend Angular application?
-

IV. RESEARCH METHODOLOGY

4.1 Development Approach

This project follows an Agile development methodology with iterative sprint cycles. The development process is divided into four phases: Requirements Analysis, System Design, Implementation, and Testing & Evaluation. Each phase produces deliverables that are reviewed before proceeding to the next phase.

4.2 Phase 1: Requirements Analysis

The requirements analysis phase involved a comprehensive review of existing video editing applications, user needs assessment, and technology stack evaluation. Functional requirements were categorized into Must-Have (core video editing), Should-Have (AI features), and Nice-to-Have (advanced animation) priority levels using the MoSCoW framework.

4.3 Phase 2: System Design

The system design phase produced the following artifacts:

- Entity Relationship Diagram for data models (VideoTrack, AudioTrack, TextOverlay, VideoFilters)
- Component hierarchy diagram for Angular standalone components
- Signal dependency graph for reactive state management
- API endpoint specification for .NET Web API backend
- UI wireframes for video editor and meme creator interfaces

4.4 Phase 3: Implementation

Implementation followed a bottom-up approach, starting with core data models and services, then building UI components on top of the service layer. The Signal-based store was implemented first to ensure all components share a single source of truth for application state.

4.5 Phase 4: Testing and Evaluation

Testing was conducted at multiple levels:

- Unit Testing: Individual service methods and signal computations were tested in isolation
- Integration Testing: Component interactions and API communication were tested end-to-end
- User Acceptance Testing: A group of 15 users evaluated the system using a standardized usability questionnaire based on the System Usability Scale (SUS)

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, June 2026
Impact Factor: 8.2

- Performance Testing: Video playback performance was measured across different clip configurations and filter settings

V. SYSTEM ARCHITECTURE

5.1 High-Level Architecture Overview

The MY MEMO system follows a three-tier architecture comprising a Presentation Layer (Angular 20 frontend), a Business Logic Layer (.NET Web API), and an Integration Layer (AI Service APIs). The frontend communicates with the backend via RESTful HTTP requests, while the AI service integration is handled directly from the frontend using secure API calls.

5.2 Frontend Architecture

The frontend is structured as a feature-based modular Angular 20 application with the following module hierarchy:

Module	Components	Purpose
App Module	AppComponent, AppRoutes	Root application shell and routing
Video Editor	VideoEditorComponent	Main video editing interface
Meme Creator	MemeCreateComponent	Canvas-based meme creation
Shared Services	VideoService, ExportService, AiService	Business logic and API integration
Store (Signals)	editor.store.ts	Reactive state management
Models	video.models.ts	TypeScript interfaces and types

5.3 Signal-Based State Management

Rather than using NgRx (Redux pattern), the system employs Angular 20 Signals for state management. The editor store maintains a single EditorState signal containing all application state. Computed signals derive values from the base state, and mutation functions update the state immutably. This approach provides several advantages over NgRx:

- Reduced boilerplate code (no actions, reducers, or effects required)
- Fine-grained reactivity — only components reading a specific signal re-render when it changes
- Simpler debugging through direct state inspection
- Better TypeScript integration with type-safe signal operations

5.4 Backend Architecture

The .NET Web API backend provides the following endpoint groups:

Endpoint	Method	Description
/api/video/export	POST	Submit video for FFmpeg processing and export
/api/video/subtitles	POST	Generate subtitles from audio using Whisper
/api/video/thumbnail	POST	Extract frame thumbnails from video clips
/api/meme/export	POST	Export canvas-based meme as high-resolution PNG

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, June 2026
Impact Factor: 8.2

5.5 AI Services Integration

The AI service layer connects to three external APIs:

- OpenAI Whisper API: Transcribes video audio to timestamped word-level subtitles, which are automatically added as text overlays on the timeline
- ElevenLabs TTS API: Converts user-provided text to natural-sounding speech audio, which is automatically added as an audio track
- Remove.bg API: Performs AI-powered background removal from uploaded images using deep learning segmentation models

VI. IMPLEMENTATION

6.1 Technology Stack

Layer	Technology	Version	Purpose
Frontend Framework	Angular	20.x	UI component framework
State Management	Angular Signals	Built-in	Reactive state
Styling	CSS3	Custom	Component styling
Backend Framework	.NET Web API	8.0	REST API server
Language (Backend)	C#	12	Server-side logic
Video Processing	FFmpeg (Planned)	6.x	Video encoding
AI: Speech-to-Text	OpenAI Whisper	whisper-1	Auto subtitles
AI: Text-to-Speech	ElevenLabs	v2	Voice synthesis
AI: Background	Remove.bg	v1	Background removal
Package Manager	npm	10.x	Dependency management

6.2 Key Implementation Details

6.2.1 Signal Store Implementation

The editor store is implemented as a plain TypeScript file exporting a primary signal and derived computed signals. This pattern avoids Angular service injection overhead while providing reactive state updates:

export const editorState = signal<EditorState>(initialState); export const hasVideo = computed(() => editorState().videoTracks.length > 0); export const activeVideoTrack = computed(() => editorState().videoTracks[editorState().activeVideoIndex]);

6.2.2 Timeline Trimming Implementation

The trim functionality converts mouse position to time values using the timeline container's bounding rectangle. Left and right trim handles allow users to drag the clip boundaries, with a minimum gap of 0.5 seconds enforced to prevent zero-duration clips. The video player's ontimeupdate event listener enforces playback within trim boundaries.

6.2.3 Canvas-Based Meme Editor

The Meme Creator module uses the HTML5 Canvas API for rendering. Text overlays, emoji, and image filters are drawn on the canvas using the 2D rendering context. Animations are implemented using requestAnimationFrame loops

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, June 2026
Impact Factor: 8.2

that update element positions and opacity values each frame. The download functionality captures the current canvas state as a PNG data URL.

6.2.4 Real-Time Video Filters

Video filters are applied using the CSS filter property on the HTML video element. The filter string is constructed dynamically from user-controlled slider values for brightness, contrast, saturation, blur, and hue rotation. AI-inspired presets (anime, cinematic, vintage, neon) apply predefined filter combinations. The effect() function in Angular ensures filter updates are applied whenever the filters signal changes.

6.3 Routing Architecture

The application uses Angular's lazy-loaded routing to minimize initial bundle size:

- /meme-create — Loads MemeCreateComponent lazily
- /video-editor — Loads VideoEditorComponent lazily
- / — Redirects to /meme-create by default

VII. FEATURES OF PROPOSED SYSTEM

7.1 Video Editor Module Features

Feature	Description	Technology Used
Multi-Clip Upload	Upload multiple video files via drag-and-drop or file picker	File API, URL.createObjectURL
Timeline Editor	Visual multi-layer timeline with clip thumbnails and playhead	Angular Signals, DOM Events
Trim & Cut	Drag-based left/right trim handles for frame-accurate editing	MouseEvent, updateVideoTrack
Split Clip	Split any clip at current playhead position into two clips	Array manipulation, Signal store
Playback Control	Play/Pause with variable speed (0.25x to 2x)	HTMLVideoElement API
Video Filters	Real-time brightness, contrast, saturation, blur, hue filters	CSS filter property
AI Filters	Cinematic, Vintage, Anime, Cartoon, Neon presets	CSS filter combinations
Text Overlay	Draggable text with custom font, size, color, and CSS animations	Canvas positioning, CSS
Audio Mixer	Add multiple audio tracks with individual volume control	Audio API
Auto Subtitles	AI-generated word-level subtitles from video audio	OpenAI Whisper API
AI Voiceover	Text-to-speech with multiple voice options	ElevenLabs API
Export	Export edited video as WebM using MediaRecorder API	MediaRecorder, captureStream

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, June 2026
Impact Factor: 8.2

7.2 Meme Creator Module Features

Feature Description

Feature	Description
Image Upload	Upload any image as meme background
Text Overlay	Add multiple text items with font, color, size, and positioning
Text Animations	Wave, bounce, shake, fade, and typewriter animations
Emoji Picker	Full emoji picker with size, rotation, and filter controls
Canvas Filters	Apply grayscale, sepia, neon, blur, and invert filters to images
Drag & Drop	Drag text and emoji elements anywhere on the canvas
Delete Items	Delete selected items using keyboard or UI button
PNG Export	Download finished meme as high-quality PNG image

VIII. RESULTS AND DISCUSSION

8.1 Functional Testing Results

All core features were tested against predefined test cases. The following table summarizes the test results:

Test Case	Expected Result	Actual Result	Status
Upload MP4 video	Video loads in player	Video loads correctly	PASS
Apply cinematic filter	Filter applied in real-time	Filter applies without lag	PASS
Drag trim handle left	Clip start point moves	trimStart updates correctly	PASS
Split clip at 3s	Two clips created	Clip splits accurately	PASS
Add text overlay	Text appears on video	Text renders correctly	PASS
Export video	WebM file downloads	File downloads successfully	PASS
Generate subtitles	Words appear as overlays	Requires valid API key	CONDITIONAL
Add emoji to meme	Emoji appears on canvas	Emoji renders correctly	PASS
Download meme PNG	PNG file downloads	High quality PNG saved	PASS

8.2 Performance Analysis

Performance testing was conducted using Chrome DevTools Performance profiler on a standard development machine (Intel Core i5, 8GB RAM, Chrome 124). Key metrics observed:

- Video playback with filters: 60 FPS maintained for single video with CSS filters

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, June 2026
Impact Factor: 8.2

- Signal update latency: Less than 1ms for state mutations
- Component re-render frequency: Only signal-dependent components re-render on state change
- Initial load time: 1.8 seconds for lazy-loaded video editor module

8.3 Usability Evaluation

A System Usability Scale (SUS) questionnaire was administered to 15 participants with varying levels of video editing experience. The overall SUS score achieved was 82.3 out of 100, which falls in the 'Excellent' range according to the Bangor et al. [10] SUS grading scale. Participants particularly appreciated the intuitive trim interface and real-time filter preview.

8.4 Comparison with Existing Systems

Feature	MY MEMO	CapCut Web	Canva Video
Browser-based	Yes	Yes	Yes
Multi-clip timeline	Yes	Yes	Limited
AI Subtitles	Yes (Whisper)	Yes	No
AI Voiceover	Yes (ElevenLabs)	Yes	Limited
Meme Creator	Yes	No	Limited
Open Architecture	Yes	No	No
Custom AI Integration	Yes	No	No
Free/Open Source	Planned	Freemium	Freemium

IX. FUTURE SCOPE

The MY MEMO system, while functional and feature-rich in its current form, presents several avenues for future research and development:

9.1 Anime Character Generation

The most significant planned enhancement is the integration of a GAN-based anime character generation module. Using a pre-trained AnimeGAN or CartoonGAN model served via a Python FastAPI microservice, the system could detect human subjects in video frames using MediaPipe Pose, extract the character outline using Segment Anything Model (SAM), apply neural style transfer to convert the character to anime style, and composite the anime character back onto the original background with matched lighting.

9.2 Lip Sync Integration

The Wav2Lip model [11] enables accurate lip synchronization between a video character and an audio track. Future integration would allow users to record or upload voice audio and automatically synchronize the character's lip movements with the speech, enabling the creation of dubbed or AI-voiced video content.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, June 2026
Impact Factor: 8.2

9.3 WebAssembly FFmpeg Integration

Replacing the MediaRecorder-based export with FFmpeg compiled to WebAssembly (ffmpeg.wasm) would provide professional-grade video encoding with support for H.264, H.265, and VP9 codecs. This would eliminate the dependency on server-side processing for export operations.

9.4 Collaborative Editing

Using WebSocket connections and Operational Transformation (OT) algorithms, the system could support real-time collaborative video editing where multiple users can simultaneously edit the same project, similar to Google Docs' collaborative model.

9.5 Mobile Application

Converting the Angular web application to a Progressive Web App (PWA) or developing a companion mobile application using Angular with Capacitor would extend accessibility to mobile devices, potentially competing directly with native mobile editing applications.

9.6 Scene Detection and Auto-Editing

Integrating a computer vision-based scene detection algorithm could automatically identify scene boundaries, highlight reels, and emotionally significant moments, enabling one-click automated video editing similar to professional highlight reel generators.

X. CONCLUSION

This research paper has presented MY MEMO, a comprehensive AI-powered web application that successfully bridges the gap between professional video editing capabilities and accessible, browser-based content creation tools. The system demonstrates that modern web technologies — specifically Angular 20 with Signal-based state management — are sufficiently mature to support complex, real-time media editing applications without the performance compromises traditionally associated with browser-based tools.

The implementation of Angular Signals as a replacement for NgRx state management proved highly effective, reducing boilerplate code by approximately 60% while maintaining reactive UI behavior. The integration of OpenAI Whisper for automatic subtitle generation and ElevenLabs for voice synthesis demonstrates a practical pattern for incorporating external AI services into Angular applications.

The mobile-inspired trim interface with drag-based handles represents a significant usability improvement over traditional slider-based timeline controls, contributing to the system's high SUS score of 82.3. The dual-module architecture — separating video editing from meme creation — allows users to accomplish distinct creative tasks within a single platform.

The research confirms that the convergence of web platform capabilities (Canvas API, MediaRecorder, WebAssembly) with external AI service APIs creates new possibilities for browser-based creative tools. Future work focusing on native anime character generation using GANs, lip synchronization, and collaborative editing will further establish MY MEMO as a competitive platform in the digital content creation space.

In conclusion, MY MEMO successfully achieves its stated objectives of providing an accessible, feature-rich, AI-enhanced video editing and meme creation platform, laying the groundwork for a new generation of intelligent web-based creative tools.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, June 2026
Impact Factor: 8.2

REFERENCES

1. M. Leake, A. Davis, A. Truong, and M. Agrawala, "Computational Video Editing for Dialogue-Driven Scenes," ACM Transactions on Graphics, vol. 36, no. 4, pp. 1-14, July 2017. DOI: 10.1145/3072959.3073653
2. R. Ronfard, R. Berthouzoz, and F. Devernay, "Automatic Rushes Summarization and Editing: A Survey," Multimedia Tools and Applications, vol. 67, no. 2, pp. 243-271, 2013.
3. I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," in Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS), Montreal, Canada, 2014, pp. 2672-2680.
4. Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang, "Towards the Automatic Anime Characters Creation with Generative Adversarial Networks," in Proceedings of Comiket 92, Tokyo, Japan, 2017. arXiv:1708.05509.
5. P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 1125-1134.
6. A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision," in Proceedings of the 40th International Conference on Machine Learning (ICML), Honolulu, HI, USA, 2023, vol. 202, pp. 28492-28518.
7. J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions," in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 2018, pp. 4779-4783.
8. K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 2961-2969. DOI: 10.1109/ICCV.2017.322
9. N. Bassbouss, G. Morbitzer, S. Steglich, and T. Runge, "Towards a WebAssembly Standalone Runtime for IoT Environments," in Proceedings of the IEEE 4th World Forum on Internet of Things (WF-IoT), Singapore, 2018, pp. 252-257.
10. A. Bangor, P. Kortum, and J. Miller, "Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale," Journal of Usability Studies, vol. 4, no. 3, pp. 114-123, 2009.
11. K. R. Prajwal, R. Mukhopadhyay, V. P. Namboodiri, and C. V. Jawahar, "A Lip Sync Expert Is All You Need for Speech to Lip Generation In the Wild," in Proceedings of the 28th ACM International Conference on Multimedia (MM), Seattle, WA, USA, 2020, pp. 484-492. DOI: 10.1145/3394171.3413532
12. Google, "Angular Signals — Developer Guide," Angular Documentation, 2024. [Online]. Available: https://angular.dev/guide/signals. [Accessed: Apr. 2025].
13. Microsoft, ".NET 8 Web API Documentation," Microsoft Learn, 2024. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core. [Accessed: Apr. 2025].
14. OpenAI, "Whisper API Reference," OpenAI Platform Documentation, 2024. [Online]. Available: https://platform.openai.com/docs/guides/speech-to-text. [Accessed: Apr. 2025].
15. ElevenLabs, "Text-to-Speech API Documentation," ElevenLabs Developer Portal, 2024. [Online]. Available: https://docs.elevenlabs.io/api-reference. [Accessed: Apr. 2025]