TANUKI LEAF: REAL-TIME DIFFUSION AVATAR GENERATION

Tan Eng Hui, Wang Tao, Hu Lei, Ho Zi Hao Timothy

NUS-ISS, National University of Singapore, Singapore 119615

ABSTRACT

The project aims to create an intelligent avatar generation system using AI-powered tools such as computer vision and diffusion models. Motivated by the inefficiencies in current avatar creation processes, our approach offers customisable, real-time avatars driven by natural language prompts and hand gestures. The proposed system integrates gesture recognition, image generation, and real-time rendering. We demonstrate improved accessibility and performance for digital creators, gamers, and livestreamers.

Keywords: avatar generation, computer vision, realtime diffusion, image generation

1. INTRODUCTION

With the increasing popularity of virtual livestreaming and digital content creation, there is growing demand for expressive, customisable digital avatars. However, existing tools for avatar creation are often complex, fragmented, and resource-intensive. Users typically need to navigate multiple platforms—ranging from 3D model design to rigging and configuration—making the process time-consuming and inaccessible for casual users or newcomers.

This project, *Tanuki Leaf*, aims to address these challenges by introducing a real-time AI-driven system for avatar generation and control. The system takes a video feed and a natural language prompt as input. It then applies image-to-image diffusion models—guided by the prompt—to generate a visually consistent, stylised avatar. Hand gestures captured via webcam are recognised using Mediapipe and lightweight ML models, and are used to drive avatar behaviours in real time.

The highlights of our solution are summarised as follows:

- **Real-time image-to-image avatar generation**: Avatar visuals are generated from webcam frames using diffusion models fine-tuned via LoRA and conditioned on prompts.
- **Natural language personalisation**: Prompts enable fine-grained control over avatar appearance and style with simple textual input.
- **Gesture-based control**: Hand gestures captured through webcam are recognised and mapped to avatar actions or behaviours.
- **Seamless integration**: The system combines image processing, model inference, and gesture recognition in a unified, low-latency pipeline.

2. LITERATURE REVIEW

Our project builds upon recent advancements in computer vision, diffusion-based generative models, and real-time streaming frameworks. The relevant literature and tools can be grouped into three main categories: (1) gesture recognition, (2) image generation using diffusion models, and (3) system-level integration for real-time applications.

2.1. Hand Landmark Detection

We evaluated two main approaches for hand landmark detection: **Mediapipe** and **YOLOv8**. Mediapipe offers a high-fidelity, dedicated hand tracking module that outputs 21 precise landmark points per hand in real time. YOLOv8, while more general-purpose, was adapted for keypoint detection by training it on annotated hand images.

Mediapipe was ultimately preferred for its ease of integration, stability, and landmark precision. YOLOv8, although flexible and customisable, required more training data and did not consistently detect fine hand movements as accurately as Mediapipe.

Strengths: Mediapipe provides reliable landmark extraction with minimal setup. YOLOv8 is adaptable for custom datasets and multi-purpose detection.

Weaknesses: Mediapipe has limited support for training or extending gesture vocabularies. YOLOv8 underperforms in detail-sensitive tasks like finger positioning.

2.2. Gesture Recognition

Once the hand landmarks were extracted, gesture recognition was performed using traditional machine learning techniques. We experimented with several approaches and settled on using **cosine similarity** between input landmark vectors

and predefined gesture templates. This allowed for fast, interpretable classification with minimal model overhead.

The main challenge was ensuring that gestures were both intuitively designed and sufficiently distinct in vector space to minimise misclassification. Gesture performance varied based on camera angle, hand orientation, and lighting conditions.

Strengths: Lightweight implementation, fast inference, and easy customisation.

Weaknesses: Sensitive to input noise and minor inconsistencies in hand pose or angle.

2.3. Diffusion-Based Image Generation

Diffusion models have become a cornerstone in image synthesis tasks due to their state-of-the-art performance in generating high-quality, diverse, and realistic images[1]. Their ability to iteratively refine noisy data into structured outputs makes them particularly suitable for tasks requiring stylised and consistent visual generation. Consequently, we chose Stable Diffusion models, enhanced via LoRA (Low-Rank Adaptation) and DreamBooth fine-tuning, for generating avatars. These models enable high-quality, stylised image generation with minimal input data. Specifically, we employed an image-to-image pipeline where each video frame is transformed based on conditioning prompts to produce consistent avatars.

Strengths: LoRA and DreamBooth allow personalised fine-tuning with relatively few training images, enabling concept or character embedding without full retraining.

Weaknesses: Output consistency across frames is challenging, and generation can be computationally intensive without optimisation.

2.4. Real-Time Integration and Streaming

The core of our system's runtime performance is driven by the open-source **StreamDiffusion** framework, which supports real-time diffusion-based rendering pipelines. For downstream streaming and rendering, we integrated tools like **OpenCV** for image processing and **FFmpeg** for stream output.

Strengths: StreamDiffusion supports asynchronous image generation with minimal latency. The modularity of the framework allows for easy integration with diffusion models.

Weaknesses: Real-time performance still depends heavily on hardware (e.g., GPU VRAM) and requires fine-tuning to avoid latency spikes.

2.5. Prompt Adaptation and Fine Control

Recent advances in personalized image and video generation have introduced various approaches to enabling base models to learn new target concepts efficiently. Among them, **IP-Adapter**[2], **DreamBooth**[3], and **InstantID**[4] represent three distinct strategies with different trade-offs in terms of fidelity, flexibility, and deployment cost.

IP-Adapter leverages CLIP-based image embeddings to guide generation without requiring model fine-tuning. It is lightweight and highly flexible, suitable for multi-reference or style-driven tasks. However, its ability to maintain identity consistency is relatively limited. InstantID addresses this issue by focusing on preserving facial identity through an ID encoder, enabling fast and reliable facial image generation. Nevertheless, its applicability is primarily confined to human faces, making it less suitable for broader concept learning. In contrast, DreamBooth offers high-fidelity, subject-specific generation through fine-tuning on a small number of images. While it requires more computational resources and training time, it provides the most reliable performance when precise identity preservation and concept control are required—particularly in scenarios involving dynamic user input and custom character generation.

3. PROPOSED APPROACH

The proposed system consists of an end-to-end pipeline for real-time avatar generation using multimodal inputs—namely camera feed and gesture controls. The overall architecture is illustrated in Fig. 1.

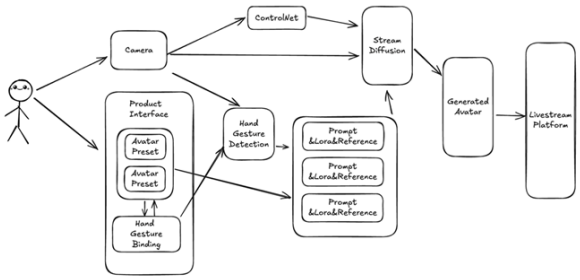

Fig. 1. System Architecture of Tanuki Leaf Avatar Generation Pipeline

3.1. Camera and Control Input

The system begins with a webcam input, capturing live video frames of the user. This video feed serves as the source image for avatar generation and as input for gesture recognition. An optional **ControlNet** module refines the image-to-image diffusion process by introducing additional control signals (e.g., edge maps or pose guidance).

3.2. Product Interface and Preset Management

The **Product Interface** layer allows users to select from preconfigured avatar presets and bind specific gestures to actions

or prompt changes. This user-friendly module supports visual preview and simple mapping of gestures to system behaviours.

3.3. Hand Gesture Detection

A dedicated gesture recognition module processes the video feed using **Mediapipe**'s hand landmark detection. Keypoints are analysed using lightweight machine learning models (e.g., cosine similarity classifiers) to detect gesture classes. Recognised gestures are then passed to trigger bound actions or preset switches.

3.4. StreamDiffusion-Based Generation

To achieve high-quality, identity-consistent generation while addressing the training efficiency bottleneck of DreamBooth, we adopt a lightweight fine-tuning strategy based on *LoRA*[5] (Low-Rank Adaptation). By injecting low-rank trainable adapters into the model's attention layers, LoRA significantly reduces the number of trainable parameters, enabling faster adaptation to new targets with minimal resource consumption

The **StreamDiffusion** engine is used for generating avatars from the video frames using the prompts and fine-tuned LoRA weights. This engine supports real-time inference with frame-wise continuity optimisations. The output is a stylised, animated avatar image stream.

3.5. Prompt Engineering

Each avatar generation instance is guided by a structured prompt, LoRA weights and an optional reference image. These prompts contribute to the semantic and stylistic framework for the output. The system supports modular combinations of **Prompt and LoRA and Reference**, providing for real-time changes in visual themes or effects contexts without model retraining. Changes to the generated avatar visuals are activated via gestures and system settings.

Prompt engineering is key to guiding the diffusion model toward high-fidelity, anime-style outputs. In place of using natural language, the prompts are structured using weighted tokens [6], which assign the emphasis of visual traits (e.g. `(masterpiece:1.3), short spiky white hair, (dynamic pose:1.1))` [7]. These tokens shape the model's attention during denoising, allowing precise control over lighting, composition, and anatomical dimensions.

To ensure visual precision and character fidelity, the prompts use categorical descriptors (e.g. `'1 guy', 'blue eyes'`) and domain-specific labels (e.g. `(official art:1.1), (ultra-detailed)`) [7]. This layered structure ensures higher consistency in outputs in aspects such as hair colour, clothing, and accessories. For example, emphasis on `(sorcerer outfit:1.3)` reduces uncertainty in rendering of costume details.

We envision empirical tuning to be vital in improving consistency. It is expected that, in our Gojo Satoru case study, unstructured prompts lead to inconsistencies in hair colour, blindfold appearance, and energy effects. Transitioning to weighted structured prompts should improve visual consistency, achieving increased sample accuracy in key traits. Final prompts used precise tags (e.g., `(official art:1.1),blue energy with swirling cursed motifs`) [8] to further reduce background variations, and is validated with internal samples and qualitative scoring.

LoRA (Low-Rank Adaptation) modules are integrated using syntax like `lora:add_detail:0.4` [8], enabling fine-grained control without retraining. These modifiers enhance detail or style intensity—e.g., combining a glow-enhancing LoRA with "blue energy" to enable a cinematic presentation.

Maintaining visual consistency, as observed in real-time use, requires calibrated utilisation of key prompt tokens and suppression of undesired features through negative prompts. Tokens like `ultra-detailed`, `best quality`, and `fantasy:1.2` [8] maintain image tone and clarity, while negative prompts prevent issues like malformed limbs or low resolution.

In summary, prompt engineering acts as a control interface that combines structured language, model alignment, and empirical tuning. It facilitates expressive, high-quality anime avatars and supports dynamic generation pipelines. Such modularity provides for flexibility in future modifications, allowing potential addition of automated prompt generation systems that adapt based on user feedback to increase levels of user personalization.

3.6. Automatic Speech Recognition (ASR)

To enhance multimode interaction, the system features an Automatic Speech Recognition (ASR) module that enables voice-activated control of anime model generation. Users can turn on ASR functionality through the interface, which provides continued microphone listening when enabled. The ASR pipeline listens for a predefined activation phrase (e.g."activate"), enabling it to switch into command recognition mode.

Voice commands are processed using **Google's Speech Recognition API**, with the captured audio converted into text and passed through a correction layer [9]. This layer references a configurable dictionary of domain-specific substitutions (e.g., mapping "ram" to "rem") to improve recognisability for avatar-specific names and gestures. Once corrected, the text is matched against a library of known gestures using both direct and fuzzy string matching techniques to increase range of recognition.

The correction layer emulates the algorithm proposed by Campos-Sobrino et al., which utilizes Levenshtein distance on phonemes to reduce errors in speech recognition systems, especially for domain-specific applications. This approach

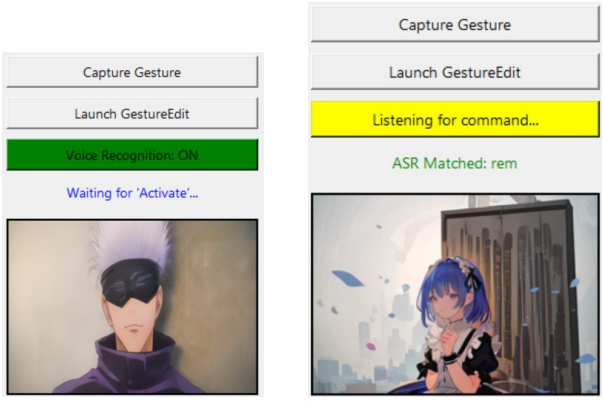

Fig. 2. Snapshot of ASR activation of diffusion model

notably enhances the accuracy of recognising unique terms by leveraging phonetic similarity metrics. [10]

If a match is found, the associated gesture action is launched in real-time, resembling those models initiated via hand gestures. This allows users to switch avatar presets or trigger animations hands-free. Visual feedback is provided in the UI to indicate audio recognition success or failure, creating a responsive voice control experience. The ASR component thus expands available functions for user to alternate between avatar generations in the pipeline.

Fig. 3. Dictionary and Fuzzy Matching of audio

3.7. System Integration

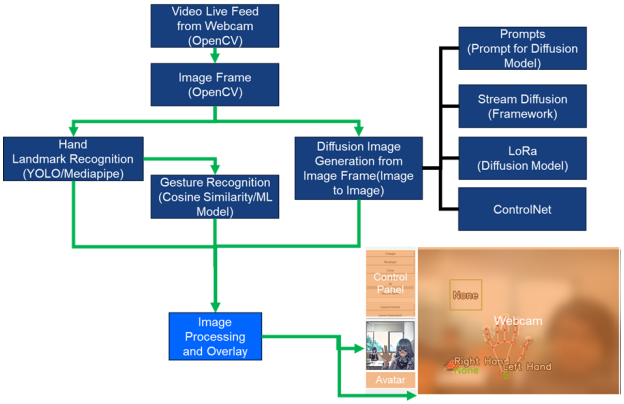

Fig. 4. Overall system architecture and integration flow.

The system captures a live video feed from a webcam using OpenCV, continuously extracting image frames for further processing. Each image frame is simultaneously passed into two main processing streams:

- **Hand Landmark Recognition and Gesture Recognition**: Hand landmarks are detected using YOLO or MediaPipe models. These landmarks are subsequently fed into a gesture recognition module, which uses cosine similarity or a machine learning model to classify the hand gestures.
- **Diffusion Image Generation**: In parallel, the image frame is processed through a diffusion model pipeline comprising Stream Diffusion, LoRA fine-tuning, and optionally ControlNet. Prompts are applied to condition the image generation based on the desired outcome.

The outputs from the gesture recognition and diffusion model are integrated within an Image Processing and Overlay module. This module overlays gesture labels, control panels, and avatar augmentations onto the live feed, resulting in the final visual output displayed to the user in real-time.

3.8. Streaming Integration and Broadcasting

In the final stage of the pipeline, the generated Tanuki Avatar output and the live gameplay footage are combined using Open Broadcaster Software (OBS).

As shown in Figure 6, OBS is configured to manage multiple media sources, overlaying the avatar onto the gameplay screen.

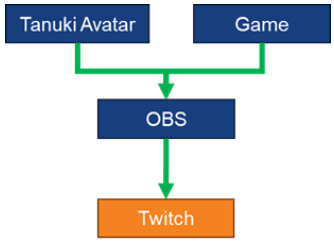

Fig. 5. Overall streaming workflow integration with OBS and Twitch.

The overall streaming workflow is illustrated in Figure 5. Both the avatar feed and the gameplay video are merged inside OBS, which acts as the main composition tool.

The final output is then streamed live to platforms such as Twitch, providing an interactive and engaging viewing experience for audiences.

This integration ensures a seamless broadcast where both real-time game actions and avatar reactions are synchronised and delivered with minimal latency.

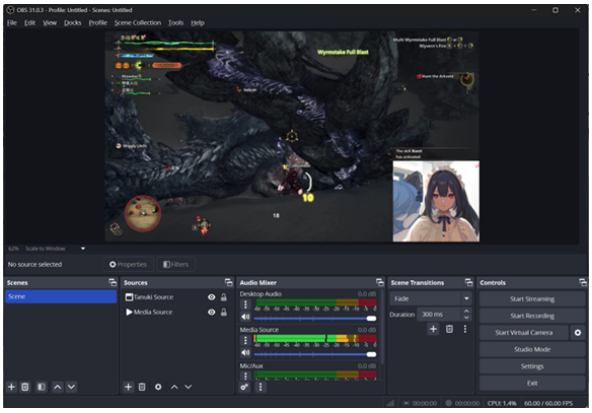

Fig. 6. OBS setup combining the gameplay and avatar feed.

4. COMPONENT IMPLEMENTATION AND EXPERIMENTAL ANALYSIS

This section presents the core technological modules developed and integrated into the project. Each module addresses a critical function necessary for the overall system pipeline:

- **Hand Landmarks Detection**: The detection of key points on human hands, forming the basis for gesture interpretation and further image processing.
- **Hand Gesture Recognition**: The classification of hand poses and movements into meaningful gestures using extracted landmark features.
- **Diffusion and LoRA-based Avatar Generation**: The generation of avatar imagery using diffusion models fine-tuned with LoRA techniques to balance quality and efficiency.
- **Prompt Engineering**: The design and optimisation of text prompts to control and guide the behaviour of the diffusion model during avatar generation.

For each module, the following subsections present the relevant datasets, implementation approaches, experimental findings, and key observations.

4.1. Hand Landmarks Detection

4.1.1. Dataset

The dataset contains images of human hands annotated with 21 key points representing anatomical landmarks, including the wrist and joints of each finger. These annotations are visualised as dots overlaid on the images to reflect the structure and pose of the hand in various natural positions.

Figure 7 shows an example of a labelled image from the dataset, where the key points are clearly marked to indicate joint locations. This visual representation offers an intuitive understanding of the data used for training and evaluation.

Table 1. Hand Images and Landmarks dataset used

Task	Dataset Type	Instances	Description
Hand Landmarks Detection	Public	30k instances	Requires citation and used under fair use guidelines. [11] [12] [13]

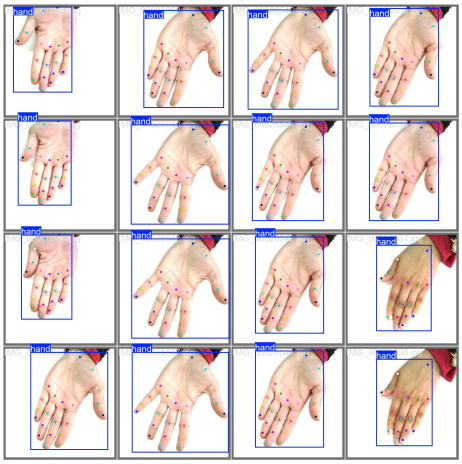

Fig. 7. Example of a labelled hand image with key points.

4.1.2. Implementation Details

For hand landmarks detection, the YOLO 8n model was fine-tuned. The training configuration is categorised as follows:

Model Training Setup

The training configuration defines the number of training steps, image processing size, and initial setup to balance model accuracy and training efficiency.

- **Epochs**: 100. A sufficient number to allow convergence without overfitting.
- **Batch Size**: 16. Selected to fit GPU memory while maintaining stable gradient updates.
- **Image Size**: 640. Balances input resolution for detailed hand landmark detection with real-time processing requirements.
- **Pretrained Weights**: Used. Leverages transfer learning to improve convergence speed and accuracy.
- **AMP (Automatic Mixed Precision)**: Enabled. Reduces memory usage and speeds up training with minimal accuracy loss.

Optimisation Settings

These parameters control how the model learns from data during training, ensuring stability and effective convergence.

- **Optimiser**: Auto. Automatically selects the best optimiser configuration based on task and hardware.

- **Initial Learning Rate** (`lr0`): 0.01. A standard starting point for YOLO models, balancing stability and convergence speed.
- **Learning Rate Factor** (`lrf`): 0.01. Controls the decay of the learning rate over time.
- **Momentum**: 0.937. Helps accelerate gradients in the relevant direction, smoothing the updates.
- **Weight Decay**: 0.0005. Prevents overfitting by penalising large weights.
- **Warmup Epochs**: 3.0. Stabilises training in the early stages by slowly ramping up the learning rate.
- **Warmup Momentum**: 0.8. Gradually increases momentum during the warmup phase.
- **Warmup Bias Learning Rate**: 0.1. Specific to bias parameters, ensuring stability at the start of training.

Data Augmentation Techniques
Data augmentation was applied to improve generalisation and make the model robust to real-world variations in hand poses and appearances.

- **Translation Range**: 0.1. Simulates small shifts in hand position within the frame.
- **Scaling Range**: 0.5. Allows the model to handle variations in hand size and distance to the camera.
- **Horizontal Flip Probability** (`fliplr`): 0.5. Improves robustness to left and right hand symmetry.
- **HSV Hue Adjustment**: 0.015. Simulates slight lighting colour shifts.
- **HSV Saturation Adjustment**: 0.7. Increases variability in colour saturation for better generalisation.
- **HSV Value Adjustment**: 0.4. Adjusts brightness variations.
- **Auto Augment Policy**: RandAugment. Applies random augmentations to diversify the training data automatically.
- **Random Erasing Probability**: 0.4. Simulates occlusions and missing parts in images to increase robustness.
- **Mosaic Augmentation**: Enabled. Combines four images into one to expose the model to different contexts and scales.
- **Mixup Augmentation**: Disabled. Disabled to preserve fine-grained hand landmark structures critical for keypoint detection.
- **Copy-Paste Augmentation**: Enabled (Flip mode). Adds augmented object instances to training images, increasing variability.

Hardware and System Setup
Training was conducted on a GPU-enabled environment to ensure feasible training time, as CPU-only training would have been prohibitively slow for 100 epochs with augmentation-heavy pipelines. Additional system settings were configured to optimise data loading and improve computational efficiency.

- **Device**: GPU. A GPU was used to accelerate model training, significantly reducing training time compared to CPU.
- **GPU Model**: NVIDIA RTX 3060. Used for model training to accelerate computation and ensure practical training time for 100 epochs.
- **Workers**: 8. Multiple CPU workers were used for parallel data loading to prevent bottlenecks.
- **Precision**: AMP (Automatic Mixed Precision) enabled. Reduced memory usage and increased computation speed with minimal impact on model accuracy.
- **Deterministic Training**: Enabled. Ensured reproducibility by controlling random number generators across training processes.

Validation and Export Settings
After training, the model was validated and exported with the following settings:

- **Validation Split**: `val`
- **Validation Enabled**: True

For Mediapipe, no training or fine-tuning was required; the pretrained hand landmarks detection model was used directly.

4.1.3. Performance Metrics

Metrics used to evaluate the models include:

- **Precision (B, P)**: Measures how many of the detected objects are correct.
- **Recall (B, P)**: Measures how many actual objects were correctly detected.
- **mAP@50 (B, P)**: Mean Average Precision at IoU = 0.5, indicating how well the model detects objects.
- **mAP@50-95 (B, P)**: Mean Average Precision across multiple IoU thresholds (0.50 to 0.95), providing a more rigorous evaluation.

4.1.4. *Experimental Results*

For hand landmarks detection, YOLO fine tuned models and Mediapipe models were evaluated. YOLO models, although capable of detecting hands, did not perform well in identifying precise hand keypoints and was not sensitive to small regional changes. Based on observations from processed live video feeds, Mediapipe Landmark Detection was deemed more suitable for the use case, due to its higher precision and sensitivity to keypoint details, which are crucial for downstream gesture recognition.

Metric	YOLOv8n	YOLOv11n
Class	all	all
Images	7808	7808
Instances	7808	7808
Box Precision (P)	0.979	0.974
Box Recall (R)	0.983	0.970
mAP@50 (Box)	0.993	0.991
mAP@50–95 (Box)	0.930	0.890
Pose Precision (P)	0.920	0.846
Pose Recall (R)	0.884	0.786
Pose mAP@50	0.918	0.809
Pose mAP@50–95	0.801	0.573

Table 2. Comparison of evaluation metrics for YOLOv8n and YOLOv11n on the hand-keypoints dataset.

4.1.5. *Ablation Study*

An experimental comparison between YOLO 8n and Mediapipe was conducted. It was observed that YOLO 8n's detection of hand keypoints lacked accuracy and detail, affecting the performance of gesture recognition. Mediapipe provided more accurate and sensitive keypoints, supporting better gesture recognition performance. Based on actual observation results, Mediapipe was selected as the preferred model for hand landmarks detection.

The comparison in Figure 8 further illustrates the differences between YOLO 8n and Mediapipe in hand landmarks detection accuracy, which impacts gesture recognition downstream.

4.1.6. *Discussions and Limitations*

While Mediapipe was more suitable for hand landmarks detection compared to YOLO 8n, there were limitations observed. YOLO 8n, despite fine-tuning, was not sensitive enough to small regional changes, resulting in inaccurate keypoint positioning. The selection of Mediapipe was based on the need for high precision and sensitivity, critical for the performance of gesture recognition downstream. However, no further detailed limitations beyond this comparison were discussed in the report or presentation.

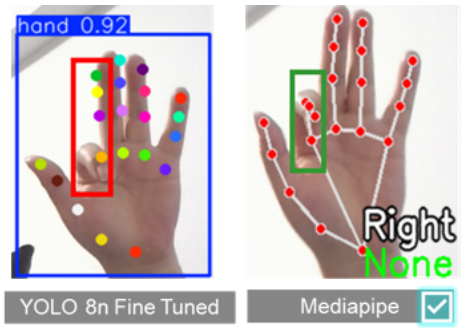

Fig. 8. Visual comparison of detected hand keypoints between YOLO 8n Fine-Tuned (left) and Mediapipe (right). YOLO 8n shows less accurate keypoint placement, whereas Mediapipe detects finer, structured landmarks necessary for gesture recognition.

4.2. Hand Gesture Recognition

4.2.1. *Dataset*

Table 3. Gesture Recognition Datasets Used

Task	Dataset Type	Instances	Description
Gesture Recognition	Manual Collection	300–600 (minimal 100 per class, min. 3 classes)	Captured using webcam and Mediapipe hand landmarks.

To enable gesture recognition, vectors representing the spatial relationships between hand key points were extracted from the captured hand landmark data. These vectors capture the relative positions of joints and fingertips, providing a compact and informative description of a hand's pose at a given moment.

The vectors were derived by computing the displacement between specific landmark pairs, resulting in two-dimensional (x, y) difference vectors. These structured representations form the foundation for both gesture recognition model training and cosine similarity matching. In cosine similarity matching, the system compares the current hand pose against stored gesture templates by evaluating the angle between corresponding vectors.

The extracted vectors are systematically stored in a JSON format for efficient access during both training and inference phases.

An illustration of the vectors used for gesture recognition is shown in Figure 9.

This vector-based approach provides robustness against variations in absolute hand position and orientation, focusing instead on the relative configuration of the fingers to improve recognition accuracy.

Fig. 9. Sample illustration of hand vectors for gesture recognition.

4.2.2. Captured Hand Gesture

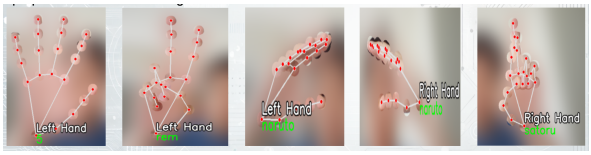

Fig. 10. Sample illustration of captured hand gesture.

4.2.3. Implementation details

For gesture recognition model training, each captured hand landmark vector was annotated with a corresponding gesture label. The data is then fit to several traditional classification models, for example multi-layer perceptron, decision tree, random forest, KNN were tested. The vector data preprocessing and training implementation and configuration is as follows.

Data preprocessing
One hot encoding is used to convert categorical data in json file into a one-hot encoded format, one additional step is added to filter all zero values after one-hot encoding to ensure there are no invalid labels in dataset. An additional label encoder step is used before models training like Random Forest to convert categorical string labels (e.g., 'saturo','rem') into numerical values (0,1,2, etc.). The data set is divided into training 80% and testing 20% to evaluate the performance of the model on unseen data. Random state is set to 42 to ensure the reproducibility of the split.

Model Optimization
Data augmentation

- **Position Noise**: Adds small Gaussian noise ($\sigma = 0.02$) to landmark positions for variation.
- **Rotation Transformations**: Random rotation between -0.15 and 0.15 radians.
- **Scaling Variation**: Random scaling between 0.9-1.1 for size invariance.
- **Perspective Distortion**: Simulates different viewing angles through non-uniform scaling.
- **Multiple Augmentations**: Creates 5 augmented samples per original sample.

Model-Specific Hyperparameter Optimization

Random Forest

- **Tree Count Optimization**: Tests 50, 100, and 150 estimators.
- **Depth Control**: Variable depths (4, 6, 8, None) prevent overfitting.
- **Node Split Quality**: Minimum samples for splits (2, 5, 10) and leaves (1, 2, 4).
- **Feature Randomization**: Tests both sqrt(n) and 50
- **Feature Selection**: Evaluates different feature subset sizes (all, 80

Decision Tree

- **Pruning Parameters**: `Optimisesmax_depth` set to 3, 5, 7, or `None` to control tree complexity.
- **Leaf Size Control**: Enforce minimum samples per leaf $(3, 5, 10)$ for generalization.
- **Split Threshold**: Minimum samples required for node splitting: $[5, 10, 15]$.
- **Feature Subsets**: Tests sqrt(n) and 80

K-Nearest Neighbors

- **Neighbor Count Optimization**: Test different k values $(3, 5, 7, 9)$.
- **Distance Weighting**: Compares uniform vs. distance-weighted voting.
- **Distance Metrics**: Manhattan distance tests (p = 1) and Euclidean distances (p = 2).

Multi-Layer Perceptron

- **Architecture Search**: Tests various network topologies (single and multi-layer).
- **Regularization Tuning**: Alpha values from 0.0001 to 0.1 control weight penalties.

- **Early Stopping**: Prevents overfitting by monitoring validation performance.
- **Batch Size Variation**: Tests 32 and 64 sample batches.
- **Learning Rate Tuning**: Tests 0.001 and 0.0001 initial rates.

Several techniques are used to prevent overfitting

- **Early stopping in MLP**
- **Grid search parameters that favor regularization**
- **Monitoring train vs. test performance gaps**
- **Creation of ensembles when models show suspiciously high accuracy**

4.2.4. Performance metrics

For the traditional machine learning model performance evaluation, `predict_proba` is used, a detailed classification report (precision, recall, F1-score) is generated.

- **Precision (B, P)**: Measures how accurate the model predict a specific hand gesture.
- **Recall (B, P)**: Measures how many actual instances of a gesture were correctly detected.
- **F1-score**: The harmonic mean of precision and recall, providing a single balanced metric.
- **Support**: The number of actual occurrences of each gesture class in test dataset.

4.2.5. Experimental results

Table 4. Performance results for traditional machine learning models

Model	Test Acc.	CV Acc.	F1	Prec.	Recall
RF*	0.9977	0.9924	0.9977	0.9977	0.9977
DT*	0.9988	0.9965	0.9988	0.9988	0.9988
KNN	0.9965	0.9974	0.9965	0.9965	0.9965
MLP	0.9977	0.9988	0.9977	0.9977	0.9977

*RF: Random Forest, DT: Decision Tree

4.2.6. Ablation study

An experimental comparison between different traditional machine learning models. According to the performance evaluation, all models tested have similar performance, which is higher than 0.99. Given that we have small dataset for each class (100 - 200 set) and we use hand gesture landmarks instead of capturing and labeling on raw image, we recommend to use decision tree based on its simplicity and higher performance.

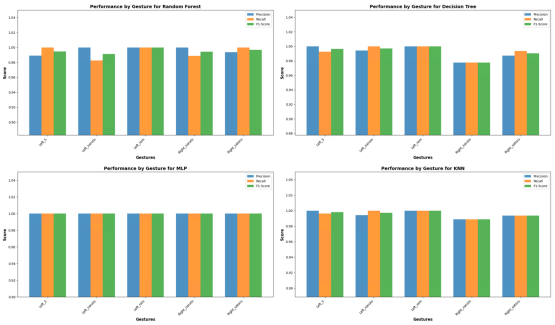

Fig. 11. Performance results by gestures per models.

4.2.7. Discussions and limitations

Generally, the trainings of traditional machine learning models are faster and less resource-intensive. The sklearn models can run on CPU-only systems. They also work well with smaller datasets, less prone to overfitting when training example are scarce. Traditional machine learning models often have smaller model sizes, offering better performance and the ability to run on resource-constrained devices with minimal overhead. ML models also work well with flattened vectors and do not require complex tensor reshaping. Although deep learning CNN approach preserves spatial relationships and can automatically extract features, batch normalization and dropout helped preventing overfitting and more robust to variations in hand size and position. However, there is no significant difference was observed between the performance of traditional machine learning models and also because we are using hand gesture landmarks as dataset instead of raw image of hand gestures. The dataset is also relatively small, no significant differences in training time were observed among models.

4.3. Diffusion and LoRA

4.3.1. Dataset

Table 5. Images Datasets for LoRA Used

Task	Dataset Type	Instances	Description
Low-rank Adaptation (LoRA) Model Training	Manual Collection	10–20 images per model	Fair use (non-commercial purposes), downloaded from the internet.

To support model adaptation via Low-Rank Adaptation (LoRA), a small set of sample images was curated to fine-tune the model towards a specific subject or visual style. These images serve as the training data for LoRA to learn personalised visual features without retraining the full model.

Figure 12 showcases a few of these images used during the LoRA training process.

Fig. 12. Sample images used for LoRA model training.

These samples enable the model to adapt to the target subject with minimal data and training time, making LoRA a highly efficient fine-tuning method for subject-driven generation tasks.

4.3.2. Implementation details

We implemented the Dreambooth LoRA model training using the Diffusers library. The model was fine-tuned with a low-rank adaptation (LoRA) approach, utilizing the following configuration:

- **Resolution**: All input images were resized to a resolution of 512×512 pixels.
- **LoRA Rank**: The rank of the LoRA adaptation was set to 128, balancing model capacity and computational efficiency.
- **Text Encoder Training**: The text encoder was jointly trained with the diffusion model to enhance text-image alignment.
- **Training Hyperparameters**:
  - Batch size: 1
  - Gradient accumulation steps: 1
  - Learning rate: 2×10^{-4}
  - Learning rate scheduler: Cosine with restarts
  - Warmup steps: 0
  - Maximum training steps: 15,000

Training was performed using the `accelerate` framework to optimise resource utilisation and scalability. The script `train_dreambooth_lora.py` was executed with the aforementioned parameters to ensure stable and effective fine-tuning of the model.

4.3.3. Ablation study

We conducted an ablation study to evaluate Dreambooth and LoRA. Without Dreambooth, generated images deviated from the target character. Without LoRA, training time increased 5x (from 1h to 5 hours on an NVIDIA A4090), though output quality remained similar. Both components are crucial for quality and efficiency.

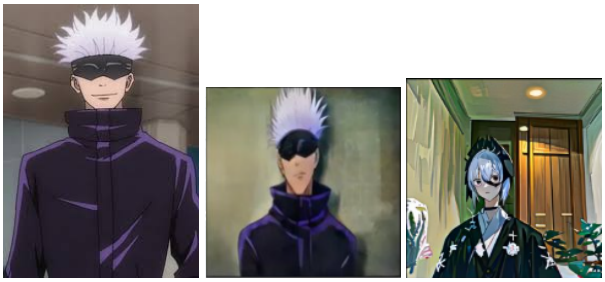

Fig. 13. Sample/with LoRA/without LoRA

4.3.4. Discussions and limitations

In selecting a foundation model for our project, we prioritized a balance between performance, quality, and development time. After careful consideration, we opted for Stable Diffusion 1.5 (SD1.5) due to its robust performance and well-established ecosystem, which aligns with our need for efficient development and reliable outcomes. However, in terms of image quality, more recent models such as SDXL[14] and FLUX[15] demonstrate superior capabilities and warrant further exploration for future iterations. On the performance front, models like SD-Turbo[16] and SDXL-Lightning[17] present promising alternatives, offering potential improvements in speed and efficiency. These options, while compelling, require additional development to ensure compatibility with frameworks like StreamDiffusion and thorough testing to validate their real-world effectiveness.

In our exploratory experiments, we attempted to integrate ControlNet[18] into the StreamDiffusion framework with the aim of enhancing the controllability of the generated results. However, the experimental findings indicated that the incorporation of ControlNet did not yield a significant improvement in controllability and instead resulted in a noticeable drop in frame rates. Considering the trade-off between performance and controllability, we opted not to use ControlNet in the final version. Nevertheless, we believe that further enhancing the controllability of StreamDiffusion during the generation process and improving the consistency of its output with input images or actions remains a promising direction for future research and warrants in-depth exploration. For example, future work could investigate alternative solutions such as T2I adapter[19], which may offer a better balance between control and robustness while minimising performance

overhead.

4.4. Prompt Engineering

The project evaluated three progressive approaches to prompt engineering, using Gojo Satoru from *Jujutsu Kaisen* as our case study. Each approach demonstrates incremental improvements in character consistency and output quality.

4.4.1. Initial Approach

> *"Create an image of Gojo Satoru from Jujutsu Kaisen, featuring his signature white hair, blindfold or dark sunglasses, and his black high-collared outfit. Position him confidently, with a playful smile, against a dynamic background of blue energy and swirling cursed motifs, emphasizing his powerful and enigmatic presence, ultra-detailed, 8K."* [8]

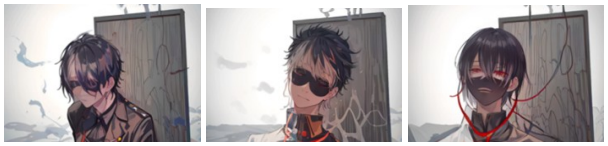

Fig. 14. Initial prompt outputs showing notable variations

1. **Hair Consistency**: 0/3 accuracy — outputs showed blonde, gray, and off-white variants
2. **Eye Colour**: 0/3 accuracy — included red eye colour
3. **Outfit Colour**: 2/3 accuracy — jacket colours ranged from black, to ivory to white
4. **Accessories**: 0/3 accuracy — sunglasses appeared or were more rectangular like blind folds
5. **Background Elements**: 0/3 accuracy — energy effects oscillated between wispy and geometric

This initial prompt, although specified to high detail, lacked specific structural components or emphasis weighting to guide the diffusion model effectively. The natural language phrasing allowed creative flexibility, but it also allowed for ambiguity, leading to an uneven range of interpretations across avatar generations. In addition, the lack of model-friendly tag formatting such as parentheses or class token usage made it difficult for the generator to prioritize core visual traits.

4.4.2. Engineered Approach

> *"1guy, (masterpiece), (ultra-detailed), (best quality), blue eyes, (dynamic pose:1.1), short spiky white hair, dressed as a black high-collared sorcerer, surrounded by blue energy and swirling cursed motifs..."* [7]

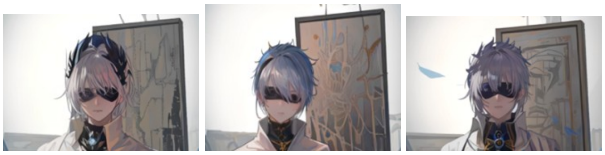

Fig. 15. Engineered prompt outputs showing improved consistency

1. **Hair Consistency**: 2/3 accuracy — length varied between chin-length hair and bob cut
2. **Eye Colour**: 3/3 accuracy — mostly dark and mysterious quality
3. **Outfit Colour**: 2/3 accuracy — consistent white with proper collar formation
4. **Accessories**: 3/3 accuracy — blindfold appeared in 3 generations
5. **Background Elements**: 2/3 accuracy — energy patterns showed similar swirling behaviour

This intermediate prompt employed parenthetical tags such as (masterpiece) and (ultra-detailed) to boost quality and detail fidelity. The use of (dynamic pose:1.1) helped influence the model toward a more action-oriented pose, while the direct tagging of visual attributes like "short spiky white hair" mitigated prior inconsistencies. Although some deviations persisted, the structured and semi-tokenized approach significantly narrowed the model's interpretive creativity and increased model fidelity.

4.4.3. Engineered Approach (Specific)

> *"1guy, (masterpiece:1.3), (ultra-detailed:1.2), (best quality), (official art:1.1), blue eyes, (dynamic pose:1.1), short spiky white hair, (sorcerer outfit:1.3): black high-collared..."* [7]

1. **Hair Consistency**: 3/3 accuracy — maintained spiky white style within 5cm of hair length variance
2. **Eye Colour**: 3/3 accuracy — perfect blue with consistent glow intensity

Fig. 16. Specific engineered prompt outputs with highest consistency

3. **Outfit Colour**: 3/3 accuracy — texture and collar height showed negligible variation
4. **Accessories**: 3/3 accuracy — dark eye surrounding appeared in all generations
5. **Background Elements**: 3/3 accuracy — energy patterns matched frequency and density

The final iteration demonstrated the highest level of consistency across all key attributes. Weighted emphasis, as seen in syntax terms like (masterpiece:1.3) and (sorcerer outfit:1.3), provided stronger conditioning signals for avatar generation. Tagging "official art" ensured a visual style emulative of canon illustrations. This form of prompt engineering strikes a balance between artistic freedom and precise recreation. Thus, the specific approach was observed to reduce hallucinations in non-prioritised traits.

4.4.4. Discussions and Limitations

Our incremental approach to prompt engineering demonstrated notable improvements in generating consistent and accurate depictions of Gojo Satoru from *Jujutsu Kaisen*. The final engineered prompt achieved increased accuracy across the key characteristics, including hair, eye colour, outfit, accessories, and background elements. This success highlights the effectiveness of structured prompts with weighted emphasis and specific tagging in guiding diffusion models toward desired outputs.

However, there remains some slight limitations. First, while the final prompt (specific approach) reduced hallucinations in non-prioritized traits, it did not completely eliminate them. [20] For example, very subtle variations in the texture of Gojo's outfit or slight differences in the swirling energy patterns were still present across generations. This suggests that while weighted emphasis enhances consistency, it may not fully constrain the model's creative freedom, leading to minor variations.

Second, the process of prompt engineering is naturally iterative and time-consuming [21]. Each refinement requires careful analysis of the generated images, adjustment of prompt elements, and re-evaluation, which can be manual-intensive. Additionally, the effectiveness of a prompt can vary across different diffusion models, necessitating further experimentation and adaptation, making the prompts less universally applicable to different diffusion models.

In conclusion, while our engineered approach significantly improved the consistency and accuracy of the avatar generation of Gojo Satoru, challenges related to creative variability and the iterative nature of prompt engineering remain. Further research should focus on developing methods to balance model creativity with user specification, as well as streamline the prompt engineering process to raise efficiency.

5. CONCLUSIONS AND FUTURE WORK

The project presents a practical and innovative approach to real-time avatar generation through the integration of gesture recognition, prompt-based image-to-image diffusion, and system-level stream integration. Our team successfully developed a working prototype capable of generating stylised avatars in real time based on webcam input and user-defined prompts, with additional support for gesture-triggered changes in avatar behaviour or appearance.

Throughout the development process, we faced several constraints that shaped our design choices. Limited computational resources—particularly GPU memory—necessitated lightweight inference pipelines and selective use of pre-trained diffusion models. Time constraints also influenced the scope of gesture training and system polish. Despite these limitations, we achieved key milestones such as LoRA-based fine-tuning, prompt switching via gesture detection, and StreamDiffusion integration for low-latency performance.

From a team perspective, this project was a valuable interdisciplinary experience that blended concepts from computer vision, machine learning, human-computer interaction, and systems engineering. We encountered practical challenges in tuning ML models for real-time performance, managing asynchronous modules, and optimising visual coherence in avatar generation.

Looking forward, several directions stand out for future development:

- **Improved model stability:** Enhancing temporal coherence in the diffusion output to minimise frame inconsistencies during continuous generation.
- **Expanded gesture set:** Training on a broader and more dynamic set of hand gestures to allow more nuanced avatar control.
- **User interface refinement:** Developing a visual interface for non-technical users to configure prompts, presets, and gesture bindings intuitively.
- **Scalability and deployment:** Exploring edge deployment strategies or web-based streaming options to make the system accessible beyond local GPU machines.
- **Multimodal input:** Incorporating audio cues or facial tracking for more immersive and expressive avatars.

If given more time and resources, we would focus on improving robustness, automating data set collection, and exploring real-world deployment via live streaming platforms like OBS. Ultimately, we envision Tanuki Leaf as an accessible and modular toolkit for creators looking to engage their audiences through expressive digital identities.

6. AUTHOR CONTRIBUTIONS

The development of the Tanuki Leaf system was a collaborative project, with each team member taking responsibility for distinct modules while also contributing to system integration, testing, and iterative improvements. Below is a breakdown of the roles based on individual and shared contributions.

6.1. Individual Roles and Focuses

- **Tan Eng Hui** served as the team leader and was responsible for the overall system architecture and integration. In addition to coordinating task distribution and managing project milestones. Responsible for ensuring that the various components and user interface were properly integrated and functioning as a cohesive system. Implemented the hand landmark classification and hand gesture recognition based on cosine similarity on the landmark vectors.

- **Wang Tao** led the exploration of LoRA and diffusion related technology and development. He trained the fine-tuned DreamBooth model for custom avatar generation and also experimented with different methods and weight configurations to optimise visual fidelity during image-to-image diffusion.

- **Hu Lei** worked on the gesture recognition pipeline, which involved collecting a set of custom gesture data using Mediapipe and training traditional ML classifiers to support real-time gesture mapping.

- **Ho Zi Hao Timothy** focused on prompt engineering, optimising prompt structure and using positive and negative prompts to refine image generation. In addition, implemented the automatic speech recognition (ASR) module to support voice-based interaction with the application.

6.2. Common Tasks and Collaboration

All members contributed to the literature review, UI/UX design decisions, and testing in different runtime environments. Collaborative efforts included fine-tuning diffusion outputs, evaluating gesture mappings, debugging and performance tuning , especially for performance bottlenecks and integration issues.

6.3. Challenges and Learning Curve

A key challenge for the team was the steep learning curve associated with emerging technologies such as DreamBooth, LoRA, StreamDiffusion, and gesture-based control. Due to limited documentation and lack of community support, which stemmed from the emerging and niche nature of the technologies used, the team had to quickly understand, implement, and test each component within a very short time frame. This was necessary to ensure proper system integration and seamless cooperation between all components.

Resource limitations, particularly in GPU memory and compute time, constrained our ability to scale training and experiment with advanced configurations.

Despite these challenges, the team remained adaptable, demonstrating strong collaborative problem-solving and resilience under time pressure.

7. REFERENCES

[1] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang, "Diffusion models: A comprehensive survey of methods and applications," 2022.

[2] H. Ye et al., "Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models," *arXiv preprint arXiv:2308.06721*, Aug. 2023.

[3] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman, "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[4] Qixun Wang, Xu Bai, Haofan Wang, Zekui Qin, and Anthony Chen, "Instantid: Zero-shot identity-preserving generation in seconds," *arXiv preprint arXiv:2401.07519*, 2024.

[5] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022.

[6] Electroverted, "Prompt writing guide - part 1: Basic," CIVITAI, June 2024, Retrieved April 22, 2025.

[7] Generation data, "Kohaku v2," CIVITAI, Jan. 2023, Retrieved April 22, 2025.

[8] kblueleaf, "Kohaku v2," CIVITAI, Aug. 2023, Retrieved April 28, 2025.

[9] Google Cloud, "Speech-to-text documentation," 2025, Accessed: 2025-04-28.

[10] Diego Campos-Sobrino, Mario Campos-Soberanis, Iván Martínez-Chin, and Víctor Uc-Cetina, "Fixing errors of the google voice recognizer through phonetic distance metrics," *arXiv preprint arXiv:2102.09680*, 2021.

[11] M. Afifi, "11k hands: Gender recognition and biometric identification using a large dataset of hand images," *Multimedia Tools and Applications*, vol. 78, no. 15, pp. 20835–20854, Aug. 2019.

[12] Ritika Giri, "2000 hand gestures dataset," Kaggle Dataset, 2022, Accessed: 2025-04-19.

[13] Sparsh Agrawal, "Gesture recognition dataset," Kaggle Dataset, 2022, Accessed: 2025-04-19.

[14] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach, "Sdxl: Improving latent diffusion models for high-resolution image synthesis," 2023.

[15] Black Forest Labs, "Flux," `https://github.com/black-forest-labs/flux`, 2024.

[16] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach, "Fast high-resolution image synthesis with latent adversarial diffusion distillation," 2024.

[17] Shanchuan Lin, Anran Wang, and Xiao Yang, "Sdxl-lightning: Progressive adversarial diffusion distillation," 2024.

[18] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala, "Adding conditional control to text-to-image diffusion models," 2023.

[19] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie, "T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models," *arXiv preprint arXiv:2302.08453*, 2023.

[20] Tanmoy Chakraborty and Shams Masud, "The promethean dilemma of ai at the intersection of hallucination and creativity," *Artificial Intelligence and Machine Learning*, aug 2024, Retrieved April 28, 2025.

[21] X. Chen et al., "Learning from mistakes: Iterative prompt relabeling for text-to-image diffusion model training," *ACL Anthology*, vol. 2024, no. 165, pp. 2937–2952, 2024.