

Final report and research findings

By Parinda Pannoon

Introduction

The fashion industry is experiencing a significant transformation through AI-powered virtual try-on technology, reshaping how consumers shop and brands engage with customers.

The current market trend of virtual try-on technology has moved from novelty to necessity in fashion retail. Major brands and retailers are investing heavily in AI solutions that allow customers to visualize clothing, accessories, and makeup on themselves without physical fitting rooms. The technology addresses one of e-commerce's biggest challenges: the inability to try before buying, which traditionally led to high return rates and customer dissatisfaction.

The advancement of virtual try-on

Recent AI breakthroughs have dramatically improved virtual try-on accuracy. Deep learning models now better understand fabric physics, draping, and how garments interact with different body types. Computer vision and generative AI enable realistic rendering of textures, patterns, and how clothing moves. Body measurement algorithms have become more precise, offering better size recommendations alongside visual previews.

The technology is making online shopping more interactive and personalized. Customers can upload photos or use live camera feeds to see how items look on their actual body shape and skin tone. Some platforms offer avatar-based try-ons where users create digital representations of themselves. Mobile integration has been crucial, with AR-enabled apps allowing instant try-ons while browsing.

Brands report reduced return rates, increased conversion rates, and improved customer confidence in purchasing decisions. Virtual try-on also generates valuable data about customer preferences and fit issues, helping companies optimize their designs and inventory. The sustainability angle is significant too, fewer returns mean reduced shipping emissions and waste.

Emerging Trends of AI image generation

Hyper-personalization is the next frontier, with AI learning individual style preferences and body characteristics. The trajectory is clear: AI virtual try-on is becoming standard infrastructure in fashion retail, fundamentally changing the relationship between consumers and clothing. However, despite progress, the industry faces hurdles around accuracy across all body types, realistic fabric simulation, and privacy concerns regarding body data.

The Technology Behind the AI VTON

VTON technology core lies in a refined blend of artificial intelligence disciplines working seamlessly together:

- **Computer Vision:** Serving as the AI's "eyes," this system analyzes the user's photo to map body geometry, detect pose, and segment clothing. It enables precise garment replacement by understanding shape, proportion, and spatial alignment. This helps to ensure the garment is placed correctly and naturally on the user's body.
- **Generative AI (Diffusion Models):**
At the heart of the generation process is the *stable diffusion model*. Diffusion models generate photorealistic images by transforming random digital noise into coherent visuals under specific conditions — here, the user's body shape and garment attributes. The model effectively *inpaints* new clothing onto the person's image, preserving facial identity while simulating realistic texture, drape, and fit ([Skywork, 2025](#))

Objectives:

The current state in VTON using AI image generation, still faces a limitation on detecting eye positions and ratios. Moreover, the primary challenge is generating glasses with transparent lenses while maintaining the original eye position, color, and characteristics, ensuring natural and consistent integration (<https://civitai.com/models/1268453/comfyui-glasses-virtual-try-on>). Here FashlyAI defines the following goals to achieve for our MVP.

1. How to develop a sunglasses VTON model combined with the user's input face.
2. What approach can preserve human faces and sunglasses details precisely?

Feasibility study

Assess existing Virtual try-on models [both commercial and non-commercial]

1. **IDM-VTON** (<https://huggingface.co/yisol/IDM-VTON>) Or Improved Diffusion Models for Virtual Try-ON.

It consists of two different components: 1) the image prompt adapter (IP-Adapter) that encodes the high-level semantics of the garment, and 2) the UNet encoder, which is GarmentNet, that extracts low-level features to preserve fine-grained details.

License: Creative Commons Attribution Non Commercial Share Alike 4.0 (CC-BY-NC-SA-4.0)

Components: [IP-Adapter](#) for base codes.

[OOTDiffusion](#) and [DCI-VTON](#) for masking generation.

[SCHP](#) for human segmentation.

[Densepose](#) for human densepose.

2. **Virtual Try-On tool using IP-Adapter** (<https://github.com/tencent-ailab/IP-Adapter>)

IP-Adapter, an effective and lightweight adapter to achieve image prompt capability for the pretrained text-to-image diffusion models. The key design of our IP-Adapter is decoupled cross-attention mechanism that separates cross-attention layers for text features and image features. IP-Adapter is reusable and flexible. IP-Adapter trained on the base diffusion model can

be generalized to other custom models fine-tuned from the same base diffusion model. Moreover, IP-Adapter is compatible with other controllable adapters such as ControlNet, allowing for an easy combination of image prompt with structure controls. Components: utilize the open-source SD model as our example base model to implement the IP-Adapter. SD is a latent diffusion model conditioned on text features extracted from a frozen CLIP text encoder. The architecture of the diffusion model is based on a UNet with attention layers. License: Apache License Version 2.0

3. SM4LL-VTON

Release only demo. The sm4llvton family consists of several lightweight models, each an expert in a specific VTON domain. This specialization allows them to achieve state-of-the-art results on relatively small, targeted datasets. Inference is handled via ComfyUI, in an environment that mirrors the training conditions.

Components: They don't explicitly tell what base models have been used, no code or model released.

License: non-commercial license

<https://sm4ll-vton.github.io/sm4llvton/>

4. LoRA approach

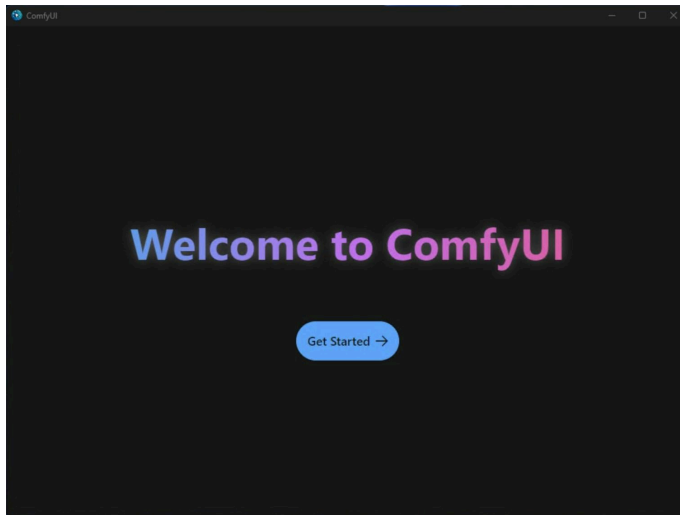
Low-rank adaptation (LoRA) is a technique used to adapt machine learning models to new contexts. It can adapt large models to specific uses by adding lightweight pieces to the original model rather than changing the entire model. LoRA: Low-Rank Adaptation Of Large Language Models ¹. In the paper, they showed that models reduced and retrained using LoRA outperformed base models on a variety of benchmark tasks. The model performance might be improved without requiring full fine-tuning and by using a significantly smaller number of trainable model parameters. Instead of retraining the whole model, LoRA freezes the original weights and parameters of the model as they are. Then, on top of this original model, it adds a lightweight addition called a low-rank matrix, which is then applied to new inputs to get results specific to the context. The low-rank matrix adjusts for the weights of the original model so that outputs match the desired use case (Noble, n.d.). This makes it possible to create custom styles or subjects by simply adding the LoRA file and a specific trigger word to your prompt.

In the feasibility study, LoRA can be developed through ComfyUI GUI to facilitate the components construction rather than complicated code. Additionally, LoRA is just a technique of deep learning, you can develop LoRA by any GUI such as Kohya-ss library, or directly on python.

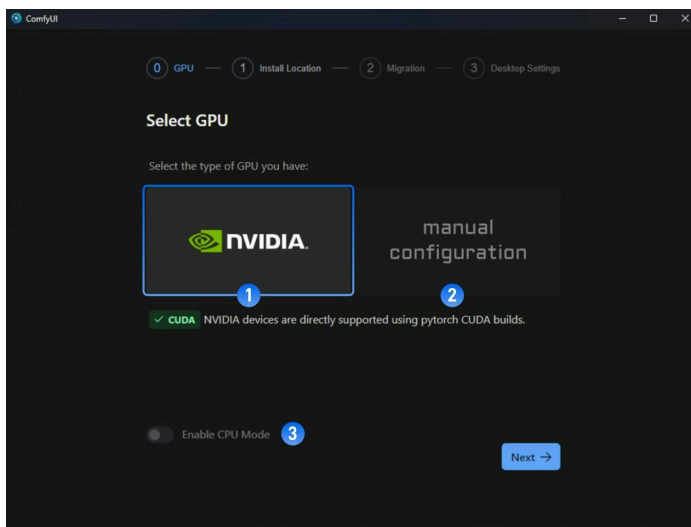
ComfyUI Desktop Initialization Process

1. Download ComfyUI via <https://docs.comfy.org/>

ComfyUI Desktop hardware requirements: NVIDIA GPU



2. Select GPU

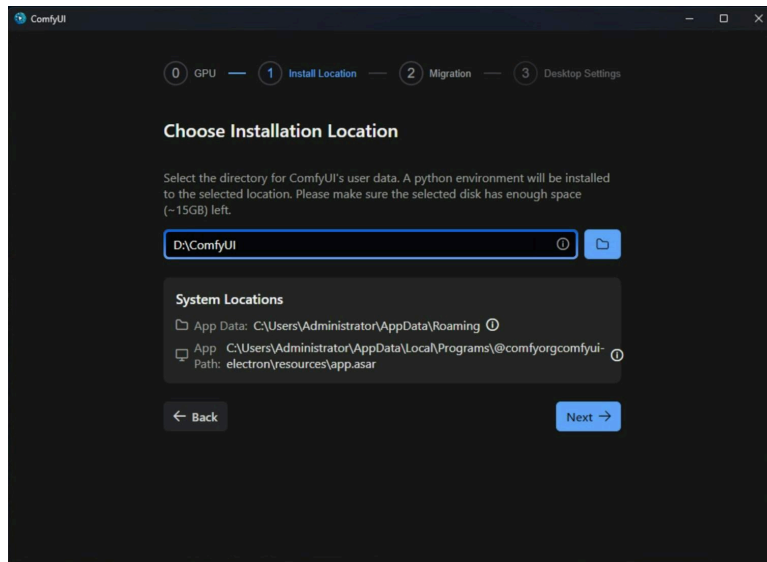


The three options are:

1. Nvidia GPU (Recommended): Direct support for pytorch and CUDA
2. Manual Configuration: You need to manually install and configure the python runtime environment. Don't select this unless you know how to configure
3. Enable CPU Mode: For developers and special cases only. Don't select this unless you're sure you need it.

Unless there are special circumstances, please select NVIDIA as shown and click Next to proceed.

3. Install location



In this step, you will select the installation location for the following ComfyUI content:

- Python Environment
- Models Model Files
- Custom Nodes Custom Nodes

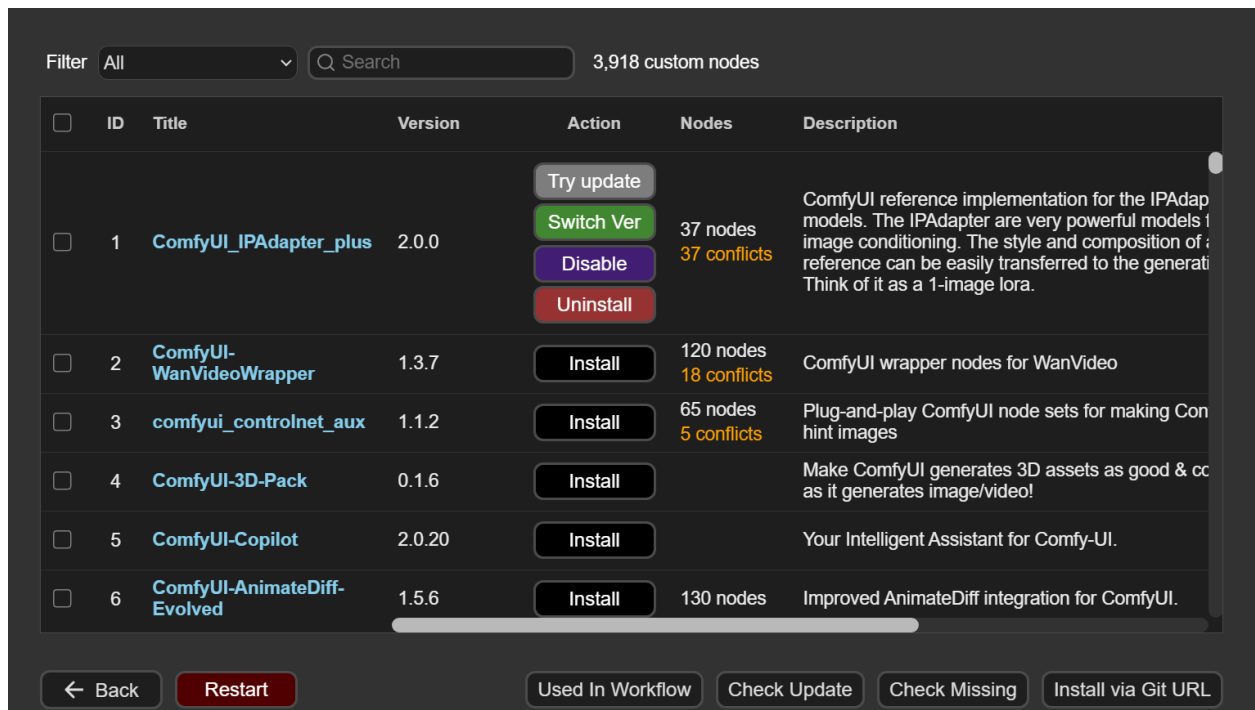
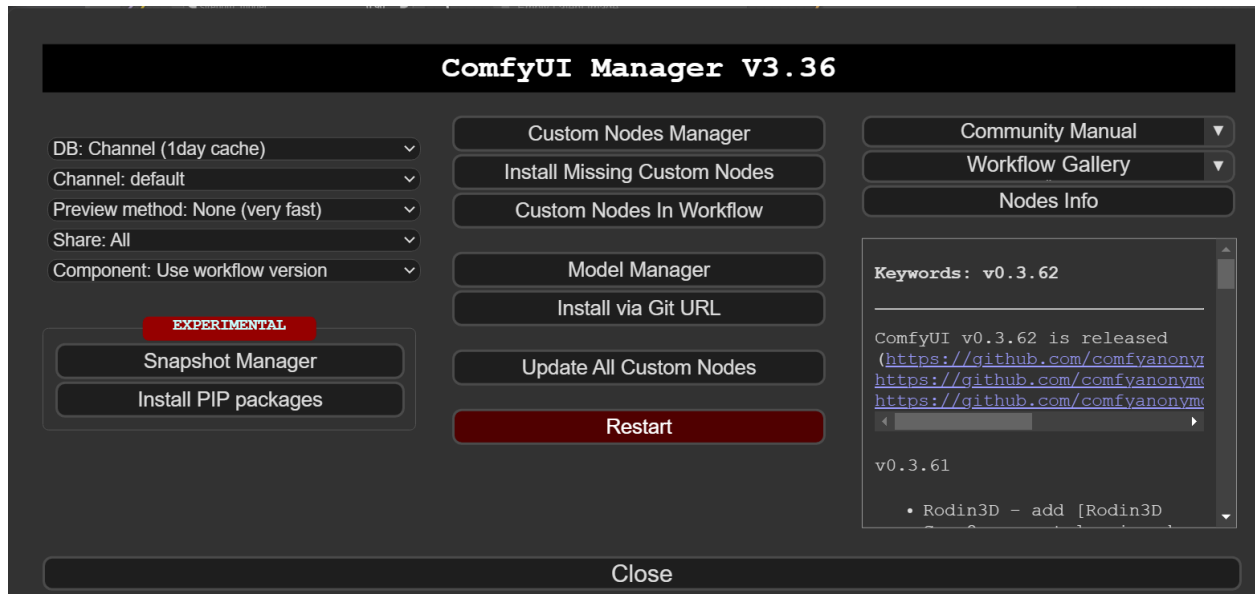
Recommendations:1. Please select a solid-state drive as the installation location, which will increase ComfyUI's performance when accessing models.2. Please create a separate empty folder as the ComfyUI installation directory. 3. Please ensure that the corresponding disk has at least around 15G of disk space to ensure the installation of ComfyUI Desktop

4. Complete the installation

If everything is correct, click next until install the software, the installer will complete and automatically enter the ComfyUI Desktop interface, then the installation is successful

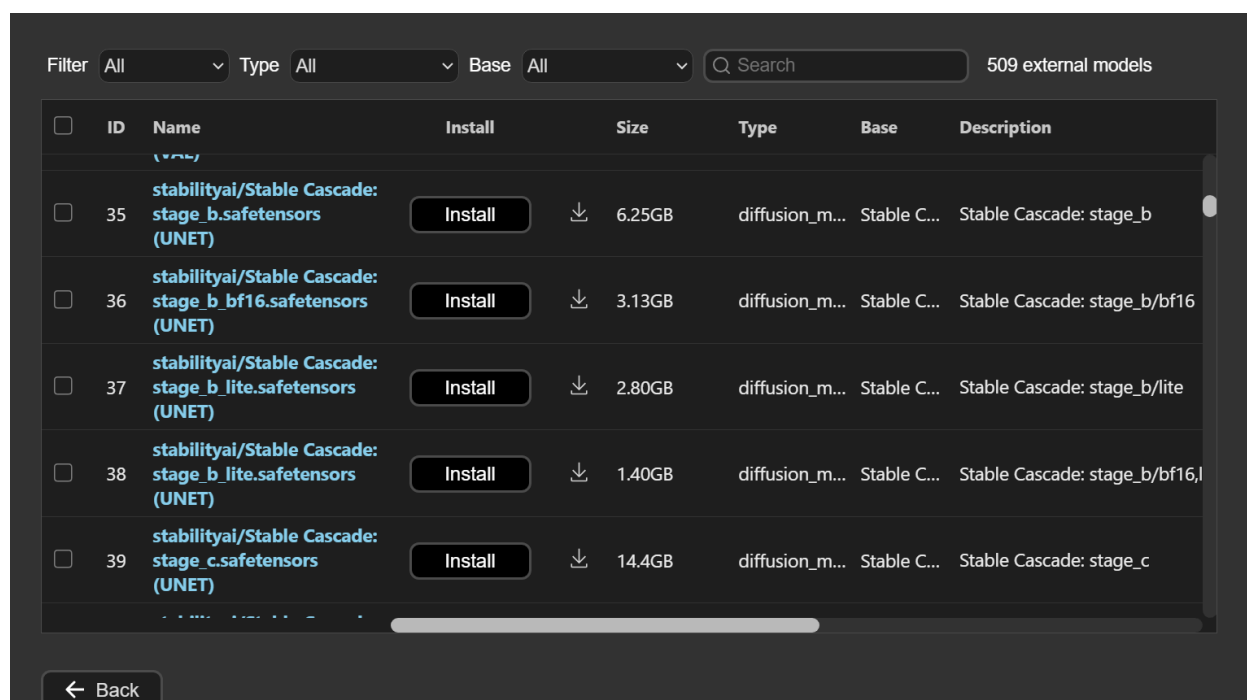
5. Custom Nodes manager

If there are missing nodes or you need to install additional nodes, please go to Manager>Custom nodes manager or Install missing nodes. Then install the preferred version and restart the Comfy UI to activate those new nodes.



6. Model manager

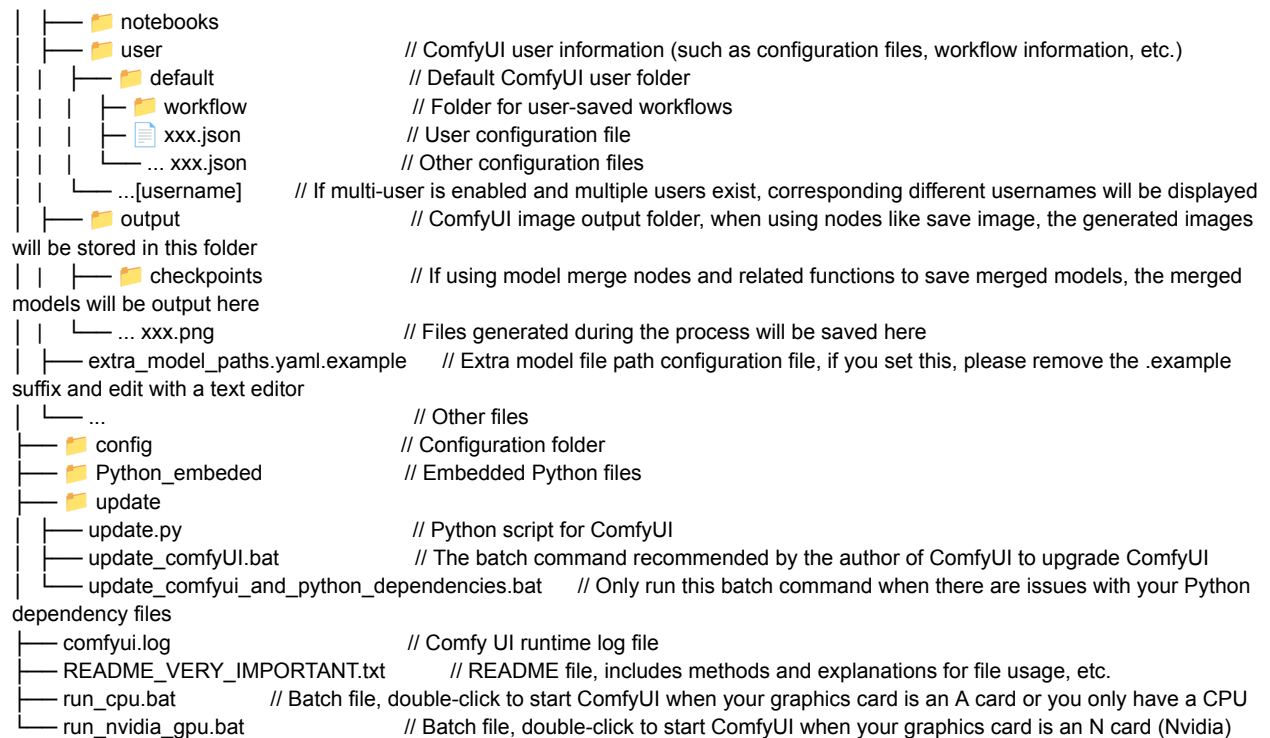
If you need some base models, IPAdapter, ControlNet or CLIP models, you can directly download through Huggingface or you can download and install the models through Model manager. Note that some models are very heavy weight like 16 GB so in this case you can select alternative versions.



Folder structure

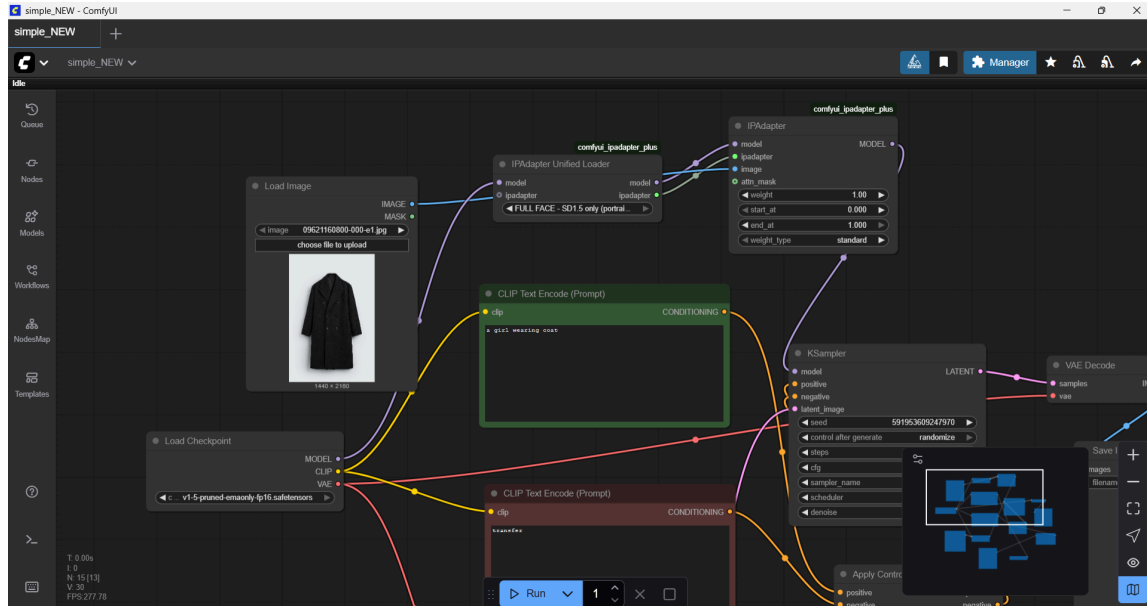
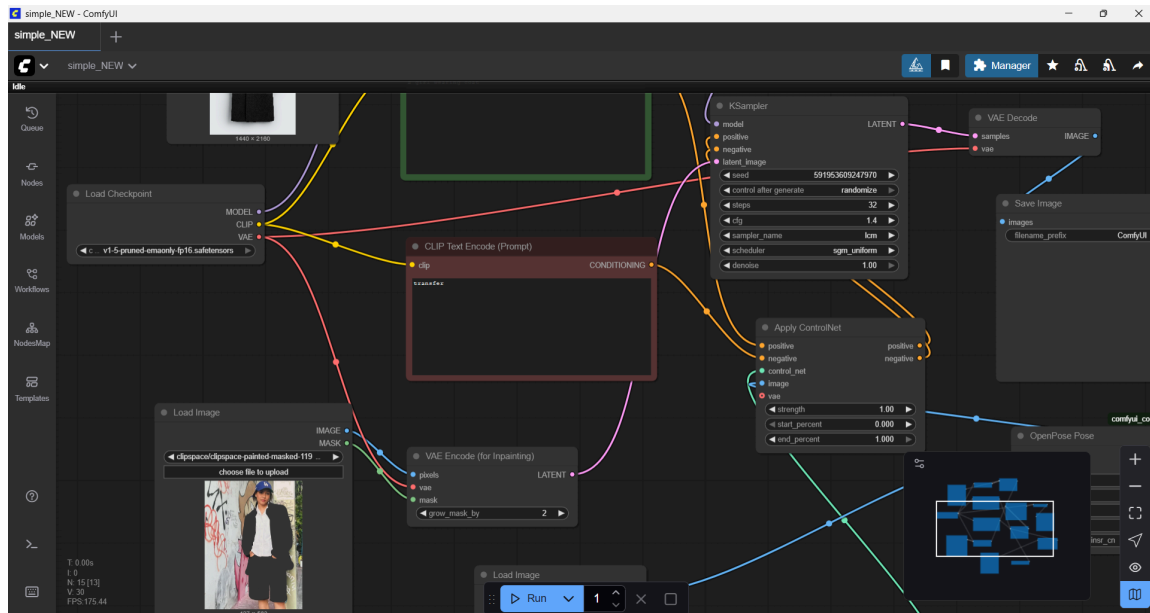
The following is a breakdown of the roles of some files in the ComfyUI installation directory (<https://comfyui-wiki.com/en/interface/files>).

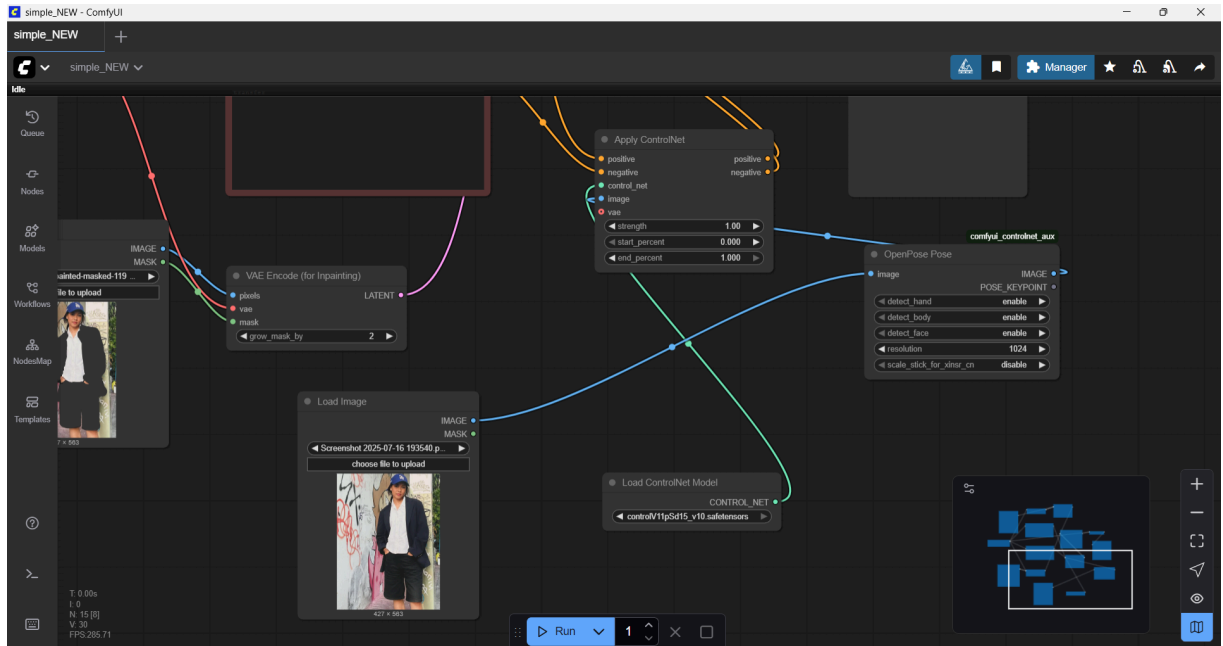
ComfyUI_windows_portable	
ComfyUI	// Main folder for Comfy UI
.git	// Git version control folder, used for code version management
.github	// GitHub Actions workflow folder
comfy	
comfy_extras	
custom_nodes	// Directory for ComfyUI custom node files (plugin installation directory)
input	// ComfyUI upload folder, when you use nodes like load image, the corresponding uploaded
images will be stored in this folder	
models	// Corresponding model file configuration folder
checkpoints	// Path for storing large model checkpoint files
clip	// Path for storing CLIP files
clip_vision	// Path for storing CLIP_vision files
configs	
controlnet	// Path for storing ControlNet model files
diffusers	
embedding	// Path for storing embedding model files
gligen	
hypernetworks	// Path for storing hypernetwork models
loras	// Path for storing Lora model files
style_models	
UNET	
upscale_models	// Path for storing upscale model files
vae	// Path for storing VAE model files
vae_approx	



Experiment1 (Cloth virtual tryon using Open pose controlnet)

This workflow combines Stable Diffusion XL, IPAdapter, and ControlNet (OpenPose) to virtually put clothes on a person's image, essentially a Virtual Try-On (VTON) system. It uses 2 image inputs which are human image + clothes and another one is used for masking.





Workflow Breakdown

1. Model & Components

- **LoadCheckpoint**: Loads the base SD 1.5 model (`v1-5-pruned.safetensors`).

2. Input Images

- **LoadImage**: Loads the *masked person image* (the area where new clothes will appear). Provides both IMAGE and MASK. Loads the *clothing reference image*. Loads the *original person reference* (for pose extraction).

3. Preprocessing

- **OpenPosePose**: Extracts pose keypoints from the person reference. This pose processor will guide ControlNet for realistic body alignment. In other words, it is a human pose detection library that works by detecting multiple "keypoints" in a human body and converting that information into a consistent "skeleton" representing the person (<https://openposes.com/>).

4. Control Network

- **ControlNetLoader**: Loads controlnet model that support SD 1.5
- **ControlNetApply**: Merges:
 - Positive/Negative CLIP conditioning,
 - Pose map,
 - SD1.5's conditioning and VAE.

- This ensures the generated clothing follows the subject's pose and body proportions.

5. CLIPTextEncode:

- One encodes the positive prompt — e.g. *“a girl wearing black coat”*.
- The other encodes the negative prompt — e.g. *“distorted”*. These guide generation quality and avoid visual artifacts.

6. IPAdapter Integration

IP-Adapter is a lightweight adapter designed to integrate image-based guidance with text-to-image diffusion models. The adapter uses an image encoder to extract image features that are passed to the newly added cross-attention layers in the UNet and fine-tuned (https://huggingface.co/docs/diffusers/en/using-diffusers/ip_adapter).

- **IPAdapterUnifiedLoader** loads the adapter preset (**VIT-G medium strength**).
- **IPAdapter** applies the clothing reference image as a visual guide. This transfers clothing *style, texture, and fit* into the final image.

7. Latent Processing

- **VAE-Encode (For inpaint)**: Converts the masked person image + mask into latent space for inpainting (filling the masked area with generated clothes).
- **KSampler**: Main diffusion process that combines model from IPAdapter: ControlNet conditioning and inpainted latent image, producing new latent with the outfit applied.
- **VAEDecode**: Converts the latent back into an image.

8. Output

- **SaveImage**: Saves the final try-on image. You can find the output in the output folder.

Key findings

Most virtual try-on models on Hugging Face focus on garment try-on, not accessories (e.g., sunglasses). Many are released under non-commercial licenses, limiting use cases.

For model limitations, few or no pretrained models exist specifically for sunglasses try-on.

OpenPose and related components lack eye detection and facial proportion mapping, reducing accuracy for face-related try-ons.

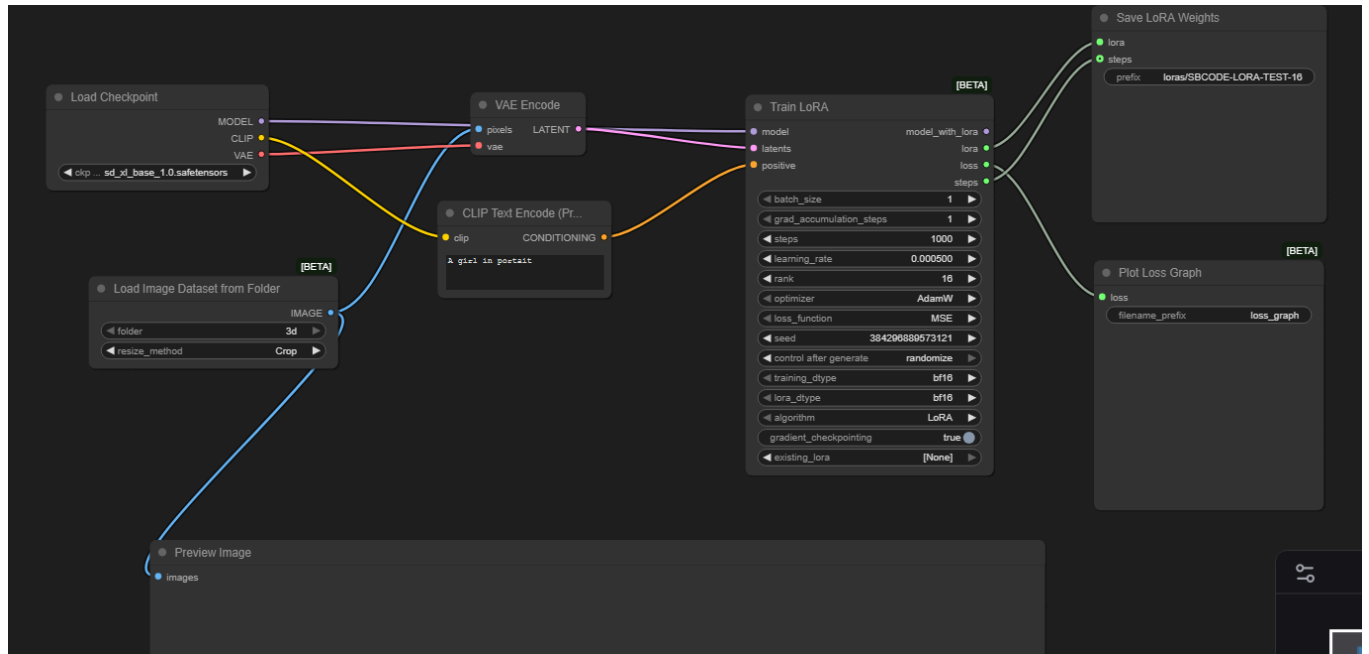
OpenPose is helpful for pose and masking guidance, but it does not automate the masking process. Users must manually define the mask areas for clothing regions.

For the output quality, it generates basic clothing textures (e.g., simple shirts), but lacks detail preservation and size accuracy, produced garments are non-hyperrealistic, appearing more stylized than photorealistics.

Experiment2 (LoRA finetuning human faces)

LoRA Workflow for training dataset

It fine-tunes specific layers of the model on a small dataset of human portraits, making the model learn consistent face structure, proportions, and details.



Workflow Breakdown

1. Base Model Loading

- **CheckpointLoader**: this node loads the main SDXL base model—`sd_xl_base_1.0.safetensors`

2. Training Data Input

- **Load Image Dataset From Folder**: this node loads a folder of face images (the training dataset). Optionally you can set the resize option or crop option for uniformity.

3. Latent Preparation

- **VAE Encode**: helps to convert real training images into latent tensors using the VAE. This helps the model understand the compressed features (shapes, edges) of the faces.

4. Prompt Conditioning

- **CLIPTextEncode**: encodes the descriptive text (e.g., "A girl in portrait"). Provides semantic conditioning during training — the model learns to connect text prompts with

the visual style of the dataset.

5. LoRA Training Process

- **TrainLoraNode** is the core of the workflow. This node takes a base model (from SDXL), Latent images (from VAE), Conditioning text (from CLIP). Then trains a LoRA module on the facial dataset. The suitable parameters for settings the LoRA trainer so far are in the following details;

Batch size: 1
Steps: 500
Learning rate: 0.0004
Rank: 256
Optimizer: AdamW
Loss: MSE
Algorithm: LoRA

The LoRA model output will be exported in the folder output/loras

6. Loss Tracking

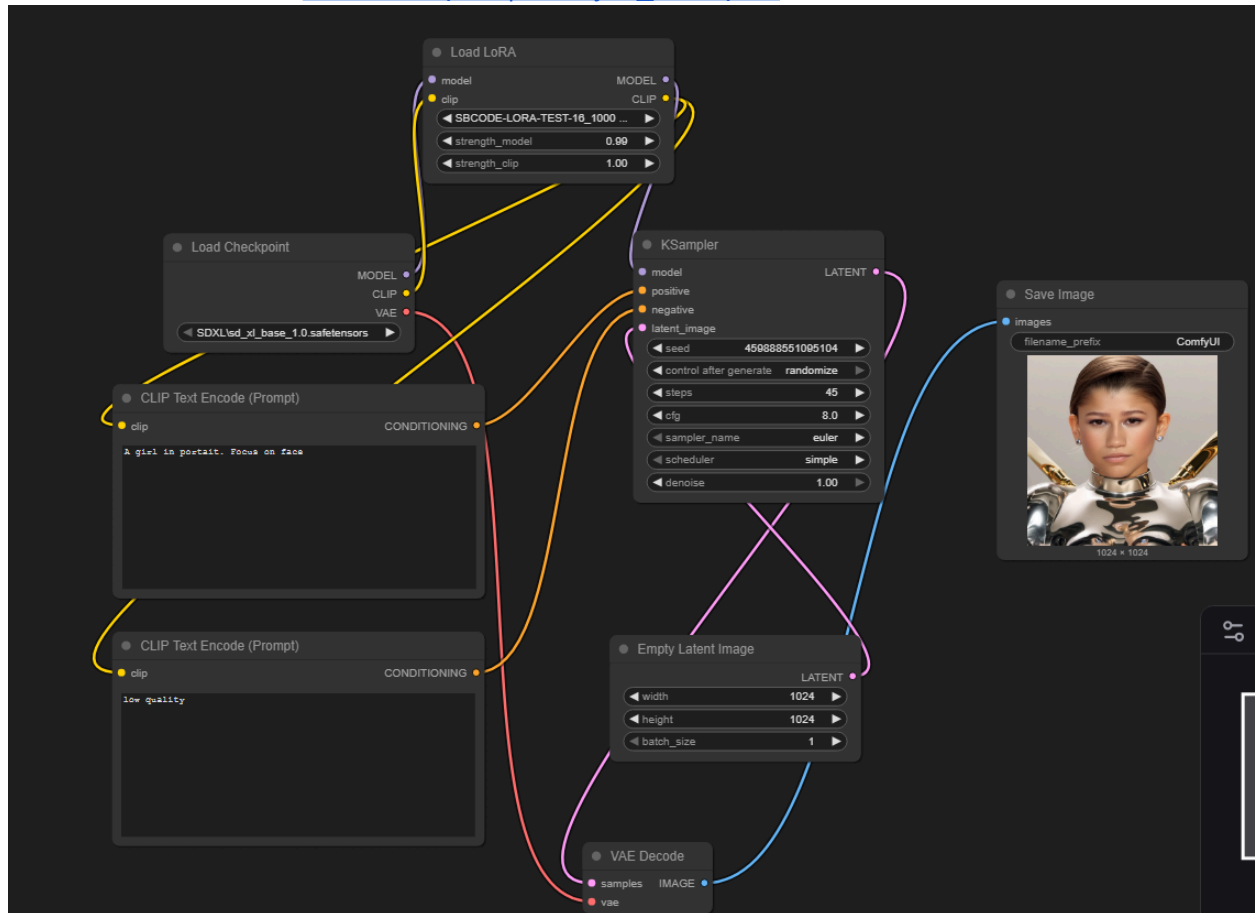
- **LossGraphNode**: Displays a real-time loss curve to evaluate how well the LoRA is learning facial consistency.

NOTE: The current workflow retrieved from the following link: Credit [SBCODE Tutorials](https://sbcode.net/genai/train-lora-node/)
Copyright © Sean Bradley <https://sbcode.net/genai/train-lora-node/>

LoRA Workflow for visualizing the trained model

This workflow is used for plot images that were trained in the previous step. This loads SDXL base, injects the trained LoRA, encodes positive/negative prompts, samples from an empty latent at a chosen size, decodes, and saves the preview image. It's for quickly checking if our LoRA preserves faces as intended.

Find more details here [Lora Examples | ComfyUI examples](#)



Workflow for executing LoRA model (output)

Workflow Breakdown

*Make sure you move the lora model from output folder to models/lora folder

- CheckpointLoader**: the node loads SDXL base (sd_xl_base_1.0.safetensors) and its CLIP/VAE.
- LoraLoader**: Injects the LoRA model (SB CODE-LORA-TEST-16_1000_steps_01001.safetensors) with strengths model=0.99, clip=1. You can readjust these to see identity/style balance as prefer.

3. **CLIPTextEncode (positive/negative):** Positive prompt is “A girl in portrait. Focus on face” (in the case of training a woman images). The negative prompt is “low quality”. Use clear face-centric phrasing to focus identity features.
4. **EmptyLatentImage:** Creates a blank latent canvas at 1024×1024 (batch 1). Change size to match your training/target use. In this case SDXL base model supports 1024 px.
5. **KSampler:**
 - Sampler: Euler,
 - Steps: 45,
 - CFG: 8,
 - Denoise: 1.
 - Keep seed fixed to compare runs while you vary LoRA strength.
6. **VAEDecode links to SaveImage node:** this decodes the latent and shows the preview image.

Key findings

A well-tuned LoRA in ComfyUI can significantly improve portrait realism and face preservation with minimal data, provided consistent image preparation and balanced hyperparameters. The best hyperparameters so far that achieved better image quality and reduced face distortion are:

- Steps: 500
- LR: 0.0004
- Rank: 256

Prompts used:

- In training: “a woman in portrait, half body”
- For visualization (positive): detailed portrait, direct gaze, gray background
- For visualization (negative): “low quality, multiperson”

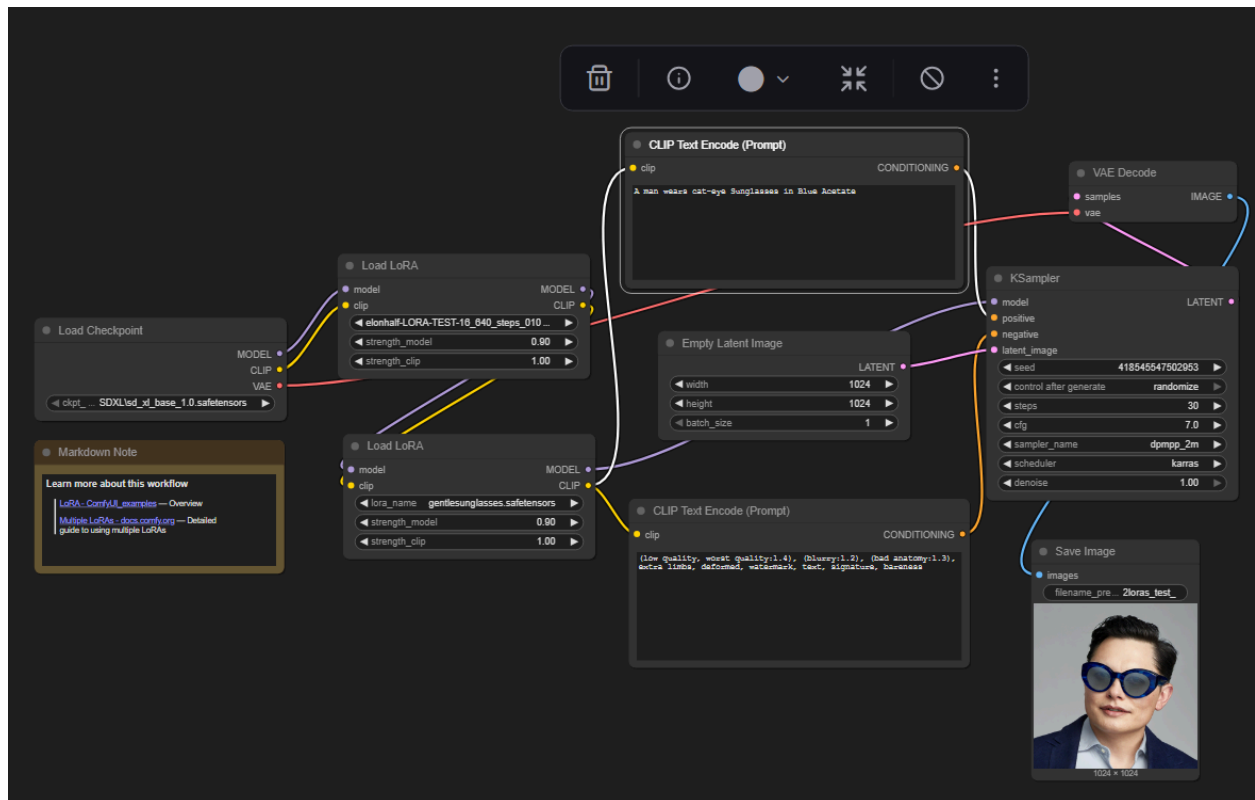
The finding found that 3–5 training images are sufficient for stability, but more images improve consistency. Image inputs should be cropped to the same extent and size - Image background should not be removed - The image inputs from the same set/outfits/backgrounds help the output more stable

Experiment3 (Multiple LoRAs)

This workflow allows you to apply and visualize multiple LoRAs at once (for example, one LoRA trained for facial identity and another for accessories like sunglasses).

It's a compositional inference setup, combining styles or learned details from several LoRAs in a single generation.

This workflow is only for visualizing 2 loras altogether, you need to separately train 2 loras in the previous workflow. This workflow retrieves from Comfy ui template – [ComfyUI Multiple LoRAs Example - ComfyUI](#),



Workflow Breakdown

*Make sure you move both lora models from output folder to models/lora folder

1. Base Model Loading

- CheckpointLoader: Loads the base model of sd_xl_base_1.0.safetensors

2. First LoRA Application

- LoraLoader: Loads LoRA 1 which is elonhalf-LORA-TEST-16_640_steps_01001.safetensors
-strength_model = 0.9

-strength_clip = 1.0

Adds this LoRA's learned weights (e.g., facial preservation or portrait fine-tuning) to the base model.

3. Second LoRA Application

- Loads LoRA 2 which is gentlesunglasses.safetensors

-strength_model = 0.9

-strength_clip = 1.0

This blends additional details (e.g., sunglasses style or accessory features).

4. Prompt Conditioning

- CLIPTextEncode (Positive Prompt): "A man wears cat-eye Sunglasses in Blue Acetate"
Keep the prompt simple and descriptive for the desired combination. This guides the model toward generating a person with distinct eyewear style.
- CLIPTextEncode (Negative Prompt): (low quality, worst quality:1.4), (blurry:1.2), (bad anatomy:1.3), extra limbs, deformed, watermark, text, signature, bareness. This filters out distortions, blur, and unwanted artifacts.

5. Latent Initialization

- EmptyLatentImage: Creates a blank latent canvas at 1024x1024, batch size = 1.

6. Sampling & Generation

- KSampler parameters
Steps: 30
CFG: 7
Sampler: dpmpp_2m
Scheduler: karras
Denoise: 1.0
Seed: randomize

This produces the combined output using both LoRAs' learned attributes.

7. Decoding and Output

- VAEDecode: Converts the latent result into a visible RGB image.
- SaveImage: Saves the result in folder outputs.

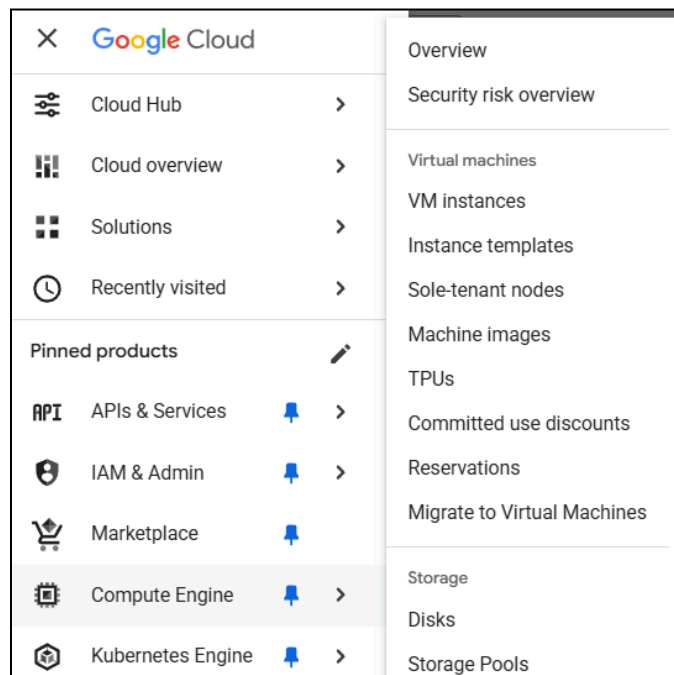
Key findings

The multiple LoRA setup effectively combines two components: facial features and sunglasses into a result. This demonstrates a promising approach for preserving both human facial identity and product details. Future improvements could involve integrating ControlNet to capture eye positions and facial proportions more accurately, allowing the sunglasses to align more naturally and realistically with the face.

GCP Set up and account setting

1. Go to GCP console, access IAM&ADMIN > VM Instances

1. Go to GCP console, access IAM&ADMIN > VM Instances



2. Select instance020250926-124146 where has gcp nvidia

VM instances

Create instance

Import VM

Refresh

Instances

Observability

Instance schedules

VM instances

Filter







Enter property name or value





<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	●	instance-20250926-124146	asia-east1-c			10.140.0.4 (nic0)		RDP <div></div>
<input type="checkbox"/>	●	instance-20251012-025458	asia-southeast1-a			10.148.0.3 (nic0)		SSH <div></div>

Related actions

3. You will be able to edit the VM information, this VM location is asia-east1-c because it supports the NVIDIA T4 server. Please check available location:

<https://docs.cloud.google.com/compute/docs/gpus/gpu-regions-zones>

Basic information	
Name	instance-20250926-124146
Instance Id	8746953638716060394
Description	None
Type	Instance
Status	 Stopped
Creation time	Sep 26, 2025, 2:50:14 PM UTC+02:00
Location 	asia-east1-c
Boot disk source image	windows-server-2025-dc-v20250913
Boot disk architecture	X86_64
Boot disk license type	PAYG (Pay-as-you-go)
Instance template	None
In use by	None
Physical host 	None
Maintenance status 	—
Reservations	Automatically choose
Labels	goog-ops-a... : v2-x86-tem...
Tags 	— 
Deletion protection	Disabled

Deletion protection	Disabled
Confidential VM service 	Disabled
Preserved state size	0 GB
Machine configuration	
Machine type	custom-6-26624-ext (6 vCPUs, 26 GB Memory)
CPU platform	Unknown CPU Platform
Minimum CPU platform	None
Architecture	x86/64
vCPUs to core ratio 	—
Custom visible cores 	—
All-core turbo-only mode 	—
Display device	Disabled Enable to use screen capturing and recording tools
GPUs	1 x NVIDIA T4
Resource policies	
Networking	
Public DNS PTR Record	None
Total egress bandwidth tier	—

4. You can set up or edit the GPU under machine configuration and edit GPU Type and number of gpu as you prefer. In this case we selected N1, NVIDIA T4. If you want to increase the spec, please request or increase the quota to GCP customer service.

How to Set up

1. VM instance> click to our VM> Edit> Machine configuration> select GPUs > Machine type custom> 6 vCPUs and 26 GB
2. Boot disk> Image > Window system 2025 data center
3. Quotas & System Limits > search for NVIDIA T4> request 1 GPU to our predefined region

Machine configuration

General purpose

Compute optimized

Memory optimized

Storage optimized

✓ GPUs

Graphics processing units (GPUs) accelerate specific workloads on your instances such as machine learning and data processing. [Learn More](#)

GPU type
NVIDIA T4

Number of GPUs
1

☐ Enable Virtual Workstation (NVIDIA GRID)

Series ?	Description	vCPUs ?	Memory
<input checked="" type="radio"/> N1	Balanced price & performance	1 - 96	1.8 - 624


Machine type

Choose a machine type with preset amounts of vCPUs and memory that suit most workloads. Or, you can create a custom machine for your workload's particular needs. [Learn more](#)

Preset

Custom

n1-standard-1 (1 vCPU, 3.75 GB memory)



vCPU

1

Memory

3.75 GB

5. The rest of information

Network interfaces

Name ↑	Network	Subnetwork	NIC type	Primary internal IP address	Alias IP ranges	IP stack type	External IP address	Network tier ?
nic0	default	default		10.140.0.4		IPv4	Ephemeral	Premium

Storage

Filter

Enter property name or value

?

|||

Name ↑	Size (GB)	Type	Data protection	Mode	When deleting instance
instance-20250926-124146 (Boot Disk)	256	Balanced persistent disk	default-schedule-1 Every day, starts between 4:00 PM and 5:00 PM	Read/write	Delete disk

Backup plan ?

Managed by [Backup and DR Service](#).

CREATE ON-DEMAND BACKUP

Backup schedule state ?	Not configured
Backup plan	—

Security and access

Shielded VM ?

Secure Boot ?	Off
vTPM ?	On
Integrity Monitoring ?	On

SSH Keys

SSH keys	None
Block project-wide SSH keys	Off

API and identity management

Service account	371712967612-compute@developer.gserviceaccount.com
Cloud API access scopes	Allow default access

✓ [Show details](#)

Management

Data Encryption

Key ID	—
Key name	—

Availability policies

VM provisioning model ?	Standard
Max duration ?	None
Preemptibility	Off (Recommended)
On VM termination ?	—
Graceful shutdown maximum duration	—
On host maintenance	Terminate VM instance
Host error timeout ?	—
Automatic restart	On (Recommended)
Customer Managed Encryption Key (CMEK) revocation policy	Do nothing

Sole-tenancy

Affinity labels	None
CPU Overcommit	Disabled

Set up password and username

The screenshot shows the Google Cloud Platform console interface. At the top, there's a header with a question mark icon and a hamburger menu icon. Below the header, there's a table with two columns: 'External IP' and 'Connect'. The first row shows an external IP of '35.194.171.127' with a link '(nic0)' and a dropdown menu set to 'RDP'. The second row shows an external IP of '35.194.171.127' with a link '(nic0)' and a dropdown menu set to 'SSH'. A context menu is open over the 'SSH' dropdown, showing four options: 'Set Windows password', 'View gcloud command to reset password', 'Download the RDP file', and 'Learn about Windows auth'. Below the table, there's a section titled 'Set up firewall rules' with a subtext 'Control traffic to and from a VM'.

External IP	Connect
35.194.171.127 (nic0)	RDP
35.194.171.127 (nic0)	SSH

- Set Windows password
- View gcloud command to reset password
- Download the RDP file
- Learn about Windows auth

Set up firewall rules
Control traffic to and from a VM

Access remote desktop connection through Window

Set up username and password following the initial GCP set up

- Add external IP: 34.81.169.184
- Username: parindapannoon
- Password: F\$bU)mMX,h[b;WD

Note: This account information can be changed anytime under setting Windows password*

Reference

https://huggingface.co/docs/diffusers/en/using-diffusers/ip_adapter

<https://sbcode.net/genai/train-lora-node/>

https://comfyanonymous.github.io/ComfyUI_examples/lora/

<https://docs.comfy.org/tutorials/basic/multiple-loras>

<https://docs.cloud.google.com/compute/docs/gpus/gpu-regions-zones>

<https://comfyui-wiki.com/en/interface/files>

<https://docs.comfy.org/>

<https://www.ibm.com/think/topics/lora>

<https://sm4ll-vton.github.io/sm4llvton/>

<https://github.com/tencent-ailab/IP-Adapter>

<https://huggingface.co/yisol/IDM-VTON>

<https://civitai.com/models/1268453/comfyui-glasses-virtual-try-on>

<https://skywork.ai/skypage/en/miragic-virtual-try-on-ai-fashion/1977569128918085632>

<https://openposes.com/>