**SOURCE CODE**

import gradio as gr

import numpy as np

import torch

from PIL import Image

import os

import datetime

from diffusers import StableDiffusionControlNetPipeline, ControlNetModel, UniPCMultistepScheduler

import mediapipe as mp

# Device

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

# Load ControlNet and Stable Diffusion Pipeline

controlnet = ControlNetModel.from\_pretrained(

"lllyasviel/sd-controlnet-openpose",

torch\_dtype=torch.float32

)

pipeline = StableDiffusionControlNetPipeline.from\_pretrained(

"runwayml/stable-diffusion-v1-5",

controlnet=controlnet,

safety\_checker=None,

torch\_dtype=torch.float32

)

pipeline.scheduler = UniPCMultistepScheduler.from\_config(pipeline.scheduler.config)

pipeline.to(device)

# Save folder

save\_folder = "saved\_outfits"

os.makedirs(save\_folder, exist\_ok=True)

# Options

genders = ["female", "male"]

# Pose Estimation Function

def estimate\_pose(image\_pil):

try:

mp\_pose = mp.solutions.pose

pose = mp\_pose.Pose(static\_image\_mode=True, model\_complexity=2)

image\_rgb = np.array(image\_pil.convert("RGB"))

results = pose.process(image\_rgb)

if not results.pose\_landmarks:

print("No landmarks.")

return None

annotated = image\_rgb.copy()

mp\_drawing = mp.solutions.drawing\_utils

mp\_drawing.draw\_landmarks(

image=annotated,

landmark\_list=results.pose\_landmarks,

connections=mp\_pose.POSE\_CONNECTIONS

)

return Image.fromarray(annotated)

except Exception as e:

print("Pose error:", e)

return None

# Generation Function

def generate\_outfit(image, gender, prompt):

if image is None:

return None, "Please upload or capture an image."

pose\_image = estimate\_pose(image)

if pose\_image is None:

return None, "Pose detection failed. Try a full-body image."

full\_prompt = f"{gender} outfit"

if prompt:

full\_prompt += f", {prompt}"

try:

resized = pose\_image.resize((512, 512))

result = pipeline(prompt=full\_prompt, image=resized, num\_inference\_steps=10)

if not result or not hasattr(result, "images") or len(result.images) == 0:

return None, "Generation failed."

output = result.images[0]

return output, "Outfit generated!"

except Exception as e:

print("Generation error:", e)

return None, f" {e}"

# Save Output

def save\_outfit(image):

if image is None:

return None

filename=os.path.join(save\_folder,f"outfit\_{datetime.datetime.now().strftime('%Y%m%d\_%H%M%S')}.png")

image.save(filename)

return filename

# Gradio UI

def ui():

with gr.Blocks() as demo:

gr.Markdown("AI Fashion Outfit Generator + Virtual Try-On")

with gr.Row():

with gr.Column():

input\_image = gr.Image(label="Live Webcam or Upload", type="pil")

gender = gr.Dropdown(genders, label="Gender", value="female")

prompt = gr.Textbox(label="Custom Style Prompt", placeholder="e.g. red leather jacket")

generate\_btn = gr.Button("Generate Outfit")

with gr.Column():

output\_image = gr.Image(label="Result")

status = gr.Textbox(label="Status", interactive=False)

save\_btn = gr.Button("Save Outfit")

saved\_path = gr.Textbox(label="Saved Path", interactive=False)

generate\_btn.click(fn=generate\_outfit, inputs=[input\_image, gender, prompt], outputs=[output\_image, status])

save\_btn.click(fn=save\_outfit, inputs=[output\_image], outputs=saved\_path)

demo.launch(share=True)

ui()