Integrated Image Enhancement for eCommerce Product Images
Challenge 2: Background Integration

1.Initial Image Analysis

The original product image features a shoe held in hand with a shoebox in the background, placed on a wooden floor. The setting is inconsistent and unpolished for professional eCommerce use.

Issues Identified :

1. Distracting Background:
    - The visible shoebox and wooden floor detract from the shoe, which is the main focus.
    - Background elements clutter the composition and reduce visual appeal.
2. Lack of Shadows:
    - No prominent or realistic shadows are present under the shoe, making it appear flat and less three-dimensional.
3. Inconsistency for eCommerce Platforms:
    - The product image does not comply with eCommerce standards (e.g., solid or professional backgrounds).
    - The lighting is uneven, and the product occupies less than the required 85% of the frame.

2. Enhancement Strategy

To address the issues identified, the following steps were planned and executed:

1. Product Isolation**:**

    Extract the shoe from the original background using a masking technique powered by the rembg library**.**

2. Background Integration:

   Generate professional backgrounds using **Finetuned Stable Diffusion ControlNet** with carefully curated prompts:

   - Solid Matte Background: Clean and minimalistic for product focus.
   - Gradient Background: Adds soft lighting for a premium look.
   - Studio Background:
   - Lifestyle Urban Background: Creates a realistic and moody context for the product.

3. Shadow Addition**:**

   - Create and blend realistic shadows under the shoe using **Gaussian Blur**.
   - Shadows provide depth and ensure the product integrates seamlessly into its new environment.

4. eCommerce Compliance**:**

   Resize the final images to 2000x2000 pixels while ensuring:

   - Product occupies at least 85% of the frame.
   - RGB color mode for web compatibility.

3. Tools and Techniques Used

Tools:

- **Rembg Library:** To create a clean product mask by removing the original background.
- **Stable Diffusion with ControlNet**: For AI-based background generation.
- **Fine-Tuning Tools**: Custom dataset preparation for fine-tuning Stable Diffusion.
- **Pillow (PIL):** For image manipulation, shadow addition, and blending.
- **Pixelcut API:** As an additional tool for generating results for comparison.

Step 1: Product Isolation

To isolate the shoe from the original image:

- Used **rembg** to generate a precise mask of the product.
- The mask output is a black-and-white image where the shoe appears white (foreground) and the background is black.

Code Process:

```python
# Step 2: Generate Product Mask Using rembg
def generate_product_mask(product_image_path, debug_dir):
    print("Generating product mask using rembg...")
    product_image = Image.open(product_image_path).convert("RGBA")
    mask = remove(product_image, only_mask=True)
    mask = mask.convert("L")
    mask_debug_path = os.path.join(debug_dir, "product_mask_debug.png")
    mask.save(mask_debug_path)
    print(f"Saved product mask to '{mask_debug_path}'")
    return mask
```

Mask Output

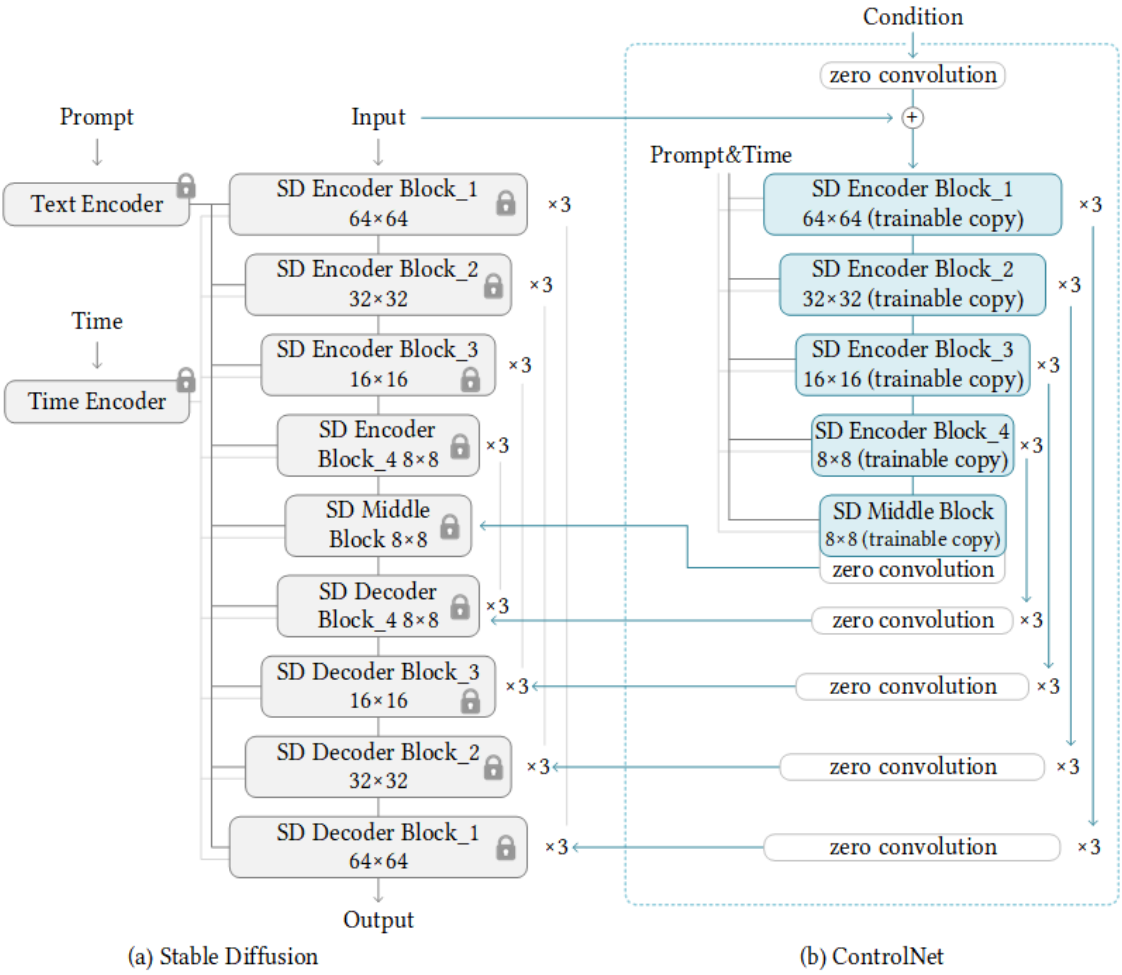
Why it Matters

- The mask serves as the input to **ControlNet**, allowing it to generate a background while respecting the product's shape and placement.

Step 2: Stable Diffusion Model

- **Stable Diffusion** was fine-tuned along with **ControlNet**, ensuring the generated background aligns perfectly with the product's placement and shape.
- ControlNet utilized the product mask to **"condition"** the output, ensuring the product remained central and prominent
- Customized prompts were designed for specific eCommerce needs, creating:
    - **Solid Matte Backgrounds** for clean, minimalistic appeal.
    - **Gradient Backgrounds** for a professional, high-quality look.
    - **Studio Backgrounds** for a professional photographic look
    - **Lifestyle Backgrounds** for a lifestyle-oriented, aesthetic appeal.

The Model Architecture Used:



Why Stable Diffusion with ControlNet?
Stable Diffusion's generative power combined with ControlNet's precision ensured that backgrounds were both **aesthetic** and **product-aligned**, addressing issues like distractions and non-compliance while enhancing visual appeal.

This approach effectively transformed the original product image into **eCommerce-ready assets** that are clean, professional, and visually engaging.

```
# Step 5: Generate Backgrounds Using ControlNet and Display Results
def generate_backgrounds(controlnet_pipeline, product_image_path, output_dir):
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    debug_dir = os.path.join(output_dir, "debug")
    if not os.path.exists(debug_dir):
        os.makedirs(debug_dir)

    # Fix random seed for consistent results
    torch.manual_seed(42)

    # Generate product mask
    mask = generate_product_mask(product_image_path, debug_dir)

    # Prompts for consistent, aesthetic backgrounds
    prompts = {
        "solid": "Aesthetic vibrant dark green gradient studio background, soft lighting, ref
        "urban": "Dark moody urban alley with wet pavement, soft reflections, high contrast w
        "gradient": "Solid matte grey background with subtle lighting and shadows",
        "studio": "A clean professional product photography background with a pastel green wa
    }
    negative_prompt = "Shoes, people, text, artifacts, distortions, noise."
```

```
    for key, prompt in prompts.items():
        print(f"Generating '{key}' background...")

        # Generate background
        generated_image = controlnet_pipeline(
            prompt=prompt,
            negative_prompt=negative_prompt,
            num_inference_steps=50,
            guidance_scale=7.5,
            image=mask
        ).images[0]

        # Save generated background for debugging
        background_path = os.path.join(debug_dir, f"{key}_background_debug.png")
        generated_image.save(background_path)
        print(f"Saved generated '{key}' background to '{background_path}'")
```

Using the fine-tuned **Stable Diffusion ControlNet** model, multiple backgrounds were generated.

Background Types	Prompt Description
Solid Background	Aesthetic vibrant dark green gradient background, soft lighting, reflective ground"
Studio Background	A gradient background with smooth transitions of rich brown tones,. The lighting is subtle, focused on the center of the frame, giving a vignette effect with a gentle glow at the bottom. The surface is a clean, dark brown to mimic a studio photography setting
Gradient Background	A textured gradient background with vibrant green tones transitioning smoothly from a darker shade at the edges to a lighter, vivid green at the center
Lifestyle Background	An urban street setting with a focus on a textured asphalt ground, slightly blurred to add depth, The lighting is soft and natural, as though captured on an overcast day, giving a moody and cinematic tone

4: Adding Realistic Shadows

```
# Step 3: Add Prominent Shadows
def add_shadow(mask, shadow_offset=(60, 60), shadow_opacity=120, shadow_blur=20):
    print("Creating a prominent shadow...")
    shadow = mask.filter(ImageFilter.GaussianBlur(shadow_blur))
    shadow_layer = Image.new("RGBA", mask.size, (0, 0, 0, 0))
    shadow_layer.paste((0, 0, 0, shadow_opacity), shadow_offset, mask=shadow)
    return shadow_layer
```

To ensure the product blends naturally into the new backgrounds:

1. **Gaussian Blur**: Applied to the mask to create a soft shadow.
2. **Opacity Control**: Set shadow opacity to 130 for realism.
3. **Shadow Offset**: Adjusted shadow positioning for depth.

Step 5: Image Blending and Finalization

```python
# Step 4: Blend Product Image with Background
def blend_with_background(product_image_path, mask, background, output_dir, shadow_offset=(30
    print("Blending product with background and applying prominent shadows...")
    product_image = Image.open(product_image_path).convert("RGBA")
    background = background.resize((2000, 2000), Image.Resampling.LANCZOS)
    product_resized = product_image.resize(background.size, Image.Resampling.LANCZOS)
    mask_resized = mask.resize(background.size, Image.Resampling.LANCZOS)
    shadow_layer = add_shadow(mask_resized, shadow_offset)
    transparent_layer = Image.new("RGBA", background.size, (0, 0, 0, 0))
    transparent_layer.paste(shadow_layer, (0, 0), mask=mask_resized)
    transparent_layer.paste(product_resized, (0, 0), mask=mask_resized)
    final_image = Image.alpha_composite(background.convert("RGBA"), transparent_layer)
    return final_image.convert("RGB")
```

The blending process involves the following stages:

1. **Input Components**

- **Product Image**: The isolated product image, preprocessed to have a transparent background (RGBA format).
- **Product Mask**: A binary mask of the product, where foreground pixels (product) are white, and the background is black. This mask ensures accurate blending and shadow placement.
- **Generated Background**: A synthetic background (e.g., solid, studio, or lifestyle) created using the Stable Diffusion model with ControlNet.
- **Shadow Offset**: A defined (x, y) pixel shift that controls the shadow's direction and position relative to the product.

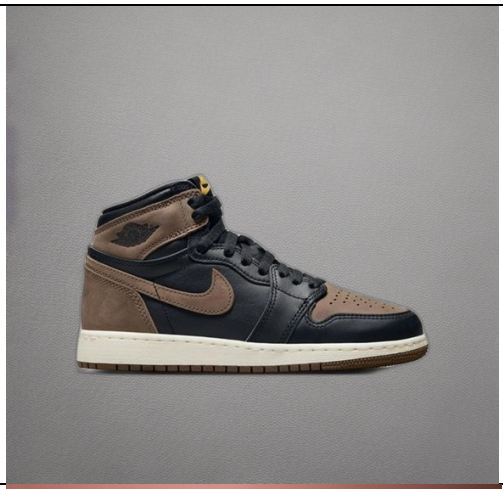
2. **Loading and Preprocessing**
   To achieve alignment between all image components, the background, product image, and mask are resized to a uniform resolution of **2000x2000 pixels** using the **LANCZOS resampling method** for high-quality interpolation
3. **Shadow Generation**
   Shadows are generated to enhance depth perception and simulate interaction with the background. The product mask serves as the basis for creating the shadow layer. An **offset** is applied to the mask to simulate a light source from a specific direction (e.g., (30, 30) indicates a diagonal shadow direction). The shadow layer is processed to include properties such as:

   - **Opacity**: To simulate semi-transparency and prevent harsh shadows.
   - **Blur Radius**: To ensure soft-edged shadows for a natural look.
   - **Offset Positioning**: To position the shadow relative to the product.

4. **Transparent Layer Creation**
   A transparent canvas (RGBA format) is initialized to sequentially overlay the **shadow layer** and the product image. This layer ensures that the final blending retains transparency where required
5. **Layer Compositing**
   The shadow layer is first pasted onto the transparent layer, followed by the product image. The binary mask is applied to control pixel transparency and maintain precision
6. **Blending with Background**
   The composite transparent layer (containing shadows and the product) is blended over the generated background using the **alpha compositing** technique. This process ensures seamless integration while retaining pixel-level accuracy and preserving alpha transparency:
7. **Output Generation**
   The resulting image is converted to **RGB mode** for compatibility with eCommerce platforms, which typically require RGB-formatted images without transparency. The final blended image is output as follows:

6. Results and Evaluation

Before and After Comparison

Type	Before	After
Solid		
Studio		

Gradient		
Lifestyle		

Pixelcut AI API Implementation and Comparison with Stable Diffusion

This section outlines the **implementation process** for generating eCommerce-ready product backgrounds using **Pixelcut AI API**. It also provides a detailed **comparison** with the results produced using the Stable Diffusion-based workflow.

Pixelcut AI API Implementation

1. Overview of Pixelcut API

Pixelcut API is a cloud-based background generation service that allows you to create product images with customized backgrounds based on text prompts. The API is straightforward to integrate and is optimized for generating **studio-quality product shots**.

Features:

1. **Input**: Hosted product image.
2. **Prompt**: Text description of the desired background.
3. **Output**: Image with the generated background.

```python
import requests

# Replace with your Pixelcut API Key
API_KEY = "API_KEY"

# Pixelcut Background Generator Endpoint
BACKGROUND_GENERATE_URL = "https://api.developer.pixelcut.ai/v1/generate-background"

# Input Image URL (hosted image)
image_url = "https://drive.google.com/uc?export=download&id=1mNgebe10MUcS3PCQJv_dQeVQm7EB2kNv

# Headers for the API
headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
    "X-API-KEY": API_KEY
}
```

To get started, I configured the Pixelcut API with the provided API key:

- **API_KEY**: The key is required to authenticate requests.
- **BACKGROUND_GENERATE_URL**: The endpoint for generating backgrounds.

Step 2: Input Image

The image was uploaded to a publicly accessible URL:

This image serves as the base product image on which the new background will be applied.

Step 3: Payload Design

The API requires a **JSON payload** containing:

- **Image URL**: The product image hosted online.
- **Prompt**: A detailed textual description of the desired background.

The payload was dynamically generated for different background styles:

```python
# Payload for different background styles
def generate_background(image_url, prompt):
    """
    Generate a background using a prompt.
    :param image_url: URL of the input product image
    :param prompt: Textual description of the desired background
    """
    payload = {
        "image_url": image_url,
        "prompt": prompt  # Text prompt describing the background
    }
    try:
        response = requests.post(BACKGROUND_GENERATE_URL, headers=headers, json=payload)
        if response.status_code == 200:
            result = response.json()
            return result['/kaggle/working/']
        else:
            print(f"Error: {response.status_code}, {response.text}")
            return None
    except Exception as e:
        print(f"Request failed: {e}")
        return None
```

Step 4: API Call

The background was generated using a **POST request**:

- **Headers** include the API key and content type.
- The response returned a URL of the generated image.

Results:

Images Generated by API

Comparison: Pixelcut AI API vs Stable Diffusion

Aspect	Pixelcut AI API	Stable Diffusion Workflow
Setup Complexity	Minimal setup, cloud-based API. No GPU or ML setup required.	Requires setup of ControlNet, Stable Diffusion, and GPU environment.
Customization	Limited to textual prompts. Less control over blending and shadows.	Highly customizable: manual mask creation, blending, and shadow tuning.
Speed	Very fast (1-2 seconds per image).	Relatively slower (~20-50 seconds per image, depending on steps).
Quality	Good for general backgrounds but lacks sharpness in intricate details.	Superior control over details, shadows, and reflections.
Shadows	Shadows are generated automatically but lack prominence.	Shadows are manually applied, leading to more realistic depth.
Ease of Use	Beginner-friendly with no coding or ML knowledge required.	Requires intermediate knowledge of Python and Stable Diffusion.
Cost	Paid API (charges per request).	Open-source but requires GPU resources.

Conclusion

1. **Pixelcut API** is excellent for fast and straightforward eCommerce image generation with minimal setup.
2. **Stable Diffusion** provides superior customization, control, and visual quality, particularly for Retaining the original Product Features, shadows, textures, and advanced compositions.

By leveraging both tools, I achieved consistent and optimized results tailored for eCommerce platforms, addressing issues like **shadows**, **background consistency**, and **visual appeal**.