Open in app ↗

Medium Search

Latent Space 操作

Rich
7 min read · Just now

Share ••• More

stable diffusion model

版本一

為了找到需要的latent space vector 我打算比較2張圖片在latent space的差異，並假設此差異為需要的vector，因此先把2張圖丟進laten
面

```
def encode_image_to_latent(image_path, vae_encoder, vae_scale_factor, device):
    image = Image.open(image_path).convert("RGB").resize((width, height))
    image_np = np.array(image).astype(np.float32) / 255.0
    image_tensor = torch.from_numpy(image_np).permute(2, 0, 1).unsqueeze(0).to(device)
    image_tensor = (image_tensor - 0.5) * 2  # Normalize to [-1, 1]

    with torch.no_grad():
        latent_dist = vae_encoder.encode(image_tensor.half())
        latent_sample = latent_dist.latent_dist.sample() * vae_encoder.config.scaling_factor
    return latent_sample

z1 = encode_image_to_latent(image_path_1, vae, vae_scale_factor, device)
z2 = encode_image_to_latent(image_path_2, vae, vae_scale_factor, device)
```

之後相減得到之間的差(z_diff)

```
z_diff = z2 - z1
```

再降維觀察亮暗的變化

```
z_diff_magnitude = torch.abs(z_diff)
diff_magnitude_img = z_diff_magnitude.mean(dim=1, keepdim=True)
diff_min = diff_magnitude_img.min()
diff_max = diff_magnitude_img.max()
if diff_max > diff_min:
    diff_magnitude_img = (diff_magnitude_img - diff_min) / (diff_max - diff_min)
else:
    diff_magnitude_img = torch.zeros_like(diff_magnitude_img)

diff_img_pil = Image.fromarray((diff_magnitude_img[0, 0].cpu().numpy() * 255).astype(np.uint8))
diff_img_pil.save("latent_difference_magnitude.png")
print("潛在差異圖已保存為 latent_difference_magnitude.png")
```

接著嘗試把z_diff 轉回成圖片

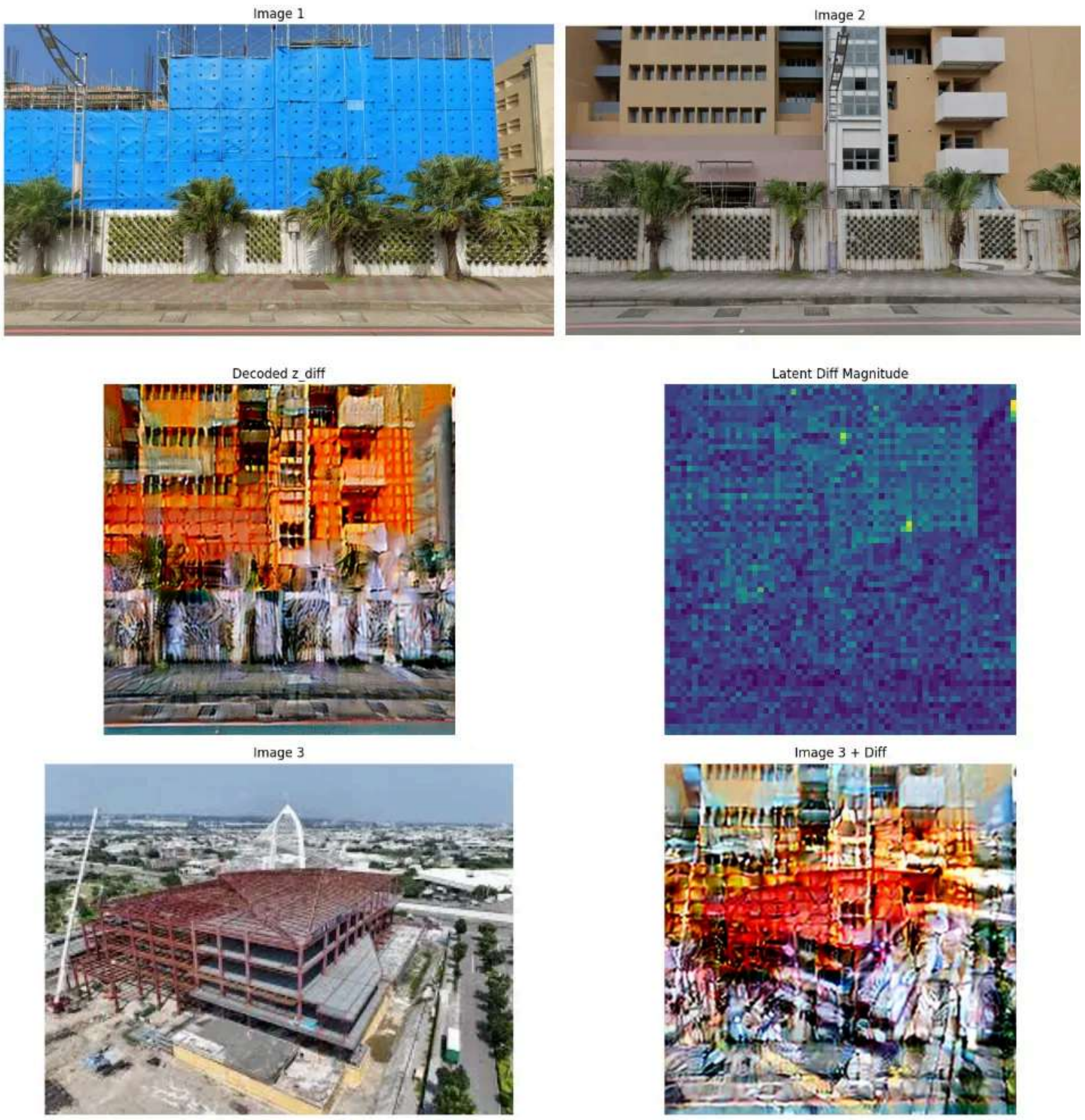
```
with torch.no_grad():
    z_diff_decoded = vae.decode(z_diff / vae.config.scaling_factor).sample
    z_diff_decoded = (z_diff_decoded / 2 + 0.5).clamp(0, 1)
    diff_img = z_diff_decoded[0].permute(1, 2, 0).cpu().numpy()
    Image.fromarray((diff_img * 255).astype(np.uint8)).save("decoded_diff.png")
    print("z_diff 還原圖片已保存為 decoded_diff.png")
```

之後就是再拿一張圖片在latent space 加上此z_diff看效果會是如何

```
z_new = encode_image_to_latent(image_path_3, vae, vae_scale_factor, device)

# 將差異加到新圖片的潛在向量
with torch.no_grad():
    z_new_plus_diff = z_new + z_diff
    z_new_plus_decoded = vae.decode(z_new_plus_diff / vae.config.scaling_factor).sample
    z_new_plus_decoded = (z_new_plus_decoded / 2 + 0.5).clamp(0, 1)
    new_img_plus_diff = z_new_plus_decoded[0].permute(1, 2, 0).cpu().numpy()

    # 儲存並顯示結果
    Image.fromarray((new_img_plus_diff * 255).astype(np.uint8)).save("new_image_plus_diff.png")
    print("新圖片加入差異向量後的圖片已保存為 new_image_plus_diff.png")
```



版本二

先透過prompt 用 stable diffusion model 生出兩張圖片在latent space

```
prompt_unfinished = "unfinished building under construction, rough structure, incomplete walls"
prompt_finished = "completed building with modern design, polished walls, windows, fully constructed"
```

之後一樣產生z_diff

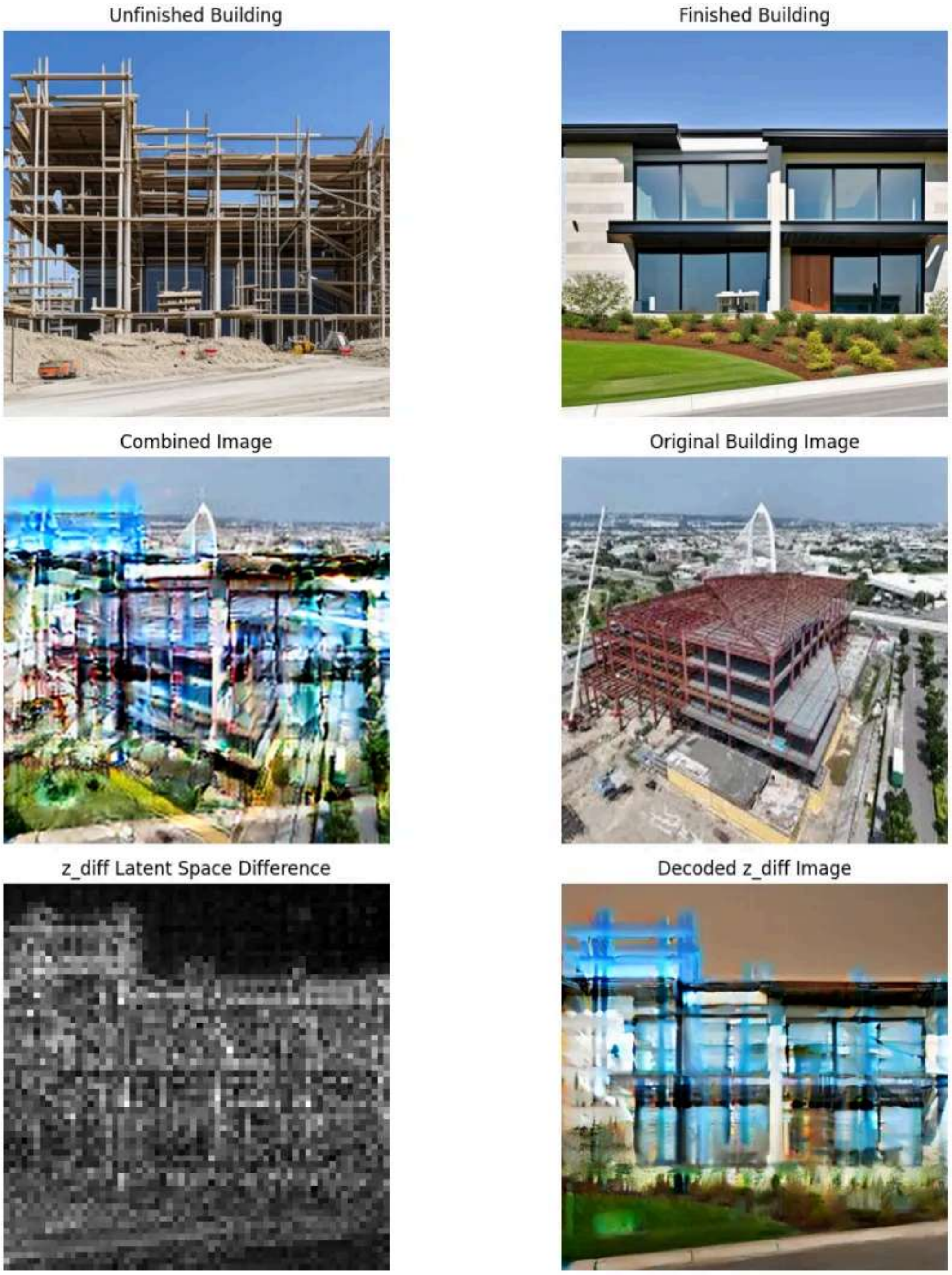
```
z_unfinished = run_diffusion(prompt_unfinished, negative_prompt, guidance_scale_prompted, latents_initial, num_inference_steps
z_finished = run_diffusion(prompt_finished, negative_prompt, guidance_scale_prompted, latents_initial, num_inference_steps, pi
```

同樣的拿一張新圖加上z_diff

```
z_difference = z_finished - z_unfinished
z_combined = latent_existing + z_difference
```

最後透過decoder轉回去原圖

```
image_unfinished = postprocess(decode_latent(z_unfinished, vae, vae.config.scaling_factor))
image_finished = postprocess(decode_latent(z_finished, vae, vae.config.scaling_factor))
combined_image = postprocess(decode_latent(z_combined, vae, vae.config.scaling_factor))
decoded_z_diff_image = postprocess(decode_latent(z_difference, vae, vae.config.scaling_factor))
```



R

Written by Rich

2 Followers · 6 Following

No responses yet

R Rich