TP-2 Modèles de Diffusion
Expérimentation avec Hugging Face Diffusers

December 12, 2024

Introduction

Les modèles de diffusion sont devenus une avancée majeure dans le domaine de la génération d'images à partir de descriptions textuelles. Ces techniques permettent de créer des visuels d'une qualité remarquable, ouvrant de nouvelles perspectives dans la création de contenu. Le framework **Hugging Face Diffusers** offre une interface conviviale pour exploiter ces modèles avec des pipelines adaptés à divers besoins, comme l'inpainting ou la transformation d'image. Ces travaux pratiques illustrent les puissantes capacités des modèles de diffusion pour la création d'images. En explorant divers pipelines, en ajustant les paramètres et en apprenant à optimiser les performances, vous disposez des bases pour intégrer ces techniques dans vos projets de génération visuelle. Pour aller plus loin, explorez des approches comme le fine-tuning ou l'intégration à des applications interactives.

Dans ces travaux pratiques, vous apprendrez à :

- Installer et configurer les outils nécessaires à l'utilisation de Diffusers.
- Générer des images à partir de prompts textuels et comprendre les paramètres influençant les résultats.
- Explorer des techniques avancées comme l'inpainting et la transformation Img2Img.
- Optimiser les performances pour réduire les temps d'inférence.
- Intégrer des données personnalisées pour des applications spécifiques.

Chaque section est accompagnée d'explications détaillées et d'exercices pratiques pour renforcer vos compétences.

1 Exercice 1 : Installation et configuration

Objectif

Installer les bibliothèques nécessaires et préparer l'environnement pour utiliser Stable Diffusion.

Instructions

Exécutez les commandes suivantes pour installer les bibliothèques et importer les modules requis :

```
!pip install diffusers transformers torch accelerate

import torch
from diffusers import StableDiffusionPipeline
```

Remarque : Assurez-vous d'utiliser un environnement avec un GPU disponible (par exemple, Google Colab avec une instance GPU). Si aucun GPU n'est disponible, la génération d'images sera extrêmement lente.

Explication

Stable Diffusion est un modèle de génération d'images basé sur des techniques de diffusion, qui modélise le processus inverse de diffusion pour transformer du bruit en une image cohérente à partir d'un prompt textuel.

2 Exercice 2 : Génération d'image simple

Objectif

Utiliser un pipeline Stable Diffusion pour générer une image à partir d'un *prompt* textuel.

Instructions

1. Chargez le pipeline pré-entrainé `"runwayml/stable-diffusion-v1-5"`.
2. Configurez le pipeline pour utiliser un GPU.
3. Générez une image à partir d'un *prompt* textuel.

```
pipe = StableDiffusionPipeline.from_pretrained("runwayml/
    stable-diffusion-v1-5", torch_dtype=torch.float16)
pipe = pipe.to("cuda")

prompt = "un chat jouant du piano dans un studio d'
    enregistrement"
image = pipe(prompt).images[0]
image.save("chat_pianiste.png")
```

Explication : Le prompt fournit une description textuelle de l'image que vous souhaitez générer. Le modèle utilise cette description pour produire un contenu visuel.

Tâche supplémentaire : Essayez de changer le prompt pour explorer d'autres scènes créatives. Par exemple, `"un dragon volant au-dessus d'une ville futuriste"`.

3 Exercice 3 : Exploration des paramètres

Objectif

Comprendre comment ajuster les paramètres du pipeline pour influencer les résultats.

Explications

- **num_inference_steps** : Nombre d'itérations pour générer l'image. Un nombre élevé donne une meilleure qualité, mais augmente le temps d'inférence.
- **guidance_scale** : Contrôle la balance entre la fidélité au prompt et la créativité. Des valeurs élevées (typiquement 7.5-10) favorisent une meilleure correspondance au prompt.
- **negative_prompt** : Permet d'indiquer des éléments à éviter dans l'image générée.

Instructions

Générez une image avec des paramètres modifiés :

```
result = pipe(
    prompt="un paysage de montagne au coucher du soleil",
    num_inference_steps=50,
    guidance_scale=7.5,
    negative_prompt="nuages, brouillard"
)
result.images[0].save("paysage_montagne.png")
```

Tâche supplémentaire : Expérimentez avec différentes valeurs de `guidance_scale` et `num_inference_steps` pour observer leurs impacts sur le résultat.

4 Exercice 4 : Inpainting

Objectif

Appliquer la fonctionnalité d'inpainting pour modifier une partie d'une image.

Explications

L'inpainting utilise une image originale et un masque indiquant les zones à modifier. Le modèle génère des contenus conformes au *prompt* dans les zones masquées. Cela peut être utilisé pour supprimer ou remplacer des objets.

Instructions

```
from diffusers import StableDiffusionInpaintPipeline
from PIL import Image

pipe_inpaint = StableDiffusionInpaintPipeline.
    from_pretrained("runwayml/stable-diffusion-inpainting")

image = Image.open("image_originale.png")
mask_image = Image.open("masque.png")

prompt = "un bouquet de fleurs sur une table"
result = pipe_inpaint(prompt=prompt, image=image, mask_image
    =mask_image).images[0]
result.save("image_modifiee.png")
```

Exercice : Créez un masque différent pour modifier un autre objet dans l'image. Par exemple, remplacez une chaise par une lampe futuriste.

5 Exercice 5 : Génération conditionnelle (Img2Img)

Objectif

Transformer une image existante en s'appuyant sur un *prompt* textuel.

Explications

La méthode Img2Img applique des modifications à une image de référence, en combinant les informations visuelles de l'image initiale et le contenu du prompt. Le paramètre `strength` détermine dans quelle mesure l'image initiale est modifiée.

Instructions

```
from diffusers import StableDiffusionImg2ImgPipeline

pipe_img2img = StableDiffusionImg2ImgPipeline.
    from_pretrained("runwayml/stable-diffusion-v1-5",
    torch_dtype=torch.float16)
pipe_img2img = pipe_img2img.to("cuda")

init_image = Image.open("image_initiale.png")

prompt = "un paysage futuriste avec des b\^atiments volants"
images = pipe_img2img(prompt=prompt, image=init_image,
    strength=0.75, guidance_scale=7.5).images
images[0].save("paysage_futuriste.png")
```

Tâche supplémentaire : Modifiez `strength` (valeurs de 0.1 à 1.0) pour contrôler l'impact de l'image initiale.

6 Exercice 6 : Optimisation des performances

Objectif

Utiliser des techniques d'optimisation pour réduire les temps d'inférence.

Explications

Les pipelines peuvent être optimisés en remplaçant leur scheduler par des algorithmes plus efficaces, comme `DPMSolverMultistepScheduler`. Cela permet de conserver une qualité correcte tout en accélérant le processus de génération.

Instructions

```
from diffusers import DPMSolverMultistepScheduler

pipe.scheduler = DPMSolverMultistepScheduler.from_config(
    pipe.scheduler.config)

prompt = "un robot dansant dans une discoth\‘eque"
image = pipe(prompt, num_inference_steps=20).images[0]
image.save("robot_dansant.png")
```

Exercice : Comparez les temps d'exécution avant et après optimisation en utilisant un chronomètre Python (module `time`).