TRAINING AND TESTING OF GLIDE BY OPENAI

PEGHEADS.ORG
GLOBAL DIGITAL AI EDUCATIONAL ORGANIZATION

What is GLIDE?

GLIDE (Guided Language-to-Image Diffusion for Generation and Editing). This diffusion model achieves performance comparable to DALL-E despite utilizing only one-third of the parameters.

The classifier is first trained on noised images, and during the diffusion sampling process, gradients from the classifier are used to guide the sample towards the label.

GLIDE needs more minor sampling delay and does not require CLIP reordering.

In our previous tutorial, we trained the DALLE model to do the same and it did provide us good results.

1. Create the data folder for testing and training the glide
2. Train the GLIDE model with our dataset
3. Run the test script to check if model is generating images
4. Generate the images with our trained and tested model
5. Change the parameters in the model to generate better results

GOAL

Train and test the GLIDE model with our BotPeg dataset to create images using the text prompt

STEP 1

Create the data folder for testing and training

We have used a total of 92 images for our experiment, each image is a .jpg image with dimensions of 256 x 256. And along with these images we have also provided 92 text files explaining the image respectively.

The naming given to these files is very important for the code to work properly.

If the image name is "Singing_0000000.jpg", then the text file name should be "Singing_0000000.txt".

All these image and text files should be stored in a folder called data.

STEP 2

Train the GLIDE model with our dataset

Clone the repository from our Github > Install the required libraries and Login to WANDB > INPUT: Data Folder > Train the Model: Run the train_glide.py > OUTPUT: Set of .pt files

Clone the repository from our Github

Cloning is a process of creating an identical copy of a Git Remote Repository to the local machine.

You can run the following command to clone our code .

```
git clone  https://github.com/PegHeads-Inc/PegHeads-Tutorial-5.git
```

Install the required libraries

Install the following libraries.

All the libraries are listed in the requirements.txt file. You just need to run this file using the following command and the libraries will be automatically installed for you hassle-free.

```
python -m pip install -r requirements.txt
```

Login to WANDB

Steps for registering are as follows.

- Go to **wandb.ai/site** and click sign up. now you can sign up with Google or GitHub or email and password.
- After signing up, you will have to fill out some information about yourself.
- Once you finish you will be taken to the home page and can copy the wandb API key. Don't share it with anyone but you can reset it.
- Feel free to take look at docs: **https://docs.wandb.ai/quickstart**
- Run the following command to install and login to wandb in colab

```
import wandb
!wandb login
```

INPUT

You can use our BotPeg dataset for trying out the code, the following link can be used to download our dataset.

https://drive.google.com/file/d/1svCu920Yb2adiO6XPQX3ipl-eBzZ7bYC/view?usp=sharing

unzip the folder and add it to the glide folder downloaded from our GitHub or you could create your own data folder as explained in step1.

Train the Model

To train the model we need to run `train_glide.py`, and use the following command to do so.

```
python train_glide.py --data_dir "./data" --use_captions --epochs 20
--project_name "glide-finetune" --batch_size 4 --learning_rate 1e-04
--side_x 64 --side_y 64 --resize_ratio 1.0 --uncond_p 0.2
--checkpoints_dir "./checkpoints"
```

as you can see in this command we are running the `train_glide.py` with the following arguments

Argument	Used for
`--data_dir`	gives the location of the data folder
`--use_captions`	whether to use captions or not
`--epochs`	gives the number of epochs for running the code, we can start with 20
`--project_name`	gives the project name
`--batch_size`	specifies the number images to be generated
`--learning_rate`	specifies the learning rate of the model
`--side_x`	size of the image(width)
`--side_y`	size of the image(height)
`--resize_ratio`	specifies the image resize value
`--uncond_p`	The base model should be tuned for "classifier free guidance". This means you want to randomly replace captions with an unconditional (empty) token about 20% of the time. This is controlled by the argument --uncond_p, which is set to 0.2 by default
`--checkpoints_dir`	name of the directory to store the .pt files

OUTPUT

The code will now generate a list of .pt files and save it in a folder called checkpoints. Please note that each .pt file will be in gigabytes and might take up a lot of space in your memory, so if you're running for more than 20 epochs make sure to delete the first few .pt files from the checkpoints folder, you can use the last .pt file generated for the testing process.

STEP 3

Run the test script to check if model is generating images

Import the libraries > Insert the sample parameters > Run the code

Run the glide_testing.ipynb file for testing, its divided into the following:

Import the libraries

The first block has the code for importing all the libraries like `PIL,IPython,torch, glide_text2im.`

Insert the sample parameters

Parameter	Value
`prompt`	{type: "string"} **The text prompt used to generate the image for example. "BotPeg Singing"**
`batch_size`	{type:"number"} **Number of images to be generated**
`base_timestep_respacing`	{type:"string"} example 40 **use 40 diffusion steps for fast sampling**
`sr_timestep_respacing`	'fast27' **use 27 diffusion steps for very fast sampling**
`glide_path`	**Path of the .pt file for example** './checkpoints/glide-ft-87x799.pt'

Run The Code

The rest of the blocks have the code to train the model and generate the images as per the text prompt provided.

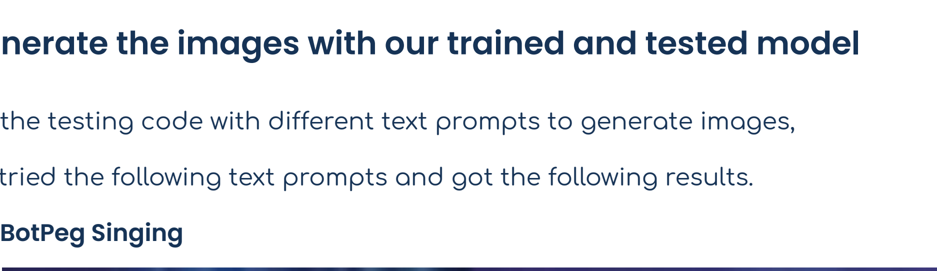
STEP 4

Generate the images with our trained and tested model

Try the testing code with different text prompts to generate images,

We tried the following text prompts and got the following results.

1. **BotPeg Singing**



STEP 5

Change the parameters in the model to generate better results

You can change the batch size, and epochs parameters for testing.

if you try out the code for the same text prompts for different number of epochs you will see the difference in the generated images, we found that the more number of epochs you run it with the better images it produces.

We tried the prompt "BotPeg Singing" with 200 epochs, when compared to the image shown above which was the result of 20 epochs the images shown below are better.

PLEASE STAY TUNED FOR OUR UPCOMING TUTORIAL ON DALL-E2

For more information check out:

https://github.com/afiaka87/glide-finetune

PEGHEADS.ORG
GLOBAL DIGITAL AI EDUCATIONAL ORGANIZATION