MANDY HATHAWAY

Build an Image Generation Model

Instructor Guide · AI and Machine Learning Curriculum

Objectives	Resources
• Understand how AI can create images • Understand how to use a GPU in Google Colab • Know about the StableDiffusers library • Use PyTorch to build an AI model to process input requests	• Completed notebook: Simple Image Generation Model.ipynb • Google Colab: colab.research.google.com • GitHub: github.com/mandyhathaway/ImageGeneratorforStudents

Description

In this project, students build their own simple image generation model using PyTorch, StableDiffusers, and Matplotlib. Working in Google Colab with GPU acceleration, they select an open-source Stable Diffusion model, write prompts, and generate original images. By the end, students have a working model they can keep and continue experimenting with.

Overview

- Phase 1: Connect to a GPU
- Phase 2: Build the Model
- Phase 3: Experiment with Your Model

Instructor Note: Show students the completed version of the project before getting started. This project works best after students have already experimented with tools like Adobe Firefly or Runway, once they've generated images in another tool, they're motivated to build their own model. Remind them that this model will be theirs to keep and use for free, whenever they want.

Phase 1: Connect to a GPU

Goals and Learning Objectives

- Understand how to connect to a GPU in Google Colab
- Understand why GPU acceleration is used for image generation

Discussion

For this project, we are processing image data and generating images. Because our models live in the cloud, we can use Colab's free GPUs (graphical processing units) to give the model the processing power it needs.

Why a GPU? A CPU (central processing unit) is the standard processor in most computers, it's designed to handle a wide variety of tasks quickly, but largely one at a time. A GPU was originally designed to render graphics for video games, which requires doing millions of small math operations *simultaneously*, It turns out that running AI models requires exactly the same kind of parallel math: multiplying enormous matrices of numbers all at once. This makes a GPU dramatically faster than a CPU for AI workloads. A task that might take 30+ minutes on a CPU can often complete in under a minute on a GPU. Google Colab provides free access to a cloud-based GPU, so students can run powerful models without needing specialized hardware.

Walkthrough

From the **Runtime** menu on the top toolbar, select **"Change runtime type."** In the dialog that opens, select an option ending in **GPU** (e.g. T4 GPU), then click **Save**. The environment will reconnect and you are now using a remote graphics processing unit.

Phase 2: Build the Model

Goals and Learning Objectives

- Select an image generation model
- Use PyTorch to create a machine learning pipeline
- Connect the model to the GPU for faster processing

Walkthrough

Step 1: Import libraries

Import the three libraries used in this project: PyTorch (to build and run the model), StableDiffusers (for AI image generation), and Matplotlib (to display the image).

```
import torch
from diffusers import StableDiffusionPipeline, DPMSolverMultistepScheduler
import matplotlib.pyplot as plt
```

Step 2: Clear the cache

Clear the GPU cache to ensure there is enough memory to load the model.

```
torch.cuda.empty_cache()
```

Step 3: Select the model

Choose Stable Diffusion 2.1, an open-source generative model from Stability AI.

```
model_id = "stabilityai/stable-diffusion-2-1"
```

Step 4: Load and configure the pipeline

Load the model and connect it to the GPU ("cuda") for faster processing.

```
pipe = StableDiffusionPipeline.from_pretrained(model_id, torch_dtype=torch.float16)
pipe.scheduler = DPMSolverMultistepScheduler.from_config(pipe.scheduler.config)
pipe = pipe.to("cuda")
```

Step 5: Set your prompt

Define the text prompt the model will use to generate an image.

```
prompt = "a woman walking in the woods"
```

Step 6: Generate the image

Pass the prompt to the pipeline and retrieve the generated image. Width and height can be adjusted.

```
image = pipe(prompt, width=1000, height=1000).images[0]
```

Step 7: Display the image

Use Matplotlib to render the image in the notebook.

```
plt.imshow(image)
plt.axis('off') # Hides axis labels
plt.show()
```

Phase 3: Experiment with Your Model

Goals and Learning Objectives

- Experiment with a self-built image generation model
- Try changing the output image size
- Save images directly from the notebook

Walkthrough

Students now have a working image generation model. Encourage them to experiment freely with different prompts. To save an image they like, right-click the generated image and select "Save Image As." To change the output size, adjust the **width** and **height** values in the pipeline call:

```
image = pipe(prompt, width=1000, height=1000).images[0]
```

Discussion and Reflection

- How is your model different from popular tools like Midjourney or DALL·E?
- Does it perform better with certain types of prompts?
- Can you influence the style of the output through your prompt?

- Share your favorite generated image with the class.

OPTIONAL EXTENSION

Building a Web Interface with Gradio

Once the basic image generation model is working, students can take it further by wrapping it in a simple web interface using Gradio. This transforms the notebook from a script into an interactive app with a text input field, a generate button, and an image display area, all running inside Colab.

What is Gradio?

Gradio is a Python library that lets you build simple web interfaces for machine learning models with just a few lines of code. Rather than building a full web application, you define an input type (in this case, text), an output type (an image), and the function that connects them. Gradio handles the rest and generates a shareable link so others can try the model too.

Additional Import

Add Gradio to the imports at the top of the notebook:

```
import gradio as gr
```

Wrapping the Model in a Function

Instead of running the pipeline directly, wrap the image generation call in a named function that Gradio can call each time a user submits a prompt. See below:

```
def generate_image(prompt):
    image = pipe(prompt, width=1000, height=1000).images[0]
    return image
```

Launching the Interface

Pass the function to Gradio and launch the interface. The title and description appear at the top of the app and can be customized by each student:

```
gr.Interface(
    fn=generate_image,
    inputs="text",
    outputs="image",
    title="My Personal Stable Diffusion Generator",
    description="Enter a prompt to bring your imagination to life!"
).launch()
```

Discussion

- What is the difference between running code in a notebook cell and using an interface like this?
- Who could you share this app with using the Gradio link? What would they need to use it?

- What other inputs could you add to give users more control (style, size, number of images)?
- How does wrapping a model in an interface change who can access and use it?

Full notebook available at: github.com/mandyhathaway/ImageGeneratorforStudents