Assignment 3: Generative AI

By Talha Farhat and Abbas Bukhari

December 13, 2024

Question 1

Why did you choose the Playground model?

We chose the Playground model (based on Stable Diffusion XL) because, according to the Hugging Face model card, it was one of the best models available among those provided by the instructor. Additionally, other models explicitly mentioned their limitations in generating realistic fingers, which was a critical requirement for our task.

What did we expect from the model outputs?

We expected the outputs to be suboptimal, as generating realistic and anatomically accurate fingers is challenging even for state-of-the-art models.

What did we learn from the model outputs?

From the outputs, we learned that current image generation models still face significant challenges in creating realistic fingers. This difficulty highlights the need for models to better understand human anatomy and physical principles.

What are some ideas to improve the outputs?

To enhance the outputs, training the model on an extensive dataset focused on human hands and incorporating a deeper understanding of general physics could help reduce unrealistic or distorted images.

Question 2

What are the different strategies of fine-tuning that we explored?

1. Initially, we attempted to fine-tune the model directly on Google Colab, but resource constraints made this unfeasible.

2. We then explored LoRA (Low-Rank Adaptation) fine-tuning with various permutations of hyperparameters such as resolution, batch input size, and mixed precision. However, time constraints required us to significantly reduce the dataset size.

What issues did we face?

1. **No Fine-tuning Guide for Playground Model:** Since there was no specific fine-tuning guide for the Playground model, we had to adapt guides for Stable Diffusion XL, which required additional adjustments.

2. **Large Model Parameters:** The model parameters were too large to load on the available resources in Colab or Kaggle. To address this, we used LoRA and experimented with different hyperparameter combinations.

3. **Dataset Size and Time Constraints:** The provided dataset had 11,000 images. Training on the full dataset required 23 hours, but online notebook environments only allowed 12-hour connections. As a result, we reduced the dataset size to 1,000 images.

4. **Training Instability:** After fine-tuning for 50,000 steps, the model initially produced pure noise. We discovered that the learning rate was too high and had to lower it significantly to achieve any sensible results.

How did we overcome these issues?

- For the lack of a Playground-specific guide, we adapted instructions for Stable Diffusion XL, tweaking steps to suit our model.

- To manage resource limitations, we opted for LoRA fine-tuning and conducted hyperparameter testing.

- The dataset was reduced to 1,000 images, balancing time and resource constraints.

- By iterating on the learning rate and other hyperparameters, we addressed the issue of noisy outputs and improved model stability.

What are some takeaways from the fine-tuning?

The fine-tuning process highlighted that:

- Using small datasets and limited resources leads to marginal improvements and inconsistent results.

- Fine-tuning requires careful hyperparameter selection, particularly for learning rates and batch sizes, to avoid instability.
- A more substantial dataset and longer training durations could significantly enhance the model's performance.

Can we make the model even better?

Yes, with access to greater computational resources and time, the model could be fine-tuned more effectively. Additionally, integrating a better understanding of physics and anatomy into the training data could help address inherent challenges, such as generating realistic fingers.

Question 3

Architecture Diagram

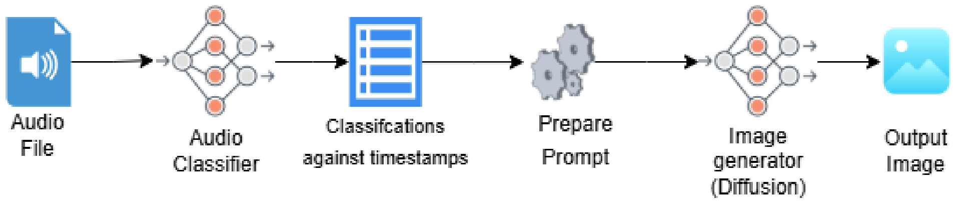
Audio File
Audio Classifier
Classifcations against timestamps
Prepare Prompt
Image generator (Diffusion)
Output Image

Figure 1: A high-level architecture diagram illustrating the workflow from audio input to image output.

Workflow from Audio to Image

1. **Audio Input:** Accept audio files in `.wav` or `.mp3` formats.
2. **Neural Network Classification:** Use MediaPipe's YamNet to classify audio into categories.
3. **Processing Outputs:** Extract top categories from the classification and prepare a text prompt.
4. **Image Generation:** Pass the prompt to the Playground model to generate an image.
5. **Output Image:** Display the generated image.

How it works and what it does

1. Download, install, and import necessary libraries.
2. Convert audio files (if required) from `.mp3` to `.wav` using a utility function.
3. Provide the audio file path to the classifier function, which uses YamNet for classification.
4. YamNet outputs category predictions for each timestamp, e.g.:

```
Timestamp 0: Dog (0.26)
Timestamp 1: Domestic animals, pets (0.20)
Timestamp 1: Domestic animals, pets (0.21)
```

5. Use the `get_top_categories()` function to process the output and create a text prompt.
6. Send the text prompt to the Playground model to generate and retrieve the image.