Weekly Project Report – Week 1
Unified Generative AI Models via Command-Line Interface

Group 6 Section: 05
Araf Tahsan Pavel ID: 2212079042
Md Tanveer Hossain Nihal ID: 2211935042
Ismot Ara Emu ID: 2212108042

***Abstract*—This week, the foundational components of the unified command-line interface were successfully initiated. The team implemented the Bash automation script and developed the core Python handler for CLI operations. Additionally, we finalized the selection of generative models and gathered the required source code into the `CodeP` folder for modular development. This phase focused on building the groundwork and understanding the model integration process in depth.**

I. Introduction

The aim of this project is to design and implement a unified command-line interface that can seamlessly execute various generative AI models—spanning text, speech, and image domains—through a consistent and user-friendly system.

II. This Week's Progress

- Completed the Bash script (`run_exp.sh`) to automate CLI-based execution and testing.
- Developed the core CLI handler in `main.py` to parse commands and route them to model-specific pipelines.
- Identified and finalized the models to be integrated: Transformer, TinyBERT, Coqui TTS, and a diffusion model.
- Integrated `runner.py` in `codeP` directory to manage which specific model and which specific dataset it will work on. It is imported in `main.py` and used its `run_pipeline()` function
- Gained a comprehensive understanding of the selected models' input/output formats and configuration needs.

III. Challenges Faced

- Variations in model architecture and dependencies led to complexities in standardizing their integration.

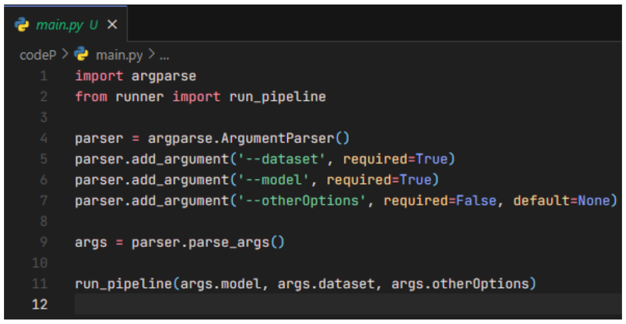
IV. Solutions and Decisions

- Opted to use lighter or quantized versions of models where possible to ensure compatibility with available hardware.

V. Next Week Plan

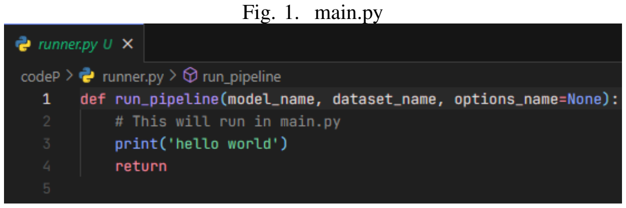
- Begin integration of Coqui TTS and the diffusion model into the CLI framework.
- Implement logging and result-saving features within the unified interface.
- Draft initial usage documentation for new users and contributors.

My Code Screenshot



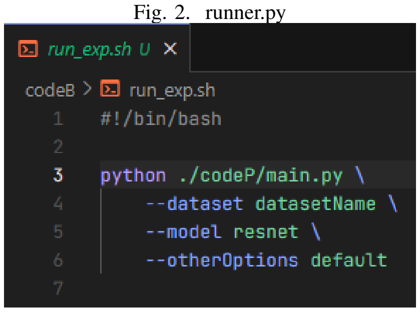
```python
import argparse
from runner import run_pipeline

parser = argparse.ArgumentParser()
parser.add_argument('--dataset', required=True)
parser.add_argument('--model', required=True)
parser.add_argument('--otherOptions', required=False, default=None)

args = parser.parse_args()

run_pipeline(args.model, args.dataset, args.otherOptions)
```

Fig. 1. main.py



```python
def run_pipeline(model_name, dataset_name, options_name=None):
    # This will run in main.py
    print('hello world')
    return
```

Fig. 2. runner.py



```bash
#!/bin/bash

python ./codeP/main.py \
    --dataset datasetName \
    --model resnet \
    --otherOptions default
```

Fig. 3. bash file