**CogView: Mastering Text-to-Image Generation via Transformers**

**Project Team:**

Leela Prasanna

Mahitha

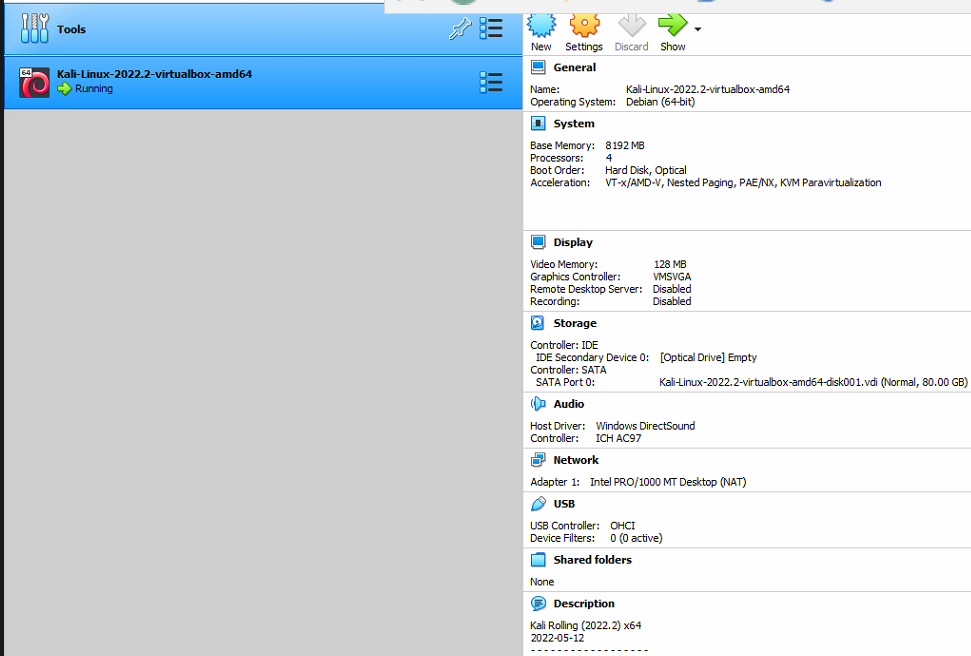
Lija George

George Emmanuel

Prathyush Turaga

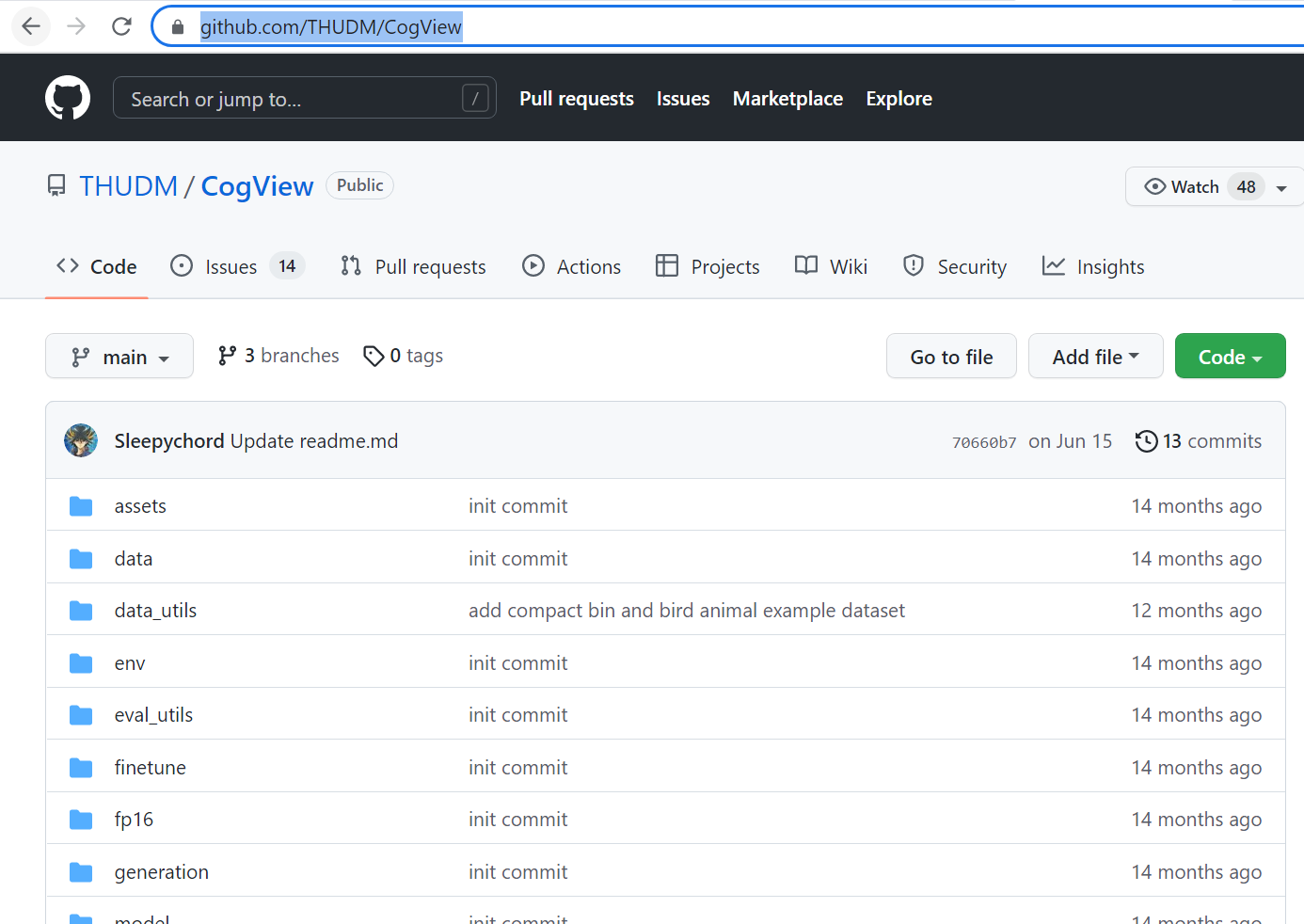
**Implementation Process**

1. As part of this process, we will be implementing CogView project on a VM (Kali Linux).
2. For this purpose, we will take a virtualization software like VMWare or Virtual Box.
3. Upon the virtualization software, we will mount a server-based operating system like any available Linux distribution.
4. We have taken Kali Linux here as the virtual environment. Downloaded image file from internet and created the setup on a personal computer.



1. Now we have downloaded GitHub repository code into Kali Linux downloads folder and extracted the zipped file into a new folder called “CogView-main”

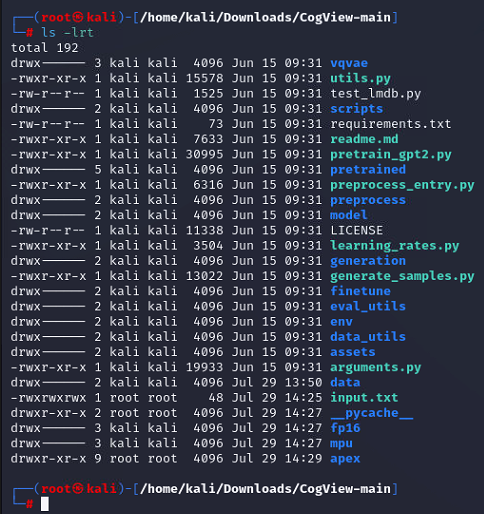
<https://github.com/THUDM/CogView>



Text

Description automatically generated

1. Open terminal, and navigate to the repository files directory “/home/kali/Downlaods/CogView-main”.
2. Now will check if Python is installed and available on the VM. Also, we will need pip as python package manager for installing required libraries.
3. Upon checking the Virtual Machine (VM), we observed that latest Python 3.10 is pre-installed along with pip with version 20+.



1. From the project directory, will install required python libraries with below command.

#pip install -r requirements.txt

1. Subsequently we will run commands as advised in GitHub repository read.me section of existing research work.
2. Before running main scripts, we will download the datasets from link as shown in below script.

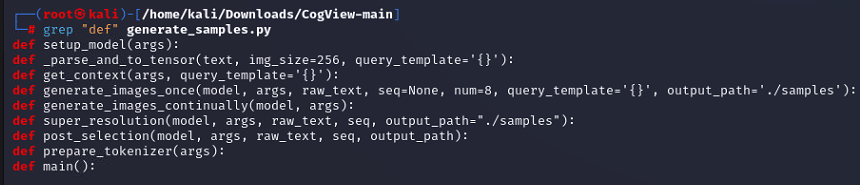
Text

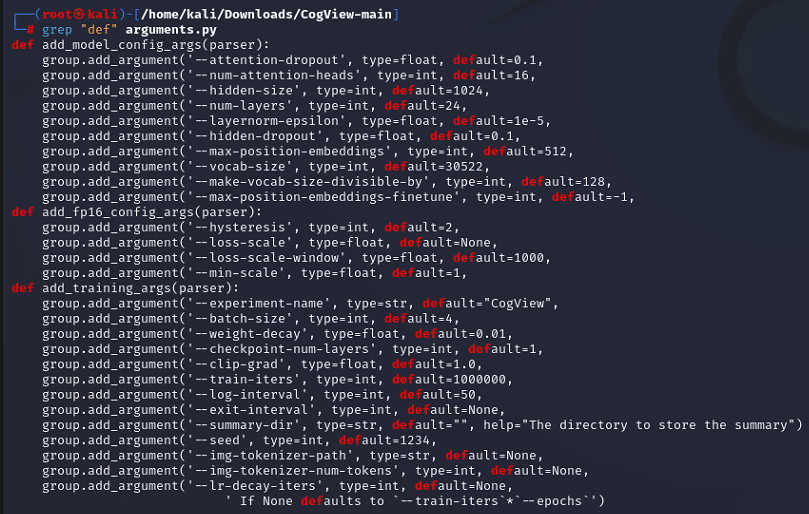
Description automatically generated

1. After downloading files, we will extract them to “…CogView-main/pretrained/cogview” sub-directory with “tar -xvf” command

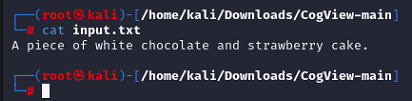


1. There are 2 files in the repository folder “generates\_sample.py” and “arguments.py” which has python functions and arguments required for running main scripts in sub-sequent sections.





1. Before moving to main program, we will create a new text file called “input.txt” which will contain the text for which program will find the right image from dataset with tokenization logic.

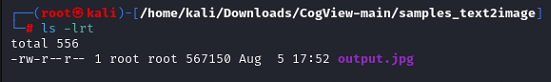


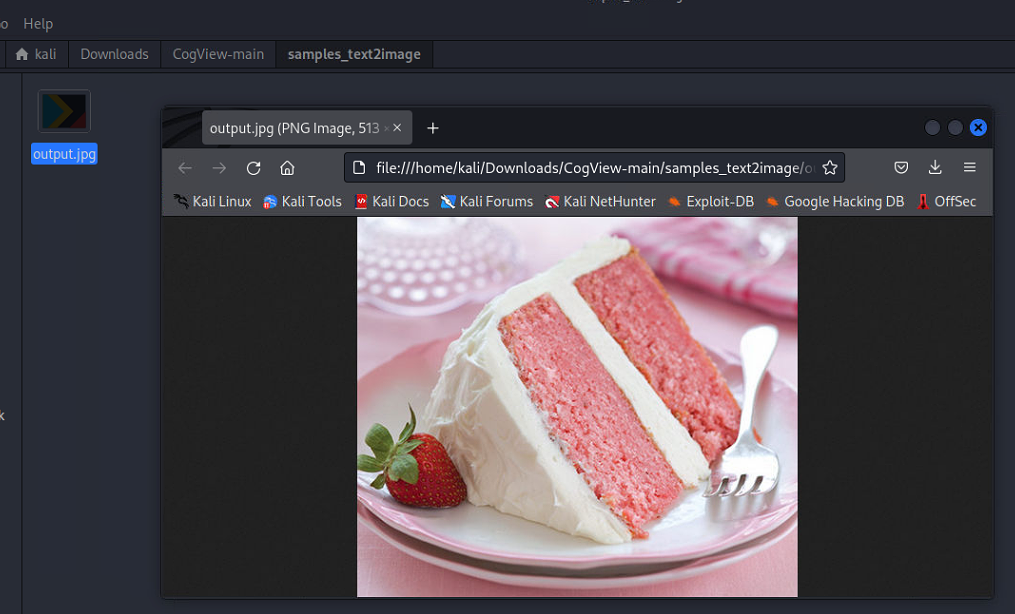
1. Now we will start running the main script “text2image.sh”. This shell script will run the main program and creates a folder called “samples\_text2image/” under same directory. This folder has the output image.

Text

Description automatically generated







1. Other scripts like “super\_resolution.sh”, “post\_selection.sh” and “pretrain\_single\_node.sh” are used for training the model, tokenizing the input data sets and correcting the missing parts of images according to base image. (like adding tail, ears to a tiger image if missing).

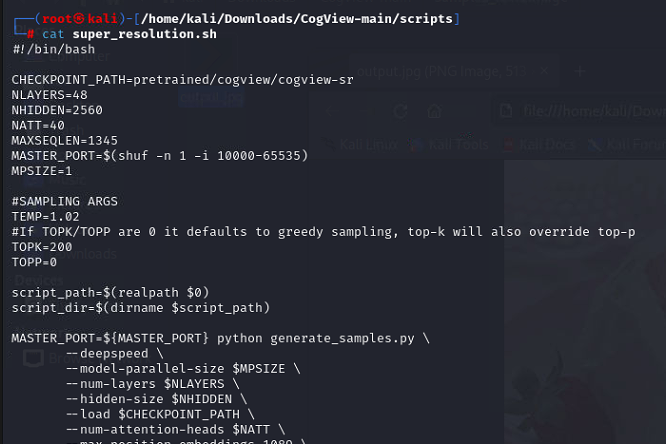




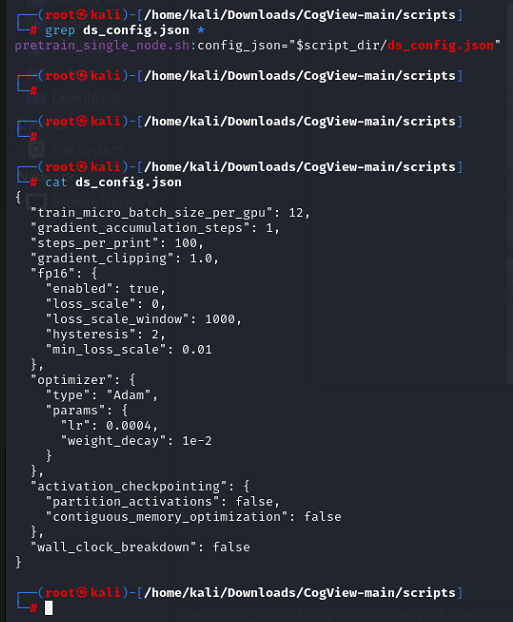




1. Enhancements:
   1. Time is one of the critical parameters for neural network model while processing large datasets. Here we tried to enhance the existing model in giving more efficient result by multiple methods.
      1. Instead of taking diverse dataset covering different type of images, we have localized the images for specific topic and observed the time it took to identify the image. This gave an improved result of nearly 20% less time than existing model.
      2. Besides, we tried to improve processing time through hardware upgrade i.e., when we tried to implement the program on a system with more RAM of 16GB and allocating more cores with 128GB hard disk capacity to the Virtual Machine, model has performed quick computation and produced result in comparatively less time.
      3. Accuracy is another aspect which is acting as a limitation for existing model. To enhance this point, we tried to change the super\_position.sh script variable values and observed small improvement in image identification process.



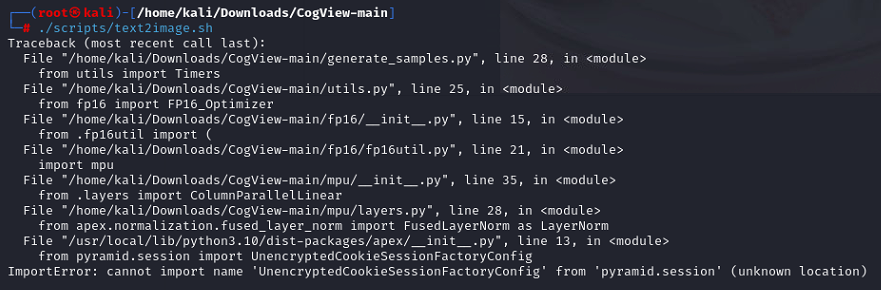
* + 1. Also, we tried to enhance model to reduce blurriness by changing variables in “ds\_config.json” file which is the configuration file used for pre-training the model.



1. Issues:

During implementation process, we encountered few issues like

* Sometimes variable value changes are not giving accurate results. So, we tried multiple times and then understood the right values to be used.
* Few python libraries latest versions are not compatible to the project. One such issue occurred while running “text2image.sh” script which is due to Apex and Cuda version compatibility issues.



We resolved this issue by googling and understanding that Apex library is not updates by developer’s community after 2018. So, with alternate solution as below, we downloaded NVIDIA apex drivers and installed with below commands.

git clone <https://github.com/NVIDIA/apex>

cd apex

pip install -v --disable-pip-version-check --no-cache-dir --global-option="--cpp\_ext" --global-option="--cuda\_ext" ./