What to do:  
  
Liquid AI:  
1) Get the models  
Download the models we are going to use:  
<https://huggingface.co/HuggingFaceTB/SmolVLM2-500M-Video-Instruct>  
and  
[https://huggingface.co/LiquidAI/LFM2-VL-450M](https://huggingface.co/LiquidAI/LFM2-VL-450M?clone=true)  
you can download using git:  
-git lfs install

-git clone <https://huggingface.co/HuggingFaceTB/SmolVLM2-500M-Video-Instruct>  
  
and

-git lfs install

-git clone <https://huggingface.co/LiquidAI/LFM2-VL-450M>

The models are big so it can take some time.  
  
2) Get the files to execute  
Download file and extract they were you want:  
- liquid.zip

3) Move the whole directories downloaded with the model form huggingface to the liquid folder extracted on the previous steps. Image to be more clear where they should be:  
A screenshot of a computer

AI-generated content may be incorrect.

4) Install python libs to run the models

Double click on the .bat file “install\_requirements.bat”  
If you have python and pip installed this will install all the requirements to run the model

5) run the models

Once the pip install is over, you can run the models.

Double click on the .bat file “run\_lfm\_model\_original.bat” to use the original liquid mode. This will run and open the browser to use the model. If it not work, open a terminal, go to the location of your file and type “py running\_model\_original\_...themodelyouwant.py”, and ctrl-click in the ip to open in the browser.

e.g.:

A screen shot of a computer

AI-generated content may be incorrect.  
A screenshot of a computer

AI-generated content may be incorrect.

To run the other model to the test, Double click on the .bat file “run\_smol\_model\_original.bat” to use the original liquid mode. This will run and open the browser to use the model.  
  
Now you should have two browsers windows open, one with the lfmv2-vl model and smolvlm2 model.  
  
5) Code carbon:

Download file and extract they were you want:  
- CarbonMonitor-2.0-win64.zip

6) Go to CarbonMonitor-2.0-win64/ CarbonMonitor-2.0-win64.zip/bin and double click on the .start\_monitor.bat file. This should open a browser window with the CapGreen code carbon.  
  
7) Monitoring the process to compare the models.  
First we need to find which models are the python process. For now we use the Task Manager on windows.  
Press the windows button and type “Task Manager”, on the Left click on Details, and on the search bar type python  
A screenshot of a computer

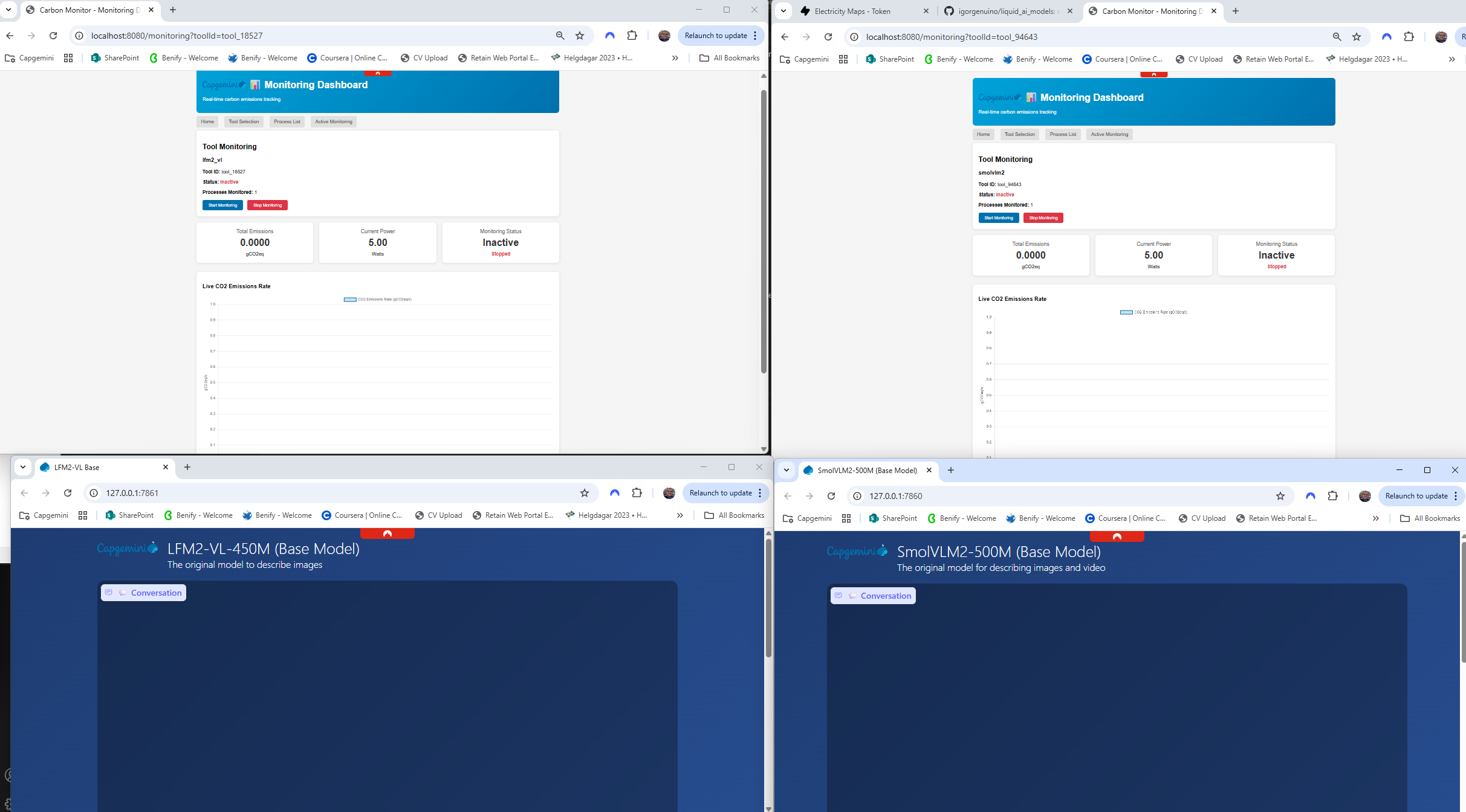
AI-generated content may be incorrect.

You would be able to see the python process corresponding to the running AI models. You need this information to seach on capgreen code carbon the application to monitor.  
  
Now that you have the PID of each process lets monitor it.  
In your browser, duplicate the tab of capgreen code carbon. Two tabs will be helpful to monitor at the same time the two process.  
  
In Capgreen carbon monitor, click in “Start Monitoring a Tool” .

As “tool name”, lets call it lfm2\_vl.  
In “select processes to monitor type the pid of the process for example, 24364 (change for the PID you find at the task manager)  
A screenshot of a computer

AI-generated content may be incorrect.  
Press Create Tool Monitoring  
  
In the other tab do the same for the other running process, call it smolvlm2 for example:  
A screenshot of a computer

AI-generated content may be incorrect.

Press Create Tool Monitoring  
  
You now must have two tabs of the Capgreen, two monitoring tools, one for each process, and the two model windows open to be used  


Press Start Monitoring on both. The chart should start to be populated. Try to start both at the same time to more accurately comparison.  
Now in the Models, you can upload a image and ask the same inference in both model for example:  
A computer screen with a blue background

AI-generated content may be incorrect.  
Firs add the inference and the image, and press Send.  
  
  
Example for results  
A screenshot of a computer

AI-generated content may be incorrect.  
  
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

About the code.

LFM2-VL: <https://huggingface.co/liquidai/LFM2-VL-450M>

SmolVLM2: <https://huggingface.co/HuggingFaceTB/SmolVLM2-500M-Video-Instruct>

There are not so many alternatives for LLM models with under 500M parameters that accept image as input. According to LFM2-VL-450M page the comparison is done with the Hugging LLM SmolVLM2. The reason is because they are the 2 vision model with less than 500M parameters. More parameters would not be a fare comparison.

**Comparison of Vision Processor Capabilities**

A review of the source code on the Hugging Face Hub reveals key architectural differences in how each model's processor handles images.

* The **LFM2-VL** processor exposes tunable parameters (min\_image\_tokens, max\_image\_tokens) that allow for fine-grained control over the image tokenization process.
* The **SmolVLM2** processor utilizes a fixed internal method, relying on its default configuration without exposing these specific parameters for user adjustment.

**Running the models**

Both models are implemented using Gradio web UI, a easy tool to run models in a web browser. The idea is to keep the python execution file as similar as possible in order to have a more fare comparison.

**Code Carbon – CapGreen – Carbon Monitoring**

It is a c++ application compiled for Windows, Mac and Linux. Since it is a c++ application it can be easily deployed as well for embedded targets using less resources.  
CapGreen contains a big base with power (watts) information from different processors (TODO: add the possibility to inform this as parameters). CapGreen retrieve the information of electricity vs carbon emissions from <https://app.electricitymaps.com/> according to the location. CapGreen carbon monitor calculates how much emissions have been produced by a process taking into account CPU, (TODO GPU) and RAM consumption. Total emissions and also emissions per second is available in a visual chart.  
TODO: save information to be used in a embedded system  
TODO: investigate the possibility of creating a lib to track specific part of code.