Project Guide

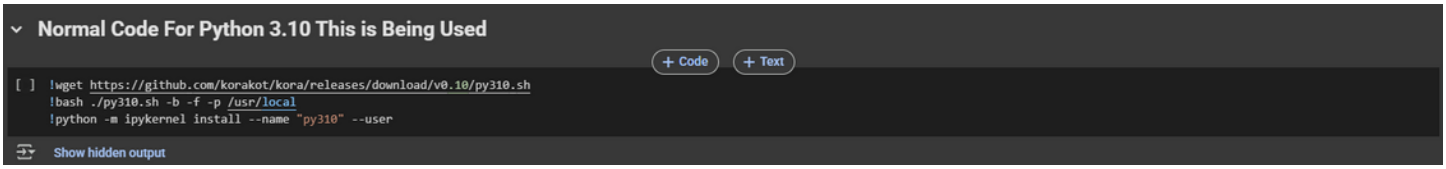
AI-DRIVEN RICE MONITORING USING EDGE COMPUTING ON ARM-M MICROCONTROLLERS

Prerequisites:

- Google Colab (For training without local GPU)
- Python Version: 3.9.x or 3.10.x
- STM32Cube IDE Version 1.16
- Cuda Version 11.8 (If there is GPU on local device)
- STEdge AI Developer Cloud Account
- Knowledge on how to use STM32Cube IDE
- Review STM32 Model Zoo GitHub directory here

Training on Google Colab:

- Upload the zip titled '`stm32-modelzoo-services-main_untouched.zip`, `new_5_balanced.zip`, `new_5_test`, and `refreshed_model_zoo.ipynb` to your Google Drive associated with your Google Colab account.
- In Google Colab, open the `refreshed_model_zoo.ipynb` notebook, connect to GPU, and run the cells starting from this cell:



until this cell:

In which you will be prompted to restart the session after these cells have finished running.

- After the session has been restarted run this cell and ensure the output shows CUDA version 11.8 is used.

```
!nvcc --version
!nvidia-smi
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Sep_21_10:33:58_PDT_2022
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
Thu Apr 24 23:19:23 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   36C    P8             9W /   70W |       0MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
```

- Navigate to the `/content/stm32ai-modelzoo-services-main/image_classification/src` and replace the contents of the `user_config.yaml` file with the the contents from the `new_config.yaml`. Note, to not change the name of `user_config.yaml` file, just replace the contents.
- Run the last two cells to begin training, and subsequently download the training output

```
import sys
import os
sys.path.append('/usr/local/lib/python3.10/site-packages')
sys.path.append('/usr/local/cuda-11.8/lib64')
sys.path.append('/usr/local/cuda-11.8/bin')
os.environ["CUDA_HOME"] = "/usr/local/cuda-11.8"
os.environ['CUDA_PATH'] = '/usr/local/cuda-11.8'
os.environ['PATH'] += ':/usr/local/cuda-11.8/bin'
os.environ['LD_LIBRARY_PATH'] = '/usr/local/cuda-11.8/lib64:' + os.environ.get('LD_LIBRARY_PATH', '')
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'  # Suppress TensorFlow warnings
%cd /content/stm32ai-modelzoo-services-main/image_classification/src
!python stm32ai_main.py
```

Show hidden output

```
[ ] import shutil
    %cd /content/stm32ai-modelzoo-services-main/image_classification/src
    shutil.make_archive('experiments_outputs', 'zip', 'experiments_outputs')
    # If running on Colab, run this cell to automatically download the outputs.zip file, else download manually.
    from google.colab import files
    files.download('experiments_outputs.zip')
```

Deploying the Model:

- Download and unzip the '`stm32-modelzoo-services-main.zip` to your local device
- Within the unzipped folder, create a virtual environment using the following commands:

```
cd stm32-modelzoo-services-main
python -m venv st_zoo
```

- Activate the virtual environment by navigating to the following directory:

  `cd stm32-modelzoo-services-main\Scripts` and use the 'activate.bat' command.

- Return to the main environment and install the required packages with the command

  `pip install -r requirements.txt`

- Setup on the local device is completed, follow the steps here to proceed with deployment on the STM32H747-i Disco Board.

Applying HSV Modification through STMCube IDE:

- Within the `\stm32ai-modelzoo-services-main\application_code\image_classification\STM32H7\Application\STM32H747I-DISCO\STM32CubeIDE` open up the '*.project*' file.
- In the '*app_camera.c*' file, modify the values of Hue, Saturation, and Brightness according to the value obtained from the '*linear_mapping.py*' script according to the correction to be applied

```
void Camera_Set_HueDegree()
{
  if (BSP_CAMERA_SetHueDegree(0,4) != BSP_ERROR_NONE)
  {
    while(1);
  }
}

void Camera_Set_Saturation()
{
  if (BSP_CAMERA_SetSaturation(0,1) != BSP_ERROR_NONE)
  {
    while(1);
  }
}

void Camera_Set_Brightness()
{
  if (BSP_CAMERA_SetBrightness(0, -2) != BSP_ERROR_NONE)
  {
    while(1);
  }
}
```

- Once modifications have been made, save, build, and run the application from the STM32Cube IDE

Utilizing Grad-CAM:

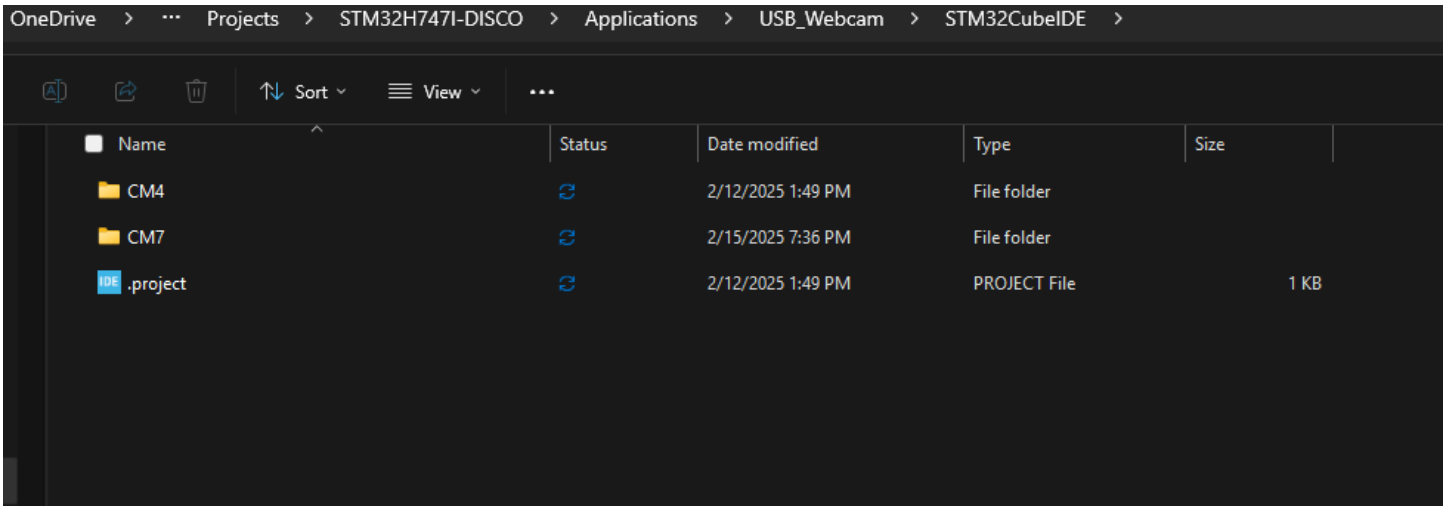
- Navigate to the `stm32-modelzoo-services-main\image_classification\src\prediction` and open the '*predict.py*' and go to the end of the *visualize_gradcam* function.
- Modify this directory to the preferred directory where you want your images to be saved:

```
gradcam_result_dir = r'G:\My Drive\trained_quantized\thesis\better\grad-cam-copare\efficientnet'
os.makedirs(gradcam_result_dir, exist_ok=True)
grad_res_filename = f'{gradcam_result_dir}/{os.path.basename(img_path)}.png'
cv2.imwrite(grad_res_filename,output)
```

- To use the prediction script, refer to the Model Zoo guide to carry out prediction here OR navigate to the `image_classification\src\prediction\README.md` and follow the '*README.md*' file.
- Make sure to use the '*.h5*' file and not the '*.tflite*' file as Grad-CAM will not work on quantized models

Camera Capture:

- Inside the Camera Capture STM Application, navigate to this folder:



- Open the *.project* file with STM32Cube IDE, build and run the project.
- Once the file is deployed on board, go to the camera app on your laptop to use that application for taking pictures.
- If you encounter any troubles, the STM32 FPVision AI package may be useful

Data Augmentation:

- Open the data_augmentation.py python code, then input the folder for your dataset. It will be better to use the same folder, so that the newly created images are kept in the same folder. Else, you will need to copy the original images to the folder.

```
# Directory paths
original_data_dir = r'C:\Users\harth\datasets\new_6_balanced'
augmented_data_dir = r'C:\Users\harth\datasets\new_6_balanced'
```

- Adjust the augmentation parameters here.

```
# Parameters
target_size = (128, 128)  # Adjust based on your image dimensions
default_num_images = 2000  # Default number of images to generate per class if not specified in num_images_dict

# Image Data Generator with Augmentation
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='nearest'
)
```

- Run the program and you will get a balanced dataset