Image Classification Game: Part 1

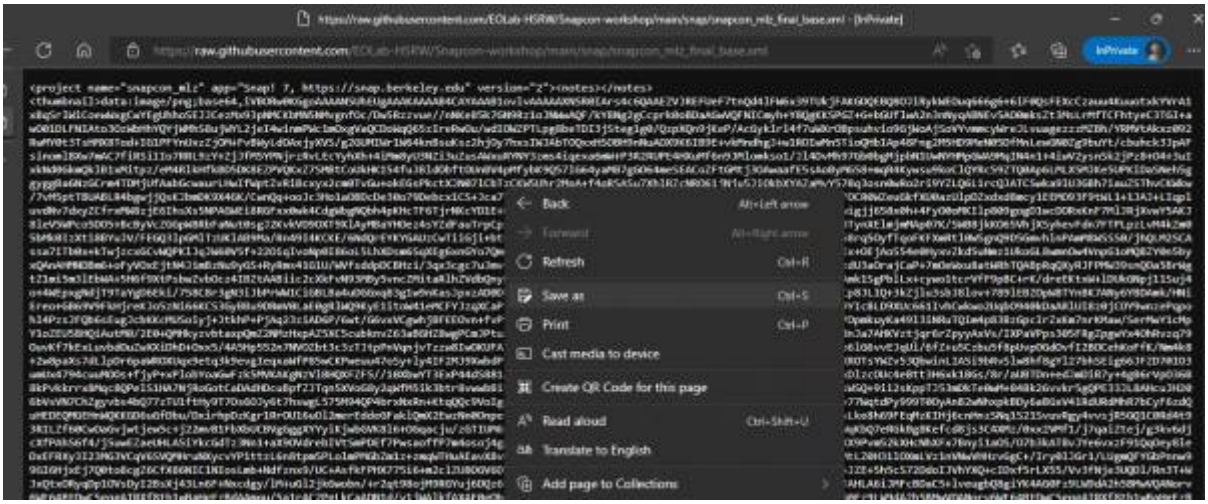
This **Snap! game** uses **Nvidia Jetson** capability to classify images.

Offline Snap! downloading

Please download and open Offline version of Snap! for our project. Go to https://snap.berkeley.edu/offline and follow the steps.

Snap! files' downloading

Please open the link Classification Game to download our project on your computer. Probably you would see the xml in raw format. Click the right button of your mouse and save it on the disk.



Web camera Image in Snap!

You can get picture from your web camera in Snap!.

- **video capture** block to enable video capturing.

- Change value of **set video transparency** block to *0* for clear image.

- **video snap on stage** block reports picture from stage.

Connection to Jetson from Snap!

If you have not imported it yet, please download jetson blocks and import it to your Snap! project.

Enable **Javascript Extensions** for following blocks.

- Use **connect to Jetson url** block to connect Jetson.

All participants need **IP address** of Nvidia Jetson in order to connect.

- Store the value of **jetson_name** in a variable.

- Store the value of **connect to Jetson url** block in a variable for later use.

Response from classification

Here we will send **video snap on stage** to Jetson for processing. Jetson will respond back class name, confidence value and class ID.

Only **class name** and **confidence value** will be used in this example. This project does not use ~~**class ID**~~.

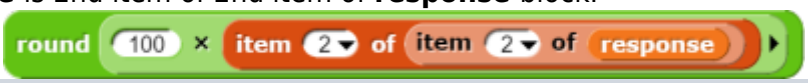
- Use **get response from Jetson** block to send **image** , and get **class name** and **confidence value**.
  - First input slot is for **jetson** variable that stores websocket data.
  - Second input slot is for **costume** you want to be classified by Nvidia Jetson.

Class name and confidence value

This section will demonstrate how to handle **response** variable to access **class name** and **confidence value.**

- **class name** is the 2nd item of 1st item of **response** block.

- **confidence value** is 2nd item of 2nd item of **response** block.



Multiply **confidence** value by 100 to get percentage of **confidence** .

You can create custom blocks, to get **class name** *get class name from response* and to get **confidence** *get confidence from response* .

Speech functionality

Speech functionality is available as a library in Snap!. Select *export libraries* from settings then choose *speech module* .

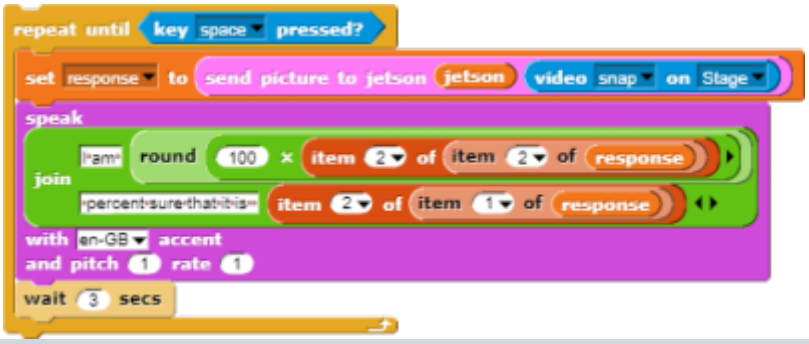
- Use **join** block to create text like *I am **confidence** percent sure, it is **class name*** .

- Use **speak and wait** block to read text a loud.

Repeat block for game

Last step is adding loop for the game.

- Use repeat block and put script inside of it.



This example used **repeat until** block to break loop when *space* key pressed.
You can download full game from Github page of EOLab-HSRW.

Image Classification Game: Part 2

Please open the link Classification Game: Extended to download the extended version of our project on your computer.

Start Camera

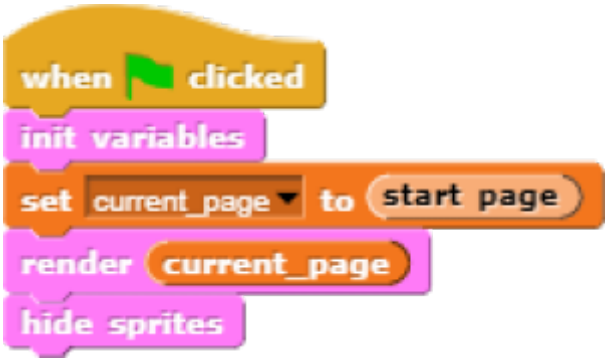
Drag and drop the **start video** custom block to start using web camera.

Connection to Jetson

Repeat the steps from Part 1 to connect to the Jetson Computer.

Game Initialization

Use **When flag clicked** , variable setting and our custom blocks for initializing the game



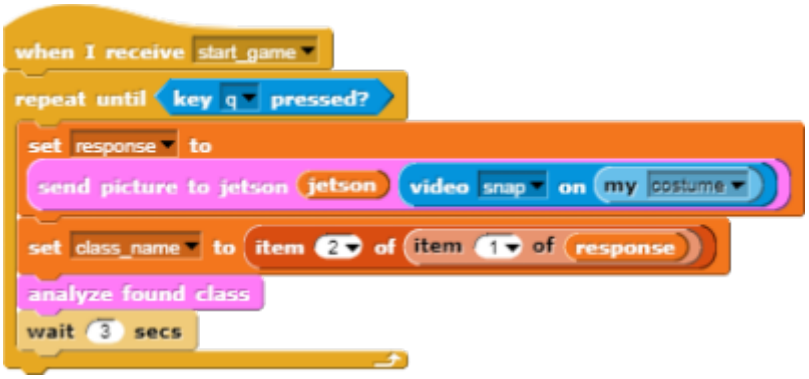
Game process control blocks

In the game process we use **when** block to catch the click event from the stage and change the current page and broadcasting to tell the other blocks that game is starting.

Also we added additional blocks to resume the game after it was stopped.

Main part: sending images to Jetson and having fun with our pets! :-)

We use familiar **when** block to listen to the start event. Then the block for receiving classification data is used in connection with **analyze** block



Inside **analyze** block we compare the class name with preset class names of food, that our pets consume, and broadcast to them. In case if the detected object is not suitable for any of them, we run **speak** block to pronounce the name of the object.

In the script part of each sprite there are already blocks, responsible for handling their choice events. We use **speak** , **play sound** , **animate** blocks to animate the sprites.

You can download full game from Github page of EOLab-HSRW.