# **Image Classification Game: Part 1**

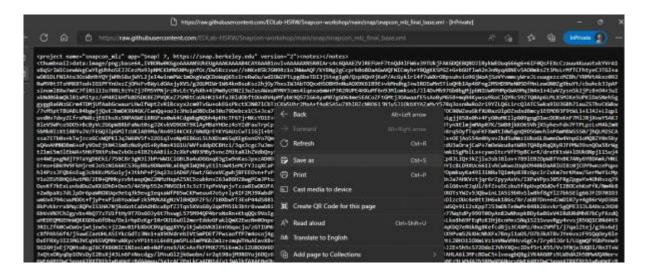
This **Snap!** game uses **Nvidia Jetson** capability to classify images.

### 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.



• 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.

```
set jetson_name to 192.168.0.48
```

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

```
etson to connect to jetson url join ws://
```

#### **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.

```
et response to send picture to jetson (jetson) video snap on Stage
```

#### 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.





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.

```
at until key space pressed?
        round 100 × item 2 of item 2 of
    percent/sure-that/it/is= | item (2 ♥ of (item (1 ♥ of
  pitch 1 rate 1
wait (3) secs
 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.

```
set jetson_name to dev-kit-20
set jetson to connect to jetson url join ws:// jets
```

#### **Game Initialization**

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

```
when clicked
init variables
set current_page to start page
render current_page
hide sprites
```

#### **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.

```
when start label is clicked

set current_page to game page

render current_page

broadcast start_game to all wait until not mouse down?
```

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

```
when space key pressed

set current_page to game page

render current_page

broadcast start_game
```

# 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

```
when I receive start_game *
repeat until key q pressed?
 et response to
 send picture to jetson jetson video snap▼ on my costume ▼
 set class_name to item 2 of item 1 of response
 analyze found class
wait 3 secs
```

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.

```
+ analyze + found + class +
if(class_name)=(an1_food))
broadcast rabbit_choice ▼ to Rabbit ▼ ◆
if (class_name) = (an2_food)
 broadcast monkey_choice ▼ to Monkey ▼
else
 if (class_name) = (an3_food)
  broadcast mouse_choice ▼ to Mouse ▼ (
  if (class_name) = (an4_food)
   broadcast squirrel_choice ▼ to Squirrel ▼ ◆
  else
   if (an5_food_list) contains (class_name
    broadcast worm_choice ▼ to Worm ▼ 4
    and pitch 🚺 rate 🚺
```

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.

```
when I receive monkey choice
speak join Oh, lithink this is class_name → with en-US ▼ accent
and pitch 🕕 rate 🕕
play sound monkey
animate
```



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

From:

https://student-wiki.eolab.de/ - HSRW EOLab Students Wiki

Permanent link:

https://student-wiki.eolab.de/doku.php?id=snapcon2022:image-classification-game&rev=1659796390

Last update: 2023/01/05 14:38

