The Glie-44 API

The Glie_44 class is composed of many functions to run predictions on various types of inputs. To satisfy the solution design, the model needed to predict each frame of a drone's recorded video. This method was decomposed into separate functions to give any developer the ability to create their own methods based on this API. The API was then improved to run on a directory of frames instead of a video to satisfy some drones' output. This API allows for customization concerning various parameters of predictions to adapt the model to anyone's needs.

Here is a documentation of each element composing the API.

CLASS Glie_44(precision, font_path)

Initializes the glie_44 class. All upcoming methods are specific to this class.

Parameters:

- ***Precision**:* (*float*) - Minimum confidence rate of predicted boxes to be considered. This ratio is between 0 and 1.
- ***Font_path***: (*String* or *PosixPath*) - Path to the saved font to be used in the model's boxes display.

Output: An instance of the Glie_44 class.

METHOD set_font_path(font_path)

Modifies the class' font path.

Parameters:

- ***Font_path***: (*String* or *PosixPath*) - Path to the saved font to be used in the model's boxes display.

Output: No output.

METHOD get_font_path()

Fetches the class' font path.

Parameters: No parameters.

Output: (*String*) - Path to the font.

METHOD set_precision(precision)

Modifies the class' precision (or minimum confidence rate).

Parameters:

- ***Precision***: (*float*) - Desired minimum confidence rate, between 0 and 1.

Output: No output.

METHOD get_precision()

Fetches the class' precision (or minimum confidence rate).

Parameters: No parameters.

Output: (*float*) - Minimum confidence rate.

METHOD set_model(model)

Modifies the class' model.

Parameters:

- ***Model***: (*PyTorch model*) - Model to be used by Glie_44.

Output: No output.

METHOD get_model()

Fetches the class' model.

Parameters: No parameters.

Output: (*PyTorch model*) - Model to be used by Glie_44.

METHOD load_model(path)

Loads the model from a given path and adds it to the class.

Parameters:

- ***Path***: (*String* or *PosixPath*) - Path to the model to be loaded into Glie_44.

Output: No output.

METHOD pred(image_tensor)

Runs the model on a list of image tensors and outputs all the predicted boxes for each image in a list.

Parameters:

- ***Image_tensor***: (*PyTorch tensor*) - Tensor of shape [image_data] where image_data is the tensor made of the image to run the prediction on.

Output: List of dictionaries with 3 keys:

- "boxes": (*FloatTensor[N, 4]*): the coordinates of the N bounding boxes in [x0, y0, x1, y1] format, with 0 <= x1 < x2 <= W and 0 <= y1 < y2 <= H
- "labels": (*Int64Tensor[N]*): the predicted label for each bounding box.
- "scores": (*FloatTensor[N]*): the scores or confidence rate of each prediction, between 0 and 1.

METHOD convert_to_tensor(image)

Converts a *PIL image* or *numpy.ndarray* into a tensor.

Parameters:

- ***Image***: (*PIL Image* or *numpy.ndarray*) - Image to be converted, each pixel should be ranged between 0 and 255.

Output: (*FloatTensor*) Tensor of shape (C x H x W) in the range [0.0, 1.0] representing the image.

METHOD run_on_image(image)

Runs the model on a *PIL image* or *numpy.ndarray.*

Parameters:

- ***Image***: (*PIL Image* or *numpy.ndarray*) - Image to run the prediction on, each pixel should be ranged between 0 and 255.

Output: (*PIL Image*) The image with the predicted boxes and labels added.

METHOD run_on_image_with_path(path)

Runs the model on an image from a given path.

Parameters:

- ***Path***: (*String* or *PosixPath*) - Path to the image.

Output: (*PIL Image*) Image with the predicted boxes and labels added.

METHOD run_on_folder(folder_path, output_folder, output_path, framerate)

Runs the model on all images from a given directory, compiles and saves the images into a new predicted video. The images names must be written in the right order as they will be sorted.

Parameters:

- ***Folder_path***: (*String* or *PosixPath*) - Path to the directory to run the prediction on.
- **Output_folder**: (*String* or *PosixPath*) - Path to the directory to save the new video.
- **Output_path**: (*String*) - Name of the new video.
- **Framerate**: (*Float*) - Desired framerate for the video.

Output: No output.

METHOD run_on_video(video_path, output_folder, output_path)

Runs the model on a given video and saves the predicted video.

Parameters:

- ***Video_path***: (*String* or *PosixPath*) - Path to the video to run the prediction on.
- **Output_folder**: (*String* or *PosixPath*) - Path to the directory to save the new video.
- **Output_path**: (*String*) - Name of the new video.

Output: No output.

UI

Firstly, the UI allows users to select the model to use for predictions on their machine via a file explorer. The models can be downloaded from the Google Drive of which a link available on the Poe Dameron's GitHub page.

The user can then choose what prediction to do depending on the input: on image, on a video or on a directory (or folder). Each of these possibilities are represented in tabs, each independent from one another. Indeed, each tab has its own memory therefore it stays the same after a tab switch.

A button appears to select the input through a file explorer. Once selected, the image/video is displayed to confirm the selection of the user. Some controls are added in the case of displaying a video such as a play/pause button and a back to beginning button. The browse button remains accessible if needed.

A text input to choose the minimum confidence rate is displayed. In the case of a prediction on a directory, a text input concerning the framerate is displayed as no framerate can be deducted from the original input.

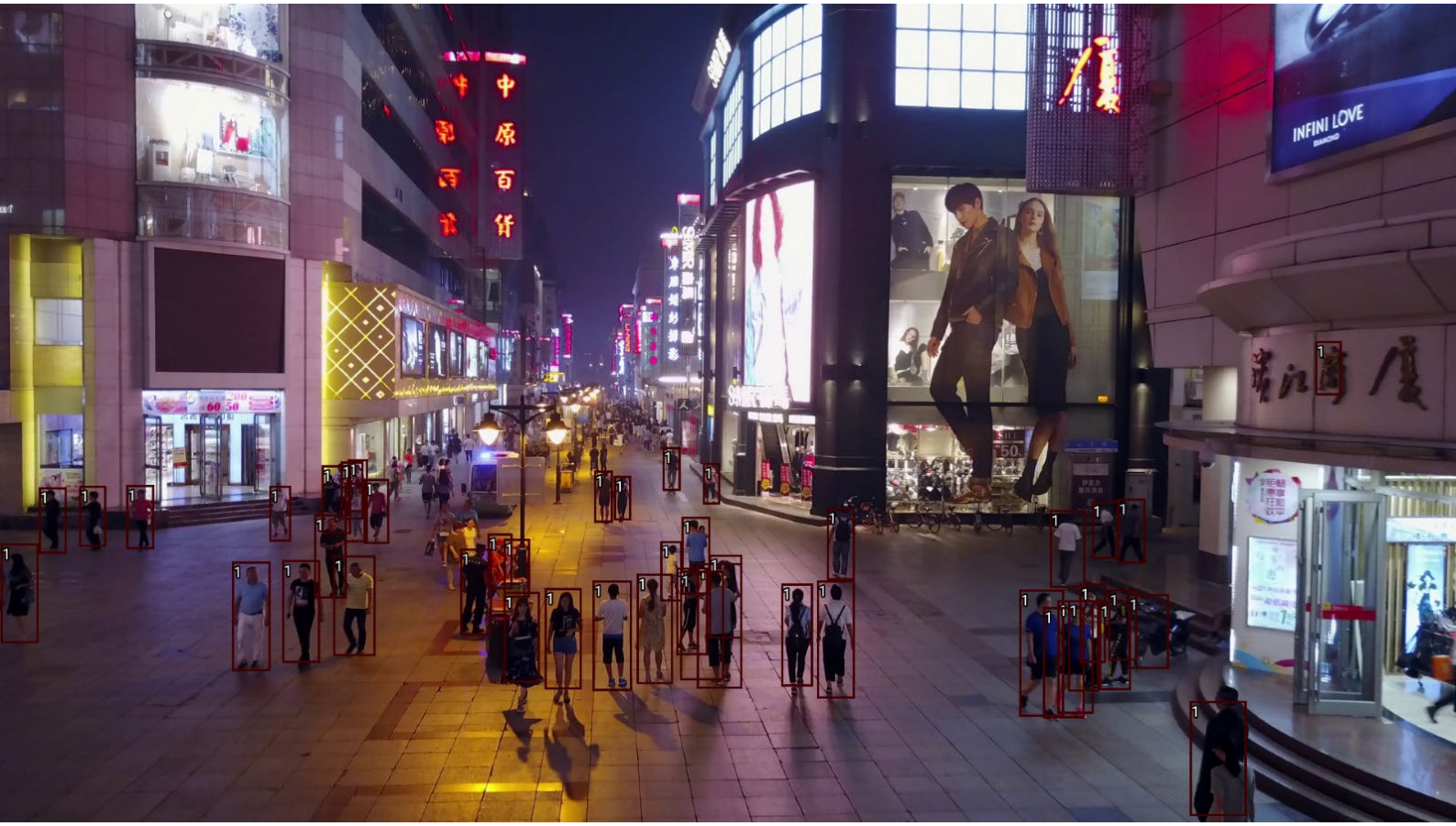
Another button is displayed to begin the prediction. In the case of images, the prediction is run, and the result is displayed immediately. Then, the button changes to save the model, in which case the user can select an output directory and a filename. Concerning videos, the output is automatically saved so clicking on the prediction will ask for the output directory and filename. After clicking the confirm button, the prediction begins.

The screen will hold while running the prediction. To have more information on the ongoing execution of the program, running the file through a console is proven more interesting to better understand what the program is doing.

The first tab is a legend to explain the different numbers representing the labels.

Testing

The required functionality of the initial solution is implemented well in the run_on_video method of the API, and the video tab for the UI (cf. output_test.avi in the given files). The result of the run_on_image method used in all predictions is:



The use of rectangles around the predicted objects allows for clear identification. The use of numbers instead of the names of the classes prevents overwriting excessively on images and therefore losing information. A legend is available in the first UI tab to explain the significations of each number.