

Training model

Training model

1. Development environment
2. Code analysis
3. Start training mode
4. End kernel

1. Development environment

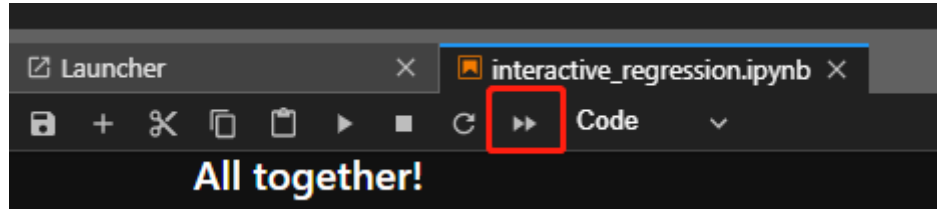
This training model needs to use TensorFlow and other related tools. Due to the use of many tools and complex deployment, the ROSMASTER car factory image has deployed the development environment of the training model, which can be used directly without repeated construction.

2. Code analysis

Please check the specific code: Rosmaster/auto_drive/interactive_regression.ipynb

3. Start training mode

Open jupyter lab with a browser, find the interactive_regression.ipynb file and open it, click the button to run all cells, then pull to the bottom, and wait for the control to be displayed.



You can see that this part of the content is displayed at the bottom, among which

dataset: Indicates the database, generally choose A;

category: Indicates the type, only apex can be selected;

count: Indicates the number of data sets, that is, how many pictures;

epochs: Indicates the number of training times, the more training times, the longer the time, the more accurate the model (I usually input 30 times);

progress: Indicates the training progress;

loss: Indicates the objective function value, which normally decreases slowly with the increase of training times, and then tends to be stable;

train: Start training;

evaluate: Evaluate dataset accuracy;

model path: Model name, the default is road_following_model.pth, if it is changed, it will affect the calling model;

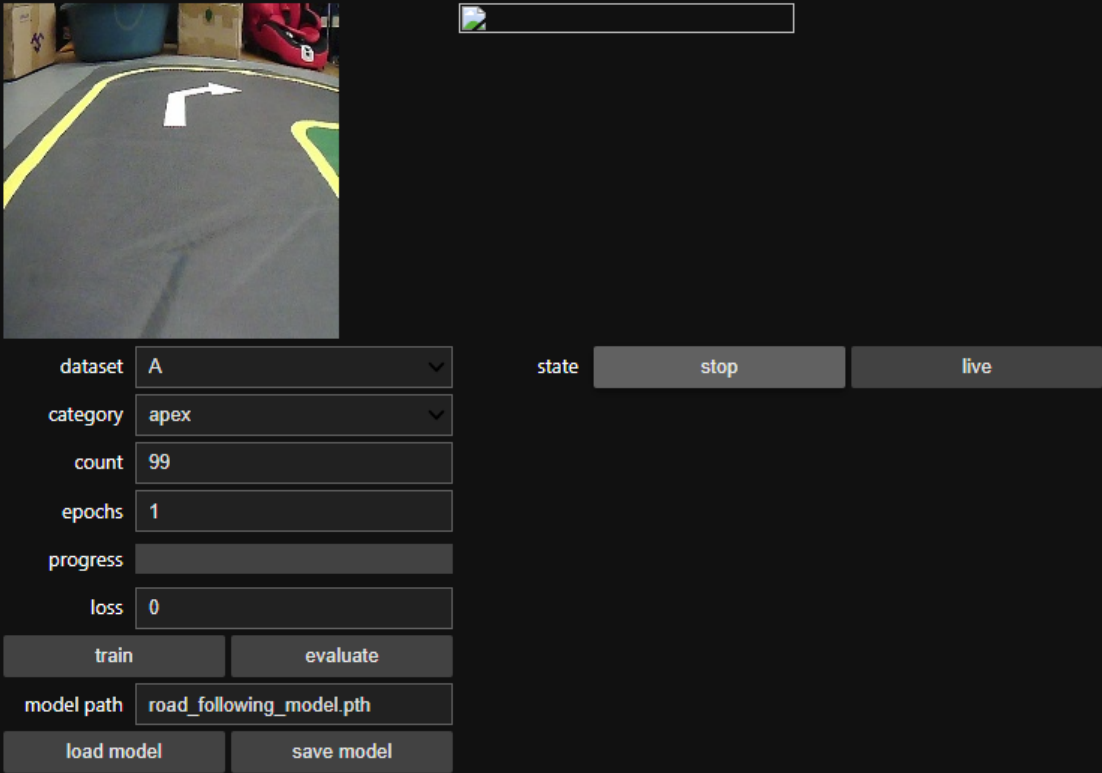
load model: Load the model, you can load the trained model;

save model: Save the model, this is very important, because the browser will not automatically save the model, please click to save the model after the training is complete;

state: stop means to disable the real-time image evaluation, and live means to enable the real-time image evaluation.

```
[7]: all_widget = ipywidgets.VBox([
      ipywidgets.HBox([data_collection_widget, live_execution_widget]),
      train_eval_widget,
      model_widget
    ])

display(all_widget)
```



The screenshot displays a JupyterLab interface with a dark theme. At the top, a code cell shows the execution of IPython widgets to create a dashboard. Below the code, the dashboard is visible, featuring a video feed of a road with yellow lane markings and a white arrow. To the right of the video is a small progress bar. Below the video, there are several interactive elements: dropdown menus for 'dataset' (set to 'A') and 'category' (set to 'apex'), input fields for 'count' (99), 'epochs' (1), and 'loss' (0). There are also buttons for 'train', 'evaluate', 'load model', and 'save model'. On the right side, there are two buttons labeled 'stop' and 'live' under the 'state' label.

Since the data collection in the previous section is completed, you can see that the count of the data set is 99, so you only need to modify the epochs to 30, and then click train to start training. After the training is completed, click save model to save the model.

4. End kernel

Since the camera is used in this training, the kernel needs to be closed after the training is over. Before closing the kernel, please confirm whether the model training is complete and save it. If you do not click to save the model, it will cause an irreparable error.

Click on the kernel management bar, find interactive_regression.ipynb and click the X on the right to end.

