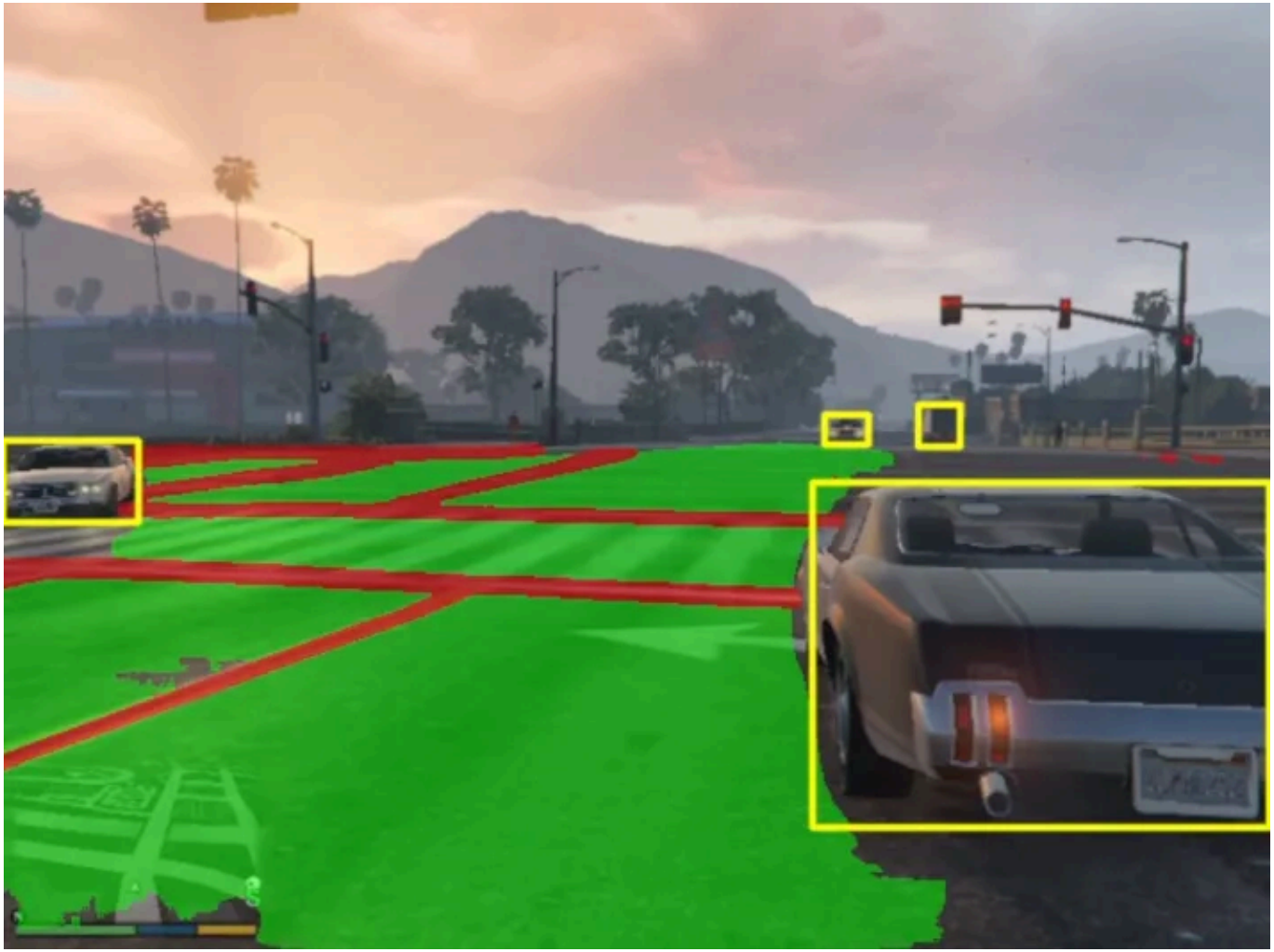
Detect Roads and Vehicles With YOLOPv2 In Grand Theft Auto 5

Install and set up YOLOPv2 so you can use it for drivable surfaces, lane lines, and vehicle detection with the video game GTA V

Jes Fink-Jensen Follow 6 min read · Oct 16, 2022



Below, I will show you how to install and set up the YOLOPv2 project from GitHub. In addition, I'll show you how to test the installation. After that, I'll review the code for using YOLOPv2 with the video game Grand Theft Auto 5. Finally, I'll show you the results of running the code with the game.

Installing and setting up YOLOPv2

You can find a description of YOLOPv2 on Github here.

To install the project files, you can do the following:

```
git clone https://github.com/CAIC-AD/YOLOPv2.git
cd yolopv2
pip install -r requirements.txt
```

The directory layout of the project looks like this:

```
D:.
├───data
│   ├───demo
│   ├───log
│   └───weights
└───utils
```

Download the `yolopv2.pt` model file from here and place it in the `weights` directory.

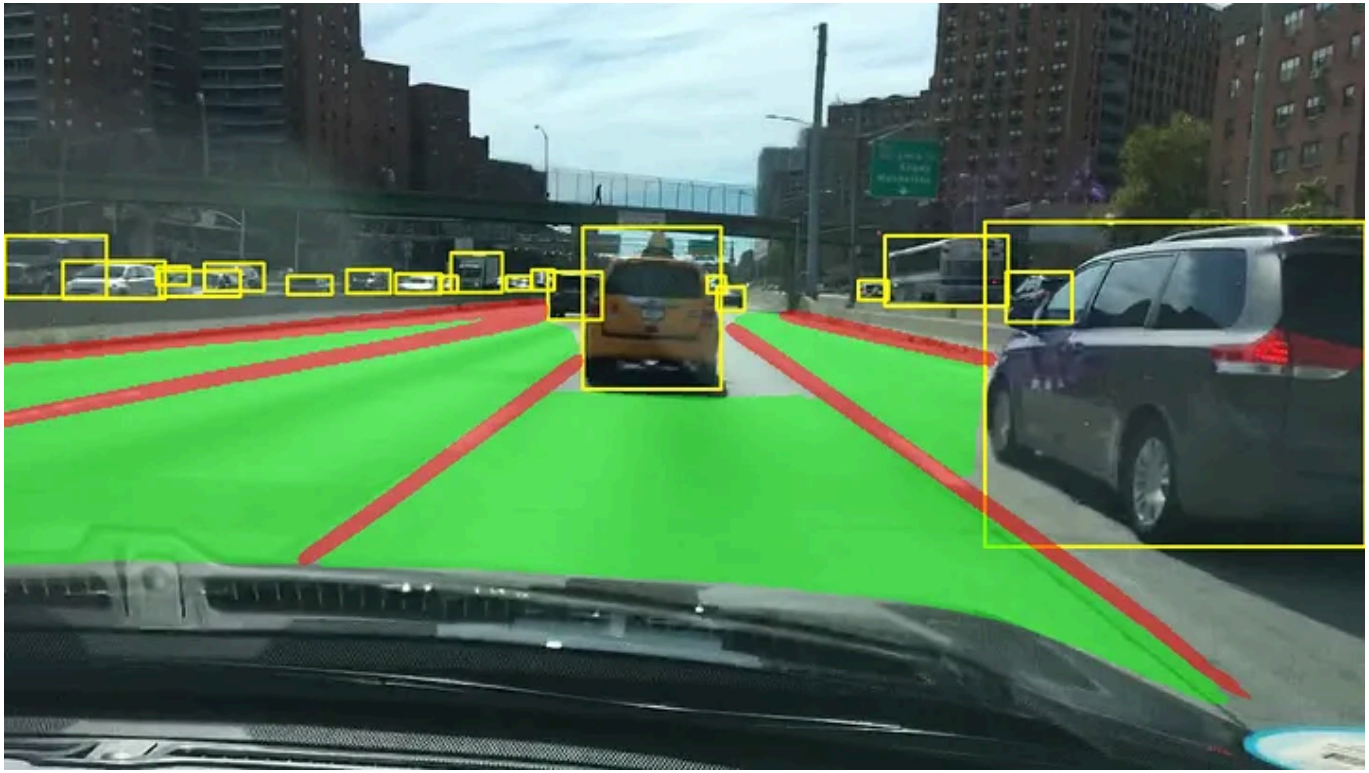
You can run the demo to check that the installation was performed correctly:

```
python demo.py
```

You can find the result in the `/runs/detect/exp` directory.

```
D:.
├───data
│   ├───demo
│   ├───log
│   └───weights
├───runs
│   └───detect
│       ├───exp
└───utils
```

The result should look like this:



The code for using YOLOPv2 with GTA V

To perform the detections on the video stream from the game, you can use the code below:

```
import mss
from win32gui import FindWindow, GetWindowRect, SetForegroundWindow
import torch
import cv2 as cv
import numpy as np
from time import time, sleep

from utils.utils import letterbox, driving_area_mask, lane_line_mask,\
 split_for_trace_model, non_max_suppression, plot_one_box, scale_coords, clip_coords

window_handle = FindWindow(None, "Grand Theft Auto V")
window_rect   = GetWindowRect(window_handle)
SetForegroundWindow(window_handle)

x0, y0, x1, y1 = window_rect
x_corr = 6
y_corr = 29
w, h = x1 - x0 - x_corr, y1 - y0 - y_corr

def Screen_Shot(left=0, top=0, width=1920, height=1080):
        stc = mss.mss()
        scr = stc.grab({
                'left': left,
                'top': top,
                'width': width,
                'height': height
        })

        img = np.array(scr)
        img = cv.cvtColor(img, cv.IMREAD_COLOR)

        return img

```

```
model_file_path = "D:/JFJ/Projects/bots/YOLOPv2/data/weights/yolopv2.pt"
model = torch.jit.load(model_file_path)
model.cuda()
model.half()
model.eval()

imgsz = 640
model(torch.zeros(1, 3, imgsz, imgsz).cuda().type_as(next(model.parameters())))


loop_time = time()
with torch.no_grad():
        while(True):

                screenshot = Screen_Shot(x0 + x_corr, y0 + y_corr, w, h)

                img0 = screenshot.copy()
                img = cv.resize(img0, (640,480), interpolation=cv.INTER_NEAREST)

                output = img.copy()
                out_w, out_h = output.shape[1], output.shape[0]

                img = img.transpose(2, 0, 1)
                img = torch.from_numpy(img).cuda()
                img = img.float().half()
                img /= 255.0
                if img.ndimension() == 3:
                        img = img.unsqueeze(0)
                [pred,anchor_grid],seg,ll = model(img)

                masking = True
                obj_det = True
```

```
66
67              if masking:
68
69                      da_seg_mask = seg
70                      _, da_seg_mask = torch.max(da_seg_mask, 1)
71                      da_seg_mask = da_seg_mask.int().squeeze().cpu().numpy()
72
73                      ll_seg_mask = ll
74                      ll_seg_mask = torch.round(ll_seg_mask).squeeze(1)
75                      ll_seg_mask = ll_seg_mask.int().squeeze().cpu().numpy()
76
77                      color_area = np.zeros((da_seg_mask.shape[0], da_seg_mask.shape[1], 3), d
78
79                      color_area[da_seg_mask == 1] = [0, 255, 0]
80                      color_area[ll_seg_mask == 1] = [255, 0, 0]
81                      color_seg = color_area
82                      color_seg = color_seg[..., ::-1]
83                      color_mask = np.mean(color_seg, 2)
84                      output[color_mask != 0] = output[color_mask != 0] * 0.5 + color_seg[colo
85
86              if obj_det:
87                      pred = split_for_trace_model(pred,anchor_grid)
88                      pred = non_max_suppression(pred)
89                      pred0 = pred[0]
90
91                      img0_shape = output.shape
92                      clip_coords(pred0, img0_shape)
93
94                      for det in pred0:
95                              *xyxy, _, _ = det
96                              plot_one_box(xyxy, output)
97
98              cv.imshow("YOLOPv2", output)
```

```
99
100                 print("FPS {}".format(1.0 / (time() - loop_time)))
101                 loop_time = time()
102                 key = cv.waitKey(5)
103                 if key == ord("q"):
104                         cv.destroyAllWindows()
105                         break
```

gtav_yolopv2.py hosted with ❤ by GitHub view raw

Place the above code in the upper level of the YOLOPv2 project directories.

Make sure also to install `mss` and `win32gui` for Python, if they are not already installed on your system. You can do that like this:

```
pip install mss
pip install pywin32
```

In lines 8 and 9, we import some functions from the `util.py` file in the `util` directory of the original YOLOPv2 project.

In lines 11–13, we find the window containing the GTA V game and make it visible in the foreground. The latter is necessary for the screenshot function.

In lines 15–18, the position and size of the game window are determined. In addition, we add some corrections to remove the menu bar and the borders from the screenshot coordinates.

In lines 20–32, we define a function that will take a screenshot of any rectangle on the screen. The input parameters determine this rectangle.

In lines 34–38, the YOLOPv2 model is loaded. I had to write the full path to get it to find the model file `yolopv2.pt`. Subsequently, the model file is sent to the GPU (cuda), the precision is halved to speed up the inference process, and the model is evaluated.

In line 40, the image size `imgsz` is set to 640. This is because the YOLOPv2 neural network takes an input of 640x640. In the following line, the network is 'warmed up.'

In line 45, we set `torch.no_grad()` to reduce memory usage and speed up computations.

In line 46, an infinite loop is set up to constantly take screenshots from the game and perform the detections on each screenshot.

In line 48, we take a screenshot of the game. Here we make sure not to take screenshots of the menu bar etc.

In line 51, we reduce the screenshot size to fit the YOLOPv2 network. As mentioned before, the network can take an image of a maximum of 640x640 pixels as input. Our screenshot is therefore reduced to 640x480.

In lines 56–61, we adapt the image data more. The dimensions are transposed, the image matrix is converted into a Pytorch tensor, sent to the GPU, the numbers are converted to floating point, the precision is halved, the numbers are normalized...

In line 62, the image is finally input into the YOLOPv2 model. The outputs are the vehicle detections in `[pred, anchor_grid]`, the drivable surface (road) in `seg`, and the lane lines in `ll`.

In lines 67–84, we do the masking of the drivable surface and the lines on the road on the output image. This is based on `seg` and `ll`.

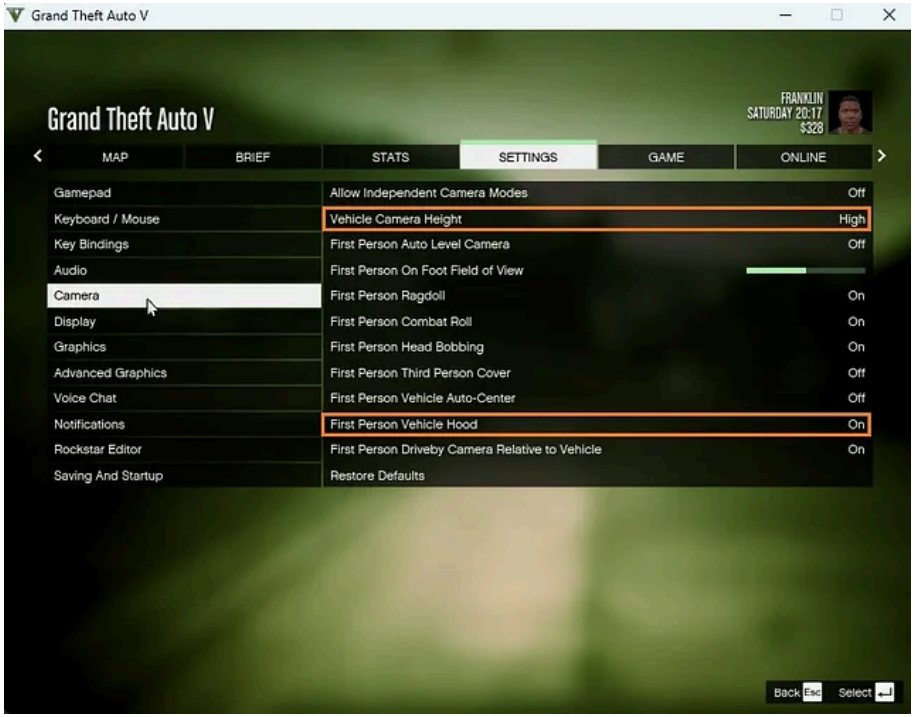
In lines 86–96, the vehicle detections are added to the output image. Here, several existing functions from the original YOLOPv2 project are reused. This is based mainly on `pred`.

In line 98, the final output image is shown on our screen in a window using OpenCV.

In lines 102–105, there is some standard OpenCV code for handling the showing of the output and waiting for "q" keypresses to close the output window.

Running YOLOPv2 with GTA V

I have set GTA V to run windowed at a resolution of 800x600 pixels. From my knowledge, the settings inside the game that are shown below are sufficient.



Once the game has loaded and we've entered into "story mode," our character can enter a car somewhere by pressing "f." Then, you can press "v" several times to get the view from the car's hood.

Unfortunately, the view is not centered on the hood but in front of the driver.

> *Note once you click inside the game window and are running the game, the cursor cannot go outside the window. You need to press* `Shift-Tab` *in order to release the*

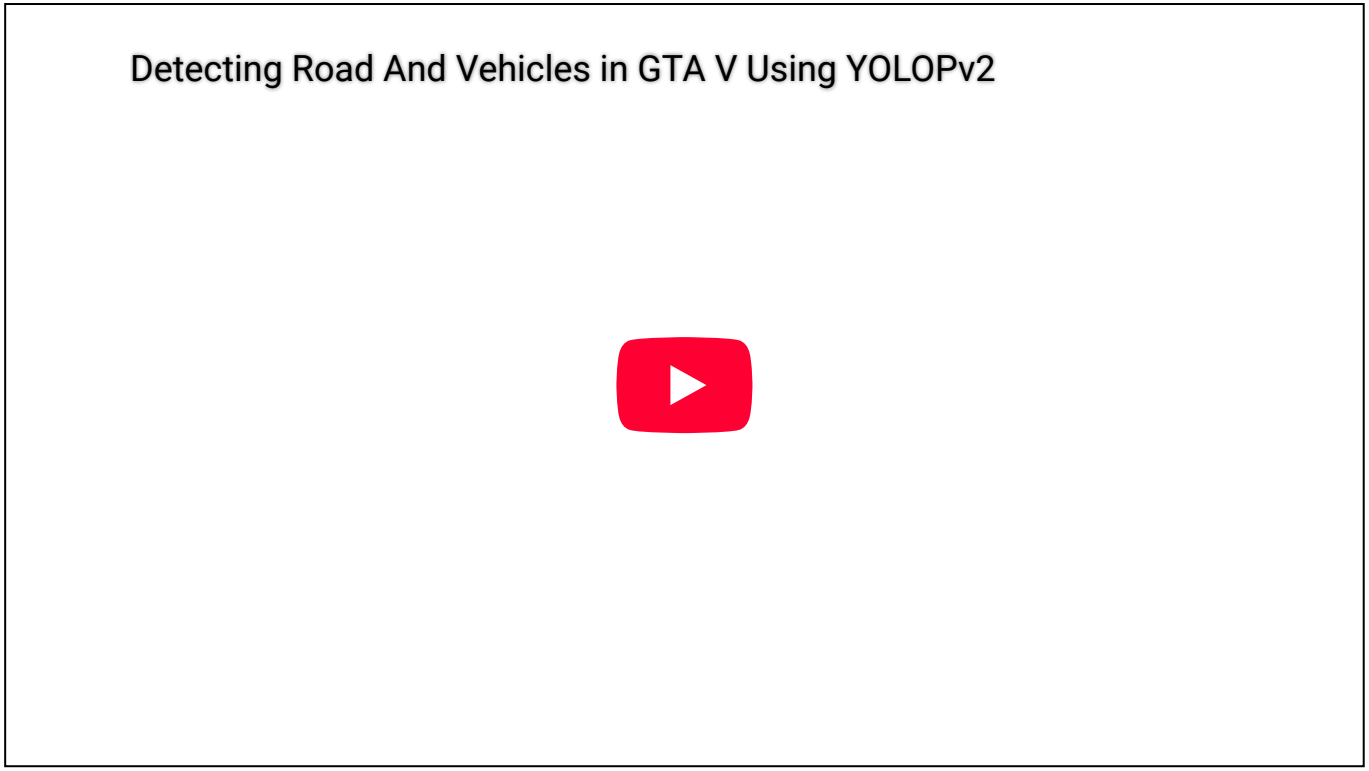
> *cursor.*

Now, we can run the above piece of Python code…

```
python gtav_yolopv2.py
```

The command line output will show the frame rate in FPS, and a window will show a video stream of the game with the YOLOPv2 detections overlayed.

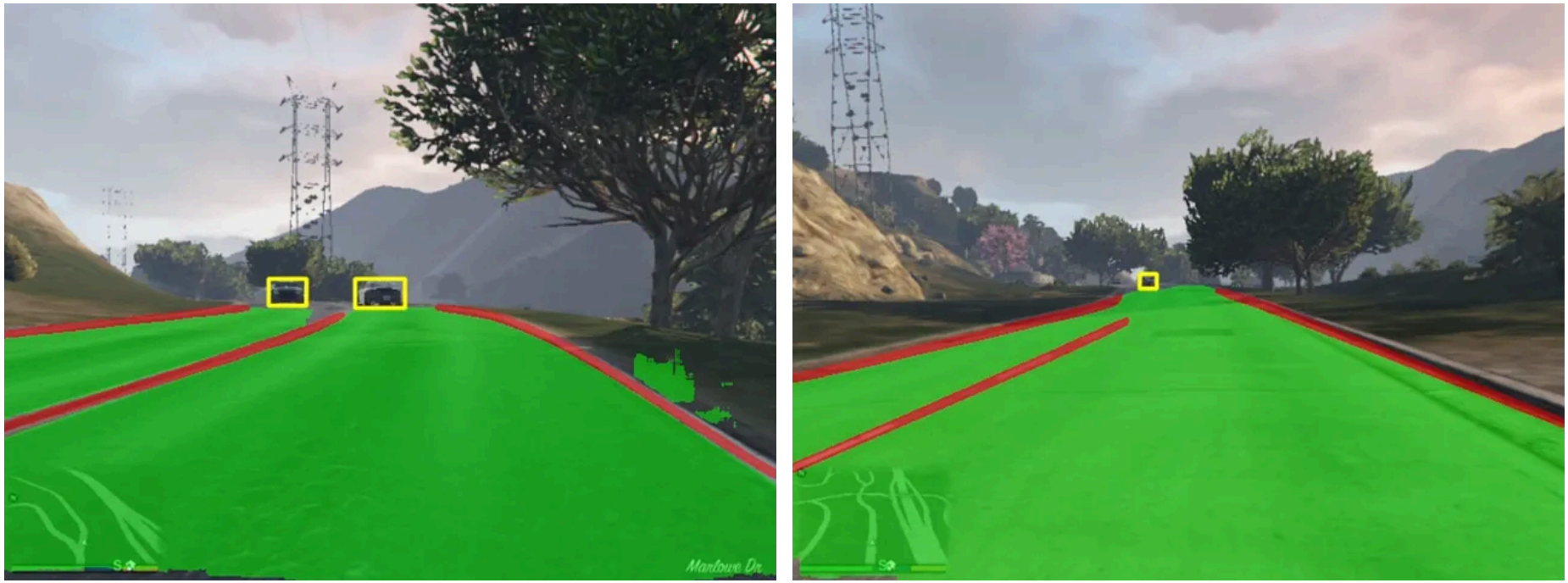
Below, you can see a video showing the result.



I just want to note that my hardware setup is an Intel NUC 10 connected to an NVidia GTX 1080 FE in a Razer Core X eGPU enclosure. The connection is via Thunderbolt.

The setup is fancy in its own way but not performance-wise. At least not to current (October 2022) standards. So, the fact that the YOLOPv2 network inference works as fast as it does is quite impressive.

Below are also a few stills from another run I did. Sometimes the road and lane line detections work perfectly, but sometimes they can be confusing.



Two very nice road and line detections with YOLOPv2

Two slightly confusing road and line detections with YOLOPv2

Conclusion

Overall, I believe the YOLOPv2 works very well, given that it wasn't made to work with a video game input and because my hardware is not particularly high-performance.

The vehicles (cars and trucks) are detected at very high precision. The drivable road and lane lines are detected very well but could be improved.

Perhaps doing some simple image processing on the screenshots before inputting them into the neural network would increase the quality even further. In this case, applying a Gaussian Blur or tweaking the HSV bounds of the screenshots may help.

References

"Can't install win32gui" on Stackoverflow

"What is the use of torch.no_grad in pytorch?" on Stackoverflow

"YOLOPv2🚀: Better, Faster, Stronger for Panoptic driving Perception" on GitHub

Computer Vision | Python | Machine Learning | Game Development | Software Development

Published in Better Programming

Follow

221K followers · Last published Nov 11, 2023

Advice for programmers.

Written by Jes Fink-Jensen

615 followers · 573 following

Follow

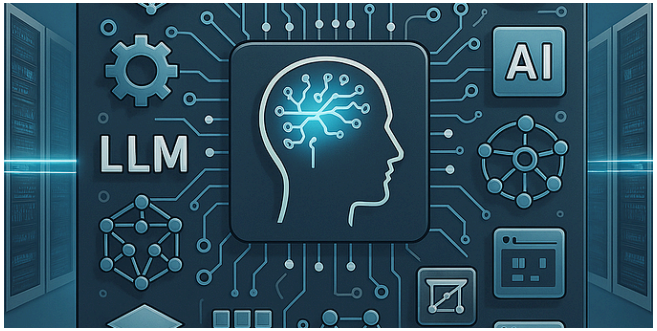
Not your average geek...

No responses yet

C Gracejhlim

What are your thoughts?

More from Jes Fink-Jensen and Better Programming



In Generative AI by Jes Fink-Jensen

Use n8n Like a Pro with Anthropic's 6 Simple Composable Patterns fo...

Step-by-Step Guide to Employing n8n's Composable Patterns for Superior LLM...

Apr 1 560 9

In Better Programming by Benoit Ruiz

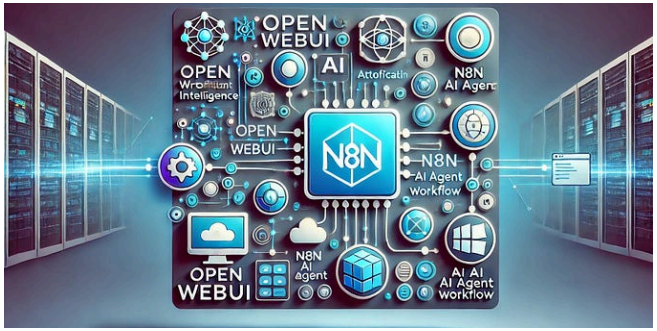
Advice From a Software Engineer With 8 Years of Experience

Practical tips for those who want to advance in their careers

Mar 21, 2023 19K 329

In Better Programming by Jason Ngan

Understanding the Offset and Cursor Pagination



In Generative AI by Jes Fink-Jensen

How to Connect Open WebUI to an n8n AI Agent Workflow

A quick look at the Pagination algorithms

Dec 6, 2022 622 5

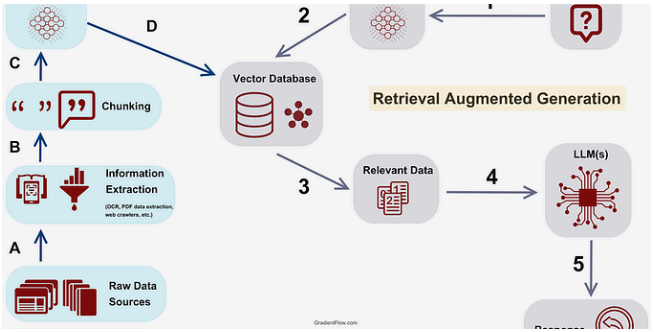
A step-by-step guide to connecting Open WebUI to a custom AI Agent Workflow in n8...

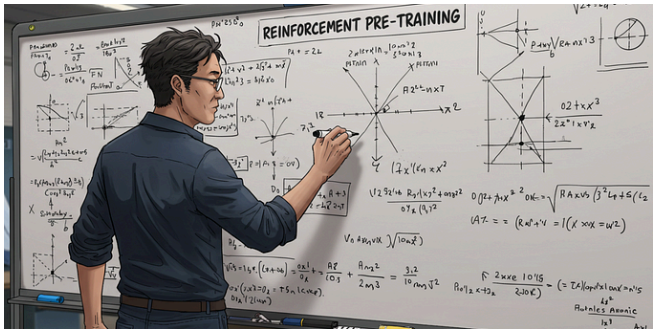
Mar 22 210 2

See all from Jes Fink-Jensen

See all from Better Programming

Recommended from Medium





In ITNEXT by Javier Ramos

You Don’t Need RAG! Build a Q&A AI Agent in 30 Minutes 🚀

Is RAG Dead? Exploring simpler AI Agents alternatives by building tools that query the...

Jun 10 1.5K 53

In AI Advances by Dr. Ashish Bamania

LLMs Can Now Be Pre-Trained Using Pure Reinforcement...

A deep dive into Reinforcement Pre-Training (RPT), a new technique introduced by...

3d ago 693 3

Sohail Saifi

Kubernetes Is Dead: Why Tech Giants Are Secretly Moving to...

I still remember that strange silence in the meeting room. Our CTO had just announced...

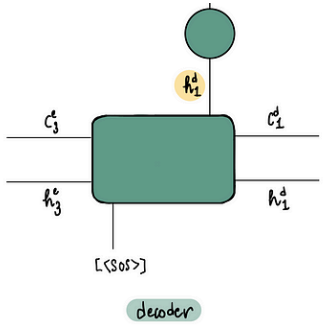
Jun 7 2.8K 115

In Generative AI by Krzyś

The Junior Developer Extinction: We’re All Building the Next...

In which we discover that the call is coming from inside the house, and we’ve been...

4d ago 666 21



In Data Science Collective by Shreya Rao

Deep Learning Illustrated, Part 7: Attention

Welcome to Part 7 of the Deep Learning Illustrated series! In this article, we'll...

6d ago 198

In JavaScript in Plain English by GeekSociety

I Replaced Cursor Agent with Claude Code, Here's What...

And why you might want to do the same (if you're a serious builder).

3d ago 363 17

See more recommendations