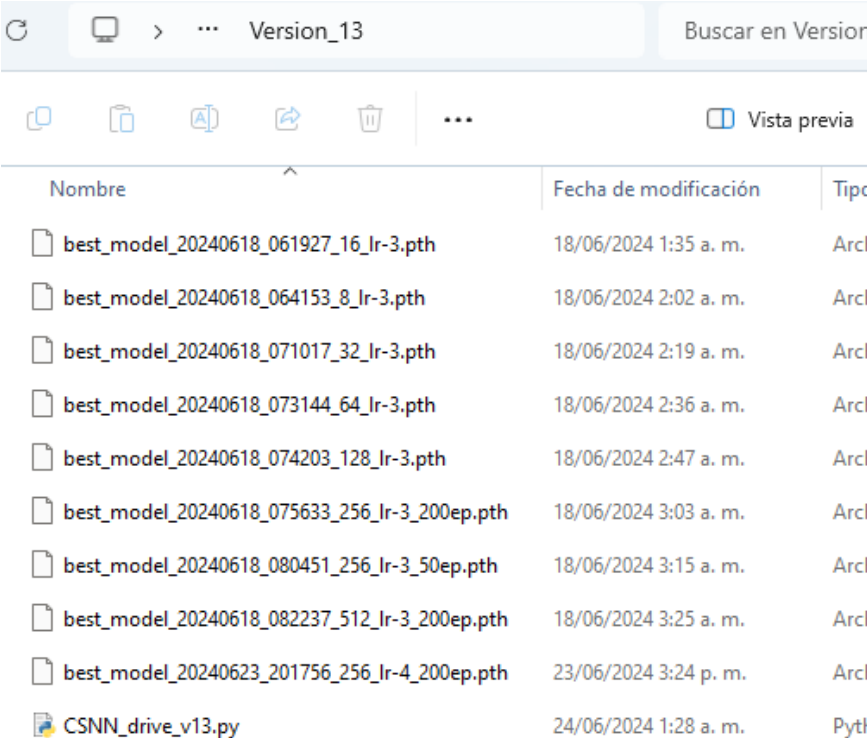


Guía para usar el modelo “pth” con Udacity

Este documento pretende ser una guía para usar el mejor modelo obtenido (archivo pth) con el simulador de Udacity.

1. Los archivos del modelo y el archivo “CSNN_drive_v13.py” se deben encontrar en el mismo directorio local



Nombre	Fecha de modificación	Tipo
best_model_20240618_061927_16_lr-3.pth	18/06/2024 1:35 a. m.	Arch
best_model_20240618_064153_8_lr-3.pth	18/06/2024 2:02 a. m.	Arch
best_model_20240618_071017_32_lr-3.pth	18/06/2024 2:19 a. m.	Arch
best_model_20240618_073144_64_lr-3.pth	18/06/2024 2:36 a. m.	Arch
best_model_20240618_074203_128_lr-3.pth	18/06/2024 2:47 a. m.	Arch
best_model_20240618_075633_256_lr-3_200ep.pth	18/06/2024 3:03 a. m.	Arch
best_model_20240618_080451_256_lr-3_50ep.pth	18/06/2024 3:15 a. m.	Arch
best_model_20240618_082237_512_lr-3_200ep.pth	18/06/2024 3:25 a. m.	Arch
best_model_20240623_201756_256_lr-4_200ep.pth	23/06/2024 3:24 p. m.	Arch
CSNN_drive_v13.py	24/06/2024 1:28 a. m.	Pyth

2. Abrir en modo de edición el archivo “CSNN_drive_v13.py” y en la sección “Inicializar el modelo” ingresar el nombre del archivo pth entre comillas simples como se presenta en la imagen y guardar cambios.

```
114 # Inicializar el modelo
115 device = 'cuda' if torch.cuda.is_available() else 'cpu'
116 model = CSNNPilotNet(beta=0.9).to(device)
117 model.load_state_dict(torch.load('best_model_20240618_075633_256_lr-3_200ep.pth', map_location=device))
118 model.eval()
119
120 # Configuración del servidor Flask y SocketIO
121 sio = socketio.Server()
122 app = Flask(__name__)
123
124 previous_image = None
125
126 # Rango de velocidad deseado
127 MIN_SPEED = 10
128 MAX_SPEED = 30
129 speed = 0
```

3. El rango de velocidad que alcanza el vehículo también se puede ajustar en la sección “Rango de velocidad deseado” (ver imagen en el punto 2) aunque la velocidad máxima que alcanza el vehículo en el simulador son 30 MPH. Si se realiza alguna modificación no olvidar guardar cambios. Probar una velocidad máxima diferente también cambia los resultados de comporta del modelo.
4. Iniciar Anaconda Prompt. Si aún no se tiene un entorno virtual de Python, se puede descargar el archivo “requirements.txt”. Navegar por el prompt hasta el directorio que contiene el archivo “requirements.txt” (se recomienda dejar en la misma carpeta con los modelos y el archivo “CSNN_drive_v13.py”) y correr las siguientes líneas de código:

```
# Crear el entorno virtual llamado "mi_entorno"
conda create --name mi_entorno
# Activar el entorno
conda activate mi_entorno
# Instalar paquetes desde el archivo requirements.txt
pip install -r requirements.txt
```

Si durante la instalación de los paquetes del archivo requirements.txt surge algún inconveniente estos se pueden instalar manualmente con el comando:

```
conda activate mi_entorno # Asegúrate de estar en el entorno correcto
pip uninstall torch # Si debes desinstalar algún paquete (ejemplo torch)
pip install torch # Instalar nuevamente el paquete
```

5. Una vez completada la instalación de los paquetes de requirements.txt. Descargar el simulador de conducción autónoma de Udacity:
<https://github.com/udacity/self-driving-car-sim>

Welcome to Udacity's Self-Driving Car Simulator

This simulator was built for [Udacity's Self-Driving Car Nanodegree](#), to teach students how to train cars how to navigate road courses using deep learning. See more [project details here](#).

All the assets in this repository require Unity. Please follow the instructions below for the full setup.

Available Game Builds (Precompiled builds of the simulator)

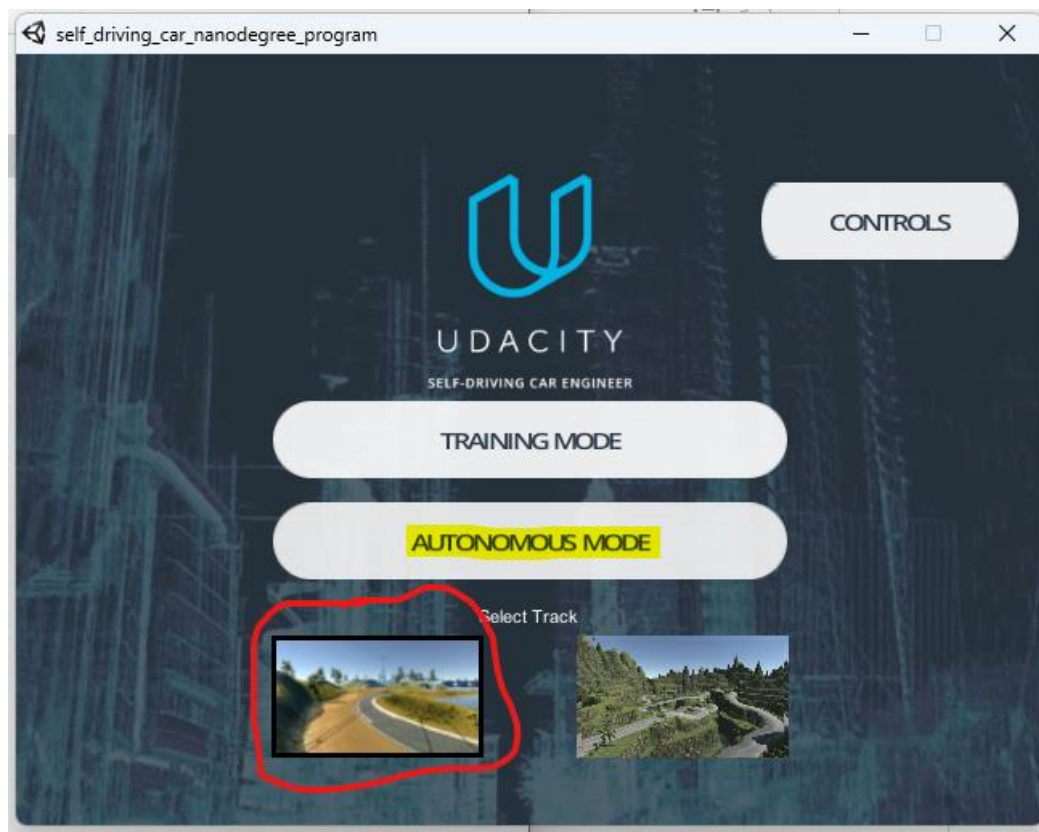
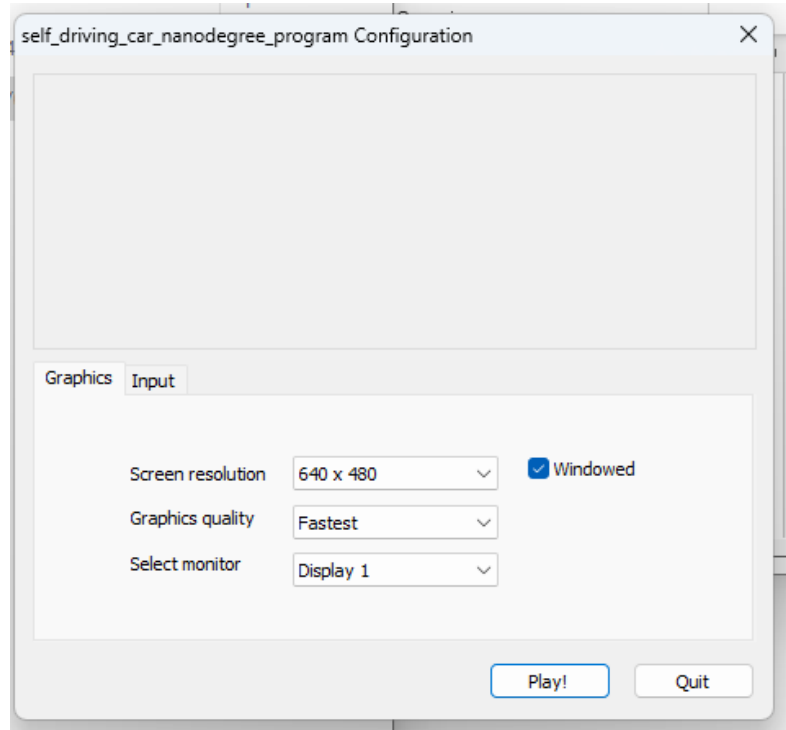
Term 1

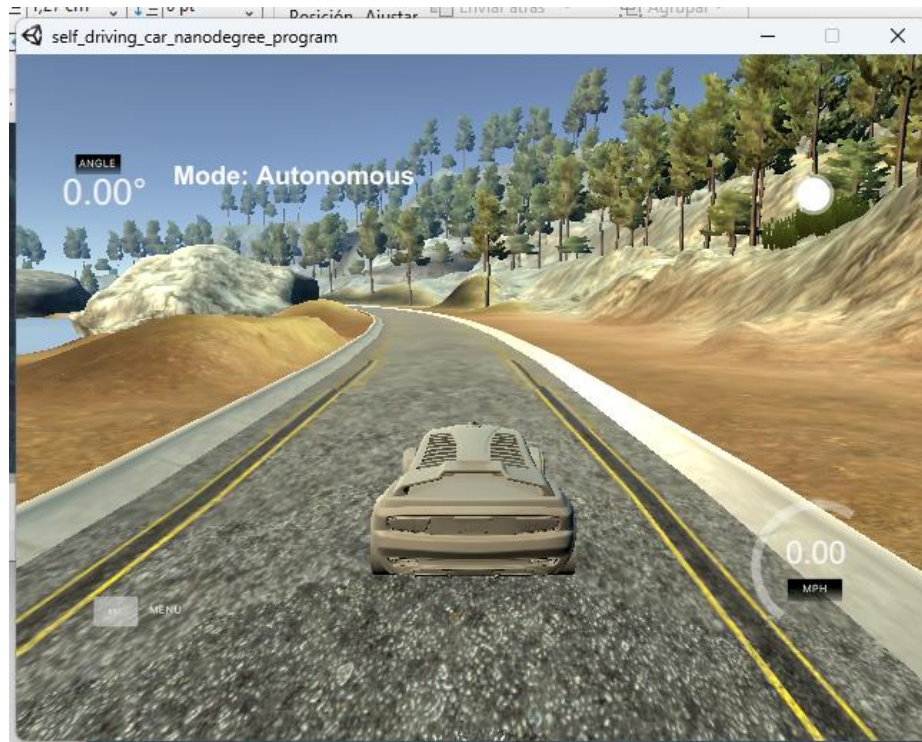
Instructions: Download the zip file, extract it and run the executable file.

Version 2, 2/07/17

[Linux](#) [Mac](#) [Windows](#)

6. Abrir el ejecutable y dar click en el botón “Play!”. Y luego seleccionar el modo “Autonomous”





- Desde Anaconda Prompt ejecutar el archivo “CSNN_drive_v13.py” con el comando:

python CSNN_drive_v13.py

El vehículo empezará a moverse

