Netlogo reinforcement-learning model for traffic simulation:   
  
This model models the movement of cars on a highway. Each car follows a simple set of rules: it slows down (decelerates) if it sees a car close ahead and speeds up (accelerates) if it doesn't see a car ahead. The model demonstrates how traffic jams can form even without any accidents, broken bridges, or overturned trucks. No "centralized cause" is needed for a traffic jam to form.

Requirements:

To run this model it is necessary to install:

* Python (version 3.12)
* Netlogo Desktop
* Tensorflow
* Keras

These dependencies can also be installed all together running “pip install -r” on the requirements file in the project folder.

How to set up:

Click on the SETUP button to set up the cars.

Set the NUMBER-OF-CARS slider to change the number of cars on the road.

Click on GO to start the cars moving. Note that they wrap around the world as they move, so the road is like a continuous loop.

The ACCELERATION slider controls the rate at which cars accelerate (speed up) when there are no cars ahead.

When a car sees another car right in front, it matches that car's speed and then slows down a bit more. How much slower it goes than the car in front of it is controlled by the DECELERATION slider.

Things to notice:

Traffic jams can start from small "seeds." These cars start with random positions and random speeds. If some cars are clustered together, they will move slowly, causing cars behind them to slow down, and a traffic jam forms.

Even though all of the cars are moving forward, the traffic jams tend to move backwards. This behavior is common in wave phenomena: the behavior of the group is often very different from the behavior of the individuals that make up the group.

The plot shows three values as the model runs:

* the fastest speed of any car (this doesn't exceed the speed limit!)
* the slowest speed of any car
* the speed of a single car (turtle 0), painted red so it can be watched.

Notice not only the maximum and minimum, but also the variability -- the "jerkiness" of one vehicle.

Notice that the default settings have cars decelerating much faster than they accelerate. This is typical of traffic flow models.

Even though both ACCELERATION and DECELERATION are very small, the cars can achieve high speeds as these values are added or subtracted at each tick.

Things to try:

In this model there are three sliders that can affect the tendency to create traffic jams: the initial NUMBER-OF-CARS, ACCELERATION, and DECELERATION.

Look for patterns in how these settings affect the traffic flow. Which variable has the greatest effect? Do the patterns make sense? Do they seem to be consistent with your driving experiences?

Set DECELERATION to zero. What happens to the flow? Gradually increase DECELERATION while the model runs. At what point does the flow "break down"?

Extending the model:

Try other rules for speeding up and slowing down. Is the rule presented here realistic? Are there other rules that are more accurate or represent better driving strategies?

In reality, different vehicles may follow different rules. Try giving different rules or ACCELERATION/DECELERATION values to some of the cars. Can one bad driver mess things up?

The asymmetry between acceleration and deceleration is a simplified representation of different driving habits and response times. Can you explicitly encode these into the model?

What could you change to minimize the chances of traffic jams forming?

What could you change to make traffic jams move forward rather than backward?

Make a model of two-lane traffic.

Netlogo features:

The plot shows both global values and the value for a single car, which helps one watch overall patterns and individual behavior at the same time. The `watch` command is used to make it easier to focus on the red car. The `speed-limit` and `speed-min` variables are set to constant values.