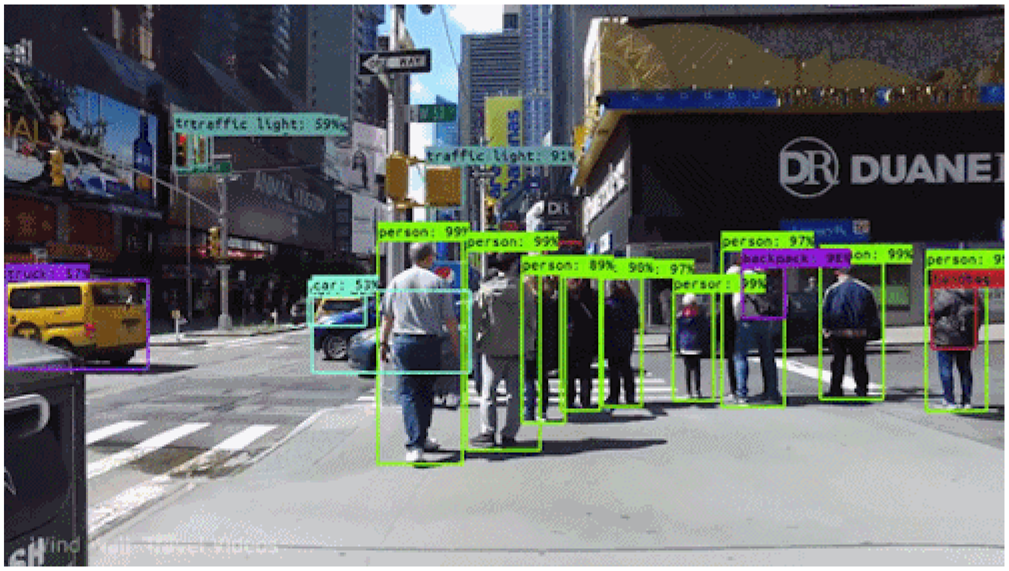
The Data Science Behind Self-Driving Cars

medium.com/@feiqi9047/the-data-science-behind-self-driving-cars-eb7d0579c80b

Fei Qi

April 22, 2019



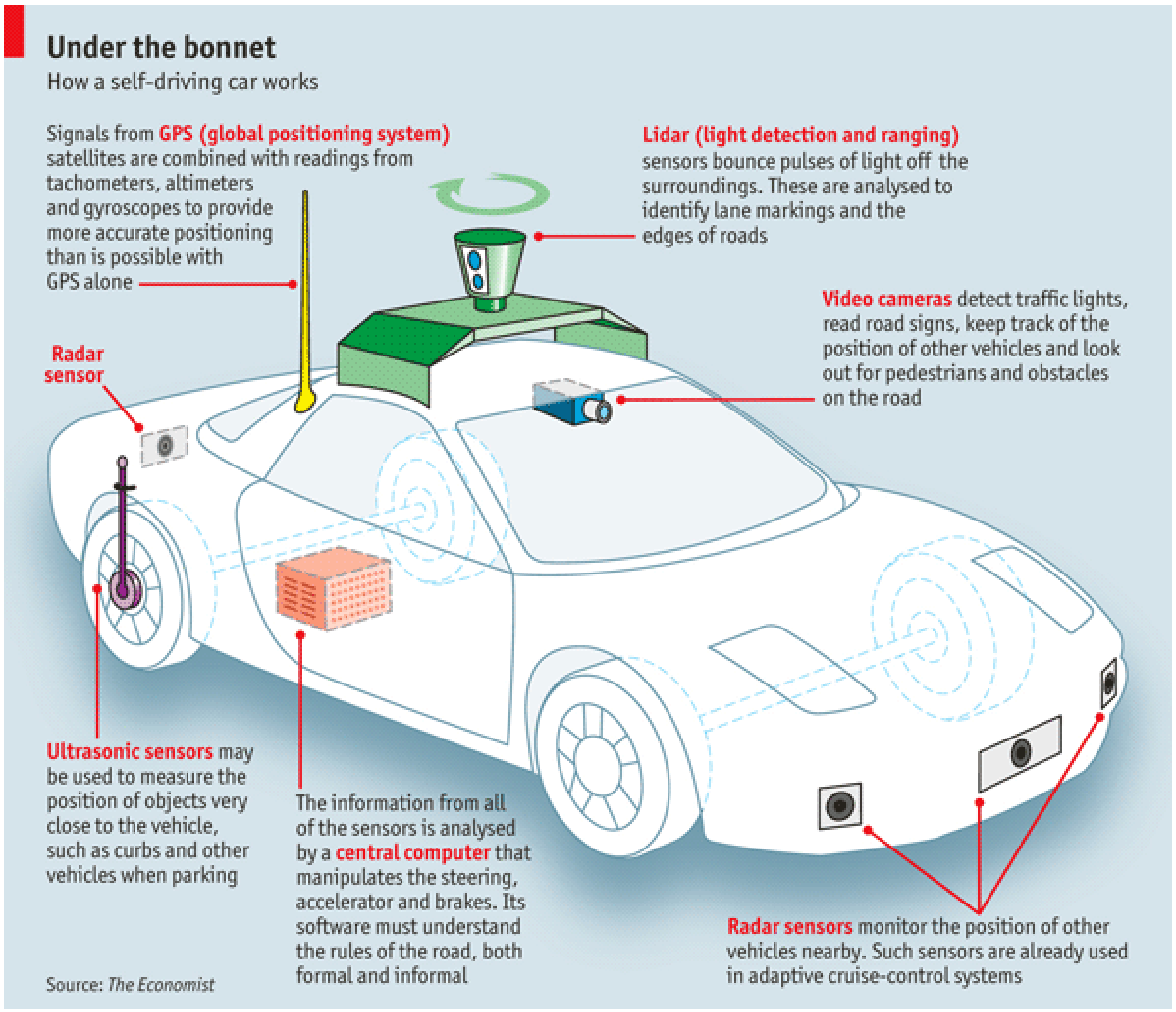
> **A Decade Ago, it was Science Fiction…**

As with most scientific discoveries, it all started with the military. The US military-industrial complex has been funneling investments into making unmanned trucks for year. In 2004, it decided to hold a million-dollar competition, inviting the world to build a robot that could drive across California's Mojave Desert. Outcomes weren't great, the best performance was only seven miles. But the aftereffect was the birth of the self-driving car.

In 2009, Google launched its self-driving car project. Within 18 months, they developed a system that could maneuver through some of California's trickiest roads without human interference. Then, Tesla joined the race by building automating features into its vehicles, such as blind spot detection, auto braking, and lane departure warnings. Uber and Lyft also began investing heavily in making ride-sharing an autonomous task. Soon after, all major automakers followed suit, as well as hundreds of start-ups sprouting up to offer improved autonomous features to help these giants in their endeavors.

Intel and Strategy Analytics estimate that the global economy will see a $7 trillion boost from this arising industry ($2 trillion for the US alone), and that the technology will save approximately 600,000 lives by 2045. The caveat here is that some 5 million truckers, cabbies, and other drivers will be put out of work.

How are Humans Replaced?

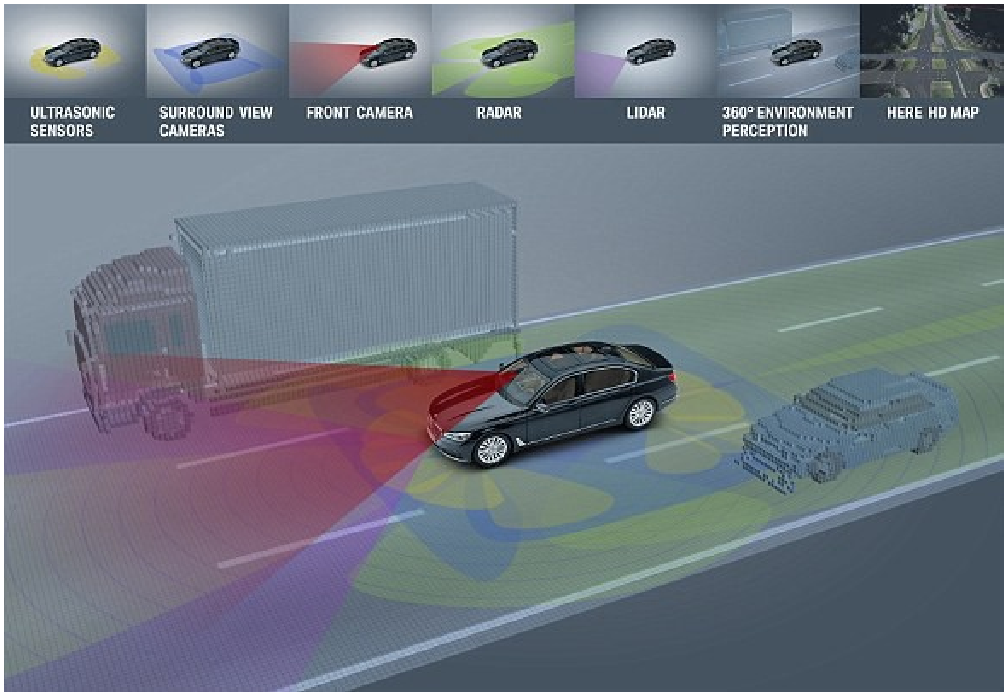

Under the bonnet
How a self-driving car works
Signals from GPS (global positioning system) satellites are combined with readings from tachometers, altimeters and gyroscopes to provide more accurate positioning than is possible with GPS alone
Lidar (light detection and ranging) sensors bounce pulses of light off the surroundings. These are analysed to identify lane markings and the edges of roads
Video cameras detect traffic lights, read road signs, keep track of the position of other vehicles and look out for pedestrians and obstacles on the road
Radar sensor
Ultrasonic sensors may be used to measure the position of objects very close to the vehicle, such as curbs and other vehicles when parking
The information from all of the sensors is analysed by a central computer that manipulates the steering, accelerator and brakes. Its software must understand the rules of the road, both formal and informal
Radar sensors monitor the position of other vehicles nearby. Such sensors are already used in adaptive cruise-control systems
Source: The Economist

To simulate the human brain and its cognitive networks, a basic self-driving car must be equipped with:

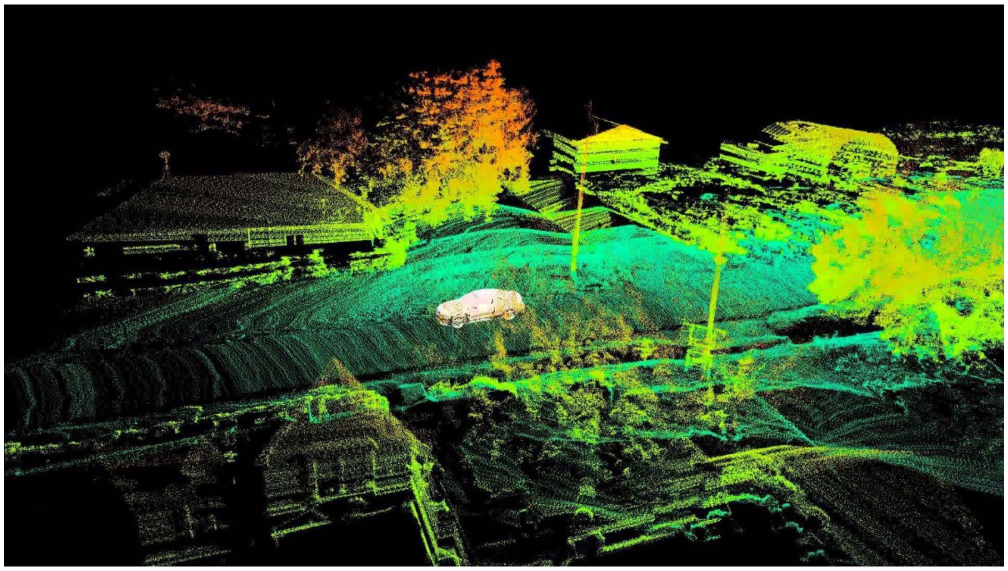
- Highly detailed maps of street features (lights, signs, curbs, etc.)
- Sensors such as cameras and LIDAR (similar to radar but uses light to create pulses instead of radio waves) short-distance 3D layout of its surroundings in real-time
- Vehicle-to-vehicle cloud communications
- Sensory inputs into the vehicle’s machine learning algorithms, to predict outcomes based on an enormous volume of data, in order to plan and act

The first three features on the list are already commercially available in many existing models. Sensors provide detailed 3D maps of surrounding environments, read signs, identify pedestrians, assist in parking, to name a few. Altimeters and Accelerometers provide more accurate positioning than GPS alone. And the more trips a car makes, the data is collected to update its knowledge of the map and environment. It is estimated that a single self-driving car could collect up to 1GB of data per second (think about the number of sensors and the instantaneous and constant need to transmit information).

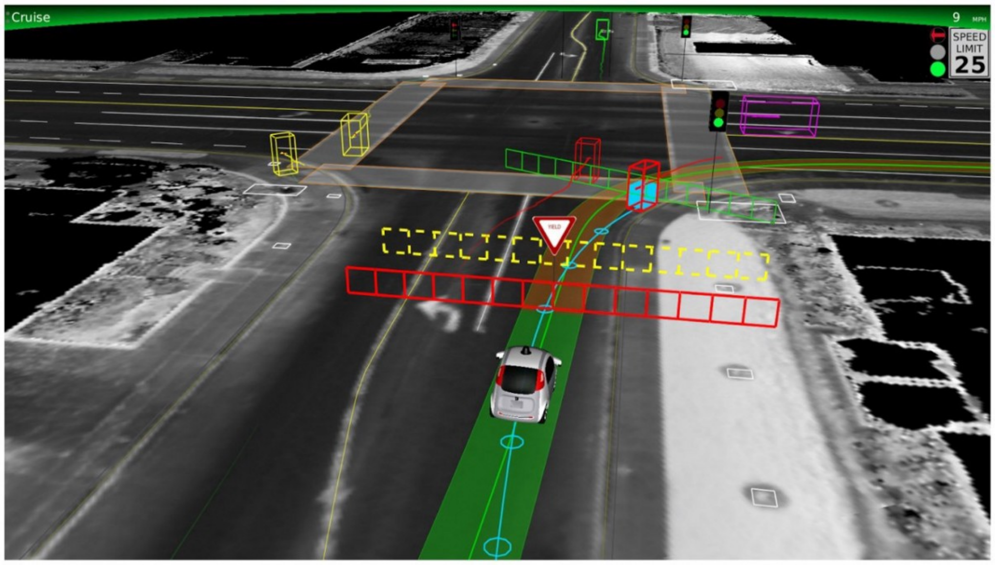
The Body and the Brain



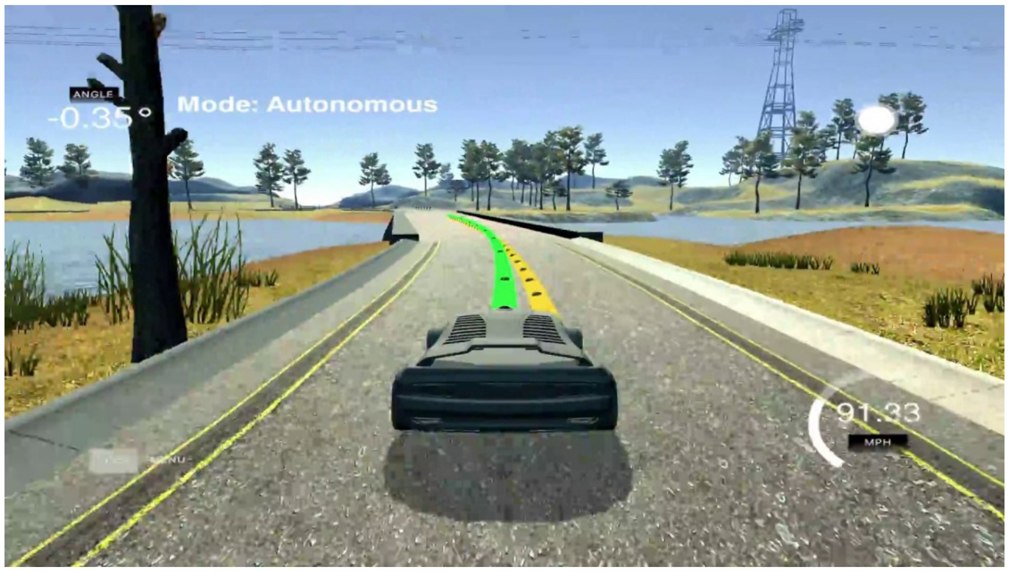
Perception



Localization



Planning



Control

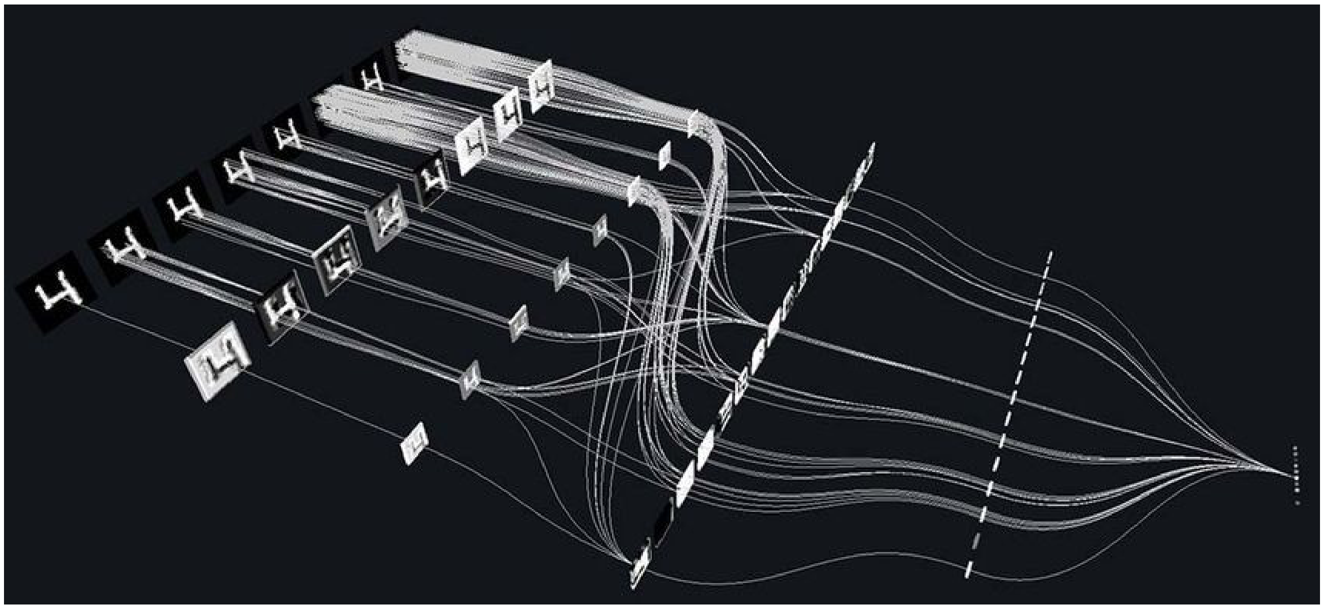
Data scientists are the pioneers behind perfecting the brain of the beast (driverless cars). We must somehow figure out how to develop algorithms that master ***Perception, Localization, Prediction, Planning, and Control*****.** How do we distinguish the difference between a stray shopping cart and a person in a wheelchair? How do we know which direction the person on the bicycle will turn? How do we differentiate between normal and abnormal situations, and when do we decide to break the law in order to save a life?

> "Perception merges several different sensors to know where the road is and what is the state (type, position, speed) of each obstacle. Localization uses very specific maps and sensors to understand where the car is in its environment at centimeter level. Prediction allows the car to anticipate the behavior of objects in its surrounding. Planning uses the knowledge of the car's position and obstacles to plan routes to a destination. The application of the law is coded here and the algorithms define waypoints. Control is to develop algorithms to follow the waypoints efficiently."

Using Machine Learning and Deep Learning, and AI, it really is an iterative process where the cycle of collecting data to interpreting the data to training the algorithm is endless. In other words, we must be able to take real-life driving experiences, turn them into programmable information, and train our models to continuously and self-sufficiently improve its understanding of the real world in order to make 'informed' decisions.

Perception and Localization

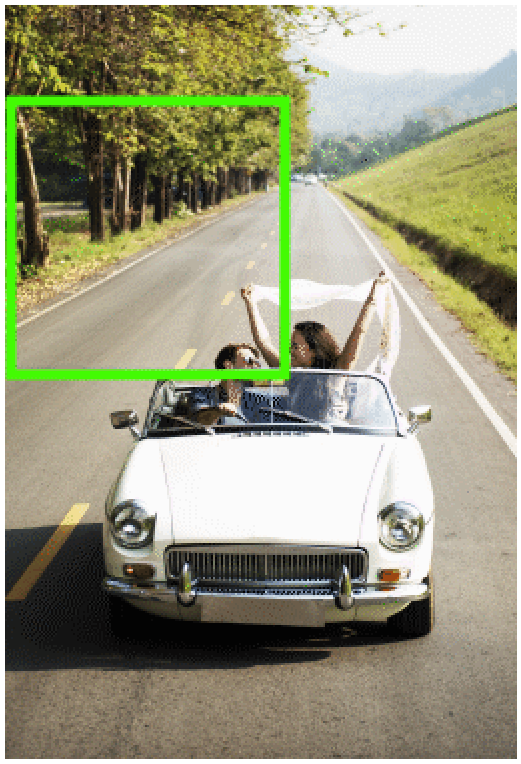
For a machine to see, it takes two steps: image classification and image localization. Image classification is determining what it is seeing. Image localization is knowing where it is seeing the object. For perform these two steps, we need to use a ***Convolutional Neural Network (CNN).*** Training the CNN will allow it to perform convolution operations on images in order to classify them and locate them.

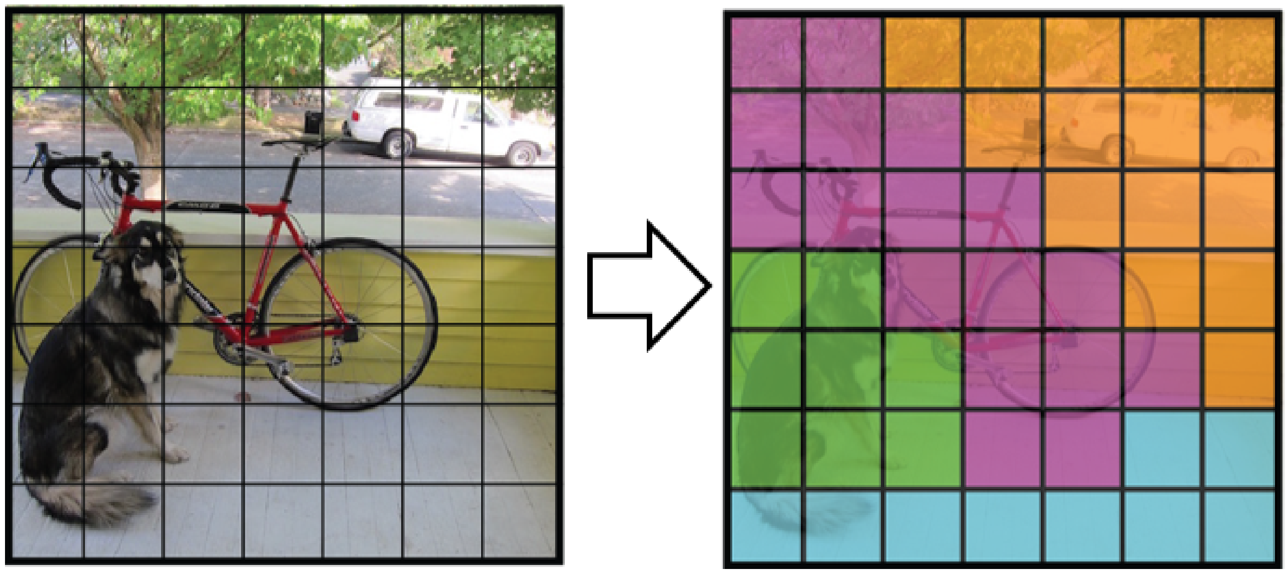


What a *Convolutional Neural Network Looks Like*

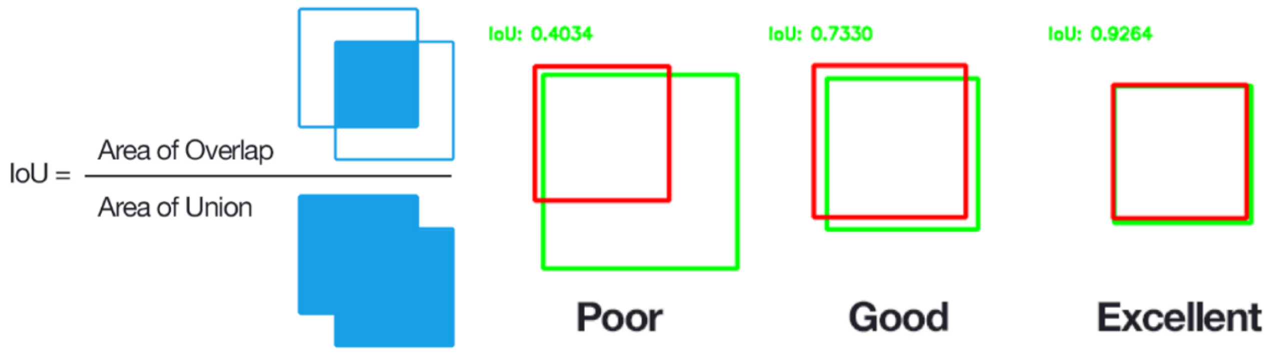
The problem with such CNN is that it can only pick up a single portion of the image at a time to perform classification on. What if there are multiple things in one image? So we write a "sliding-windows" algorithm that allows it to slide across the image to take in all portions of it. Then, each portion is run through the CNN to see if it correspond to any possible object. For anything other than a person or detectable object, the CNN will return "True", otherwise "False".

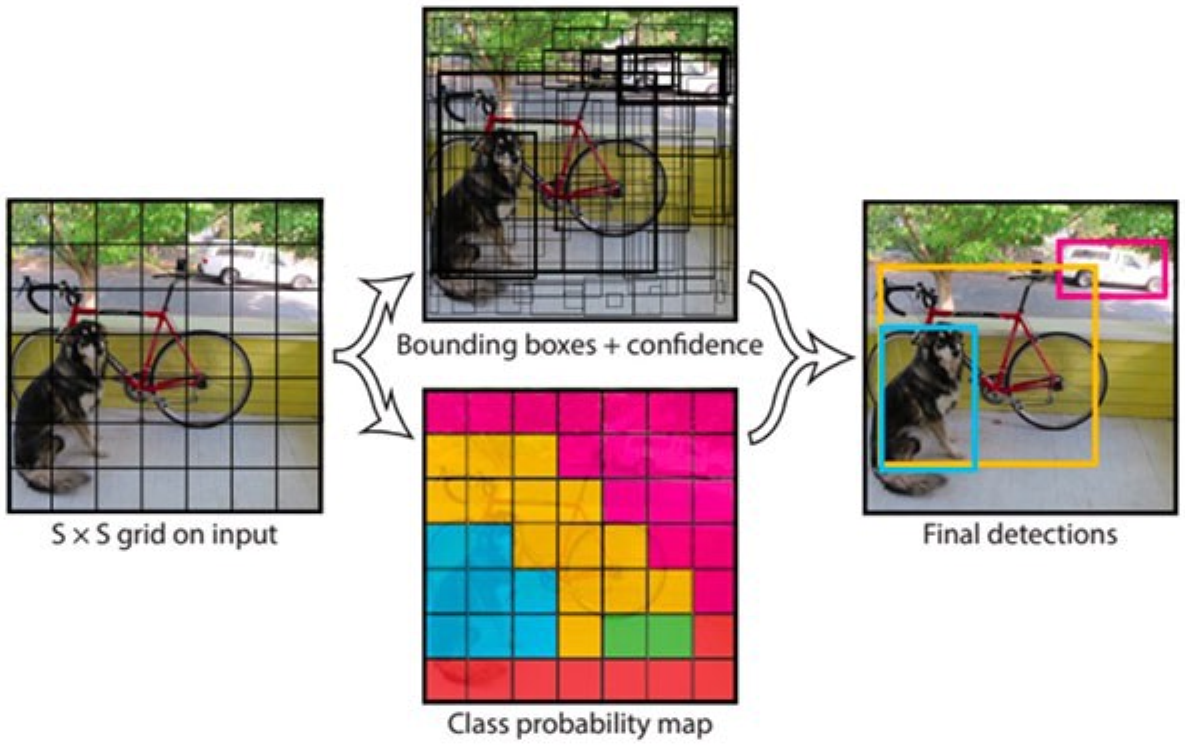
But what if the object is much bigger or smaller than the image itself? We can break the image down into grids and run the entire image through CNN. The algorithm used for this is called YOLO ("You Only Look Once"). The resulting output is a Class Probability Map, which spits out the probabilities for each grid cell being a specific object. YOLO is an enhancement of "sliding-windows" since it returns the predictions of small portions of the entire image instead of multiple run-throughs.


Sliding-Windows Algorithm


YOLO Algorithm

Given that we know whether an object is present in each grid, how do we know where it is within the grid? We use a technique called Kalman filters to find their position with the highest possible accuracy. We can use a "non-max suppression" algorithm to train the CNN. We compare the results from the CNN for each grid to the actual grid. However, we also want to consider if parts of the same object is in multiple grids, and we don't want to count the same object multiple times. Non-max suppression is a way for us to make sure that our algorithm detects each object only once. What non-max suppression does is to clean up multiple detections of one object. We define a cost function as the area of intersection / area of union of two grids. The closer the function is to 1, the better our prediction. First, we take the grid with the highest IoU (most confident detection within this grid). Then, for the remaining grids that overlap and has high IoU, we suppress them so they are not detected for the same object.





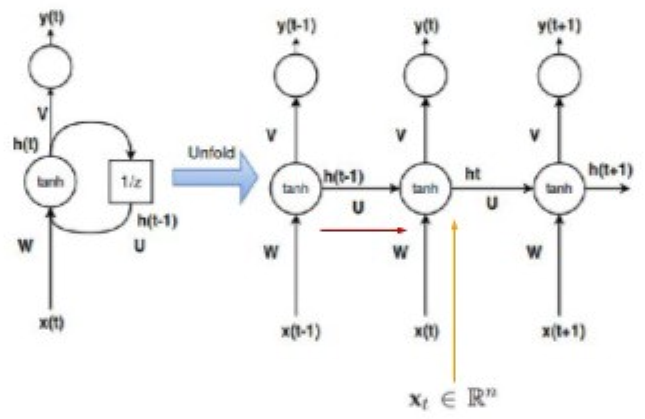
Prediction, Planning, and Control

Given perception and localization, cars can now predict the behavior of every object (vehicle or human) in their surroundings. They can anticipate how the object moves, in which direction, at which speed, what trajectory they will follow. What is commonly used for this is the ***Recurrent Neural Network (RNN)***, where RNN can learn from past behavior and forecast the future.

Recurrent Neural Network

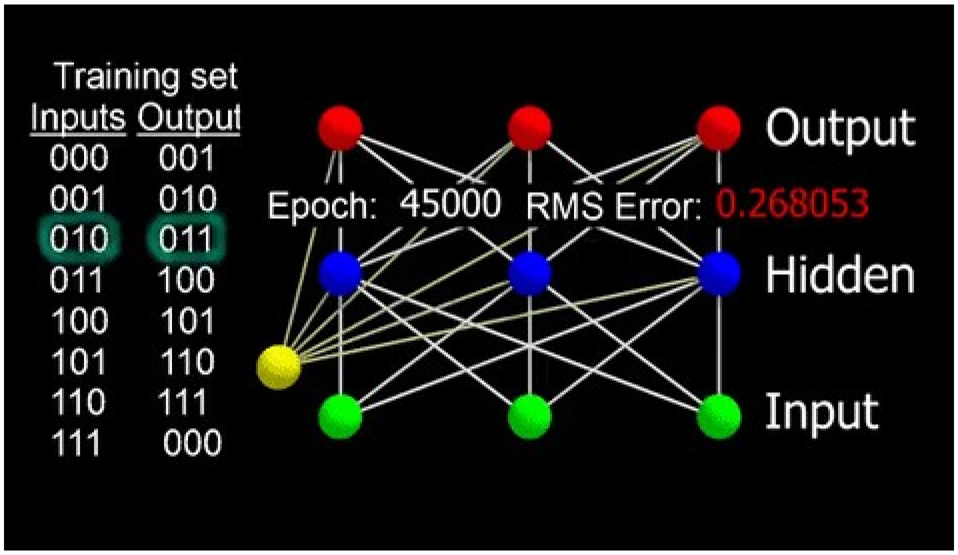
Hence we have two data flows: **Forward in layers + time** propagation

- Last time-step includes the context of our decisions recursively



$$h_t = g(W \cdot x_t + U \cdot h_{t-1} + b_h)$$

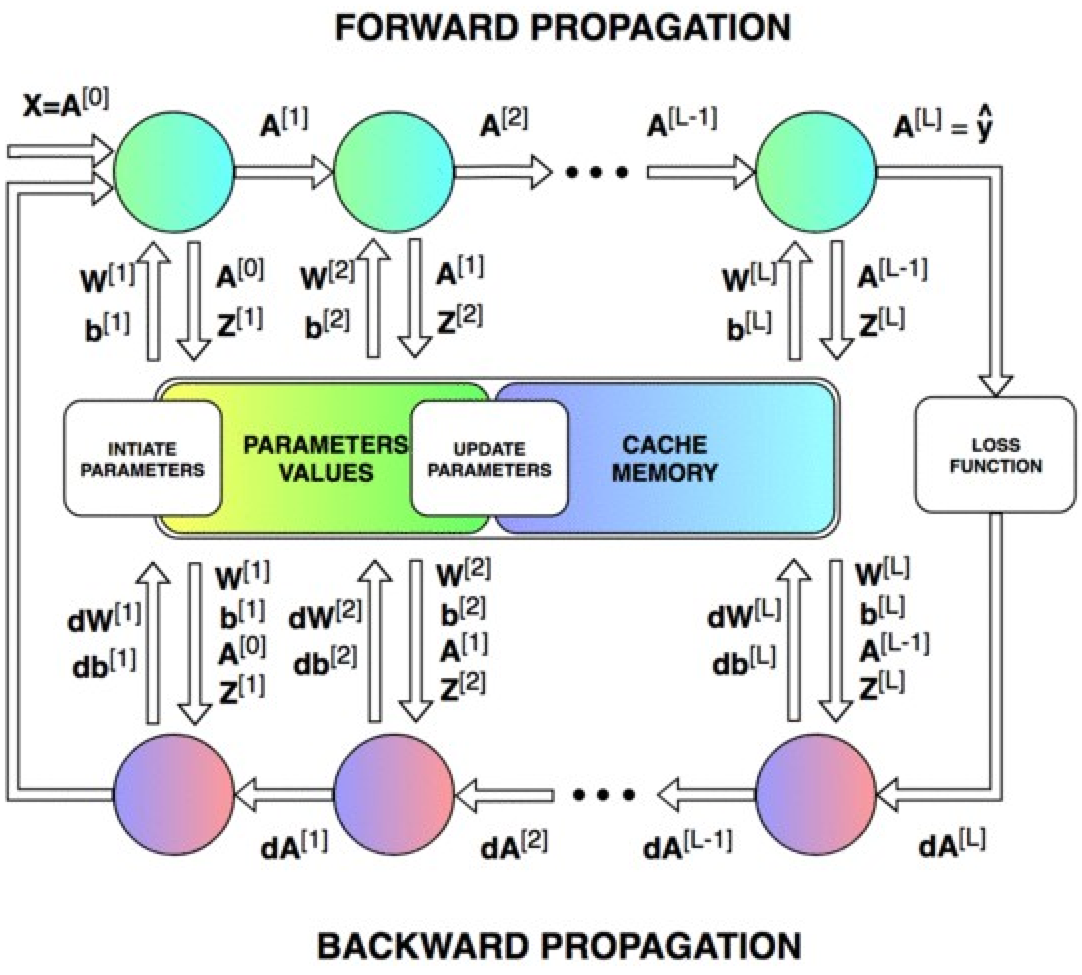
Stacked convolution operations uses back-propagation to minimize errors by replacing human fine-tuning with an algorithm that implements the chain rule of calculus. To train our model to essentially clone human behavior, we use **Behavioral Cloning** to capture human sub-cognitive skills and reproduce them in a computer program. Given the data collected, the RNN uses back-propagation algorithm to reduce error and update parameters.



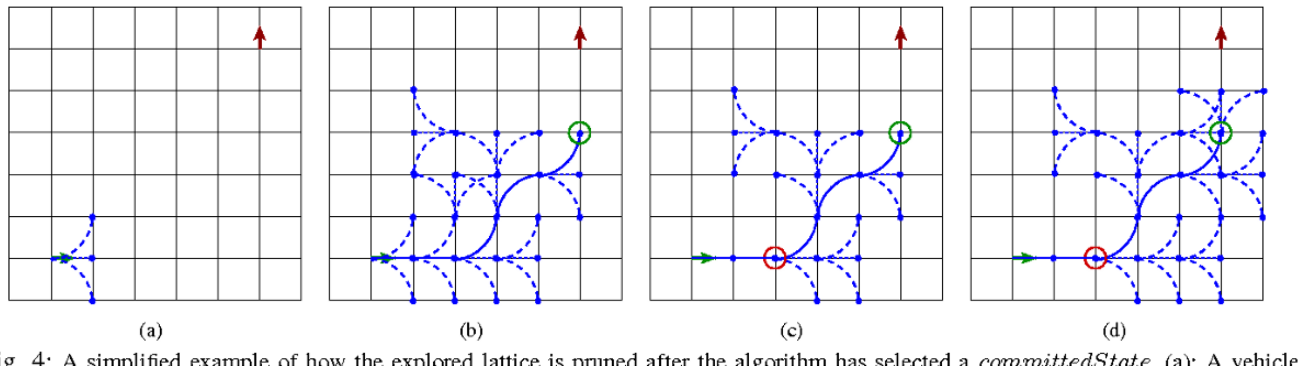
Training set

Inputs	Output
000	001
001	010
010	011
011	100
100	101
101	110
110	111
111	000

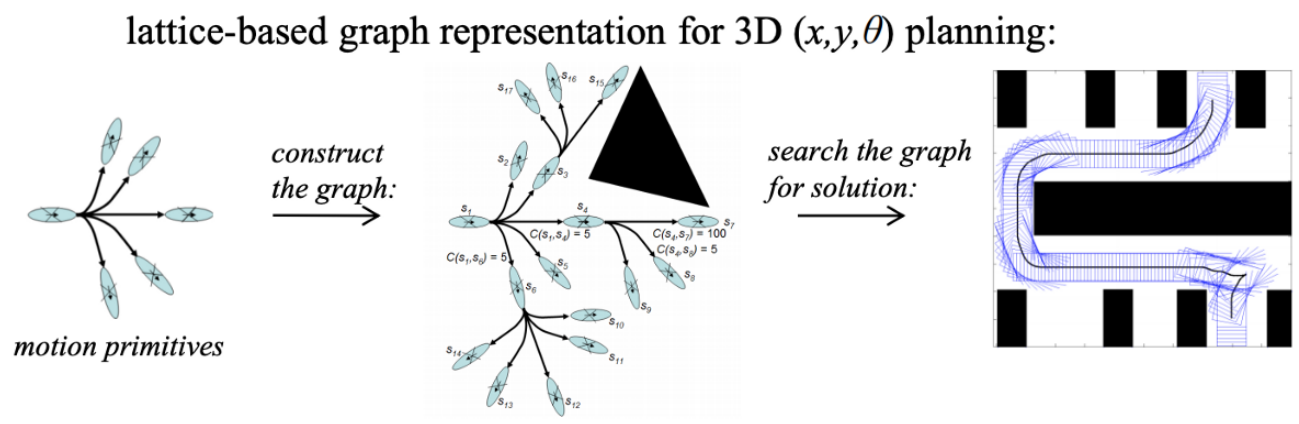
Epoch: 45000 RMS Error: 0.268053



Planning is the route to follow or generating trajectory involves search algorithms such as A*, Lattice Planning, and Reinforcement Learning. A* algorithm is one of the best technique used in path-finding and graph traversals. The algorithm works by finding the Euclidean Distance between each obstacle to generate the shortest path between starting and ending points. Lattice Planning demonstrated below:


(a)
(b)
(c)
(d)
Fig. 4: A simplified example of how the explored lattice is pruned after the algorithm has selected a committedState. (a): A vehicle

Lattice Planning



Finally, the last step is delegated to control engineers. They use the trajectory generated in the planning step to change accordingly the steering, acceleration and breaks of the car. The most common method is PID Control but there are a few others such as Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC) (but that's outside the scope of this article).

References:

https://towardsdatascience.com/self-driving-cars-the-guide-f1f427b9656b

https://towardsdatascience.com/how-do-self-driving-cars-see-13054aee2503

https://medium.com/@ODSC/self-driving-cars-generated-news-among-top-october-data-scienceresearch-f6bb04e4e573

https://www.wired.com/2017/06/impact-of-autonomous-vehicles/

https://www.wired.com/story/guide-self-driving-cars/

https://www.fi.edu/science-of-selfdriving-cars

https://www.smartdatacollective.com/data-key-autonomous-vehicle-technology-tesla-says-winning/

https://dataconomy.com/2015/12/how-data-science-is-driving-the-driverless-car/

https://blog.exxactcorp.com/atrous-convolutions-u-net-architectures-for-deep-learning-a-brief-history/

https://towardsdatascience.com/reinforcement-learning-from-grid-world-to-self-driving-cars-52bd3e647bc4

https://sergioskar.github.io/Self_driving_cars/