Improving The Throttle Control Algorithm of Reusable Rocket Engines Using Deep Learning Algorithm

Jessica Xu
Period 7
11/14/2019

1. Introduction

1.1 Problem

Modern commercial rockets are multi-stage rockets; each multi-stage rocket consists of two or more units (engines), stacked one on top of another and fired in succession. Each unit is called a stage. It can be throttled by controlling the propellant combustion rate (usually measured in kg/s or lb/s). the propellant flow entering the chamber is controlled using valves.

Currently, multi-stage rockets are not reusable and must be thrown away after being launched into space, so companies must spend millions of dollars on manufacturing brand new rockets for every single launch. It costs $60 million to make the Falcon 9 - a multi-stage rocket, and $200,000 to fuel it, according to SpaceX CEO Elon Musk.[1]

In 2017, SpaceX company successfully developed a reusable launch system technology which was able to reuse the first stage of multi-stage rockets. But this technology doesn't work for the second stage of multi-stage rockets.[2]

In order to make each stage of multi-stage rocket reusable, SpaceX company must resolve this challenging problem - "how to make the throttle control system of rocket engines react more quickly and precisely to achieve zero velocity at the same time the rocket reaches the ground".[3] It will reduce the cost of access to space dramatically if one can figure out how to resolve this issue.

1.2 Origin of Project Idea

Recently there has been significant breakthroughs in Deep Learning. Deep Learning is a branch of Artificial Intelligence and is especially useful for solving the problems like recognizing patterns ,series predictions, classification and data mining etc. It has been successfully applied for building self-driving car.[4][5]

If you consider a rocket as a flying vehicle, then the Deep Learning algorithms applied to self-driving car may also be used in the throttle control system of reusable rocket.

1.3 Hypothesis

After the first and second stages of a multi-stage rocket successfully are separated from the multi-stage rocket, they will start to fall in the order. As they fall, each of them accelerates. The landing of the first and second stages are so fast that no human pilot is able to react quickly enough to guide and ensure a smooth touchdown of each stage. Also, there is a limited amount of fuel in each stage. Once the fuel is exhausted, the stage will be out of control and fall down.

Due to unpredictable effects of the environment, precise trajectory of each stage can’t be worked out prior to launch. It means there is no way to pre-train the throttle control system of the reusable rocket and teach it how to soft-land on ground.

In recent years, Deep Learning, called Modern Artificial Intelligence, has made great breakthroughs in areas such as computer vision algorithm, natural language translation, health diagnostics, factory automation etc.

It may be possible to improve the performance of the throttle control algorithm of reusable rocket engines using Deep Learning algorithm - **Neural Networks coupled with Genetic Algorithms** and let it to pilot the landing of the first and second stages of multi-stage rockets.

Neural Networks are mathematical models inspired by information processing and distributed communication nodes in biological systems. The concept of Neural Network is the same as that of Deep Learning. The difference between Neural Networks and Deep Learning lies in the depth of the model. Deep Learning is used to describe complex Neural Networks.

According to Wikipedia, a genetic algorithm (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms. Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems.

The main reason why I decided to use **Neural Networks coupled with Genetic Algorithms** is that to make Neural Networks perform properly, we have to configure a set of parameters and set the right values for these parameters. Genetic Algorithms can quickly learn the best parameters for Neural Networks

Another advantage of **Neural Networks coupled with Genetic Algorithms** is that it doesn’t always need training data to be created ahead of time. it doesn’t need to make any assumption about how it should pilot each stage of reusable rocket. It looks like this Algorithm perfectly fit in the use case of soft landing of the first and second stages. It can learn everything on its own and decide when to fire the rocket thrusters or not to fire the rocket thrusters.

2. Background Information

2.1 History of Reusable Rocket

With the invention of rocket propulsion in the 1950s, space travel became a technical possibility. The early rockets were single-stage and developed to deliver weapons, making reuse impossible by design.

In 1981 NASA scientists overcome the problem of mass efficiency by using multiple expendable stages in a vertical-launch multistage rocket.

In early 2000s, U.S and British attempted to build reusable rockets. But they all failed because of rising cost and severe technical issues.

In 2012, SpaceX started a flight test program with experimental vehicles. These subsequently led to the development of the Falcon 9 reusable rocket launcher.[6]

SpaceX achieved the first vertical soft landing of the first stage of a reusable rocket on December 21, 2015.

As of August 2019, the only reusable operational orbital boosters are Falcon 9 and Falcon Heavy.[7]

2.2 Throttle Control Algorithm of Reusable Rocket Engine

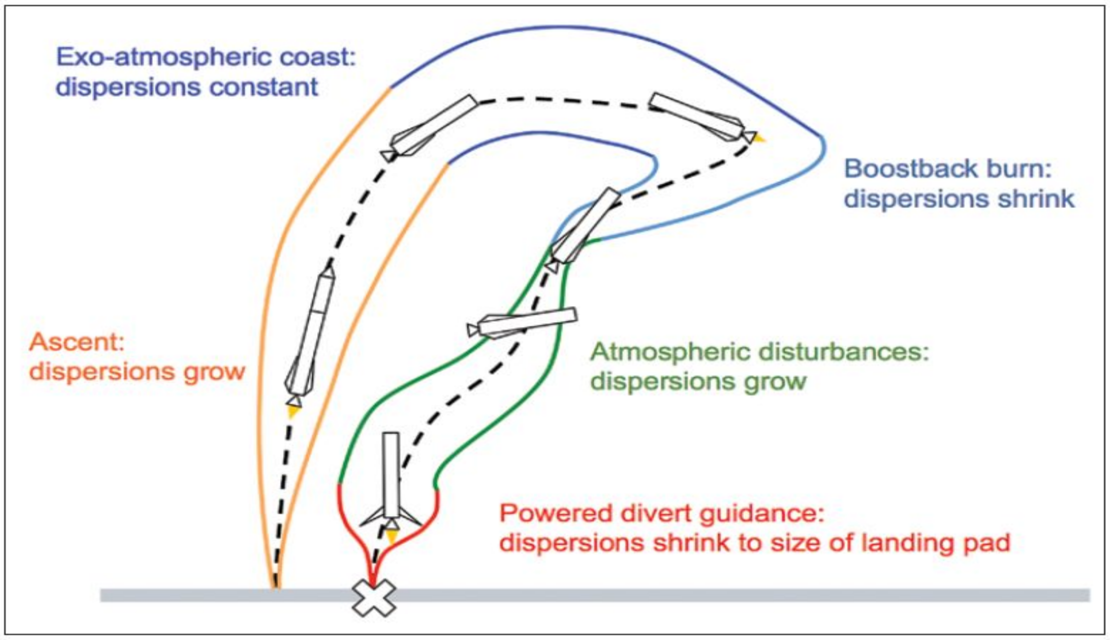
Friction with the atmosphere as a rocket falling back down can create an ionization field (charged field) capable of blocking radio signals. In this situation, it's very difficult for human pilots to communicate with falling stage of reusable rocket and pilot the landing of any reusable rocket.

In 2015, SpaceX successfully resolved the soft landing issue of the first stage of Falcon 9 reusable rocket with a computer algorithm called convex optimization algorithm.

Convex optimization algorithm involves solving a "convex optimization problem," in modern machine learning. It involves considering all the possible answers to the question of "what's the best way to get from here to the landing target without running out of fuel" as a geometric shape, and uses mathematical tools of game theory, developed first by John von Neumann, to quickly choose the best way down from that set. Convex optimization algorithm was first developed by Lars Blackmore and his colleagues in that 2009 paper on Mars landings. They both received a patent on their ideas in 2013.

Convex optimization algorithm uses mathematical reasoning to determine the best way to land. It is capable of calculating the optimal path down to the target on the ground before the first stage of reusable rocket either running out of fuel or crashing into Earth. It requires the use of computer vision to actually see and reason the best way to land. This algorithm also help calculate the thrust force required to stay on that trajectory, then it pilots the throttle control system to achieve that thrust force.[8]

However, convex optimization algorithm has some performance issues that the throttle control system doesn't react quick enough to pilot the landing process. SpaceX has been working on this issue and trying to resolve it.[9]



2.3 Brief Introduction to Deep Learning Neural Network

Human brains are made up of connected networks of neurons. Deep Learning Neural Networks seek to simulate these networks and get computers to act like interconnected brain cells, so that they can learn and make decisions in a more human like manner.

Different parts of the human brain are responsible for processing different pieces of information, and these parts of the brain are arranged in layers. In this way, as information comes into the brain, each level of neurons processes the information, provides insight, and passes the information to the next, more senior layer. It's this layered approach to processing information and making decisions that Deep Learning Networks are trying to simulate.

Deep Learning Neural Network can have only three layers of neurons: the input layer (where the data enters the system), the hidden layer (where the information is processed) and the output layer (where the system decides what to do based on the data). But Deep Learning Neural Networks can get much more complex than that, and include multiple hidden layers. Whether it's three layers or more, information flows from one layer to another, just like in the human brain[10].

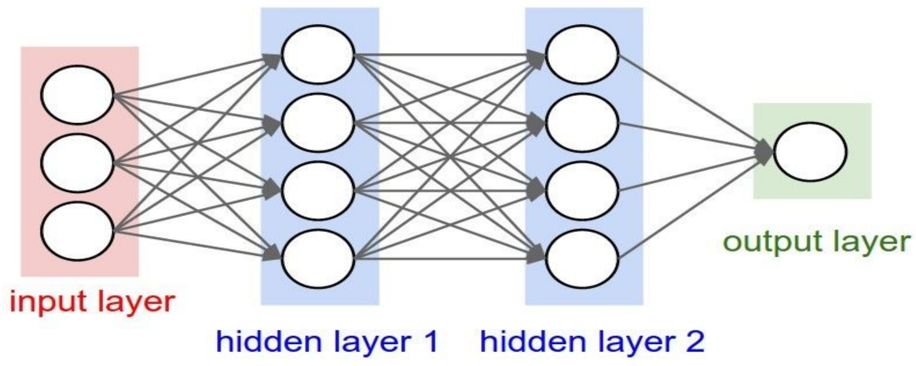


Figure 1: An artificial neural network [11]

The learning process of Deep Learning Neural Network follows the below three steps:

1. Collect an appropriate training data
2. Set up a neural net architecture — that is, a complicated system of equations, loosely modeled on the brain .
3. Repeatedly feed the data through the neural net; at each iteration comparing the result of the neural net's prediction to the correct result, and adjusting each of the neural net's parameters based on how much and in what direction it misses.

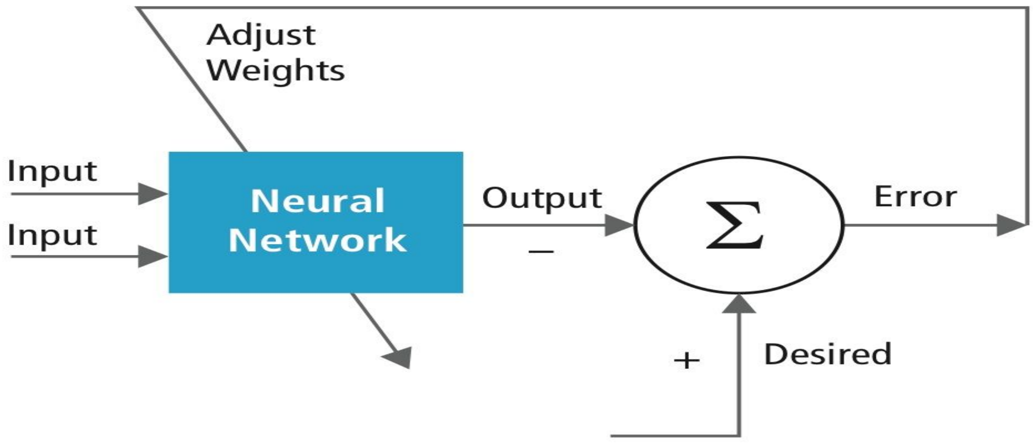


Figure 2: Training of neural networks [11]

3. Review of Literature

3.1 Recent Advances in Reusable Rocket

There are two very important capabilities to enable first and second stages of a reusable rocket to achieve soft landing.

1. The navigation system must be capable of accurately estimating the stage's state during the powered descent phase.

2. The stage's throttle control system must be capable of using these state parameters to quickly decide when to fire the rocket thrusters or not to fire the rocket thrusters to achieve soft-landing which we define as the magnitude of the landing velocity below 2 m/s.

Currently the most successful reusable rocket is Falcon 9 reusable rocket developed by SpaceX. The throttle control system of its first stage uses Convex optimization algorithm. However, convex optimization algorithm has some performance issues that the throttle control system doesn't react quick enough to pilot the landing process[9].

In the past few years, the research development in this area made some good progress. The scientists proposed a novel guidance algorithm based on convex optimization, pseudospectral discretization, and a model predictive control framework. This approach was to solve the performance problem of throttle control system by implementing online trajectory optimization in a receding-horizon manner and feeding the rocket with the most recently updated optimal control commands. A pseudospectral-improved successive convexification algorithm is adopted to solve the trajectory optimization problem due to its high solution accuracy and computation speed. Numerical experiments were conducted to demonstrate the effectiveness of the proposed algorithm[12].

More recently, several theoretical developments have given rise to the use of Deep Reinforcement Learning Algorithm for reusable launch vehicles. Deep Reinforcement Learning is the concept that optimal behaviour is reinforced by a positive reward. It is very similar to toddlers learning how to walk who adjust actions based on the outcomes they experience such as taking a smaller step if the previous broad step made them fall.

In August 2018, two scientists wrote a paper that suggested to use Convolutional Neural Networks (CNN) and Recurrent Neural Net-works (RNN) to predict the fuel-optimal control actions to perform autonomous Moon landing. This approach uses only raw images taken by on board optimal cameras. Such an approach can be employed to directly select actions without the need of direct filters for state estimation[13].

In Oct. 2018, several scientists proposed a deep reinforcement learning approach for reusable launch vehicle soft landing. In this paper, they presented a novel integrated guidance and control algorithm to learn a policy mapping the lander's estimated state directly to a commanded thrust for each engine, with the policy resulting in accurate and fuel-efficient trajectories[14].

As of now the above mentioned Deep Learning related research works were conducted through computer simulations and haven't applied to the real throttle control system of reusable rockets.

4. Using Genetic Algorithms and Neural Networks for Precision Throttling of reusable rocket engine

4.1 What is Genetic Algorithm?

Genetic Algorithms(GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics. These are intelligent exploitation of random search provided with historical data to direct the search into the region of better performance in solution space. They are commonly used to generate high-quality solutions for optimization problems and search problems.

4.2 What is Neural networks?

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

Neural networks help us cluster and classify. You can think of them as a clustering and classification layer on top of the data you store and manage. They help to group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to train on. (Neural networks can also extract features that are fed to other algorithms for clustering and classification; so you can think of deep neural networks as components of larger machine-learning applications involving algorithms for reinforcement learning, classification and regression.

4.3 Why Use both Genetic Algorithms and Neural Networks?

Neural networks are flexible and can be used for both regression and classification problems. Any data which can be made numeric can be used in the model, as neural network is a mathematical model with approximation functions. Neural networks are good to model with nonlinear data with large number of inputs. It is reliable in an approach of tasks involving many features. It works by splitting the problem of classification into a layered network of simpler elements. Once trained, the predictions are pretty fast. Neural networks can be trained with any number of inputs and layers. Neural networks work best with more data points.

Neural Networks have helped us solve so many problems. But there's a huge problem that they still have. Neural networks are black boxes, meaning we cannot know how much each independent variable is influencing the dependent variables. It is computationally very expensive and time consuming to train with traditional CPUs. Neural networks depend a lot on training data[15].

Neural Networks require a lot of data, and Genetic Algorithms didn't. Genetic Algorithms can help Neural Networks to learn better and more efficiently. Neural Networks heavily depend on Hyper-parameters to perform properly, given a problem. But These Hyper-parameters are the only values that can not be learned. We can use GAs to learn the best hyper-parameters for Neural Networks. Neural Networks coupled with Genetic Algorithms can really accelerate the learning process.

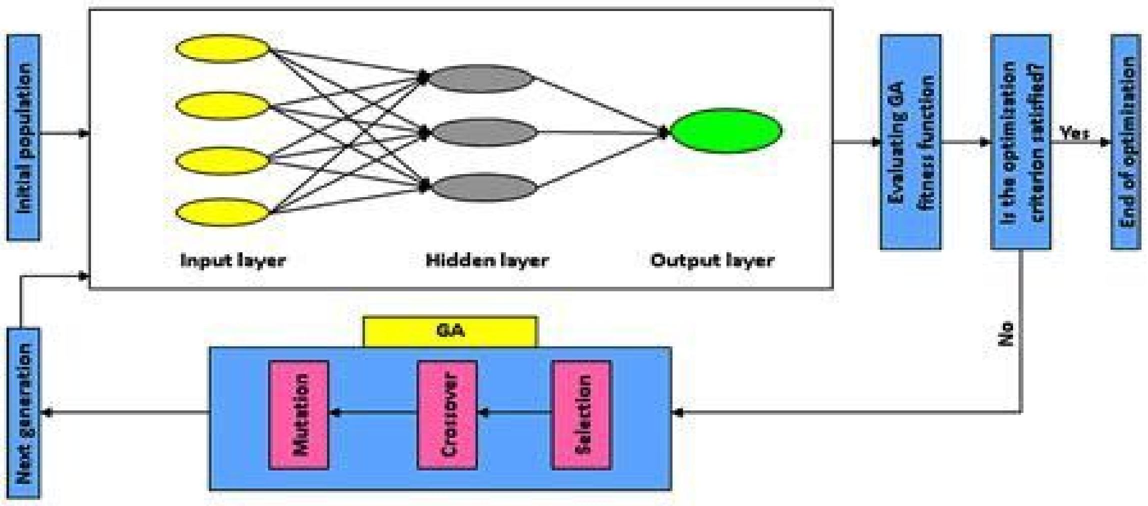


Figure 3: Genetic Algorithms coupling with neural networks

4.4 Identify Environment Parameters and State Parameters of the Throttle Control System Parameters

After a multi-stage reusable rocket is launched, each stage will carry the rocket to a certain attitude and then drop away. As it falls, it accelerates. There is a maximum velocity that each stage can reach, which is 54 m/s called the ‘terminal velocity.’ Earth gravity is 9.807 m/s^2.

The main thrusters of each stage can be applied to slow its descent. However, there is a limited amount of fuel. Once the fuel is exhausted, the stage will simply fall and crash eventually.

The throttle control system driven by Neural Network coupled with Genetic Algorithms should start piloting the landing process when it falls to 10,000 meters above the Earth. The Neural Networks algorithm will have only one option available to it. It can either decide to fire the thrusters or not to fire the thrusters. No training data will be created ahead of time and no assumptions will be made about how the neural network should pilot the falling rocket.

Even though the Neural Network coupled with Genetic Algorithms will learn everything on its own, this is still supervised training. The Neural Network coupled with Genetic Algorithms will not be totally left to its own devices. It will receive a way to score the Neural Network. To score the Neural Network, we must give it some goals and then calculate a numeric value that determines how well the Neural Network achieved its goals. These goals are arbitrary and simply reflect what was picked to score the network. The goals are summarized here:

- Land as softly as possible
- Cover as much distance as possible
- Conserve fuel

The first goal is to try to hit the ground as softly as possible. Therefore, any velocity at the time of impact is a big negative score. The second goal for the Neural Network is to try to cover as much distance as possible while falling. To do this, it needs to stay aloft as long as possible and additional points are awarded for staying aloft longer. Finally, bonus points are given for still having fuel once the craft lands. The score calculation can be seen in Equation as following:

score = (fuel · 26) + (velocity · 359)

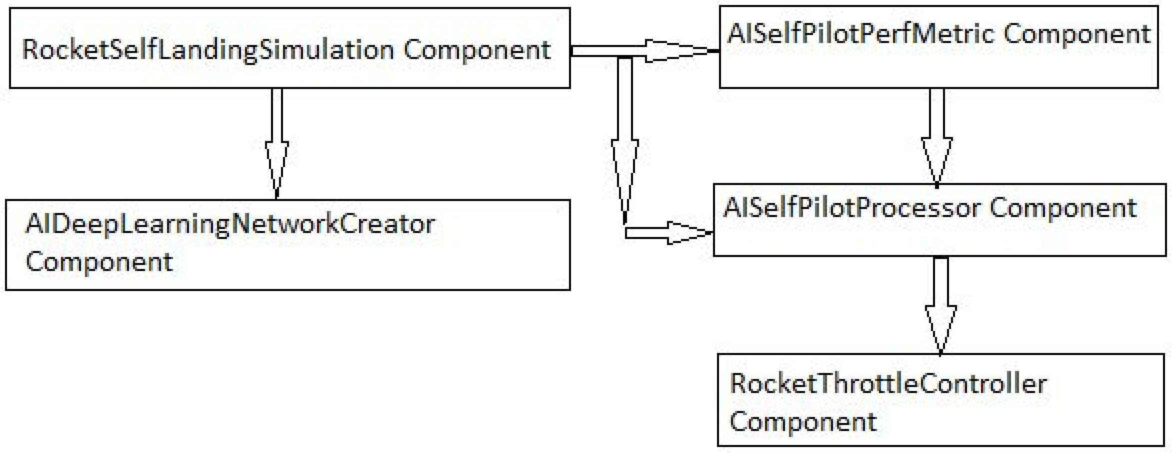
The additional assumption of this computer simulation is that each stage of a reusable rocket should have at least 100 liters fuel when it falls to an altitude to 10,000 meters above Earth's surface.

Following system state parameters are critical input parameters of the throttle control system:

- The fuel variable (holds the amount of fuel remaining).
- The time variable (holds the number of seconds aloft).
- The altitude variable (holds the current altitude in meters).
- The velocity variable (holds the current velocity in meters per second).

4.5 System Design

The application system architecture of the AI driven throttle control system of reusable rocket consists of five important components as follows:


RocketSelfLandingSimulation Component
AISelfPilotPerfMetric Component
AIDeepLearningNetworkCreator Component
AISelfPilotProcessor Component
RocketThrottleController Component

4.5.1 RocketSelfLandingSimulation Component

This module requires no arguments. Once the program begins, it will have access to AIDeepLearningNetworkCreator component to create and configure a neural network. Then kicks off the neural network immediately to start training. It will cycle through 25 epochs, or training iterations, before it is done. After the training process is complete, it will send the training model to AISelfPilotProcessor component and let AISelfPilotProcessor component to guide and control the throttle control system of reusable rocket and make sure the rocket lands on the ground as softly as possible. Please note that this text based computer simulation program only focus on resolving the three issues as follows:

- Smooth and soft touchdown on Earth
- Cover as much distance as possible
- Conserve fuel

4.5.2 AIDeeoLearningNetworkCreator Component

This module is responsible to create a neural network model which includes three layers as follows:

- Input layer which accepts three input parameters - Fuel, Time, and Altitude
- Hidden layer which consists of 100 neurons.
- Output layer which has only one neuron because we only need one output parameter to tell when to fire the rocket main thruster or not to fire the main rocket thruster.

4.5.3 RocketThrottleController Component

This module is defined to simulate the "physics" of rocket landing. The term "physics" is used very loosely. We focus more on how a neural network adapts to an artificial environment than any sort of realistic physical simulation.

This module begins by defining some constants that will be important to the Simulation.

- double EARTH_GRAVITY = 9.807 ;
- double THRUST = 20;
- double EARTH_TERMINAL VELOCITY = 54;

The EARTH_GRAVITY constant defines the acceleration on Earth that is due to gravity. It is set to 9.807 and is measured in meters per second^2. The THRUST constant value 20 m/s specifies how the number of meters per second by which the gravity acceleration will be countered. The EARTH TERMINAL VELOCITY is set as 54 m/s and is the fastest speed that the rocket can travel either upward or downward.

In addition to these constants, the simulator program need several variables to maintain rocket state. In the real world, evert reusable rocket must have fuel sensors, timers, altitude sensors and velocity sensors connected to the throttle control system of reusable rocket to provide real-time system state info.

These variables are listed below as follows:

- int fuel
- int timer
- double altitude
- double velocity

The fuel variable holds the amount of fuel remaining. The timer variable holds the number of seconds aloft. The altitude variable holds the current altitude in meters. The velocity variable holds the current velocity. Positive numbers indicate that the craft is moving upwards. Negative numbers indicate that the craft is moving downwards.

The simulator sets the values to reasonable starting values as follows:

- fuel = 2 0 0 liters;
- timer = 0 second;
- altitude = 1 0 0 0 0 meters;
- velocity = 0 m/s;

The assumption of this computer simulation is that after each stage of multi-stage rocket is separated, it will start falling at an altitude of 10,000 meters above ground and should have at least 200 liters of fuel in the tank.

The turn method processes each “turn.” A turn is defined as one second in the class module. The thrust parameter indicates whether the throttle control system wishes to thrust during this turn.

```
public void turn (boolean thrust )
```

First, increase the number of seconds elapsed by one. Decrease the velocity by the EARTH GRAVITY constant to simulate the fall.

```
second s++;
velocity −=EARTH GRAVITY;
```

The current velocity increases the altitude. Of course, if the velocity is negative, the altitude will decrease.

```
altitude+=velocity
```

If thrust is applied during this turn, then decrease the fuel by one and increase the velocity by the THRUST constant.

```
if ( thrust && fuel >0 ) {
fuel −−;
velocity+=THRUST;
}
```

Earth Terminal velocity must be imposed as it cannot fall or ascend faster than the terminal velocity. The following line makes sure that the lander is not ascending faster than the terminal velocity.

```
velocity = Math . max(−TERMINAL VELOCITY, velocity)
```

The following line makes sure that we are not descending faster than the terminal velocity.

```
velocity = Math . min (TERMINAL VELOCITY,  velocity)
```

The RocketThrottleController module also provides two utility functions. The first calculates the score and should only be called after the rocket lands. This method is shown here.

```
public int score ( ) {
        return ( int ) ( ( this.fuel ∗26 )+ (this.velocity ∗359)) ;
}
```

The score method uses fuel, and velocity to calculate the score. Additionally, a method is provided to determine if the falling rocket is still flying. If the altitude is greater than zero, it is still flying.

```
public boolean flying ( ) {
        return ( altitude >0) ;
}
```

4.5.4 AISelfPilotPerfMetric Component

This module implements the code necessary for the neural network to pilot the falling rocket. This module also calculates the final score after the rocket has landed. This module implements the CalculateScore interface.

The CalculateScore interface is used by genetic algorithms to determine how effective a neural network is at solving the given problem.

4.5.5 AISelfPilotProcessor Component

This module is responsible for how the neural network actually pilot the falling rocket. The neural network will be fed environmental information such as fuel remaining, altitude and current velocity. The neural network will then output a single value that will indicate if the neural network wishes to thrust.

The AISelfPilotProcessor constructor sets up the pilot to control the falling rocket. The constructor is passed a network to pilot the falling rocket, as well as a Boolean that indicates if telemetry should be tracked to the screen.

The falling rocket must feed the fuel level, altitude and current velocity to the neural network. These values must be normalized. To perform this normalization, the constructor begins by setting several normalization fields.

```
private NormalizedField fuelStats
private NormalizedField altitudeStats
private NormalizedField velocityStats
```

The neural pilot will have three input neurons and one output neuron. These three input neurons will communicate the following three fields to the neural network.

- Current fuel level
- Current altitude
- Current velocity

These three input fields will produce one output field that indicates if the neural pilot would like to fire the thrusters.

4.6 Running The Simulation Application and Test Data Analysis

To run this text based computer simulation application, you need to execute the main class - RocketSelfLandingSimulation by typing maven command as follows:

```
mvn clean install
```

The maven script will build the application first and then launch it.

RocketSelfLandingSimulation class is located at the following location.

```
edu.jasper.research.ai.RocketSelfLandingSimulation
```

This class requires no arguments. Once the program begins, the neural network immediately begins training. It will cycle through 25 epochs, or training iterations, before it is done. When it first begins, the score is 1667. These early attempts by the untrained neural network are hitting the ground at high velocity and are not covering much distance.

```
Epoch #1 Score:1667.0
Epoch #2 Score:1667.0
Epoch #3 Score:1667.0
Epoch #4 Score:1667.0
```

After the fourth epoch, the score begins to increase.

```
Epoch #5 Score:1779.0
Epoch #6 Score:1779.0
Epoch #7 Score:1779.0
Epoch #8 Score:2536.0
Epoch #9 Score:2536.0
Epoch #10 Score:3161.0
Epoch #11 Score:3161.0
Epoch #12 Score:3161.0
Epoch #13 Score:3161.0
Epoch #14 Score:3161.0
Epoch #15 Score:3161.0
Epoch #16 Score:3611.0
Epoch #17 Score:3611.0
```

The score will hover at 3611 for awhile, but will improve again and reach 4543 at epoch #18 and then stay unchanged to the end. You can see the Deep Learning Neural Network coupled with Genetic Algorithm converges at Epoch #18.

```
Epoch #18 Score:4543.0
Epoch #19 Score:4543.0
Epoch #20 Score:4543.0
Epoch #21 Score:4543.0
Epoch #22 Score:4543.0
Epoch #23 Score:4543.0
Epoch #24 Score:4543.0
Epoch #25 Score:4543.0
```

The training techniques used in research make extensive use of random numbers. As a result, running this example multiple times may result in entirely different scores. More epochs may have produced a better-trained neural network; however, the program limits it to 25. This number usually produces a fairly skilled neural pilot. Once the network is trained, run the simulation with the winning pilot. The telemetry is displayed at each second.

The neural network pilot kept the falling rocket aloft for 155 seconds. So, we will not show every telemetry report. However, some of the interesting actions that this neural network pilot learned are highlighted. The neural network learned it was best
to just let the craft free-fall for awhile.

```
How the winning network landed:
Elapsed: 1 s, Fuel: 200 l, Velocity: -9.8070 m/s, 9990 m
Elapsed: 2 s, Fuel: 200 l, Velocity: -19.6140 m/s, 9970 m
Elapsed: 3 s, Fuel: 200 l, Velocity: -29.4210 m/s, 9941 m
Elapsed: 4 s, Fuel: 200 l, Velocity: -39.2280 m/s, 9901 m
Elapsed: 5 s, Fuel: 200 l, Velocity: -49.0350 m/s, 9852 m
Elapsed: 6 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9794 m
Elapsed: 7 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9730 m
Elapsed: 8 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9666 m
Elapsed: 9 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9602 m
Elapsed: 10 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9538 m
Elapsed: 11 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9475 m
Elapsed: 12 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9411 m
Elapsed: 13 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9347 m
Elapsed: 14 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9283 m
Elapsed: 15 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9219 m
Elapsed: 16 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9155 m
```

```
Elapsed: 17 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9092 m
Elapsed: 18 s, Fuel: 200 l, Velocity: -54.0000 m/s, 9028 m
Elapsed: 19 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8964 m
Elapsed: 20 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8900 m
Elapsed: 21 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8836 m
Elapsed: 22 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8773 m
Elapsed: 23 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8709 m
Elapsed: 24 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8645 m
Elapsed: 25 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8581 m
Elapsed: 26 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8517 m
Elapsed: 27 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8454 m
Elapsed: 28 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8390 m
Elapsed: 29 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8326 m
Elapsed: 30 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8262 m
Elapsed: 31 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8198 m
Elapsed: 32 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8135 m
Elapsed: 33 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8071 m
Elapsed: 34 s, Fuel: 200 l, Velocity: -54.0000 m/s, 8007 m
Elapsed: 35 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7943 m
Elapsed: 36 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7879 m
Elapsed: 37 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7816 m
Elapsed: 38 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7752 m
Elapsed: 39 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7688 m
Elapsed: 40 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7624 m
Elapsed: 41 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7560 m
Elapsed: 42 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7497 m
Elapsed: 43 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7433 m
Elapsed: 44 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7369 m
Elapsed: 45 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7305 m
Elapsed: 46 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7241 m
Elapsed: 47 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7177 m
Elapsed: 48 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7114 m
Elapsed: 49 s, Fuel: 200 l, Velocity: -54.0000 m/s, 7050 m
Elapsed: 50 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6986 m
Elapsed: 51 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6922 m
Elapsed: 52 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6858 m
Elapsed: 53 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6795 m
Elapsed: 54 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6731 m
Elapsed: 55 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6667 m
Elapsed: 56 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6603 m
Elapsed: 57 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6539 m
Elapsed: 58 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6476 m
Elapsed: 59 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6412 m
Elapsed: 60 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6348 m
Elapsed: 61 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6284 m
```

```
Elapsed: 62 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6220 m
Elapsed: 63 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6157 m
Elapsed: 64 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6093 m
Elapsed: 65 s, Fuel: 200 l, Velocity: -54.0000 m/s, 6029 m
Elapsed: 66 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5965 m
Elapsed: 67 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5901 m
Elapsed: 68 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5838 m
Elapsed: 69 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5774 m
Elapsed: 70 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5710 m
Elapsed: 71 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5646 m
Elapsed: 72 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5582 m
Elapsed: 73 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5518 m
Elapsed: 74 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5455 m
Elapsed: 75 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5391 m
Elapsed: 76 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5327 m
Elapsed: 77 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5263 m
Elapsed: 78 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5199 m
Elapsed: 79 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5136 m
Elapsed: 80 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5072 m
Elapsed: 81 s, Fuel: 200 l, Velocity: -54.0000 m/s, 5008 m
Elapsed: 82 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4944 m
Elapsed: 83 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4880 m
Elapsed: 84 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4817 m
Elapsed: 85 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4753 m
Elapsed: 86 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4689 m
Elapsed: 87 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4625 m
Elapsed: 88 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4561 m
Elapsed: 89 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4498 m
Elapsed: 90 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4434 m
Elapsed: 91 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4370 m
Elapsed: 92 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4306 m
Elapsed: 93 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4242 m
Elapsed: 94 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4179 m
Elapsed: 95 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4115 m
Elapsed: 96 s, Fuel: 200 l, Velocity: -54.0000 m/s, 4051 m
Elapsed: 97 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3987 m
Elapsed: 98 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3923 m
Elapsed: 99 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3860 m
Elapsed: 100 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3796 m
Elapsed: 101 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3732 m
Elapsed: 102 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3668 m
Elapsed: 103 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3604 m
Elapsed: 104 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3540 m
Elapsed: 105 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3477 m
Elapsed: 106 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3413 m
```

```
Elapsed: 107 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3349 m
Elapsed: 108 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3285 m
Elapsed: 109 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3221 m
Elapsed: 110 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3158 m
Elapsed: 111 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3094 m
Elapsed: 112 s, Fuel: 200 l, Velocity: -54.0000 m/s, 3030 m
Elapsed: 113 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2966 m
Elapsed: 114 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2902 m
Elapsed: 115 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2839 m
Elapsed: 116 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2775 m
Elapsed: 117 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2711 m
Elapsed: 118 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2647 m
Elapsed: 119 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2583 m
Elapsed: 120 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2520 m
Elapsed: 121 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2456 m
Elapsed: 122 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2392 m
Elapsed: 123 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2328 m
Elapsed: 124 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2264 m
Elapsed: 125 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2201 m
Elapsed: 126 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2137 m
Elapsed: 127 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2073 m
Elapsed: 128 s, Fuel: 200 l, Velocity: -54.0000 m/s, 2009 m
Elapsed: 129 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1945 m
Elapsed: 130 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1881 m
Elapsed: 131 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1818 m
Elapsed: 132 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1754 m
Elapsed: 133 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1690 m
Elapsed: 134 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1626 m
Elapsed: 135 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1562 m
Elapsed: 136 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1499 m
Elapsed: 137 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1435 m
Elapsed: 138 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1371 m
Elapsed: 139 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1307 m
Elapsed: 140 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1243 m
Elapsed: 141 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1180 m
Elapsed: 142 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1116 m
Elapsed: 143 s, Fuel: 200 l, Velocity: -54.0000 m/s, 1052 m
Elapsed: 144 s, Fuel: 200 l, Velocity: -54.0000 m/s, 988 m
Elapsed: 145 s, Fuel: 200 l, Velocity: -54.0000 m/s, 924 m
Elapsed: 146 s, Fuel: 200 l, Velocity: -54.0000 m/s, 861 m
Elapsed: 147 s, Fuel: 200 l, Velocity: -54.0000 m/s, 797 m
Elapsed: 148 s, Fuel: 200 l, Velocity: -54.0000 m/s, 733 m
Elapsed: 149 s, Fuel: 200 l, Velocity: -54.0000 m/s, 669 m
Elapsed: 150 s, Fuel: 200 l, Velocity: -54.0000 m/s, 605 m
Elapsed: 151 s, Fuel: 200 l, Velocity: -54.0000 m/s, 542 m
```

```
Elapsed: 152 s, Fuel: 200 l, Velocity: -54.0000 m/s, 478 m
Elapsed: 153 s, Fuel: 200 l, Velocity: -54.0000 m/s, 414 m
Elapsed: 154 s, Fuel: 200 l, Velocity: -54.0000 m/s, 350 m
```

You can see that 6 seconds in and 9794 meters above the ground, the Earth terminal
velocity of -54 m/s has been reached. Having a terminal velocity is interesting because the neural networks learn that once this is reached, the falling rocket will not speed up. Neural network algorithm uses the terminal velocity to save fuel and “break their fall” when they get close to the surface. The freefall at terminal velocity continues for some time.

Finally, at 286 meters above the ground, the thrusters are fired for the first time.

```
Elapsed: 155 s, Fuel: 200 l, Velocity: -54.0000 m/s, 286 m
THRUST
Elapsed: 156 s, Fuel: 199 l, Velocity: -43.8070 m/s, 223 m
THRUST
Elapsed: 157 s, Fuel: 198 l, Velocity: -33.6140 m/s, 169 m
THRUST
Elapsed: 158 s, Fuel: 197 l, Velocity: -23.4210 m/s, 125 m
THRUST
Elapsed: 159 s, Fuel: 196 l, Velocity: -13.2280 m/s, 92 m
THRUST
Elapsed: 160 s, Fuel: 195 l, Velocity: -3.0350 m/s, 69 m
THRUST
Elapsed: 161 s, Fuel: 194 l, Velocity: 7.1580 m/s, 56 m
Elapsed: 162 s, Fuel: 194 l, Velocity: -2.6490 m/s, 54 m
THRUST
Elapsed: 163 s, Fuel: 193 l, Velocity: 7.5440 m/s, 41 m
Elapsed: 164 s, Fuel: 193 l, Velocity: -2.2630 m/s, 39 m
THRUST
Elapsed: 165 s, Fuel: 192 l, Velocity: 7.9300 m/s, 27 m
Elapsed: 166 s, Fuel: 192 l, Velocity: -1.8770 m/s, 25 m
THRUST
Elapsed: 167 s, Fuel: 191 l, Velocity: 8.3160 m/s, 13 m
Elapsed: 168 s, Fuel: 191 l, Velocity: -1.4910 m/s, 12 m
THRUST
Elapsed: 169 s, Fuel: 190 l, Velocity: 8.7020 m/s, 1 m
THRUST
Elapsed: 170 s, Fuel: 190 l, Velocity: -1.1050 m/s, 0 m
4543
```

The velocity is gradually slowed, as the neural network decides to fire the thrusters every few seconds. At around 1 meter above the surface, the neural network decides it should now thrust again. This slows the descent to around -1.1050 m/s when the rocket finally land.

The neural network pilot was trained using a genetic algorithm and learned some very interesting things without being fed a pre-devised strategy. The network learned what it wanted to do. Specifically, this pilot decided the following:

- Free-fall for some time to take advantage of terminal velocity.
- At a certain point, break the freefall and slow the rocket.
- Slowly lose speed as you approach the surface.

4.7 Lesson Learned

The process of setting the neural network parameters requires expertise and extensive trial and error. There are no simple and easy ways to set these parameters. It took me a lot of time to turn the parameters of the Neural Network model in my program.

As I mentioned earlier, the training techniques used in research make extensive use of random numbers. As a result, running this example multiple times may result in entirely different scores. In order to resolve this issue, what I should do when using random numbers but for the result to be repeatable is to use a *random seed*. This basically produces the same sequence of numbers each time, although they are still pseudorandom (these are a great way for comparing models and also testing for reproducibility).

In this research I have 100 neurons in the hidden layer of the Neural Network algorithm. It takes a lot of time and computation resources. This limits the implementation of this solution to only the people who are willing to put in the money and buy a lot of processing power.

4.8 Conclusion

According to the result data generated by the simulation program, I think that the neural network plus genetic algorithm gives me the best result because the touch down velocity of the falling rocket was between -2.0 m/s and +2.0 m/s. The touch down velocity of the rocket in our research experiment was -1.1050 m/s. It was a very soft velocity.

5. Bibliography

References:

[1] Loren Grush, "SpaceX’s reusable rockets will make space cheaper — but how much?", SpaceX, Published Online Dec 24, 2015.

[2] Tom Nardi, "SPACEX’S NEXT GIANT LEAP: SECOND STAGE RECOVERY", Published Online September 27, 2019.

[3] Blackmore, Lars (Winter 2016). "Autonomous Precision Landing of Space Rockets" (PDF). The Bridge, National Academy of Engineering. 46 (4): 15–20. ISSN 0737-6278. Retrieved January 15, 2017.

[4] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, Urs Muller, "Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car" (PDF). arxiv.org, arXiv:1704.07911. Published Online 25 Apr 2017.

[5] Jiman Kim, Chanjong Park,"End-To-End Ego Lane Estimation Based on Sequential Transfer Learning for Self-Driving Cars", The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2017, pp. 30-38.

[6] Lindsey, Clark (2013-03-28). "SpaceX moving quickly towards fly-back first stage". NewSpace Watch. Retrieved 2013-03-29.

[7] Mike Wall, "SpaceX Recovered Falcon Heavy Nose Cone, Plans to Re-fly it This Year" April 12, 2019 Spaceflight

[8] Tim Fernholz, "SpaceX's self-landing rocket is a flying robot that's great at math", Published Online, February 21, 2017

[9] Elon Musk, Twitter, Published Online, April 18, 2015

[10] Seth Weidman, "The 4 Deep Learning Breakthroughs You Should Know About", Dec 2017. https://towardsdatascience.com/the-5-deep-learning-breakthroughs-you-should-know-about-df27674ccdf2

[11] Samer Hijazi, Rishi Kumar, and Chris Rowen, "Using Convolutional Neural Networks for Image Recognition", IP Group, Cadence, 2015,https://ip.cadence.com/uploads/901/cnn_wp-pdf

[12] Jinbo Wang, Naigang Cui and Changzhu Wei, "Optimal Rocket Landing Guidance Using Convex Optimization and Model Predictive Control", Journal of Guidance, Control, and Dynamics, Volume 42, Number 5, May 2019

[13]Roberto Furfaro, Richard Linares, Conference Paper Conference "Deep Learning for Autonomous Lunar Landing", AAS/AIAA Astrodynamics Specialist Conference,August 2018

[14]B Gaudet, R Linares, R Furfaro "Deep Reinforcement Learning for Six Degree-of-Freedom Planetary Powered Descent and Landing" - arXiv preprint arXiv, 2018 - adsabs.harvard.edu

[15]Balaji Venkateswaran, Giuseppe Ciaburro, book title "Neural Networks with R" - Publisher: Packt Publishing, Release Date: September 2017, ISBN: 9781788397872