# VERTICAL ROCKET LANDING

PROJECT REPORT

Submitted by

**Raghuvamsi Bokka Satyajit Sen Shambhavi Halemani Shubhojit Sen**

RESEARCH INTERNSHIP

Under the guidance of

**Mr. Sajan Sudhir Technical Trainer,**

**Artificial Intelligence for Aerospace Technology**



Spartificial Innovations

(SEPTEMEBER 2022)

## ACKNOWLEDGEMENT

It gives us immense pleasure to express our deepest sense of gratitude to our respected mentor, Mr. Sajan Sudhir, Technical Trainer, Artificial Intelligence for Aerospace Technology, Spartificial Innovations, for his valuable guidance, encouragement, and help in completing this work. His useful suggestions for this whole work and co- operative behavior are sincerely acknowledged.

We also wish to thank the project coordinators, Sarthak Narnor and Ashwin Gupta, for making this internship as smooth as possible.

## ABSTRACT

A reusable launch vehicle has parts that can be recovered and reflown, while carrying payloads from the surface to outer space. Rocket stages are the most common launch vehicle parts aimed for reuse. Smaller parts such as rocket engines and boosters can also be reused, though reusable spacecraft may be launched on top of an expendable launch vehicle. Reusable launch vehicles do not need to make these parts for each launch, therefore reducing its launch cost significantly. The future of this technology lies in automating the process further. This project is a two- dimensional demonstration of the possibility of using reinforcement learning for the same in the real world. This project entails using an Advantage Actor Critic method to land a reusable rocket on a platform suspended on a water body. The aim of this project is to pave the way to explore the application of artificial intelligence to novel domains such as reusable rocket landing.

# INTRODUCTION

A reusable rocket, which is meant for carrying payloads from the surface of our planet to outer space, has parts that can be recovered and reflown. The rocket stages are the most common parts aimed for reuse. Smaller parts such as rocket engines and boosters can also be reused, though reusable spacecraft may be launched on top of an expendable launch vehicle. Typically, reusable launch vehicles do not need to make these parts for each launch, therefore reducing its launch cost significantly.

#### History of Reusable Rockets

The first reusable launch vehicles were the ones conceptualized and studied by Wernher von Braun from 1948 until 1956. The Von Braun Ferry Rocket underwent two revisions: once in 1952 and again in 1956. They would have landed using parachutes.[1][2] In 2012, SpaceX started a flight test program with experimental vehicles. These subsequently led to the development of the Falcon 9 reusable rocket launcher.[3] On 23 November 2015 the New Shepard rocket became the first Vertical Take-off, Vertical Landing (VTVL) sub-orbital rocket to reach space by

passing the Kármán line (100 km or 62 mi), reaching 329,839 ft (100,535 m) before returning for a propulsive landing.[4][5] SpaceX achieved the first vertical soft landing of a reusable orbital rocket stage on December 21, 2015, after delivering 11 Orbcomm OG-2 commercial satellites into low Earth orbit.[6] The first reuse of a Falcon 9 first stage occurred on 30 March 2017.[7] SpaceX now semi-routinely recovers and reuses their first stages, as well as reusing fairings[8] In 2019 Rocket Lab announced plans to recover and reuse the first stage of their Electron launch vehicle, intending to use parachutes and mid-air retrieval.[9] On 20 November 2020, Rocket Lab successfully returned an Electron first stage from an orbital launch, the stage softly splashing down in the Pacific Ocean.[10] China is researching the reusability of the Long March 8 system.[11] As of May 2020, the only operational reusable orbital-class launch systems are the Falcon 9 and Falcon Heavy, the latter of which is based upon the Falcon 9. SpaceX is also developing the fully-reusable Starship launch system,[12] and Blue Origin is developing its own New Glenn partially-reusable orbital rocket, as it is intending to recover and reuse only the first stage.

#### Use of Artificial Intelligence

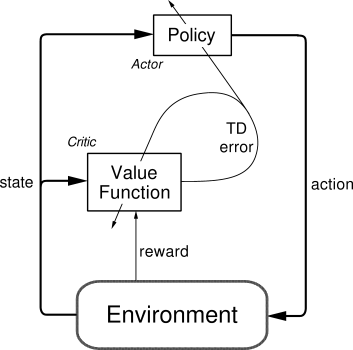
Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by animals and humans. AI research has been defined as the field of study of intelligent agents, which refers to any system that perceives its environment and takes actions that maximize its chance of achieving its goals.

The term "artificial intelligence" had previously been used to describe machines that mimic and display "human" cognitive skills that are associated with the human mind, such as "learning" and "problem-solving". This definition has since been rejected by major AI researchers who now describe AI in terms of rationality and acting rationally, which does not limit how intelligence can be articulated.

The future of the technology of robustly landing reusable rockets lies in automating the process further via artificial intelligence. This project is a two-dimensional demonstration of the possibility of using reinforcement learning for the same in the real world. This project entails using an Advantage Actor Critic (A2C) method to land a reusable rocket on a platform suspended on a water body.

#### Actor Critic (A2C) Methods

Actor-critic methods are temporal difference methods that have a separate memory structure to explicitly represent the policy independent of the value function. The policy structure is known as the actor, because it is used to select actions, and the estimated value function is known as the critic, because it criticizes the actions made by the actor.[13] Learning is always on-policy: the critic must learn about and critique whatever policy is currently being followed by the actor.[13] The critique takes the form of a temporal difference error.[13] This scalar signal is the sole output of the critic and drives all learning in both actor and critic, as suggested by Figure 1.



**Figure 1: Actor Critic Methods**

Typically, the critic is a state-value function. After each action selection, the critic evaluates the new state to determine whether things have gone better or worse than expected.[13] That evaluation is the TD error:



where is the current value function implemented by the critic. This TD error can be used to evaluate the action just selected, the action taken in state . If the TD error is positive, it suggests that the tendency to select should be strengthened for the future, whereas if the TD error is negative, it suggests the tendency should be weakened.[13]

#### Deep Q-Network (DQN) v/s Actor-Critic Method

Value based networks try to find or approximate the optimal value function, which is a mapping between an action and a value. The higher the value, the better the action.

Policy based algorithms try to find the optimal policy directly without the Q-value as a middleman.

Each method has its advantages. For example, policy-based

algorithms are better for continuous and stochastic environments, have a faster convergence, while value-based algorithms are more sample efficient and steady.

Deep Q-Networks are value based networks, whereas Actor- Critic is both value and policy based.

When value based algorithms and policy based algorithms were established in the scientific communities, the next step was to try and merge them. And those attempts yielded the Actor-Critic algorithm. It is both a policy based and value based algorithm.

The principle idea is to split the model into two. The Actor- Critic Algorithm uses two networks: an Actor network and a Critic network. The Actor determines the action when the state is given, and the Critic evaluates the value of the state or the action-value. The Actor then updates the policy distribution in the direction suggested by the Critic. Both the Critic and Actor functions are parameterized with neural networks. These two separate models keep getting better in their role as time passes. The result is that the overall architecture will learn to work more efficiently than the two methods separately.

DQNs require a target network for stability, whereas Actor- Critic methods function as required without a target

network.

Actor-Critic uses advantage estimates to calculate the value proposition for each state pair.

The biggest difference between DQN and Actor-critic is whether to use Replay Buffer. Unlike DQN, Actor-Critic does not use Replay Buffer but learns the model using state(s), action(a), reward(r), and the next state(s`) obtained at every step.

# SOFTWARE REQUIREMENT SPECIFICATION

### 2.1 Software Requirements

2.11 Python

### 2.2 Hardware Requirements

* 1. Intel® i7 or AMD Ryzen7 processor (or higher)
  2. NVIDIA GTX 1650Ti (or higher)
  3. 8 GB of RAM (16 GB recommended)
  4. 100 GB of available SSD storage space for training data

# APPROACH

#### Overview

The environment and the rocket are both shown on a standard 2D plane. In this setting, the rocket is just a conventional rigid body. For this model, we took into account the fundamental cylinder dynamics and made the assumption that the air resistance is proportional to the velocity. The thrust engine provides variable thrust in several directions.



**Figure 2: The Environment**

The action space is described by the aforementioned parameters as a set of the various engine control signals, such as thrust and angular velocity. The set of the area on the platform suspended in water that correspond to a successful landing, the areas on the platform suspended on the platform that correspond to an unsuccessful landing, and the water body itself form the state space.

* 1. **Annotations to the codebase of the project**

#### Helper Functions

There are particular functionalities that may be required in multiple places in a codebase. These functions are declared once and used everywhere in the codebase they may be required. Some of these functions may even be simple but due to the sheer degree of their use in multiple places in a single codebase, it’s only practical to declare them once as opposed to declaring them inline or anonymously multiple times. Doing so also leads to higher readability of the code. Such functions are known as helper functions.

This file utils.py contains helper functions that have been used in the rocket.py and policy.py files.

1. moving\_avg : This function helps calculate the moving average, which is an indicator commonly used in technical analysis. It is used for smoothing out the reward data by creating a constantly updated average reward.
2. load\_bg\_img : This function loads the background image in the rendered environment.
3. create\_circle\_poly : This function creates an array of 50 points on the circumference of a circle, which can then be used as an actual shape that interacts with the 2D environment.
4. create\_ellipse\_poly : This function creates an array of 50 points on the circumference of an ellipse, which can then be used as an actual shape that interacts with the 2D environment.
5. create\_rectangle\_poly : This function creates an array of the coordinates of the corners of a rectangle, which can then be used as an actual shape that interacts with the 2D environment. (All these shape-arrays will be used to create the body of the rocket.)
6. scale\_matrix : This function helps creates a scale for the environment on all the axes.
7. rotation\_matrix : This function controls the rotation of the rocket in the environment.
8. translation\_matrix : This function is responsible for the motion of the rocket from one point to another in the 2D plane.
9. create\_pose\_matrix : This function is responsible for the pose of the object as seen by the viewer, i.e., us.

#### Policy

In Reinforcement Learning (RL), a policy π: s →a is any function that returns a feasible action for a problem. For instance, any given task, one could simply take the first action that comes to mind, select an action at random, or run a heuristic. In this example, the description of the process that leads to the exact action, whatever that may be, towards the completion of said task independent of the success of the action in question to the eventual completion of said task, would be referred to as the policy employed.

The policy.py file contains the policy that’ll be used in the training of our agent in our environment. Firstly, we choose the device for running our programs. Then we list

and define a function and then some classes.

1. calculate\_returns : This function calculates the "returns" after every step using the rewards (or lack thereof) that that agent will receive with each step. It basically provides feedback to the ActorCritic.
2. class PositionalMapping : This layer map continuously inputs coordinates into a higher dimensional space and enables the prediction to more easily approximate a higher frequency function.
3. class MLP : This is a Multilayer perceptron with an embedded positional mapping layer.
4. class ActorCritic : This defines class ActorCritic, which sets the rules for the environment, but also monitors it and makes changes, if necessary. It receives an action ID and updates itself accordingly.

#### The Agent and the Environment

The file rocket.py contains the class Rocket, which consists of the rocket and the environment. The rocket is simplified into a rigid body model with a thin rod, considering acceleration and angular acceleration and air resistance

proportional to velocity. There are two tasks: hover and landing. Their reward functions are straightforward and simple. For the hover tasks: the step-reward is given based on two factors

1. the distance between the rocket and the predefined target point.
2. the angle of the rocket body (the rocket should stay as upright as possible).

For the landing task: the step-reward is given based on three factors:

1. the distance between the rocket and the predefined landing point.
2. the angle of the rocket body (the rocket should stay as upright as possible).
3. Speed and angle at the moment of contact with the ground, when the touching-speed are smaller than a safe threshold and the angle is close to 90 degrees (upright), we see it as a successful landing.

In our case, we’ve only trained the agent for the landing task, and not the hovering task.

1. class Rocket :
   1. The init function contains all the basic properties of the rocket and the environment. It is responsible for intitializing the environment. It lists all the physical parameters of the rocket and the environment. It specifies the target point.
   2. The reset function is responsible for resetting the environment when the rocket crashes.
   3. create\_action\_table creates an action table that contains all the actions that the rocket can take.
   4. get\_random\_action yields an action randomly chosen from the action table that was created by create\_action\_table.
   5. create\_random\_state creates a random state for the rocket before it begins its task.
   6. check\_crash is the function for checking the status of the rocket crashing, that is, whether it has crashed or not.
   7. check\_landing\_success is the function for checking the status of the rocket landing, that is, whether it has landed or not.
   8. calculate\_reward calculates the reward after the end of every episode.
   9. The step function takes the action\_id as input, and then yields a new state and a reward.
   10. The flatten function will flatten the state array that is given to it as an input.
   11. The render function renders the environment for us viewers to see.
   12. The create\_polygons function creates the body of the rocket using all the helper functions defined in utils.py
   13. The draw\_a\_polygon function creates the visual of the rocket using the outputs of the create\_polygons function. It creates a coloured 2D pixelated rocket that we can see when we render our model.
   14. The wd2pxl function pixelates all the points that it gets as an input.
   15. The draw\_text function puts text on the rendered window. This text shows the velocity, speed, angular inclination, simulation steps and simulation time of the rocket as it tries to land.
   16. The draw\_trajectory function draws the trajectory taken by the rocket as it tries to land.
   17. The crop\_alongwith\_camera function resizes the environment visually according to the limits that have been defined in the init function.

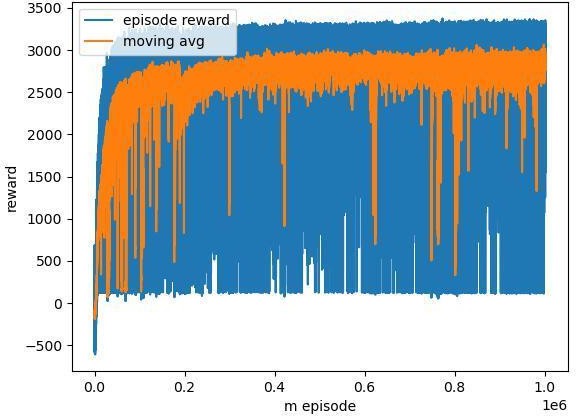
#### Training

The file example\_train.py is responsible for training the agent to land or hover, but we have used it for landing only.

It creates a folder for storing all the training checkpoints, which are basically containers for the information needed to land properly.

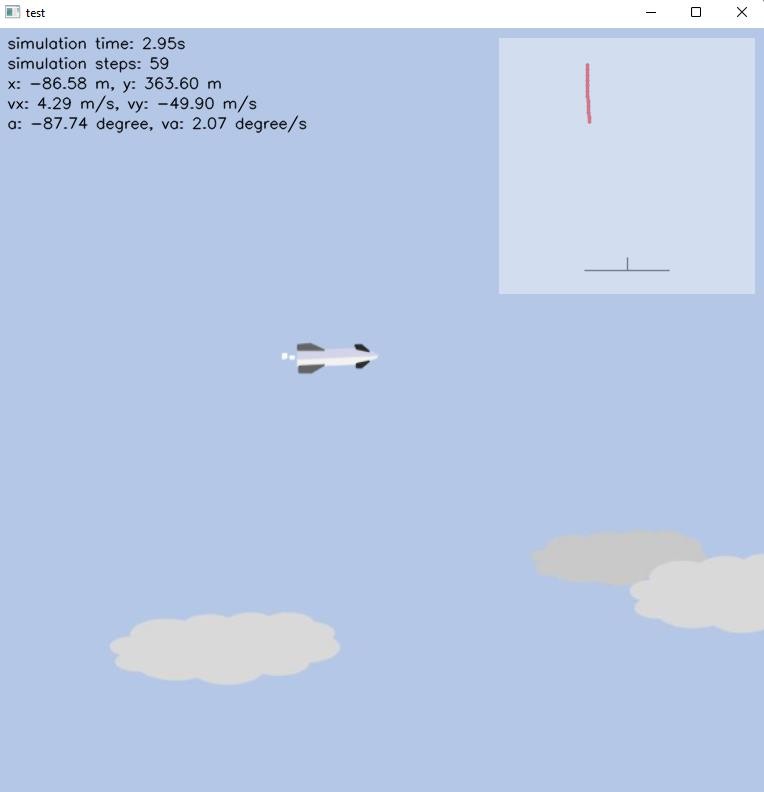
It uses functions from the previously mentioned python files and yields rewards as it trains the agent. It also plots graphs for every hundredth training episode and stores them in same folder as the checkpoints. test.py This file has been written for the purpose of testing an agent, trained or otherwise. It uses the penultimate training checkpoint from the checkpoint folder and tests the agent under those parameters.

**RESULTS**

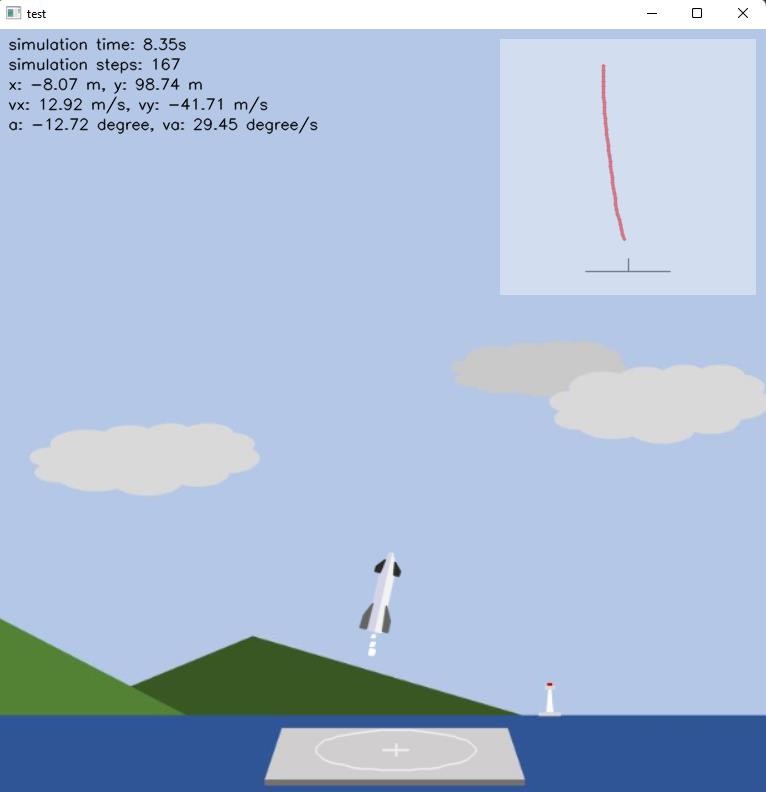


**Figure 3: Reward over number of episodes**

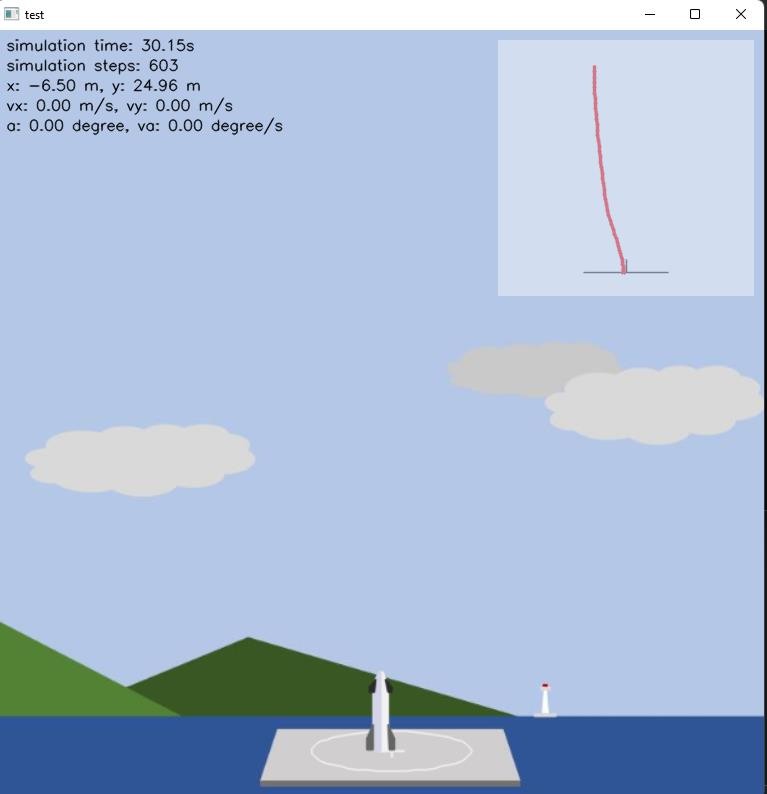
The agent has been successfully trained for 10,00,000 episodes.



**Figure 4: Simulation: Rocket descending from the sky**



**Figure 5: Simulation: Rocket hovering before landing**



**Figure 6: Simulation: Rocket having landed**

The accuracy of the agent has been calculated to be 61% based on 100 test episodes.

While this project has been a wholly collaborative endeavour, most aspects of it were compartmentalized so as to ensure the timely completion thereof.

Raghuvamsi Bokka’s major contributions were towards the creation of polygons, the drawing of the polygons, the rendering of the text in the simulation, the drawing of the trajectory of the Rocket, and the helper functions used in the primary python files.

Satyajit Sen’s major contributions were towards creating a reset feature for the model, introducing random states so as to get unbiased data, giving the model the ability to check whether it has crashed, helping in the calculation of the rewards, enabling the pixelation of all the polygons, the resizing of the environment, and the training the model.

Shambhavi Halemani’s major contributions were towards creating the action table with all the different possible actions, giving the model the ability to check whether it has landed, and the writing of the code for testing any agent, trained or otherwise.

Shubhojit Sen’s major contributions were towards adding the basic properties of the rocket and the environment to

the initialization function, the yielding of the step-ID and reward, creating the policy, and the visual rendering of the environment & the model goes.

The project can be found at: <https://github.com/shubhojitsen/vertical-rocket-landing>

## REFERENCES

1. "von Braun concept vehicle" *Astronautix.com*

<http://www.astronautix.com/v/vonbraunconceptvehicle.html>

1. "Wernher von Braun's Fantastic Vision: Ferry Rocket | WIRED"

[https://www.wired.com/2014/09/wernher-von-brauns-fantastic-vision-](https://www.wired.com/2014/09/wernher-von-brauns-fantastic-vision-ferry-rocket/) [ferry-rocket/](https://www.wired.com/2014/09/wernher-von-brauns-fantastic-vision-ferry-rocket/)

1. Lindsey, Clark (2013-03-28). "SpaceX moving quickly towards fly- back first stage"

https://web.archive.org/web/20130416030256/http://www.newspacewatch. com/articles/spacex-moving-quickly-towards-fly-back-first-stage.html

1. Cofield, Calla (2015-11-24). "Blue Origin Makes Historic Reusable Rocket Landing in Epic Test Flight". *Space.com.*

[https://www.space.com/31202-blue-origin-historic-private-rocket-](https://www.space.com/31202-blue-origin-historic-private-rocket-landing.html) [landing.html](https://www.space.com/31202-blue-origin-historic-private-rocket-landing.html)

1. Berger, Eric (2015-11-25). "Jeff Bezos and Elon Musk spar over gravity of Blue Origin rocket landing". *Ars Technica.*

[https://arstechnica.com/science/2015/11/jeff-bezos-and-elon-musk-spar-](https://arstechnica.com/science/2015/11/jeff-bezos-and-elon-musk-spar-over-gravity-of-blue-origin-rocket-landing/) [over-gravity-of-blue-origin-rocket-landing/](https://arstechnica.com/science/2015/11/jeff-bezos-and-elon-musk-spar-over-gravity-of-blue-origin-rocket-landing/)

1. "SpaceX on Twitter". *Twitter.*

<https://twitter.com/SpaceX/status/679114269485436928>

1. "SpaceX successful [sic] launches first recycled rocket – video". *The Guardian.*

[https://www.theguardian.com/science/video/2017/mar/31/spacex-](https://www.theguardian.com/science/video/2017/mar/31/spacex-successfuly-launches-first-recycled-rocket-video) [successfuly-launches-first-recycled-rocket-video](https://www.theguardian.com/science/video/2017/mar/31/spacex-successfuly-launches-first-recycled-rocket-video)

1. Wall, Mike (2019-04-12). "SpaceX Recovered Falcon Heavy Nose Cone, Plans to Re-fly it This Year (Photos)". *Space.com.*

<https://www.space.com/spacex-reuse-payload-fairing-starlink-launch.html>

1. "Rocket Lab Announces Reusability Plans For Electron Rocket".

*Rocket Lab.*

[https://www.rocketlabusa.com/updates/rocket-lab-announces-reusability-](https://www.rocketlabusa.com/updates/rocket-lab-announces-reusability-plans-for-electron-rocket/) [plans-for-electron-rocket/](https://www.rocketlabusa.com/updates/rocket-lab-announces-reusability-plans-for-electron-rocket/)

1. "Rocket Lab launches Electron in test of booster recovery".

*SpaceNews.*

[https://spacenews.com/rocket-lab-launches-electron-in-test-of-booster-](https://spacenews.com/rocket-lab-launches-electron-in-test-of-booster-recovery/) [recovery/](https://spacenews.com/rocket-lab-launches-electron-in-test-of-booster-recovery/)

1. "China to test rocket reusability with planned Long March 8 launcher". *SpaceNews.*

[https://spacenews.com/china-to-test-rocket-reusability-with-planned-long-](https://spacenews.com/china-to-test-rocket-reusability-with-planned-long-march-8-launcher/) [march-8-launcher/](https://spacenews.com/china-to-test-rocket-reusability-with-planned-long-march-8-launcher/)

1. "Making Life Multiplanetary" (video). *68th annual meeting of the International Astronautical Congress in Adelaide, Australia: SpaceX.*

<https://www.youtube.com/watch?v=tdUX3ypDVwI>

1. Sutton, Richard S. "Actor-Critic Methods" <http://www.incompleteideas.net/book/ebook/node66.html>