Heriot-Watt University

Masters Thesis

# Formal Verification of Neural Networks in Go

*Author:*
Arran Dinsmore

*Supervisor:*
Ekaterina Komendanskaya

*A thesis submitted in fulfilment of the requirements*
*for the degree of MSc. Robotics*

*in the*

School of Electrical, Electronic & Computer Engineering

School of Engineering & Physical Sciences

&

LAIV: Lab for AI Verification

March 2021

# Declaration of Authorship

I, Arran Dinsmore, declare that this thesis titled, 'Formal Verification of Neural Networks in Go' and the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: Arran Dinsmore

Date: March 2021

*“**Mistakes** are the portals of **discovery**.”*

James Joyce

# *Abstract*

As machine learning for safety critical applications such as autonomous vehicles are starting to be developed beyond proof of concepts, and enter into production within society, there is a need to ensure these systems do not fail.

Traditional rigorous testing methods are not a viable approach for such black box systems, and thus a need for formal verification methods that can prove the robustness of a system are required.

Additionally, the choice of programming language used for these tasks has grown with new machine learning extensions being developed on existing languages, and newly created languages dedicated for the use within machine learning.

This paper will investigate the feasibility of formal verifcation for a selection of the most commonly used languages within machine learning tasks, and aim to develop an understanding of generalised approaches and best practices for proving the integrity of systems that cannot afford to fail.

# *Acknowledgements*

The acknowledgements and the people to thank go here, don't forget to include your project advisor :)

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| ML | Machine Learning. 1–4, 8, 9 |
| NN | Neural Network. 2, 4 |
| PYPL | PopularitY of Programming Language Index. 6 |

# Symbols

| | | |
|---|---|---|
| $a$ | distance | m |
| $P$ | power | W (Js$^{-1}$) |
| | | |
| $\omega$ | angular frequency | rads$^{-1}$ |

*For/Dedicated to/To my...*

# Chapter 1

# Introduction

## 1.1 Context

Machine Learning (ML) algorithms are becoming increasingly present in systems that operate within shared environments with humans, or involve direct interaction with humans themselves [Pereira and Thomas, 2020]. These systems are often defined as safety-critical, such that their failures lead to unintended and potentially harmful behaviours [Amodei et al., 2016]. Examples of these systems include autonomous automotive systems, traffic control systems, medical devices, aviation software, industrial robotics, and many more cyber-physical systems that interact with our environment. Many of these systems have so far only existed as proof of concepts, but are steadily approaching commercial use within our society. Consequently, the testing and verification of ML for the use of controlling safety-critical systems has become a focused area of research in recent years.

This thesis will use the following definitions of testing and verification. Software testing, defined as the evaluation of a system under various conditions and observing its behaviour while looking for defects [Pereira and Thomas, 2020], is commonly used in ML development to ensure that a trained model generalises accurately to some previously unseen test data. Verification is defined as the process of determining whether the products of a phase of the software development process fulfill the requirements established during the previous phase [Ammann and Offutt, 2008]. Formal verification in other words, produces logical arguments that a system will not act abnormally under a wide range of circumstances, and can be used to determine not only generality, but also the robustness of a system.

The challenges regarding verification of ML models stem from the typically less deterministic and more statistically-oriented nature of their algorithms, which lead to a lower degree of understanding than software that is explicitly programmed to perform a specific task [Bishop, 2006]. These types of systems are commonly referred to as *black box* systems, where the internal mechanisms are not revealed; in other words, it is impossible to understand a model just by looking at its parameters [Molnar, 2019]. Software testing, defined as the evaluation of a system under various conditions and observing its behaviour while looking for defects [Pereira and Thomas, 2020], is commonly used in practice to ensure that a trained model generalises accurately to some previously unseen test data.

Additionally, recent research has exposed broad vulnerabilities within data driven ML algorithms, including Neural Networks (NNs); where applying small but intentional perturbations to an input which are not noticible to humans, can lead to a model outputting an incorrect classification with high confidence. [Goodfellow et al., 2014]. An example of such an attack can be seen in *Fig. 1.1* below.

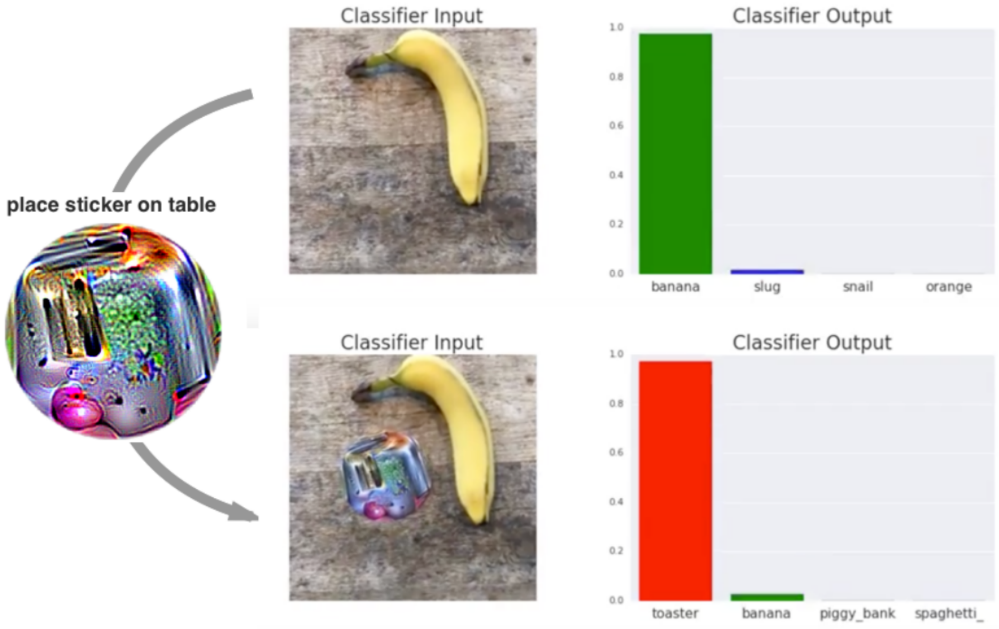


FIGURE 1.1: **Google's Adversarial Patch** – An example of a method to create targeted adversarial attacks on NNs by adding carefully designed noise via a physical patch [Brown et al., 2018].

Various languages have been developed over time that have either been exetended to, or have been intentionally designed to handle formal approaches to verifying software (Ada, Haskell etc.). However, few of these languages have been adopted by the ML community, and therefore there is a trade off decision that needs to be made; choose a language that is efficient and easy for developers to create ML models, or a language that allows developers to produce robust and verified models.

## 1.2 Motivation

Software engineers are often limited to what programming languages they can choose from to develop a project. This could be due to the constraints of a wider system used by their employer, or that their preferred language is not suitable for a specific task. This is especially true for developing ML that can be formally verified, partly because of the relatively new area of research, but also because languages adopted for ML are not as well equipped for formal verification methods. Therefore, there is a need for a generalised approach for verifying ML models accross a wider range of programming languages.

Talk about the recent machine learning advancements in Go, and that it has shown to be used for machine learning in infrastructure, and could be adopted as a language for verification of neural networks.

## 1.3 Aims & Objectives

# Chapter 2

# Background & Literature Review

This chapter will provide an introduction to the important concepts that are required by this thesis. This includes a high-level overview of programming languages that are used for ML tasks, as well as a discussion of their popularity and usage within industry and academia.

An introduction to formal verification tools and their use for verifying NNs, along with a discussion about the importance of developing tools across a wide range of programming languages will conclude this chapter.

## 2.1 Programming Language Rankings and Metrics

The following section outlines the various programming language ranking systems and metrics used to determine the popularity of, and overall usage of programming languages. A combination of the following rankings will be used to assess the current trends for each programming language discussed in this paper.

### 2.1.1 RedMonk Programming Language Rankings

RedMonk is a developer-focused industry analyst firm that curates a quarterly ranking of programming languages. The rankings are created by looking at a programming language's presence on GitHub and Stack Overflow, attempting to reflect both the usage of the language from GitHub, and the amount of discussion regarding a language from Stack Overflow [O'Grady, 2021].

This ranking uses a simple metric that can be used for gaining an overview of a programming language's popularity within industry. However, due to quantifying discussions from Stack Overflow, older languages with a small set of built-in library functions such as C can be at a disadvantage compared to newer languages and those with a large set of built-in library functions [Benson, 2017].

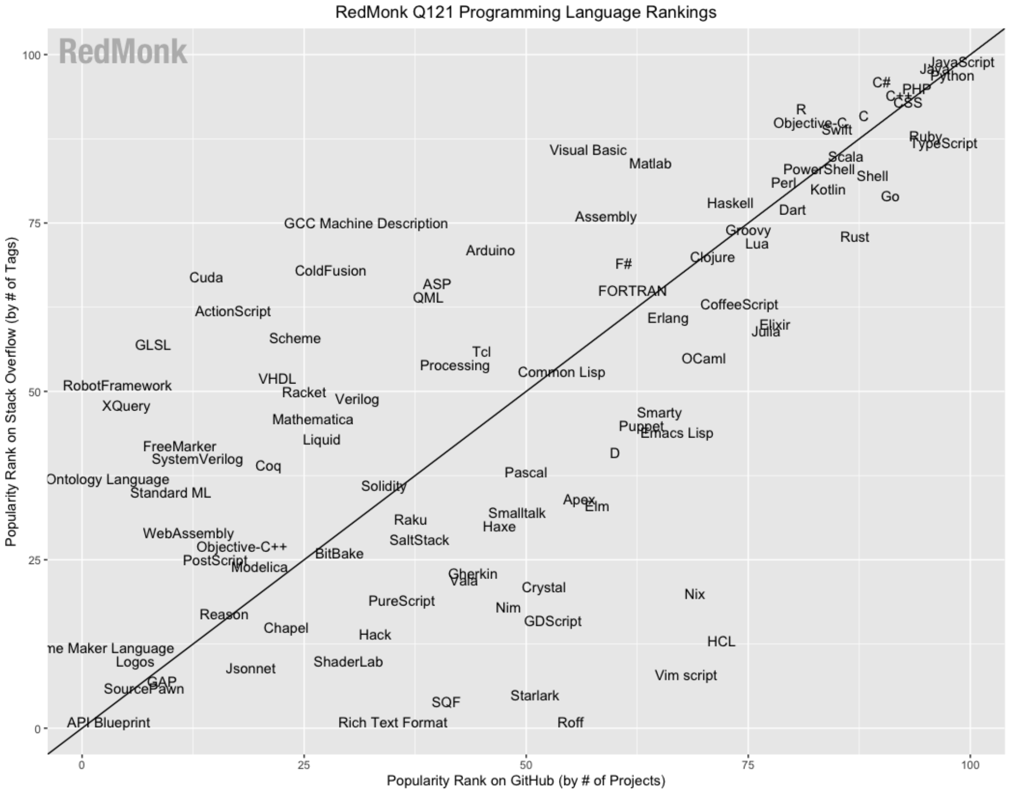


FIGURE 2.1: **RedMonk Rankings Q121** – Comparing the presence of programming languages from GitHub and Stack Overflow [O'Grady, 2021].

*Fig. 2.1* shows an example of the current visualisation tools available from this index.

### 2.1.2 PYPL PopularitY of Programming Language Index

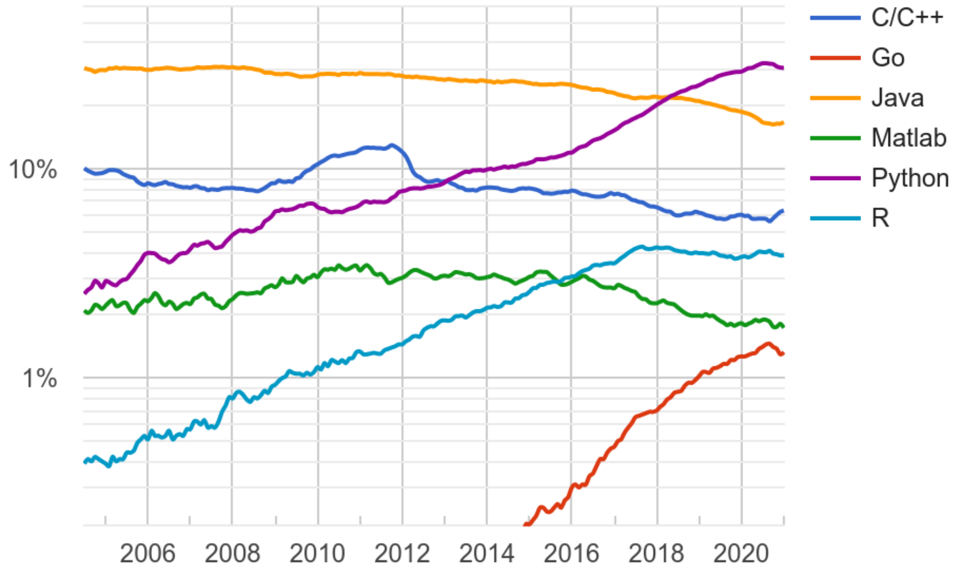
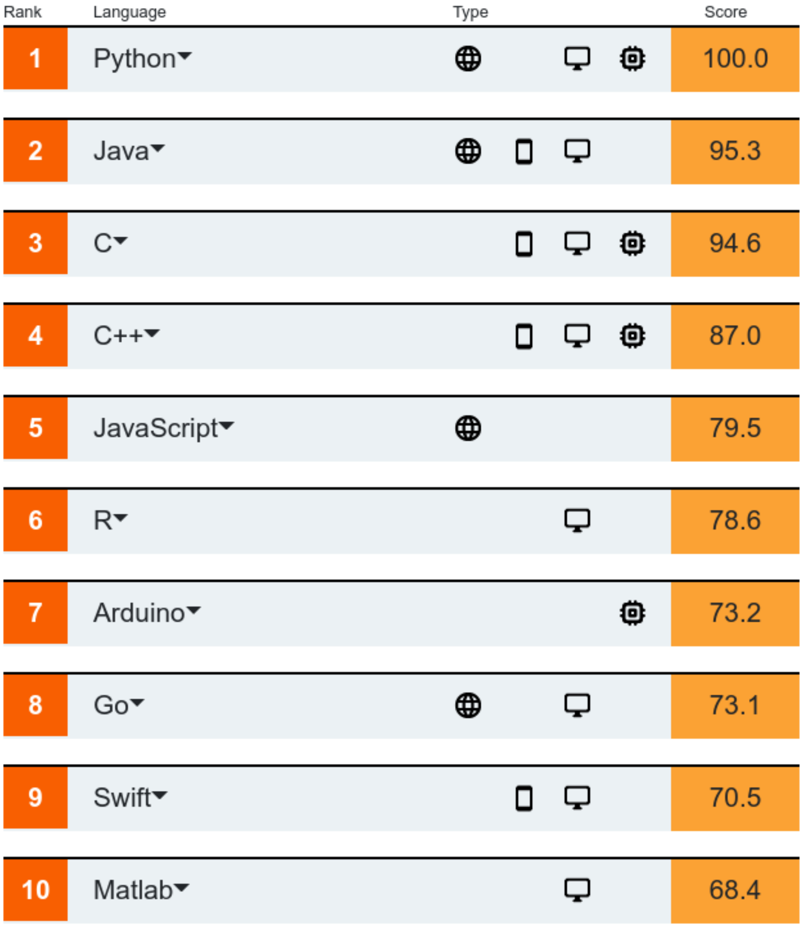


FIGURE 2.2: **PYPL Rankings** – An overview of the share of language tutorial Google searches over the last 15 years [Carbonelle, 2021]. *This chart uses a logarithmic scale.*

According to the PopularitY of Programming Language Index (PYPL), which uses the amount in which a language tutorial is searched on Google as a metric for popularity, Python has been the most popular language for the last few years (*see Fig. 2.2*), currently sitting on 30.44% of tutorial searches [Carbonelle, 2021].

Several reasons could be contributing to this popularity, including its versatility across a wide range of tasks, being platform agnostic, and easy to learn for beginners due to a large community and *readable* syntax. However, PYPL does not look at either the performance of languages or their popularity within specific tasks such as machine learning.

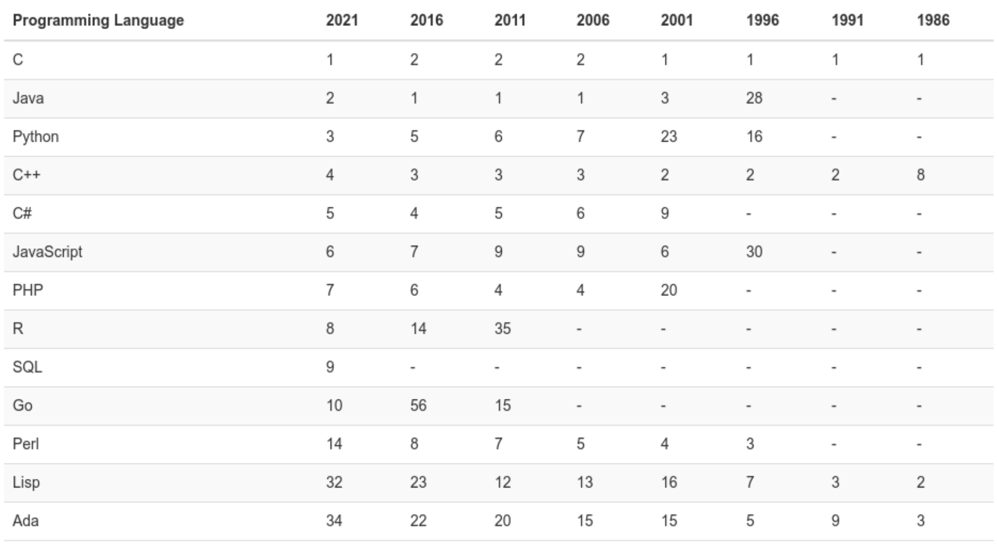
### 2.1.3 IEEE Spectrum Ranking of Programming Languages



| Rank | Language | Type | Score |
|---|---|---|---|
| 1 | Python▾ | 🌐 🖥 ▣ | 100.0 |
| 2 | Java▾ | 🌐 📱 🖥 | 95.3 |
| 3 | C▾ | 📱 🖥 ▣ | 94.6 |
| 4 | C++▾ | 📱 🖥 ▣ | 87.0 |
| 5 | JavaScript▾ | 🌐 | 79.5 |
| 6 | R▾ | 🖥 | 78.6 |
| 7 | Arduino▾ | ▣ | 73.2 |
| 8 | Go▾ | 🌐 🖥 | 73.1 |
| 9 | Swift▾ | 📱 🖥 | 70.5 |
| 10 | Matlab▾ | 🖥 | 68.4 |

FIGURE 2.3: **Top 10 Overall IEEE Spectrum Language Ranking** – Default metrics ranking top 10 programming languages [Diakopoulous et al., 2020].

### 2.1.4 TIOBE



| Programming Language | 2021 | 2016 | 2011 | 2006 | 2001 | 1996 | 1991 | 1986 |
|---|---|---|---|---|---|---|---|---|
| C | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| Java | 2 | 1 | 1 | 1 | 3 | 28 | - | - |
| Python | 3 | 5 | 6 | 7 | 23 | 16 | - | - |
| C++ | 4 | 3 | 3 | 3 | 2 | 2 | 2 | 8 |
| C# | 5 | 4 | 5 | 6 | 9 | - | - | - |
| JavaScript | 6 | 7 | 9 | 9 | 6 | 30 | - | - |
| PHP | 7 | 6 | 4 | 4 | 20 | - | - | - |
| R | 8 | 14 | 35 | - | - | - | - | - |
| SQL | 9 | - | - | - | - | - | - | - |
| Go | 10 | 56 | 15 | - | - | - | - | - |
| Perl | 14 | 8 | 7 | 5 | 4 | 3 | - | - |
| Lisp | 32 | 23 | 12 | 13 | 16 | 7 | 3 | 2 |
| Ada | 34 | 22 | 20 | 15 | 15 | 5 | 9 | 3 |

FIGURE 2.4: **TIOBE Very Long Term History Programming Language Index** – Long term history of programming languages, average positions for a period of 12 months [Jansen, 2021].

## 2.2 Overview of Programming Languages

A plethora of programming languages have been developed over time, some dedicated and others extended for the purpose of ML. This section will look at some of the most commonly used languages within the field of ML, and provide an overview of their strengths and weaknesses.

### 2.2.1 Python

Python is an interpreted, object-oriented, high-level, dynamically typed programming language with dynamic semantics. It was initially designed in 1991 by Guido Van Rossum and subsequently developed by Python Software Foundation. Python's simple syntax emphasising readability was the main purpose of its creation, making it very attractive for Rapid Application Development and reducing the cost of program maintenance [Python, 2021].

use in machine learning

Many versions of Python have since been released, and over time has seen the development of an extensive collection of community libraries used for a wide range of tasks. This has made Python one of the most versatile languages available today. Amongst these tasks, Python has become one of the most popular languages for ML and data science.

strengths

weaknesses

### 2.2.2 C

### 2.2.3 CPP

### 2.2.4 Matlab

### 2.2.5 Go

According to PYPL it has had the greatest increase in popularity since its release in 2015.

A relatively new language compared to the others used in machine learning.

Still in development, especially wrt machine learning tasks.

fast, concurrent, used a lot in system architecture and web applications.

**Gorgonia**
**Gorgo**
**Go-ML**

## 2.3 Formal Verification

Complexity of neural networks

### 2.3.1 Background

### 2.3.2 Z3

#### 2.3.2.1 Go-Z3

### 2.3.3 Sapphire

# Chapter 3

# Methodology

# Chapter 4

# Implementation

# Chapter 5

# Analysis

Chapter 6

# Conclusions

## Appendix A

# Appendix Title Here

Write your Appendix content here.

# Bibliography

Ammann, P. and Offutt, A. J. (2008). Introduction to software testing.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., and Mané, D. (2016). Concrete problems in AI safety. *CoRR*, abs/1606.06565.

Benson, D. (2017). 10 most popular programming languages - open source for you. https://www.opensourceforu.com/2017/03/most-popular-programming-languages/. (Accessed on 03/09/2021).

Bishop, C. (2006). *Pattern Recognition and Machine Learning*, volume 16, pages 140–155.

Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. (2018). Adversarial patch.

Carbonelle, P. (2021). Pypl popularity of programming language index.

Diakopoulous, N., Bagavandas, M., Singh, G., and Kulkarni, P. (2020). Interactive: The top programming languages 2020 - ieee spectrum. https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020. (Accessed on 03/09/2021).

Goodfellow, I., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv 1412.6572*.

Jansen, P. (2021). index — tiobe - the software quality company. https://www.tiobe.com/tiobe-index/?20210304%20%20%20%20%20%20. (Accessed on 03/09/2021).

Molnar, C. (2019). *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/.

O'Grady, S. (2021). The redmonk programming language rankings: January 2021.

Pereira, A. and Thomas, C. (2020). Challenges of machine learning applied to safety-critical cyber-physical systems. *Machine Learning and Knowledge Extraction*, 2(4):579–602.

Python (2021). What is python? executive summary — python.org. https://www.python.org/doc/essays/blurb/. (Accessed on 03/10/2021).