Turing Completeness of Neural Networks

D'Ambrosi Denis
dambrosi.denis@spes.uniud.it
147681

February 27, 2023

Abstract

Neural networks are a powerful machine learning technique that can be applied to a variety of applications, from image classification to natural language processing. At a basic level, neural networks consist of interconnected nodes that transmit and locally process data, but this seamingly simple architecture allows for an impressive degree of adaptability. Since nowadays they are broadly used to solve virtually any form of task, we must question they actual expressive power: are they actually Turing complete? In this paper, we will examine neural networks' Turing completeness and how it affects how they can be used for computation.

1 Introduction

> *A neural network is the second best way to solve any problem. The best way is to actually understand the problem.*
>
> *Unknown*

In 1989, Cybenko [quote] showed that for each continuous function there exists at least one neural network with a single hidden layer capable of approximating it to arbitrary accuracy. This result is significant because it implies that neural networks can be used to model a wide range of complex functions, including those with non-linear relationships between input and output variables. Multiple variations to the standard architecture have been proposed and implemented (for example increasing the number of layers, including skip connections and adding loops to the computational graph) during the years, but we must assess whenever these alternatives actually increase the expressive power of the basic topology and if so, where is the boundary of this data structure's computability power. The current essay is structured in the following way: the rest of this section will introduce the foundamental concepts and notation for the rest of the material. In section 2 [quote] we'll analyze the Turing completeness of (recurrent) neural networks from a theoretical perspective, without caring about

actual implementability of the systems described. In the following section we will instead take a look at concrete architectures that were proposed to simulate memory-bounded Turing machines. Finally, in the conclusions, we'll sum up the results and present the most obvious future work within this research field.

1.1 Turing Machines

Turing machines are a fundamental concept in the theory of computation, introduced by the mathematician Alan Turing in the 1930s [quote]. They provide a formal definition of what it means to compute a function, and have been instrumental in advancing our understanding of the limits of what can be computed by a mechanical process.

A Turing machine consists of a tape divided into discrete cells, a read/write head that can move along the tape, and a set of rules that govern how the head interacts with the tape. The tape is initially populated with a finite sequence of symbols, and the machine is in a particular (starting) state. At each step, the machine reads the symbol under the head, performs a specified action (such as writing a new symbol, moving the head left or right, or changing its state), and then moves to the next cell on the tape. The output of the machine is determined by the final state and the symbols on the tape.

Formally, a Turing machine can be represented using a quadruple

$$(Q \cup \{q_{\text{accept}}, q_{\text{reject}}\}, \Gamma, \delta, q_0) \tag{1}$$

where:

- Q is a finite set of states.
- Γ is a finite set of tape symbols, which includes the blank symbol #.
- δ is a transition function that maps $Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, where L and R represent moving the head left or right on the tape.
- $q_0 \in Q$ is the initial state.
- $q_{\text{accept}} \in Q$ is the accepting state.
- $q_{\text{reject}} \in Q$ is the rejecting state.

Before continuing, it's best that we also define the concept of istantaneous description of a Turing machine, since it will represent the starting (and ending) point of a simulation cycle by the neural networks that will be constructed.

An instantaneous description of a Turing machine is a snapshot of its current state. It provides a complete description of the machine's configuration at a given point in time, including the contents of the tape, the position of the read/write head, and the current state of the machine.

Formally, an instantaneous description can be represented as a 3-tuple (q, l, r), where:

- q is the current state of the machine.
- l is the contents of the tape to the left of the read/write head.
- r is the contents of the tape to the right of the read/write head.

For example, if a Turing machine is in state q_0, with the tape contents "00101" and the read/write head positioned over the first symbol, the instantaneous description can be represented as $(q_0, \epsilon, 00101)$, where ϵ represents the empty string.

The instantaneous description of a Turing machine is important because it allows us to reason about its behavior at a particular point in time. By examining the current state and tape contents, we can determine which transition the machine will take next, and how it will update its configuration. This allows us to analyze the computation of the machine step by step, and to understand how it processes input and produces output.

1.2 Neural Networks