Studying Knot Theory with Deep Learning

Ben Jacobsohn

17 May 2023

Abstract

In this paper I apply machine learning methods to the task of predicting knot invariants. I encode knots by their braid representations and apply various neural network architectures to predicting knot quasipositivity and coefficients of the Jones polynomial. A dense neural network is able to predict knot quasipositivity using only its braid word representation, extending the results of Mark Hughes. Different flavors of recurrent neural networks also succeed in approximating coefficients of the Jones polynomial. These and similar neural network architectures therefore show potential to predict the hyperbolic volume of knots or predict knot equivalence directly.

1 Introduction

Though knot theory is based on very concrete and intuitive ideas about knots in our physical world, the study of mathematical knots can be quite abstract and mysterious. Knot theory has shown deep connections to other fields including statistical mechanics, topology, and even quantum field theory. Yet a fundamental problem in knot theory, determining when two knots have the same structure, remains out of reach of effective algorithms for all but the simplest case. Recently, some researchers have been using machine learning to study different properties of knots, and have shown potential to make significant progress in areas where traditional mathematical analysis has struggled. This project aims to study different ways that machine learning can be applied to the analysis of knots, and what intuition about knots can be gained based on the effectiveness of different neural network architectures at predicting knot properties.

2 Knot Theory

2.1 Mathematical Knots

A mathematical knot is defined simply as an embedding of a circle in $\mathbb{R}^3$. The circle can be smoothly twisted around itself but cannot have any self in-

tersections. Two knots are equivalent if $\mathbb{R}^3$ can be continuously deformed onto itself while mapping one knot onto the other.

2.2 Knot Descriptions

While knots exist in $\mathbb{R}^3$, it is often useful to visualize them in two dimensions. A knot in $\mathbb{R}^3$ can be projected into the plane to show the full structure of the knot all at once. The only information that must be added is the knot crossings, where crossed strands must be shown to be above or below each other in a consistent fashion.

Knots are commonly described by the number of crossings. Alexander-Briggs notation labels knots by their crossing number along with an arbitrary order as a subscript, as seen below. The trivial knot with no crossings is known as the unknot.

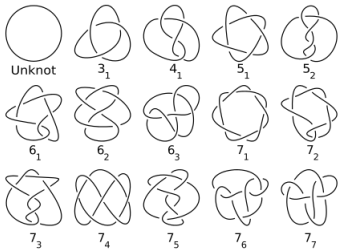

Figure 1: Knots labeled with Alexander-Briggs notation.

Every knot can also be described based on a braid. Braids are a collection of strands running in one direction, along with a list of crossings between adjacent strands. A braid can be turned into a knot by simply connecting the ends of the strands back together.

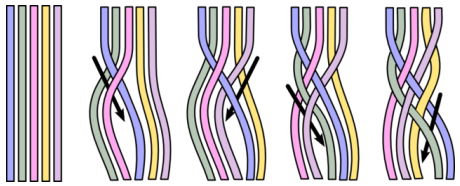

Figure 2: A braid with five strands.

2.3 Knot Equivalence

All equivalent knot diagrams can be transformed into each other through a sequence of Reidemeister moves. These are simple moves that operate on a local piece of a knot diagram. The moves are I: twist, II: poke, and III: slide, numbered based on the number of strands they operate on.

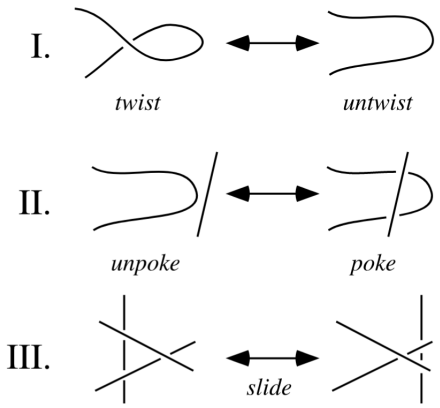

Figure 3: The three Reidemeister moves.

It is a fundamental problem in knot theory to determine whether two knots are topologically equivalent. There are currently no practical algorithms to test this directly. So, knots are often identified by various invariants. Invariants are properties of knots that do not change when the knot is deformed into different equivalent configurations. Showing that a property of a knot is unchanged by Reidemeister moves proves that the property is a knot invariant.

One simple knot invariant is called the Jones polynomial. It can be computed by defining the Jones polynomial of the unknot as 1, then recursively adding crossings to construct knots with polynomials given by a simple algebraic relation. This mean the Jones polynomial of any knot can be computed by a simple combinatorial algorithm, but with exponential time complexity.

Another knot invariant is hyperbolic volume. For some knots, the complement of the knot in the 3-sphere can be given hyperbolic structure. The hyperbolic volume of this manifold is an invariant of the knot. This can often be computed efficiently, but the algorithm for calculating it uses a limiting procedure which is not guaranteed to be finite, as some types of knots cannot have hyperbolic structure.

There is currently no way to directly compute whether two knots are equivalent. While equivalent knots always have the same invariants, having the same invariant does not guarantee that knots are equivalent. Theoretically, two equivalent knots can always be transformed into each other through a finite sequence of Reidemeister moves, but the upper bound on such a sequence is obscenely large and no effective algorithm to compute knot equivalence exists in practice.

3 Machine Learning

3.1 Neural Network Basics

Neural networks are a popular form of machine learning based loosely on biological brains which use networks of artificial neurons to approximate functions. Artificial neurons activate based on the activations of neurons they're connected to combined with a nonlinear activation function. They do this by multiplying the activations of neurons in the previous layer by the weights of their connections to the current layer, adding a bias term, then applying the activation function. In this way, any numerical inputs can be connected to layers of neurons which will transform the data into numerical outputs. These inputs and outputs can be encoded and decoded respectively to allow neural networks to perform a wide range of tasks.

3.2 Neural Network Architecture

The most basic form of a neural network uses dense layers, which are layers of neurons that each have weighted connections to all the neurons in the previous layer. In this project I also used recurrent networks, which can be applied to sequences of data. A recurrent layer takes an input sequence one element at a time and determines its activation based on the current element in the sequence, along with the layer's own activation from the previous element in the sequences. This lets the network learn patterns in data which depend on the order of a sequence. In this case, the sequence is the list of crossings in the braid word representation of a knot, in which the order is highly important. Recurrent layers can also return their activations after each element in the sequence, resulting in a list of activations instead of a single activation after the whole sequence has been processed. This list of activations can be fed into a dense layer, or used as a sequence in another recurrent layer. Different flavors of recurrent networks include GRU (Gated Recurrent Unit) and LSTM (Long Short-Term Memory) units, which implement the recurrent layer using different gates that learn how much of the layer's internal state to update based on new elements in the sequence and how much to forget based on its previous activation.

3.3 Training a Neural Network

For a neural network to accurately approximate a function, it must be trained on data so it can learn the proper weights and biases for its neurons. In this project, I use supervised learning, which learns from a set of training examples labelled with the correct output for each example. During training, the neural network is applied to a batch of these examples, and its performance is measured using a loss function which measures some form of the network's accuracy. The weights and biases of the neurons can then be updated based on each of their contributions to the overall loss of the network. This process is called

gradient descent, and functions by taking derivatives of each layer of neurons and using the chain rule to determine how the weights and biases affect the output loss, then use these derivatives to update the weights and biases for better performance. While the network is being trained, some data can be reserved as validation data. This means that the network does not train directly on the validation data, but uses it to test how well predictions can be generalized to new data. If the network's loss across training data improves but not across validation data, then the network is overfitting to the specific data it's being trained on, and is not effectively learning.

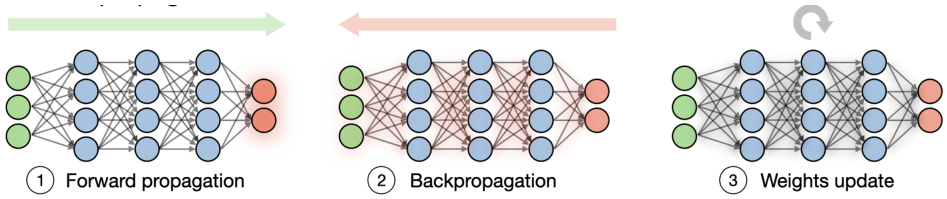

Figure 4: Training a simple neural network using backpropagation.

4 Experiments

4.1 Quasipositivity Prediction

Quaspositivity is a binary knot invariant. In *A Neural Network Approach to Predicting and Computing Knot Invariants* [1], Hughes used a simple neural network to predict knot quasipositivity. He used braid word representations of knots from the KnotInfo database up to 12 crossings, and constructed 32 equivalent braid words from each knot. He then included various pieces of classification data from the database into the input. From these inputs he used a straightforward neural network with two dense layers and a sigmoidal activation output for quasipositivity prediction. Based on this data, the trained neural network was able to correctly predict knot quasipositivity 99.3% of the time.

I reproduced the braid word generation and neural network used in this paper, but used a much smaller sample of knots up to 10 crossings. With these braid words and generated equivalent braids, I trained a similar neural network to predict quasipositivity with about 91% accuracy. Hughes also included 12 additional pieces of information about the knots in the network's input. However, one of the pieces of classification data was knot positivity, which is a stronger property than quasipositivity (all positive knots are quasipositive). With this single piece of classification data included, the neural network's prediction accuracy increased to 96%. This suggests that while some information about knot quasipositivity is encoded in the braid word representation of a knot, the

classification data that was introduced had a significant effect on the network.

4.2 Jones Polynomial Prediction

Based on the same braid word encoding as I used for quasipositivity prediction, I applied various neural network architectures to predicting the coefficients of the Jones polynomial. Since the degree of the Jones polynomial is bounded based on the number of crossings in a knot, I used a list of coefficients from the least to greatest power in the polynomial as the output. I use the 2977 knots up to 12 crossings in the KnotInfo database and generate equivalent braid words for a total of 76066 training samples. I first trained the same dense neural network as in the quasipositivity prediction on these samples, but could not recover the Jones polynomial coefficients. The loss was calculated based on mean squared error between the true and predicted lists of coefficients, which were normalized to have a standard deviation of 1. A simple recurrent network which used the sequence of crossings in each braid word did not improve past its initial random state. Networks using LSTM and GRU units were able to better predictions. Feeding the list of internal states from the GRU layer into a dense layer produced slightly better predictions, which were further improved by feeding this list into another GRU layer instead. Each network was trained until validation loss did not decrease by .01 across three network updates. The training and validation losses over each batch of samples can be seen in figure 5.

5 Future Directions

While recurrent neural networks showed some ability to predict Jones polynomials, it's possible that other neural network architectures could predict the coefficients with greater accuracy. The recurrent layers in the current network could be modified to multiply the inputs together with their hidden state instead of adding weighted multiples of them, which would more directly mirror the process of computing the Jones polynomial directly from the braid word representation of a knot. Attention and Transformer networks are also able to learn patterns in sequential data, and could be applied to this problem.

Another possible direction is to see what other knot invariants neural networks can accurately predict. The first invariant to test would be the hyperbolic volume of a knot, which requires a complex and potentially non-terminating algorithmic process to compute. Jejjala et al. have used neural networks to predict the hyperbolic volume of a knot from its Jones polynomial with high accuracy [2], so a neural network that can predict the Jones polynomial of a knot could potentially predict its hyperbolic volume as well.

Using the neural networks in this paper, knot equivalence could be predicted directly by predicting knot invariants and comparing them. Although it is possible for different knots to have identical invariants, this would allow the networks to compute some probability that two knots are equivalent. A natural

extension of this project is to instead predict knot equivalence directly from the representation of two separate knots.

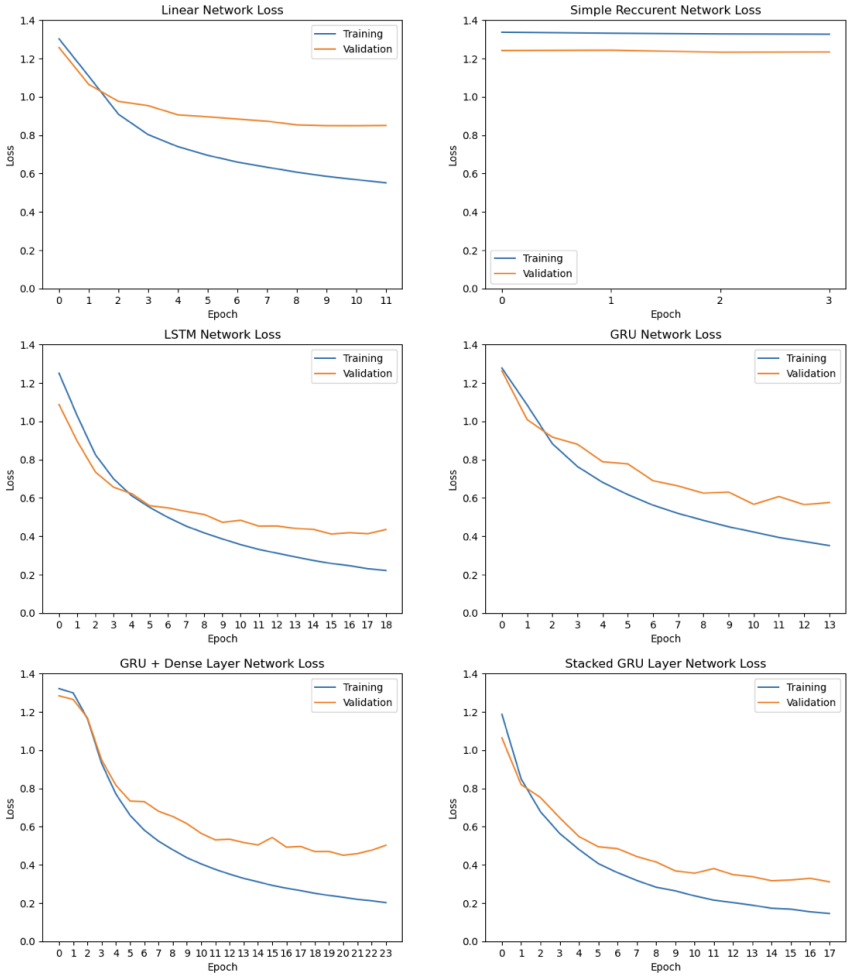

Figure 5: Training and validation loss for Jones polynomial coefficient prediction with different neural network architectures.

References

[1] Mark C. Hughes. "A neural network approach to predicting and computing knot invariants". In: (2016). arXiv: 1610.05744 [math.GT].

[2] Vishnu Jejjala, Arjun Kar, and Onkar Parrikar. "Deep learning the hyperbolic volume of a knot". In: *Physics Letters B* 799 (Dec. 2019), p. 135033. DOI: 10.1016/j.physletb.2019.135033. URL: https://doi.org/10.1016/j.physletb.2019.135033.