Learning Bridge Number of Knots: An Experiment with Kolmogorov-Arnold Networks (KANs)

Anand Shanker, Shawanwit Poomsa-ad, Benjamin Mudrak, Thi Hanh Vo

Introduction

In recent years, advancements in the field of Artificial Intelligence have democratized access to AI tools and related research. An effect of this democratization is the use of machine learning models (mainly neural networks) in adjacent fields such as math and physics. **This project focuses on applying and determining the efficacy of KANs to study the bridge numbers of knots.**

Mathematical Background

Mathematical knots are different from colloquial knots in the sense that their ends are joined, thus the knot structure cannot be unraveled. These knots are more precisely depicted with the following definitions:

Definition 1.1:
A knot is an embedding of a circle into Euclidean 3-space.

Definition 1.2:
A knot is a continuous map $\gamma: [a, b] \to \mathbb{R}^3$ such that $\gamma(a) = \gamma(b)$ and γ is injective on (a, b).

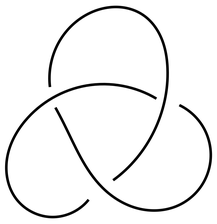

Figure 1: Planar Diagram of the Trefoil Knot

Definition 1.3:
A **Gauss code** is a sequence $(a_i)_{i=1}^{2n}$ such that $a_i \in \{\pm 1, \pm 2, \ldots, \pm n\}$ with each appearing only once.

Gauss Codes

To retrieve a gauss code from a given knot diagram:

- choose an arbitrary starting point and traverse around the knot in a chosen direction.
- On encountering a crossing, label it as a_1. If it is an under-crossing, let $a_1 = -1$. Else, let $a_1 = 1$.
- Repeat this process for each crossing
- terminate when you return to the starting point.

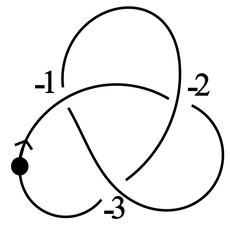

Figure 2: Labeled Trefoil Knot

Gauss codes are arguably the best way to describe a knot, but it is important to note that

- diagrams corresponding to the same gauss code (and hence, the same knot) can appear very different.
- Additionally, it is not true that every gauss code corresponds to a classical knot. For example, {1, -2, -1, 2} is not a classical knot.
- This motivated Kauffman [2] to define such non-realizable knots as **virtual knots** which form a correspondence to any given gauss code, up to **Reidemeister moves.**

Reidemeister Moves

Definition 1.4:
A Reidemeister move is some knot deformation classified as Type I, II, or III, colloquially known as a twist, a poke, and a slide (shown below). Two equivalent are equivalent knots if and only if their knot diagrams be deformed into the other by a sequence of Reidemeister moves. [3]

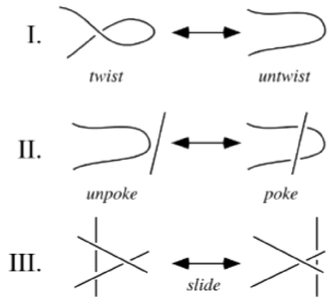

Figure 3: Illustration of Reidemeister moves from [3]

Bridge Numbers & Past Work

One important invariant of mathematical knots is the **Bridge Number,** which quantifies the minimal number of overbridges on a knot diagram. Hence, the bridge number can be thought of as encoding the structural property of a knot that is not visible from a particular diagram. Our objective is to study the bridge numbers of knots in relation to their gauss codes.

Definition 1.5:
Given a gauss code, a **strand** is a string of consecutive integers starting and ending at a negative integer. A strand containing at least one positive integer is called an **overbridge**.

Overbridges are parts of the knot diagram that start and end at under-crossings containing an overcrossing(s). Counter-intuitively, it is possible to have two diagrams corresponding to the same knot with different numbers over-bridges. This fact motivated the definition of the following invariant:

Definition 1.6:
The **bridge number** of a knot type K is the minimum number of overbridges over all gauss codes corresponding to knot type K.

Currently, given a gauss code (or knot diagram), it is difficult to determine the exact bridge number. However, algorithms exist to establish upper and lower bounds on the bridge numbers. In a recent paper, researchers [4] created a dataset by randomly generating gauss codes and selecting those which had a matching upper and lower bound.

Then, they applied standard machine learning techniques sourced from scikit-learn to classify 3 and 4 bridge knots of up to 16 crossings. Their best performing model was a Random Forest Classifier with an accuracy of 95.70% and an F1 score of 95.54%.

KANs & Applications

Kolmogorov-Arnold Networks (KANs) [5] are efficient and interpretable alternatives to Multi-Layer Perceptrons (MLPs). KANs have been shown to

- learn physical equations, solve partial differential equations,
- outperform MLPs in regression tasks,
- discover relations in knot invariants while unsupervised.

The neural scaling laws of a KAN are also far more efficient than those of an MLP, leading to small KANs being able to outperform large MLPs [5]. Thus, KANs seem promising for knot-invariant classification.

We tested numerous sizes of KANs and tuned hyperparameters to improve results. We measured accuracy, recall by bridge number, and balanced accuracy (the average of recall by bridge number) for each iteration of our models and tried to employ both classification and regression approaches.

Before applying any model architecture onto our data, we first had to manage a data imbalance. In our dataset, there were ~900,000 gauss codes corresponding to 3-bridge knots, and only ~200,000 gauss codes corresponding to 4-bridge knots. As such, we decided to oversample our 4-bridge data using **cyclic permutations** after splitting training and testing data. If we had the gauss code {1,-2,-1,2} and did 2 cyclic permutations, both {-2,-1,2,1} and {-1.2.1,-2} would get added to our training data.

KANs for Regression

Since we were using regression, we decided to establish a threshold, since our model would be predicting continuous values between 3 and 4. We decided that any predictions < 3.5 would be mapped to 3, and any ≥ 3.5 would be mapped to 4. We also tested on different thresholds to measure changes in accuracy. Our best performing model demonstrated the following results.

Threshold	Accuracy	Recall (Bridge 3)	Recall (Bridge 4)	Balanced Accuracy	F1-score (Bridge 4)
3.300	0.733	0.704	0.870	0.787	0.533
3.400	0.817	0.823	0.790	0.806	0.602
3.500	0.847	0.876	0.711	0.793	0.619
3.600	0.859	0.912	0.613	0.762	0.762
3.700	0.859	0.967	0.352	0.660	0.467

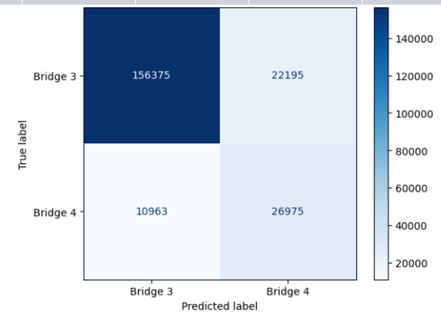

Figure 4: (Top) Table of regression model evaluation by horizontal threshold (Bottom) Confusion matrix of model at horizontal threshold 3.500

KANs for Classification

Our process for classification was generally the same as regression, with the only difference being the loss function. Our regression approach used a weighted mean-squared error loss function, while the classification approach employed a weighted binary cross entropy loss function. This allows the classification model to produce results between 0 and 1. 1 means that the model is confident the gauss code is 4 bridge, and 0 means the model is confident the gauss code is 3 bridge. Thus, thresholding in this example was done between 0 and 1. We see below that the best overall accuracy was 86.8%.

Threshold	Accuracy	Recall (Bridge 3)	Recall (Bridge 4)	Balanced Accuracy	F1-score (Bridge 4)
0.300	0.846	0.880	0.684	0.782	0.608
0.400	0.862	0.918	0.599	0.758	0.603
0.500	0.868	0.943	0.516	0.729	0.578
0.600	0.868	0.963	0.421	0.692	0.528
0.700	0.859	0.982	0.283	0.632	0.413

Figure 5: (a) Table of classification model evaluation by horizontal threshold

Conclusions & Future Work

- While both types of KAN applications produced similarly strong results when considering thresholding, we can tell by our overall accuracies that neither model was able to perform better than the models previously employed by earlier researchers [4].
- However, our work was constrained by session length and RAM limitations in Google Colab. For future researchers, it may be possible to employ larger KANs and fine-tune models to a greater degree to achieve stronger results

References

[1] H. Keese, An introduction to knot theory, 2018.
[2] L. Kauffman, "Virtual knot theory," European Journal of Combinatorics, 1999.
[3] E. W. Weisstein, Reidemeister moves, From MathWorld–A Wolfram Resource, 2025. Accessed: Sep. 29, 2025.
[4] P. Pongtanapaisan, H. Vo, and T. Nguyen, "Learning bridge numbers of knots," arXiv preprint arXiv:2405.05272, 2024.
[5] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "Kan: Kolmogorov–arnold networks," arXiv preprint arXiv:2404.19756, 2025, ICLR 2025.