Deep Learning in Knot Theory

Matt Ward
Advisors: Dr.Eric Rawdon, Dr.Scott Yilek, Dr.Rob Scharein
Summer 2019

Contents

1 Introduction 2

2 Neural Networks 3
  2.1 Describing a Neural Network as a Black-Box Algorithm . . . . . . . . 3
  2.2 The Structure of a Neural Network . . . . . . . . . . . . . . . . . . 4
  2.3 Neuron Mathematics . . . . . . . . . . . . . . . . . . . . . . . . . . 4
    2.3.1 Hidden Layers . . . . . . . . . . . . . . . . . . . . . . . . . . 5
    2.3.2 Output Layer . . . . . . . . . . . . . . . . . . . . . . . . . . 5
  2.4 The Learning Process . . . . . . . . . . . . . . . . . . . . . . . . . 5

3 Knot Classification Experiments 6
  3.1 Terminology and Key Concepts . . . . . . . . . . . . . . . . . . . . 6
  3.2 Original Experiment . . . . . . . . . . . . . . . . . . . . . . . . . 8
    3.2.1 Experiment Results . . . . . . . . . . . . . . . . . . . . . . . 9
  3.3 Initial Experiments With 64x64 Sized Images . . . . . . . . . . . . 9
    3.3.1 Perspective vs Orthographic . . . . . . . . . . . . . . . . . . 9
    3.3.2 Perspective vs Orthographic Experiment Results . . . . . . . . 11
    3.3.3 Contrasting Edge Colors . . . . . . . . . . . . . . . . . . . . 11
    3.3.4 Contrasting Edge Coloring Experiment Results . . . . . . . . 12
    3.3.5 Random Rotations . . . . . . . . . . . . . . . . . . . . . . . 13
    3.3.6 Random Rotation Experiment Results . . . . . . . . . . . . . 15
  3.4 Results From Initial Experiments . . . . . . . . . . . . . . . . . . 15
## 1 Introduction

The goal of our project was to see if neural networks could classify knots. The simplest case for such an experiment was to use 6 edge equilateral knots, of which there are only three kinds of knots: the Unknot, the -Trefoil and the +Trefoil. In this paper I will give a brief introduction to neural networks, followed by a discussion of the experiments we performed in an attempt to classify 6 edge equilateral knots with a neural network.
2 Neural Networks

2.1 Describing a Neural Network as a Black-Box Algorithm

A neural network at its simplest level can be described as a black-box algorithm. You can think of a neural network as having two primary operations, the training operation (the process in which the neural network learns) and the prediction...
operation. The training operation of neural network in a classification task takes as input vectorized data (for instance RGB pixel intensities in an image) and the data’s corresponding label (e.g. Cat, Dog, Hotdog...etc). The training operation of a neural network then uses the input data and corresponding label (truth value) to update the internal structure/state of the neural network so that the predicted output will be as close to possible as the true output (e.g. it should predict ”dog” and not ”cat” when it is fed an image of a dog). The prediction operation takes in data and predicts a label for the input data, based on its previous training experience (based on its current internal state).

2.2 The Structure of a Neural Network

![Neural Network Diagram]

The true internal structure of a neural network is that of a connected graph, where each vertex of the graph is referred to as a neuron. In a typical neural network there are three primary layer types: the input layer (which generally performs no mathematical operation, it simply stores the data), the hidden layers (the region where most of the mathematical ”magic” of a neural network takes place; the hidden layer’s job is to transform the input data into something the output layer can make sense of), and the output layer (the layer that predicts to which class a particular instance from the input dataset belongs).

2.3 Neuron Mathematics

Each neuron in a neural network takes a weighted sum of it’s input connections. Each neuron then either outputs this weighted sum, or applies a function to the weighted sum (called an Activation/Transfer Function) and outputs the result.
2.3.1 Hidden Layers

Hidden Layer neurons in a neural network typically employ a non-linear activation function on the weighted sum of their inputs. Some examples of non-linear activation functions that are typically used are:

### Activation Functions

- **Sigmoid**
  \[ \sigma(x) = \frac{1}{1+e^{-x}} \]

- **tanh**
  \[ \tanh(x) \]

- **ReLU**
  \[ \max(0, x) \]

- **Leaky ReLU**
  \[ \max(0.1x, x) \]

- **Maxout**
  \[ \max(w_1^T x + b_1, w_2^T x + b_2) \]

- **ELU**
  \[ \begin{cases} 
  x & x \geq 0 \\
  \alpha(e^x - 1) & x < 0 
  \end{cases} \]

In our neural networks we will be using the ReLU and LeakyReLU activation functions.

2.3.2 Output Layer

The output layer, otherwise known as the classification layer typically uses an activation function called the Softmax function. The classification layer (with a Softmax activation function) will output the probability that a given neural network input X belongs to each of the K possible classes (in our case Unknot, +Trefoil, -Trefoil). The class for which the network returns the highest probability of X belonging too, is the class that the network will predict X belongs to.

2.4 The Learning Process

Every edge (connection between neurons) in the neural network is assigned some initial weight (historically small random values have been used; random weight initialization). The neural network is continually fed data instances for which it will make a prediction. After the neural network has made predictions for some predefined number of data instances (called a mini-batch size), the error (percent of data instances misclassified) is calculated, and the neural network edge weights are updated in a fashion that tries to generate a lower misclassification rate during the next training iteration (mini-batch).

Source: Citations 1,2,5,6
3 Knot Classification Experiments

3.1 Terminology and Key Concepts

Throughout the following discussion of experiments performed this summer there are several terms and concepts that are important to be familiar with in order to understand our results.

1. Classification

- Is the process of identifying to which of a set of categories a new observation (e.g. a randomly selected image of a knot) belongs. For our experiments this means the following: If the neural network is given a randomly selected image or 3D model of a knot, it will attempt to correctly categorize it as an Unknot, +Trefoil, or -Trefoil. It makes this categorization with no knowledge of the true category of the observation.

2. Confusion Matrix and Classification Accuracy

Classification Accuracy = 85.33%

- A confusion matrix (above image) is a visual aid to understand how a classification algorithm is performing. The vertical labels are the true knot types and the horizontal labels are the predicted knot types.
If the confusion matrix looks like a diagonal matrix, only having non-zero elements along the diagonal, the classifier has 100% classification accuracy. All non-zero elements not along the top-left to bottom right diagonal are misclassifications.

• A classifier’s classification accuracy can be calculated by taking the sum of the diagonal of the confusion matrix, and dividing by the sum of the entire classification matrix. In this example this would be:

\[
\frac{2560}{3000} = 0.85333
\]  

(1)

We simply multiply this decimal by 100 to get a percent. \(0.8533 \times 100 = 85.33\%\) classification accuracy.

3. Training Dataset vs Validation Dataset

• Throughout this paper our ‘training dataset size’ will be discussed. In machine learning a person normally wants to produce a system that can generalize its predictive ability to unseen data. In order to estimate a model’s (in our case a neural network) ability to generalize to unseen data, we normally split the available data into ‘training data’ and ‘validation data’. The neural network is fed the training data with its true label (Unknot, T, -T etc..). The neural network then ‘trains’ by studying the patterns in this data (for which it knows the true label for every image). Once the neural network is done training, we measure the model’s ability to generalize to unseen data by having the neural network predict the knot type for every image in the ‘validation set’, and then calculating the validation data classification accuracy and confusion matrix. All of the reported accuracies and confusion matrices throughout our experiments are based off of validation data set (unseen data) performances.

4. Saliency Maps

• We can calculate the accuracy and confusion matrix of a classification task to understand our algorithm’s performance, but in the case of image classification how can you tell what aspects of an image a neural network is using to make its classifications? For instance: If you want to classify bird species based on images of birds, you would want to make sure that your neural network is making its classification based on pixels related to the birds themselves, and not simply taking advantage of the fact that bird species ‘A’ is sitting on a tree branch with green leaves, while bird species ‘B’ is sitting on a tree with purple leaves. To help visualize what aspects of an image a neural network is using in its classification task we can use a technique called a saliency map.

• To generate a saliency map we compute the gradient of an output category (all +Trefoil Knots) with respect to an input image (a single image of a knot). This shows us how the output (predicted category) changes
with respect to small changes in the input image. In the saliency maps throughout this paper, the brighter the region in the saliency map, the more this region of the original image is contributing to what category the neural network predicts the image to be.

Source: Citation 3

3.2 Original Experiment

The first experiment I performed was my implementation of the simplest experiment that had been performed last summer by the previous student researcher, Zach Sorenson. The images used in this experiment are $32 \times 32$ pixels and have no modifications made to them.

Image Size = $32 \times 32$ pixels

32 * 32 image of a Unknot  32 * 32 image of a +Trefoil  32 * 32 image of a -Trefoil

A -Trefoil knot and it’s associated Saliency map
3.2.1 Experiment Results

Original Experiment Classification Accuracy = 85.33%

After the initial experiment it was decided that the size of the image would increase to 64*64 pixels because it was very difficult for a human expert in knot theory to determine what kind of knots were present in the pictures at a resolution of 32*32; and a rule of thumb is that if a human expert cannot accomplish the classification task then machine learning models will not be able to either.

3.3 Initial Experiments With 64x64 Sized Images

3.3.1 Perspective vs Orthographic

We then decided to experiment with the camera orientation. Specifically we decided wanted to see whether a perspective view or orthographic view of the knot would result in higher classification accuracy. Using an orthographic projection means the images will be projected onto a 2D plane and objects are not affected by the angle from which you seem them.
Notice how it is easier to perceive depth in the perspective view of the rectangle, and in the orthographic view of the rectangle it looks like it has been squashed.

An image of an Unknot from a perspective view on the left, and an orthographic view on the right. Like the rectangle, notice the right side of the rightmost knot (orthographic camera view) looks squashed (projected) onto the same plane as the left side of the image. **Note these two knot images are very high resolution.

The above images are 64×64 pixel images of the Unknot from both a perspective and orthographic camera view, and their associated salience maps.
3.3.2 Perspective vs Orthographic Experiment Results

Perspective View Classification
Accuracy = 89.76%

Orthographic View Classification
Accuracy = 90.06%

Our experiment results show us that increasing image size from 32 * 32 to 64 * 64 improves classification accuracy, and that an orthographic camera view slightly outperforms a perspective view.

Source: Citations 7,8

3.3.3 Contrasting Edge Colors

After experimenting with the camera view, we decided to experiment with the coloring of the knots. The goal in experimenting with the coloring of the knots was to try and find a way to make it easier to perceive depth in the images. In knot theory the number of over or under crossings is important in determining what kind of knot you are looking at. We hypothesized that coloring the knot edges contrasting colors would add more depth to the images and therefore allow for the neural network to better recognize the locations of the knot crossings.

We decided that each edge of the knot should be a color that contrasts with its neighboring edges. In Red-Green-Blue color space the edges are colored as follow:

1. Edge 1 = 0.5/0/0
2. Edge 2 = 0.5/0.5/0
3. Edge 3 = 0/0.5/0
4. Edge 4 = 0/0.5/0.5
5. Edge 5 = 0/0/0.5
6. Edge 6 = 0.5/0/0.5
Knots that have this edge coloring scheme look like this:

High Resolution Images of Knots With Our Edge Coloring Scheme

64 * 64 Image of an Unknot With Our Edge Coloring Scheme and its Associated Salience Map

3.3.4 Contrasting Edge Coloring Experiment Results

Colored Perspective View
Classification Accuracy = 95.06%

Colored Orthographic View
Classification Accuracy = 96.96%

From this experiment we can see that our edge coloring scheme significantly improves performance for both orthographic and perspective camera views. Colored
orthographic knots lead to a classification accuracy 11.63% higher than our base case, and colored perspective knots lead to a classification 9.73% better than our base case.

### 3.3.5 Random Rotations

After experimenting with knot color, we wanted to see what impact randomly rotating the knots about their x, y, and z axes would have on neural network knot classification performance. In the first three experiments performed, the knots in the images used had their major axis aligned towards the camera position. The following figures show an Unknot starting in its initial aligned position then undergoing a random rotation about its x, y and z axes.
The above images are examples of the images we used for the 4 random rotation experiments performed:

1. Randomly-Rotated with Perspective Camera View (RP)
2. Randomly-Rotated with Orthographic Camera View (RO)
3. Colored-Randomly-Rotated with Perspective Camera View (CRP)
4. Colored-Randomly-Rotated with Orthographic Camera View (CRO)
3.3.6 Random Rotation Experiment Results

<table>
<thead>
<tr>
<th>Experiment</th>
<th>RP</th>
<th>RO</th>
<th>CRP</th>
<th>CRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Accuracy</td>
<td>65.06%</td>
<td>63.29%</td>
<td>64.36%</td>
<td>62.76%</td>
</tr>
</tbody>
</table>

Random-Rotations Experiment Results

Results of the experiment with random rotations show that randomly rotating the knots significantly decreases classification accuracy. However, it was noted that there is no real difference in classification accuracy between the four experiments (RP, RO, CRP, CRO). This is likely due to the number of times a single experiment was performed being low. I think if each experiment (RO, RP, CRP, CRO) was performed 1000+ times you would see a difference in the classification accuracy of colored-randomly-rotated knots and non-colored-randomly-rotated knots.

3.4 Results From Initial Experiments

<table>
<thead>
<tr>
<th>Experiment</th>
<th>RP</th>
<th>RO</th>
<th>CRP</th>
<th>CRO</th>
<th>Persp</th>
<th>Ortho</th>
<th>CP</th>
<th>CO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Accuracy</td>
<td>65.06%</td>
<td>63.29%</td>
<td>64.36%</td>
<td>62.76%</td>
<td>89.76%</td>
<td>90.06%</td>
<td>95.06%</td>
<td>96.96%</td>
</tr>
</tbody>
</table>

64 * 64 Experiment Results

From our experiments it can be seen that colored-orthographic knots perform the best out of the four non-rotated experiments (Persp, Ortho, CP, CO). We can also conclude that randomly rotating knots decreases classification accuracy.

3.5 Experiments With 128x128 Colored Orthographic Randomly Rotated Knots

After the eight experiments performed at the 64 * 64 image size, we decided to increase the image size to 128 * 128 pixels. For our experiments with an image size of 128 * 128 pixels we decided to see if increasing the training set size could increase the classification accuracy of randomly rotated knots. For these experiments we decided to use colored-randomly-rotated-orthographic knots (CRO) due to our previous experiments showing us that colored-orthographic knots (CO) lead to the highest classification accuracy.
3.5.1 Initial Results

Initial 128 * 128 CRO Classification Accuracy = 61.97%

From our initial 128 * 128 pixel experiment it can be seen that increasing the image size from 64 * 64 pixels to 128 * 128 pixels did not significantly improve colored-randomly-rotated knot classification accuracy. However from the confusion matrix you can see that the neural network has no problem distinguishing between Unknot and Trefoil, but has significant trouble distinguishing between +Trefoil and -Trefoil.

3.5.2 The Need For More Training Data

We next decided to experiment with the amount of training data we were using to train the neural network. We found that any increase in training data size over the baseline 12,000 training images for colored-randomly-rotated-orthographic knot
classification resulted in an increase in classification accuracy. The most notable result is that it took 32 times the baseline amount of training data to get classification accuracy of colored-randomly-rotated-orthographic knots (96.43% trained on 384,000 images) comparable to that of colored-orthographic knots (96.96% trained on 12,000 images).

<table>
<thead>
<tr>
<th>Training Set Size</th>
<th>12K</th>
<th>48K</th>
<th>240K</th>
<th>384K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Accuracy</td>
<td>61.97%</td>
<td>68.30%</td>
<td>87.70%</td>
<td>96.43%</td>
</tr>
</tbody>
</table>

Summary of CRO Knot Classification Accuracy at Various Training Data Amounts

3.6 Ongoing Experiments

3.6.1 Semi-Supervised Learning With Deep-Convolutional-Generative-Adversarial-Networks

The goal of this experiment was to produce a faster neural network that produces similar or greater classification accuracy compared to our previous results, while requiring fewer real images of knots.

3.6.2 What is a Generative Adversarial Network?

A Generative Adversarial Network (GAN) is a neural network architecture developed by Ian Goodfellow that takes some ideas from Game Theory. Our GAN has the following general architecture:
1. A Generator Network

- Our Generator Network is fed a vector containing random noise, and then proceeds to up-sample the vector using Transposed Convolution. At the end of the neural network training process, this up-sampled random vector will come out of the Generator Network looking like ”fake” knot images.

2. A Discriminator Network

- A Discriminator Network is a classification network. In our case it classifies images into four classes / categories: Unknot, +Trefoil, -Trefoil, Synthetic Image of Knot.

3. The Training Process

- The training goal of the Generator Network is to Generate synthetic images of knots that trick the discriminator into misclassifying them as real images of knots. The training goal of the Discriminator Network is to get as good as possible at always classifying synthetic images of knots as synthetic images of knots. These two networks (Generator and Discriminator) are trained (weights are optimized) synchronously and are not independent of one another. By the end of training, the Discriminator network can produce very photo-realistic synthetic images. Adding the fourth class (synthetic images) to the classification problem will result in the Discriminator Network being pushed to a higher classification accuracy without having to feed the neural network additional real images of knots.

Here are some pictures of fake/synthetic (produced from random noise by Generator Network) Unknot images generated by our Knot-GAN:
This is still a work in progress (early stages), but early experiments with just 12,000 real training images are producing a classification accuracy around 90%.

Source: Citation 9

3.7 3D-Convolutional-Neural-Networks

The final experiment I performed this summer was to try and classify knots based on their 3D shape. I used 3D-voxelized knot surfaces that look like this:

![Voxelized Unknot](image1)
![Voxelized -Trefoil](image2)
![Voxelized +Trefoil](image3)

These voxelized knots sit in the center of a $64 \times 64 \times 64$ voxel grid.
3.7.1 3D-Convolutional-Neural-Network Experiment results

3D-Voxelized-Knot Classification Accuracy = 93.37%

The 3D-knot classification technique works well, but the models in the above experiment are smooth knots. Smooth knots were used in this experiment because currently the equilateral-edge knot format is not supported by OBJ files (3D-knot file format). This problem will be resolved in the future.

4 Summary of Knot Classification Experiments

4.1 Initial Experiments

We began our experiments with our knots having their major axis aligned to the camera. We did four experiments with aligned knots: perspective camera view (P), orthographic camera view (O), colored-perspective (CP), and colored-orthographic (CO). These experiments showed us that colored-orthographic images of knots are the best performing and lead to a 7.2% increase in classification accuracy over baseline results.
We then began experimenting with random rotations of knots. The four experiments we began with were randomly-rotated perspective (RP), randomly-rotated orthographic (RO), colored-randomly-rotated perspective (CRP), colored-randomly-rotated orthographic (CRO). From these experiments we concluded that randomly rotating the knots decreases classification accuracy. However we saw no real difference in the classification accuracy of the four randomly rotated knot experiments.

<table>
<thead>
<tr>
<th>Experiment Name</th>
<th>P (Baseline)</th>
<th>O</th>
<th>RO</th>
<th>RP</th>
<th>CP</th>
<th>CO</th>
<th>CRP</th>
<th>CRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Accuracy</td>
<td>89.76%</td>
<td>90.06%</td>
<td>63.29%</td>
<td>65.06%</td>
<td>95.06%</td>
<td>96.96%</td>
<td>64.36%</td>
<td>62.76%</td>
</tr>
<tr>
<td>Improvement Over Baseline</td>
<td>0%</td>
<td>+0.3%</td>
<td>-29.47%</td>
<td>-24.7%</td>
<td>+5.3%</td>
<td>+7.2%</td>
<td>-25.4%</td>
<td>-27%</td>
</tr>
</tbody>
</table>

Results For Our Initial Experiments (Image Size = 64 * 64 pixels)

4.2 Training Dataset Size Experiments

After the eight initial experiments we decided to experiment with training dataset size to see if increasing the amount of data we trained the neural network on would increase the classification accuracy of colored-orthographic-randomly-rotated (CRO) knots. The baseline amount of training data was 12,000 images. We found that any increase in training data size leads to an increase in classification accuracy, and using 32 times the baseline amount (using 384,000 images) lead to a high knot classification accuracy of 96.43% which is a 34.36% increase in classification accuracy over baseline for CRO knots.

<table>
<thead>
<tr>
<th>Experiment Name</th>
<th>12K (Baseline)</th>
<th>48K</th>
<th>240K</th>
<th>384K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Accuracy</td>
<td>61.97%</td>
<td>68.30%</td>
<td>87.70%</td>
<td>96.43%</td>
</tr>
<tr>
<td>Improvement Over Baseline</td>
<td>0%</td>
<td>+6.33%</td>
<td>+25.73%</td>
<td>+34.36%</td>
</tr>
</tbody>
</table>

Results For Our Experiments With Colored-Orthographic-Randomly-Rotated Knots (Image Size= 128 * 128 pixels)

4.3 Knot Classification Conclusion and Future Work

In conclusion, we found that knots with our edge coloring scheme and an orthographic camera view are the best performing, and lead to a 7.2% improvement in classification accuracy over our baseline experiments. We also found that randomly rotating the knots decreases classification accuracy, however if you increase the amount of CRO knot training data to 384,000 images, you can get comparable accuracy between colored-orthographic knots (96.96% trained on 12K images) and colored-randomly-rotated-orthographic knots (96.43% trained on 384K images).

In the future we will continue to develop our 3D-Knot-Neural-Network and explore other promising techniques like Generative Adversarial Networks.
5 Minimum Rope Length

5.1 Background Information

This summer a second method to classify knots was also explored; specifically classifying knots by their minimum rope length (which is a knot invariant). In order to accomplish this 29 other knot invariants (gathered from https://www.indiana.edu/~knotinfo/) were used as the predictors for each knot’s minimum rope length. We attempted to predict the minimum rope length of knots with a crossing number of 10, after training machine learning models on invariant data from knots with a crossing number between 3 and 9.

The following definitions address terms used throughout the rest of the minimum rope length section.

**Knot Invariant**: An invariant is a property, held by a class of knots, which remains unchanged when transformations of a certain type are applied to the knots.

*Two knot invariants of interest*

**Crossing Number**: The crossing number of a knot is the smallest number of crossings of any diagram of the knot. Knots are often listed on tables by their crossing numbers

**Minimum Rope Length**: The minimal length of an ideally flexible rope that is needed to tie a given knot.

5.2 Problem Statements

Two different problems were investigated; multivariate regression to predict knots minimum rope-length and binning knots by their minimum rope length.

For the regression method the question being asked is:

*Can a Decision Tree or Neural Network predict the minimum rope length of knots with a crossing number of 10 after being trained to predict the minimum rope length of knots with a crossing number between 3 and 9?*

For the binning method the question being asked is:

*Can a neural network be used to rank knots with a crossing number of 10 by the magnitude of their minimum rope length, after being trained on invariant data from knots with a crossing number between 3 and 9?*

5.3 Regression With Decision Trees

Decision Trees are a machine learning algorithm that can perform both classification and regression tasks. One algorithm which can be used to train decision
trees is the CART (Classification and Regression Tree) algorithm. The algorithm is as follows:

1. The Decision Tree picks a single feature \( K \) (a single knot invariant in this particular experiment) and a threshold level \( t_K \).

2. When used for regression tasks (in this case predicting knots minimum rope length) the decision tree will split the dataset at a feature and threshold that best minimizes the cost function listed below. In the cost function \( m \) represents the total number of training examples and \( m_{\text{left}} / m_{\text{right}} \) represent the number of samples in the left and right child nodes of a specific node. \( \text{MSE}_{\text{left}} \) and \( \text{MSE}_{\text{right}} \) represent the mean squared error \( (1/m_{\text{node}} \cdot (\text{actual min rope length} - \text{predicted min rope length})^2) \) calculated at the leaf nodes of the left and right branch of the decision tree respectively.

\[
J(K, t_K) = \frac{m_{\text{left}} \cdot \text{MSE}_{\text{left}}}{m} + \frac{m_{\text{right}} \cdot \text{MSE}_{\text{right}}}{m}
\]

Example Decision Tree for Predicting Minimum Rope Length

One unique feature of decision trees is that they are fully interpretable. For instance in the above decision tree you can see that the tree predicted a minimum rope length of 43.865 if it saw a crossing number (knot invariant) that was less than 5.5, and a minimum rope length of 63.486 if the crossing number was less than 7.5 but greater than 5.5.

Decision Trees are a relatively weak machine learning model and have a tendency to seriously overfit the training data. This behavior can be seen in the plots below which show a decision tree performance when trying to predict minimum rope length. The decision tree is able to almost exactly predict the minimum ropelength of knots in the training dataset (knots with a crossing number between 3 and 9), but is totally
unable to predict the minimum rope length of the test dataset (knots with a crossing number of 10).

Model Performance on Training Data

![Predicted Cr(3)-Cr(9) Minimum Rope Lengths](image)

Model Performance on Testing Data

![Predicted Cr(10) Minimum Rope Lengths](image)
5.4 Regression With Neural Networks

After attempting to predict the minimum rope length of 10 crossing knots using decision trees and seeing the poor performance I decided to try and use neural networks because they often have a higher capacity to learn/recognize complex patterns in a dataset. The best performing neural network that I used was large, consisting of 31 layers and about 80 million trainable (learn-able) weights.

The neural network used is seen in figure below. The neural network has 5 input layers and 1 output node. The leftmost input to the neural network called input5 takes as input 25 of the 29 invariants for each knot. Each of the other four inputs take in a single invariant. The four invariants that have their own inputs are Crossing number, Alternating, Symmetry Type, and Small or Large. Each one of these four invariants initial representation is a 1-dimensional integer number. These four invariants each pass through their own ”embedding” layer. This layer learns to turn these integer encoded knot invariants into a vector of a predetermined size. This vector encoding allows the neural network to better learn the interaction between the knot invariants. Crossing Number is encoded as a 1-million element vector, Alternating as a 50,000 element vector, Symmetry Type as a 250 element vector, and Small or Large as a 200 element vector. These embedding layers played a vital role in the neural network’s ability to predict the minimum rope length of knots with a crossing number of 10.
Unlike the decision tree, the neural network performed quite poorly on the training data (knots with a crossing number between 3 and 9), but performed excellent on the test data (knots with a crossing number of 10). This is likely because the neural network was hand tuned (manual parameter optimization) in order to maximize performance on the test data (knots with a crossing number of 10) while ignoring the performance on the training data (knots with a crossing number between 3 and 9).

A plot of the training dataset performance and of the test dataset performance can be seen below.
Model Performance on Training Data

Predicted Cr(3)-Cr(9) Minimum Rope Lengths

- Blue line: Actual Minimum Rope Length
- Orange line: Predicted Minimum Rope Length

Predicted Minimum Rope Length

Knot Number

0 20 40 60 80
5.5 Binning Knots With Neural Networks

After experimenting with a regression approach to predicting the minimum rope length of knots, it was decided that another potential approach was to use neural networks to order knots into bins based on their minimum rope length.

A bin size of 6 was selected. For every knot 29 invariants are known and are used as predictors. We form a matrix that is 6x29 by allowing each row to correspond to a unique knot and each column to correspond to a unique knot invariant.
Keeping in mind that the rows in each 6x29 matrix correspond to a unique knot, we can order the rows of the 6x29 matrix so that from top to bottom they are ordered so that the top row corresponds to the knot with the smallest minimum rope length and the bottom row corresponds to the knot with the largest minimum rope length. The figure directly below depicts an example of a correctly ordered knot invariant matrix.

Once a matrix has its rows correctly ordered by the minimum rope length
corresponding to each row, the rows can be shuffled/permuted so they are in the \textbf{incorrect} order with respect to their minimum rope length. See the figure below.

To train the neural network 50,000 unique matrices with correctly ordered rows and 50,000 unique matrices with incorrectly ordered rows were generated from knot invariant data corresponding to knots with a crossing number between 3 and 9. The neural network is trained to differentiate (classify) between matrices with correctly ordered rows and matrices with incorrectly ordered rows.

Once the neural network had been trained on knots with a crossing number between 3 and 9, we wanted to know: \textit{when shown correctly ordered sequences of knots with a crossing number of 10, can the neural network predict that these sequences are in fact correctly ordered by the magnitude of their minimum rope length?}

There are 28 bins of 10 crossing knots that the trained neural network was asked to classify, they are listed below:

- 10.1-10.6
- 10.7-10.12
- 10.13-10.18
- ...
The neural network was able to perform this classification task very well, predicting 27/28 bins correctly – which is about 96.5 percent classification accuracy for knots with a crossing number of 10. The only bin that was incorrectly predicted was 10.133-10.138. It is not currently known why this particular bin was repeatedly misclassified.

5.5.1 Shapley Scores For Blackbox Model Explanations

In addition to achieving a high classification accuracy on the binning problem it was also desired that the reason for the high classification accuracy be understood. This is a challenging problem because neural networks are black box models, and it can be difficult to understand why they perform well in some cases and poorly in others. Shapley additive explanations are a recent method that can help overcome this challenge and explain the output of a neural network. Shapley additive explanations can produce an importance score for every predictor used in a machine learning model. In our case we use Shapley scores to evaluate which knot invariants were the most important for accurately classifying knot invariant matrices (correctly ordered or incorrectly ordered) for knots with a crossing number of 10.

The figure directly below is an example plot of Shapley scores for each of the 29 knot invariants for the 10.1-10.6 knot invariant matrix. Knot invariants with the highest shapley scores are the most important predictors and effect classification accuracy the most. In this example the four most important invariants would be:

1. Full Symmetry Group
2. Longitude Translation
3. Unknotting Number
4. 3D Clasp Number
For each of the 28 bins of knots with a crossing number of 10, the top-four most important knot invariants (with respect to classification accuracy, as determined by their shapley score) were tabulated. The results of this were as follows:

1. In 23/28 bins **3D Clasp Number** was one of the top-4 most important knot invariants with respect to classification accuracy.

2. In 22/28 bins **Full Symmetry Group** was one of the top-4 most important knot invariants with respect to classification accuracy.

3. In 17/28 bins **Determinant** was one of the top-4 most important knot invariants with respect to classification accuracy.

4. In 17/28 bins **Longitude Translation** was one of the top-4 most important knot invariants with respect to classification accuracy.

5. In 13/28 bins **Unknotting Number** was one of the top-4 most important knot invariants with respect to classification accuracy.

It can be seen that shapley scores indicate that a small group of the original 29 invariants used as predictors are actually significant factors in the neural networks predictive ability.
5.6 Minimum Rope Length Conclusion and Future Work

Predicting knots minimum rope length through multivariate regression using other knot invariants as predictors is a challenging task. Decision trees are likely too weak of a machine learning model to perform well at this task on their own although an ensemble learner consisting of decision trees may perform better. Neural Networks performed better than decision trees and are a promising machine learning model for predicting the minimum rope-length of knots (regression), with our best performing neural network achieving a mean-squared-error of 1.97. Neural networks also performed very well in the knot binning task, achieving a classification accuracy of 96.5 percent on knots with crossing number of 10. Lastly shapley scores seem to be a promising model for interpreting why a neural network is or is not able to predict knots minimum rope length.

In the future both the regression and binning neural network models will be improved, and ensemble methods will be tried with both decision trees and neural networks.

6 Technologies Used

Click the links to view technologies websites.

1. Base neural network used for all experiments (with modification)
   Neural-Network Link

2. Generative Adversarial Neural Network used (with modification)
   Generative-Adversarial-Network Link

3. Keras
   Keras Link

4. Knotplot
   Knotplot Link

7 Citations


