War Games and Dot Products

Niko Kompella

November 14, 2025

Abstract

War has been a staple of humanity's identity. Where there is life, there is conflict, and where there is conflict, there is death. The most recent development to war, however, came after World War II with the development of the atomic bomb. The nuclear deterrent has caused it to evolve beyond direct, on-the-ground conflict to more nuanced, multifaceted strategy. Now, simple land disputes or religious disputes aren't enough to start a war.

Attempting to predict conflict requires a thorough understanding of countries' histories, governments, economies, and religious stances, among others. The purpose of this paper is to use a neural network to assign importance to the different factors between countries and discern its effectiveness. Analyzing the factors that lead to a war between two factions will help isolate them for analysis.

All source code is publicly available on GitHub, with along with any figures that were used in this paper.

1 Background

World War I was one of the most important wars in human history. The introduction of machine guns and armored vehicles multiplied the scale of bloodshed and dramatically changed the strategy between defenders and attackers. This new form of warfare was put to the test in World War II, leading to an even bloodier and more drawn-out conflict. This war was brought to a close with the bombings of Hiroshima and Nagasaki, a show of power that heralded the Cold War and a new era of proxy wars.

The Cold War saw major conflicts between countries that were backed by either the United States or the Soviet Union through direct military intervention or financial and ideological support. Several conflicts defined this time, including the Soviet invasion of Afghanistan, the United States intervention in Vietnam, and the Chinese civil war.

Since this time period, almost every war between any two powers has been a proxy for a greater war between two nuclear-armed powers that will not engage in conflict due to the risk of nuclear exchange, which would ruin the world.

Neural networks (or perceptrons) were invented in the 1960s with the purpose of performing simple operations and furthering our understanding of the brain. These are the basis of artificial intelligence. The network used a structure akin to a nervous system, where "nodes" that take in an input and output another value function like neurons, and layers correspond to different regions of the brain.

Effectively, a neural network takes in a series of inputs and attempts to find a relationship (or equation) to produce the most accurate output. Through a series of matrix multiplications, the neural network adds "weights" (coefficients) to the equation through trial and error. A major issue with neural networks, however, is that it is impossible to identify what weights are assigned to what, causing them to function as a "black box," where one can only see input and output.

This paper will attempt to use a neural network to predict war between two countries and explain why neural networks in the past haven't been successful in doing this.

2 Creation of the Perceptron

For the project, a two-layer perceptron was created and tested in MATLAB. A two-layer perceptron is called this because it has one hidden layer, along with the required input and output layers, for a total of three layers.

The neural network uses a sigmoid activation function. An activation function is one that normalizes the outputs from a neural network to lie between two constraints. In this case, the sigmoid activation function (defined below) normalizes values between zero and one, which is ideal for finding a probability. It also diminishes values close to zero and one (making small, irrelevant values smaller), allowing for center values to become exponentially relevant with every use of the sigmoid function.

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

The code underwent 100 epochs (training cycles) where it checked its answer and went back again. The learning rate (alpha) was set to 0.01, which ensures that the model doesn't change its guesses drastically and ensures "slow and steady" learning.

Listing 1 was used to test a model with a random number of input layers between one and 100, ten hidden layers, and one output.

Listing 1: Example Layer Initialization

```
x = rand(100, 1); % between 1 and 100 inputs
y = sin(2*pi*x);
alpha = 0.01;      % low learning rate
epochs = 100;      % 100 training epochs
input_size = size(x, 2);
hidden_size = 10; % 10 hidden layers
output_size = 1;  % 1 output
```

The final neural network takes around 45 inputs, featuring economics, religion, ideologies, land disputes, and other information about individual countries, as well as history between the two countries and alliances that each country has. The hidden layer has 64 nodes, and these 64 nodes become values that are then assigned coefficients (weights) to turn it into one output, a probability between one and zero. Figure 1 shows the basic structure of the neural network with five inputs, ten hidden nodes, and one output.

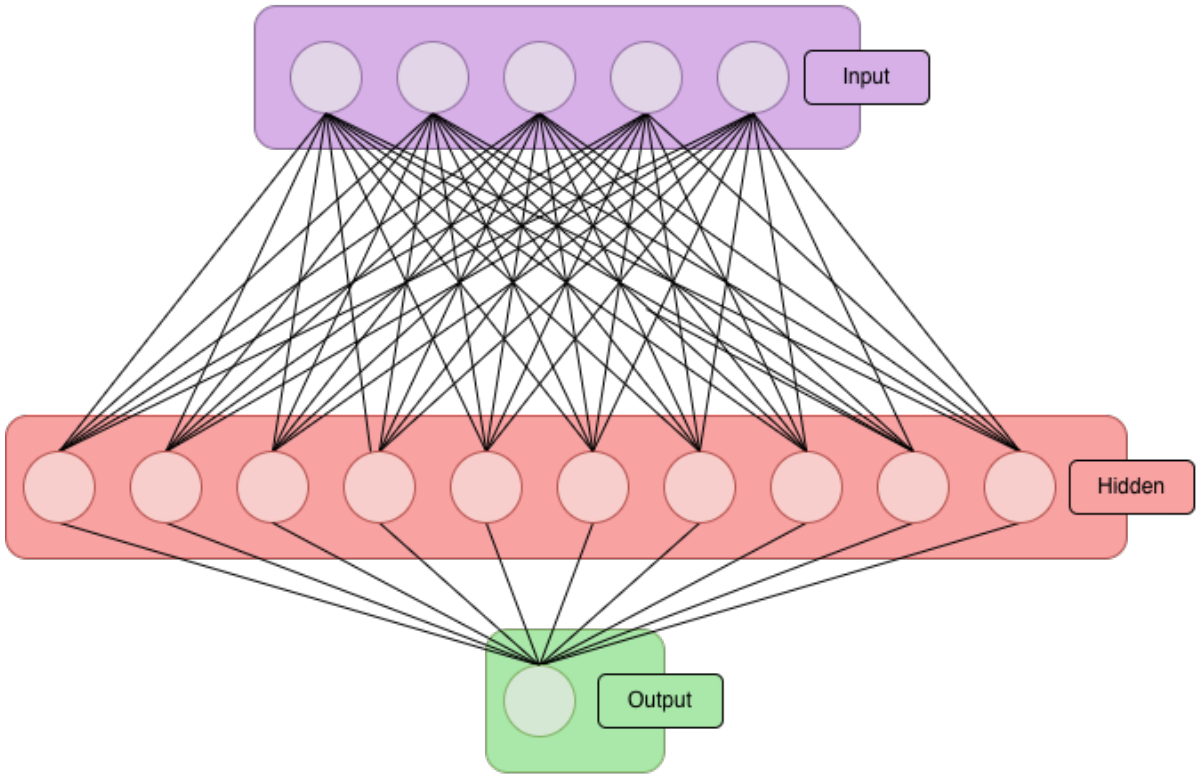

Figure 1: The Neural Network

The perceptron (another term for neural network) that was created was tested with an XOR test to prove its efficacy. The XOR (exclusive or) test is a benchmark test to prove that the hidden layer in a neural network exists and is functioning properly. The test requires a neural network to draw a curve, distinguishing the output sets of an XOR gate.

A simple neural network cannot solve this test because there is no line that separates the points – the two sets are opposite ends of a rectangle, and a hyperbola must be drawn

(which requires three-dimensional understanding, using a hidden layer). The neural network achieved this with over 99% accuracy (Fig. 2). The XOR test used two input nodes, two hidden nodes, and one output node.

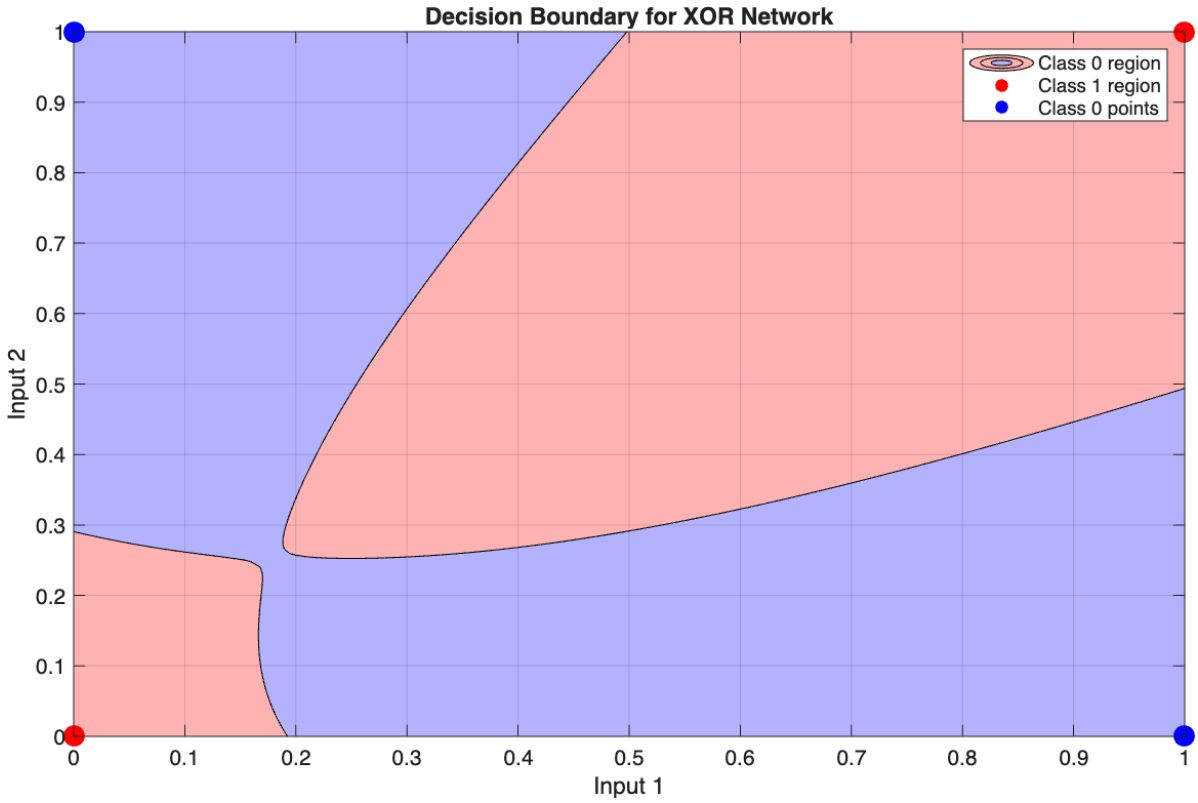

Figure 2: The XOR Test Results

3 The Training Process

The training process, the most important part of building an effective neural network, involves crunching data into a standardized set of numbers and facets. First, a web scraper had to be built to scrape several sources of various data pertaining to countries and their relations. Then, the data was put into an excel spreadsheet and exported to a CSV, where a Python script was created to format the data into readable x (input data) and Y (expected output data, or the "answer key") parameters.

Two options arose for training data: post-World War I, after the evolution of land battle,

and post-World War II, after the development of nuclear deterrents. Major wars between World War I and II, however, were mainly civil wars between factions that no longer exist, so post-World War II wars that involved countries that are still in existence were chosen.

Listing 2 shows a sample of war.csv.

Listing 2: `war.csv` (sample)

```
Year(s),Country A,Country B,Generic Reason
1947 1948,India,Pakistan,Land dispute
1950 1953,North Korea,South Korea,Ideology
1950 1953,China,South Korea,Ideology
1950 1953,North Korea,United States,Ideology
1956,Egypt,United Kingdom,Power struggle
1956,Egypt,France,Power struggle
```

Data was collected through the building of several web scrapers, that scraped economic databases, civil unrest and human rights development movements, and Wikipedia.

The data was then put into three different files: "war.csv", for conflict history, "country-data.csv", for individual information about countries, and "alliance.csv", for data on countries' biggest allies.

A python script then built dyads out for training and testing of the neural network, combining the data from the three files into one large input vector with around 60 input nodes.

Due to the incredibly small low percentage of wars to country dyads (around 50 wars for 2500 dyads), the neural network started overfitting (guessing no war achieved over 90% accuracy). Because of this, more parameters were added to the country data spreadsheet.

The train-test split (the division of data that would choose what would be used to train and test) was also altered. The training set size was decreased significantly so that around half of the included dyads would be war-probable, forcing the neural network to find patterns

and discouraging overfitting.

4 Results of the Neural Network

When the neural network has finished learning, it creates two matrices that store critical information.

The first, referred to as "w1" in the source code (and W_1 mathematically), contains the weight matrix generated by the file. The values in the matrix assign importance to the inputs in the vector, and the sigmoid activation function standardizes these values to go between zero and one. This value vector, a_1, is used to calculate W_2. Since W_2 is of size $m \times n$ and x are of size $n \times 1$, a_1 is of size $m \times 1$.

$$z_1 = W_1 x + b_1$$

$$a_1 = \phi(z_1)$$

If the neural network had more layers, this process would be repeated as many times as there are hidden layers.

The second matrix, "w2" (W_2), is a vector that uses standardized values from a_1 to create a final equation. Since W_2 is now a vector, the product is a dot product, not a matrix multiplication, and it produces a scalar output. Once again, this output is run through the sigmoid function to put it between zero and one. This output, a_2, is the final probability.

$$z_2 = W_2 a_1 + b_2$$

$$a_2 = \phi(z_2)$$

In order to do a thorough analysis, looking into W_1 is important. Linear algebra is heavily applicable here, since doing matrix operations on W_1 will reveal a lot about how the neural network thinks.

First, rank is taken. The rank of a matrix is the number of linearly independent rows in a matrix; effectively, the rank finds and removes redundant rows in a matrix and removes them, allowing one to find the number of factors the neural network actually cares about and the number of factors the neural network thinks don't add any new information.

The other operation that was done was singular-value decomposition. Singular value decomposition breaks the matrix into three that contain different information.

$$W_1 = U \cdot \Sigma \cdot V^T$$

The matrix U represents where hidden space values go. V reveals which input feature combinations the values correspond to, while Σ contains the actual values. Σ is the one that is analyzed the most, as it reveals the "strength" or "intensity" of each independent node. V^T is V transposed, effectively flipping the matrix on its side.

5 Discussion

The end results of training were broken down clearly by singular value decomposition. The neural network ended up with a 60.87% accuracy and a mean prediction of 0.4309 (guessing war 43.09% of the time). It guessed war correctly for all instances of actual war, but also had a significant amount of false flags included.

Figure 3 shows a confusion matrix, which is essentially a chart of guesses made against the correct answer. The top left and bottom right are the number of correct predictions, while the top right and bottom left (which has zero) are the number of incorrect predictions, or, in our case, false flags.

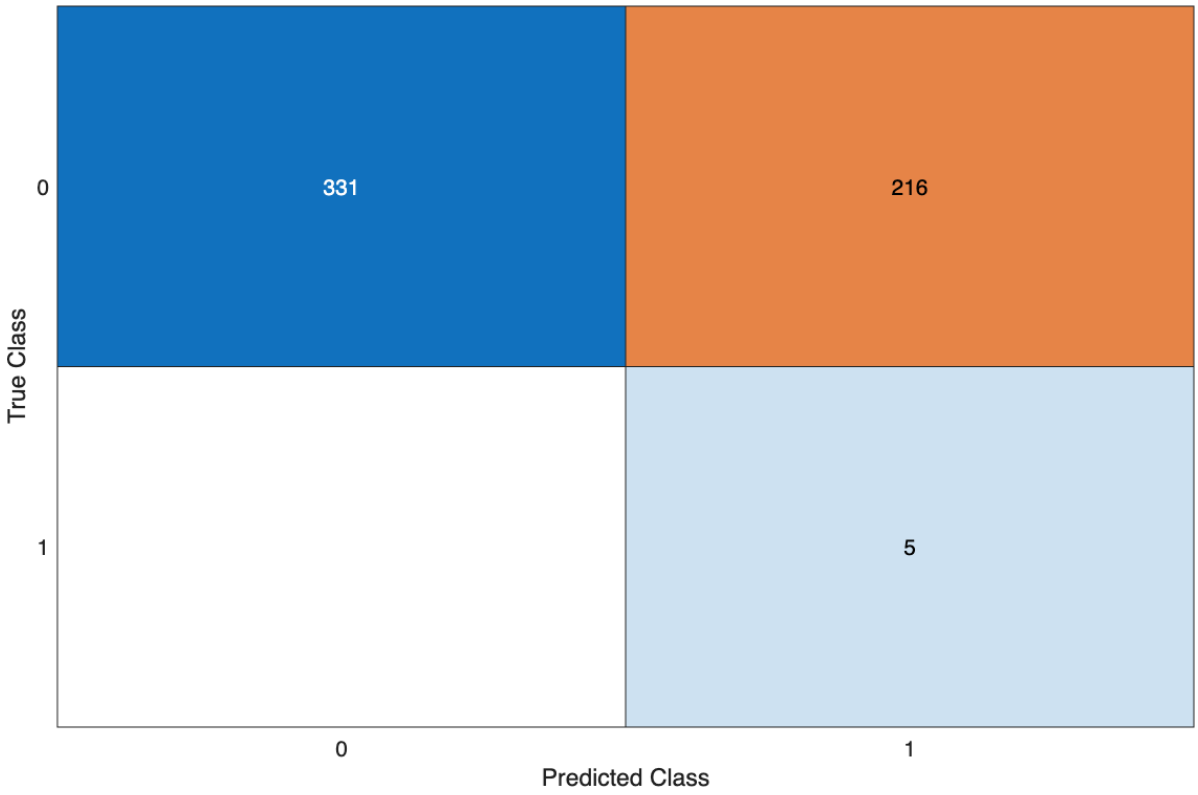

Figure 3: The Final Confusion Matrix

The rank of the matrix ended up being the same size as the number of hidden layer nodes (32 or 64, depending on which trial run was being done), implying the neural network was compact, and all nodes were used up. The real information, however, is apparent in the singular value decomposition.

Out of all nodes, the top ten singular values were printed as part of the analysis function, which broke down the weight matrix after testing for clarity.

Evidently, the top two nodes have the most significance in the final calculation – the other nodes seem to prioritize similar values, leading to a negligible effect on prediction, particularly due to the sigmoid function making small values smaller and prioritizing larger values.

Listing 3 features the top ten singular values found in this process, from most to least important.

Listing 3: `war.csv` (sample)

```
Top 10 singular values:
3.2184
2.6935
0.3563
0.3488
0.3385
0.3358
0.3274
0.3205
```

Some features were clearly marked, while others had to be searched for by-name. The list below highlights all the features, as well as importance (the lowest number is the most important).

Top Ten Most Important Features

1. `Feature_29` (-0.1633)
2. `B_CorruptionIndex_0_100_` (-0.1597)
3. `B_TradeOpenness__OfGDP_` (-0.1576)
4. `A_RecentConflict_2003_2025_` (-0.1566)
5. `Feature_27` (-0.1505)
6. `Feature_70` (-0.1494)
7. `Feature_129` (-0.1492)
8. `Feature_17` (-0.1489)
9. `Feature_164` (-0.1409)
10. `Feature_154` (-0.1391)

The obvious features, corruption, recent conflict, and trade openness, are clearly named. However, there are some unnamed features that also provide key insight. Evidently, the top

ten values are of almost equal importance; there is a drop in value after the top ten values, indicating the remaining factors are negligible in comparison.

Feature 29, the most important value, is the 'Christianity' flag, which is true if a country identifies itself as Christian, or if it has a majority Christian population. The flag would be true for both Armenia (which is a Christian nation) and the United States (which is secular but has a Christian majority).

Feature 27, similarly, is a 'Christianity/Buddhism' flag, which is true if a country has both a significant Christian and Buddhist population, a trend among East Asian countries (South Korea and Taiwan are the most prominent examples).

Feature 70 is the 'Land Dispute with Russia' flag, indicating if a country has an active (or proxy, through an ally) land dispute with Russia. For example, both Ukraine (who has land dispute in Donetsk, Luhansk, Crimea, and other oblasts) and the United States (who is fighting a proxy war through Ukraine with Russia) will have this flag set to true.

Similarly, feature 129 is a flag for majority Slavic countries, implying the neural network sees conflict in Russia and Ukraine, and envisions a conflict involving Serbia.

Feature 17 marks United Nations voting alignment, with alignment with the United States being flagged as true and alignment with China, Russia, or non-alignment being marked as false.

Feature 164 is a 'Christianity/Islam' flag, where countries that have a large number of both Christians and Muslims reside. Examples include India, Palestine, and Indonesia.

Finally, feature 154, the least important of the top ten features, is the terrorism index. Countries with a high terror index include Afghanistan, Pakistan, and Syria, all of which have had armed conflict in the last decade.

Looking at these features, the neural network seems to find pattern recognition in Christian majority states – primarily, the United States and the United Kingdom, and to a smaller extent, Armenia, with primarily Islamic countries (Syria, Afghanistan, Azerbaijan) and countries with high terror indices (Afghanistan, Pakistan, and Syria).

It also mapped conflict between Ukraine and Russia, which has been a prominent conflict for the last two decades, through flags that indicate a Slavic majority and land dispute with Russia.

Finally, conflict with China is effectively done through the 'Christianity/Buddhism' flag. The neural net predicts high war probability among countries that have this flag, the most prominent two of which are Taiwan (historically engaged in direct land conflict) and South Korea (who was and is involved in a proxy conflict through North Korea).

The neural network struggles to break a 60% accuracy due to the convoluted nature of real-life war. It is almost impossible to capture the nuance of geopolitics, and tensions escalate and deescalate quickly, particularly through the involvement of other powers.

6 Conclusions

Despite its accuracy, using artificial intelligence to predict war still has a long way to go. 60% accuracy indicates a high presence of false positives and a lack of important information in data. Capturing subtleties involves including a lot more data and emulating decisions of top world leaders in real time, a feat that is nearly impossible at the current stage of neural networks.

However, the neural network effectively found indicators that were most important in sparking war. In fact, it even went a step further and used the flags built into the data to map out the three major superpowers in the world (the United States, Russia, and China) through individual flags that are mostly unique to the regions countries operate in (UN voting alignment, presence of Islam, Slavic population, and presence of Buddhism).

References

1. GeeksforGeeks. (n.d.). *Implementation of Neural Network from Scratch using NumPy*. Retrieved from http://bit.ly/4rDXXml

2. Kompella, N. (2025). *war-prediction-nn* [Source Code for War Games and Dot Products]. GitHub. Retrieved from https://github.com/nishantksmccd/war-prediction-nn

3. World Bank. (n.d.). *Data*. Retrieved from https://data.worldbank.org/

4. International Monetary Fund (IMF). (n.d.). *Data*. Retrieved from https://www.imf.org/en/Data

5. Organisation for Economic Co-operation and Development (OECD). (n.d.). *Data*. Retrieved from https://data.oecd.org/

6. Armed Conflict Location & Event Data Project (ACLED). (n.d.). Retrieved from https://acleddata.com/

7. Peace Research Institute Oslo (PRIO). (n.d.). *Data*. Retrieved from https://www.prio.org/data

8. Transparency International. (n.d.). *Corruption Perceptions Index (CPI)*. Retrieved from https://www.transparency.org/en/cpi

9. Freedom House. (n.d.). *Freedom in the World*. Retrieved from https://freedomhouse.org/report/freedom-world

10. World Bank Group. (n.d.). *Worldwide Governance Indicators (WGI)*. Retrieved from https://info.worldbank.org/governance/wgi/

11. United Nations Population Division. (n.d.). Retrieved from https://population.un.org/

12. World Bank. (n.d.). *Data API*. Retrieved from https://api.worldbank.org/v2/