import numpy as np

import keras

from keras.models import Sequential

from keras.layers import Dense

# Input data - two players' stats

X = np.array([[73, 80, 45, 90], [93, 88, 75, 92], [89, 91, 60, 86], [96, 85, 63, 85], [70, 90, 60, 70]])

# Output data - game outcomes (win = 1, loss = 0)

y = np.array([1, 1, 0, 0, 0])

# Define model architecture

model = Sequential()

model.add(Dense(4, input\_dim=4, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

# Compile model

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train model

model.fit(X, y, epochs=100, batch\_size=5)

# Make predictions

predictions = model.predict(X)

rounded = [round(x[0]) for x in predictions]

print(rounded)

Here are a few parameters that can be adjusted to modify the behavior of the neural network:

**input\_dim**: This is the number of features in the input data and can be changed to match the number of features in your data.

**epochs:** This is the number of times the model will be trained on the same data, and increasing this value may lead to overfitting or better performance, depending on the data and the model.

**batch\_size:** This is the number of samples processed before the model's weights are updated. A larger batch size can result in faster training, but may also lead to suboptimal performance.

**optimizer:** This is the optimization algorithm used to update the model's weights. Different optimizers have different properties and may perform better or worse depending on the data and the problem.

**loss:** This is the loss function used to evaluate the performance of the model during training. The choice of loss function will depend on the nature of the problem and the desired output.

The number of hidden layers and their sizes can be adjusted to modify the capacity of the model.

These are just a few examples, and the best values for these parameters will depend on the specific problem and data. It may require some experimentation to find the best values.

**The proportion of the dataset used for validation can vary, but a commonly used split is 70-15-15, where 70% of the data is used for training, 15% for validation, and 15% for testing. The validation set is used to tune the model's hyperparameters, and the test set is used to evaluate the performance of the final model.**