GA NN: Plan

Data Preprocessing:

Load the OHLC and RSI data from Yahoo Finance or another source.

Preprocess the data, such as removing missing values, scaling, and splitting into training and testing sets.

Calculate the ATR and use it to determine the stop loss distance.

Add a new column called "input" to the data that indicates whether the price increased or decreased on the next day. (For supervised learning)

Calculate and add a new column for RSI based on the close price.

Neural Network Architecture:

Define the neural network architecture using Keras or another deep learning framework.

Include LSTM, dense, and CNN layers.

Set up the input and output layers for the OHLC and RSI data and the buy/sell signal.

Use Keras Tuner and/or NEAT to optimize the architecture and hyperparameters of the neural network.

Visualize the neural network using Netron or another library.

Training and Evaluation:

Train the neural network using the training data and appropriate loss functions, such as binary cross-entropy or mean squared error.

Evaluate the performance of the neural network using appropriate metrics, such as accuracy or profit, on the testing data.

Use early stopping to prevent overfitting and optimize the performance of the neural network.

Continuously evaluate the performance of the trading strategy and adjust the neural network architecture and risk management rules as needed to optimize performance and manage risk effectively.

Genetic Algorithm with NEAT:

Set up the NEAT genetic algorithm to optimize the neural network architecture.

Define the fitness function to evaluate the performance of each neural network.

Set up the mutation rates and other parameters of the genetic algorithm.

Use the best-performing neural network architecture found by NEAT as a starting point for Keras Tuner.

Hyperparameter Optimization with Keras Tuner:

Set up Keras Tuner to optimize the hyperparameters of the neural network based on the search space.

Define the search space to include both hyperparameters and architecture parameters.

Set up the search strategy, such as Bayesian Optimization or Random Search, and the evaluation metric, such as accuracy or profit.

Use the best-performing hyperparameters found by Keras Tuner to optimize the performance of the neural network.

GPU Acceleration:

Install the latest version of CUDA Toolkit and cuDNN on your system.

Install the latest version of TensorFlow or PyTorch with GPU support.

Configure your TensorFlow or PyTorch code to use the GPU by setting the appropriate device settings.

Monitor the GPU usage and performance using the NVIDIA System Management Interface (nvidia-smi) or another monitoring tool.

Adjust the batch size and other parameters to optimize the performance of the neural network on the GPU.

Backtesting and Visualization:

Backtest the trading strategy using the neural network on historical data.

Visualize the performance of the trading strategy using a library such as matplotlib or plotly.

Evaluate the performance of the trading strategy using appropriate metrics, such as Sharpe ratio, maximum drawdown, or total profit.

By following this plan, you can create a fully functional neural network trading bot that optimizes the architecture and hyperparameters of the neural network using both Keras Tuner and NEAT, accelerates training using GPU acceleration, and backtests the trading strategy on historical data while monitoring performance using appropriate metrics.