What is this Neural Network's architecture (layers, width, inputs, outputs)?
What is the input we use for the DNN(s)? What is the output? (explain why)

Neural Network Architecture

The Deep Q-Network (DQN) used in this implementation has the following architecture:

1. **Input Layer**:
   - The input layer receives a tensor representing the continuous yield values of the stocks. The size of the input layer is equal to the number of stocks (`input_dim`), which is derived from the number of stocks available.
2. **Hidden Layers**:
   - The first hidden layer: A fully connected (dense) layer with 64 neurons. It uses ReLU (Rectified Linear Unit) as the activation function.
   - The second hidden layer: Another fully connected layer with 64 neurons, also using ReLU as the activation function.
3. **Output Layer**:
   - The output layer is a fully connected layer with the number of neurons equal to the number of possible actions (which is also equal to the number of stocks, `output_dim`). This layer does not have an activation function since we are predicting Q-values for each action.

The architecture can be summarized as follows:

- **Input Layer**: Size = Number of stocks (N)
- **Hidden Layer 1**: Fully connected, 64 neurons, ReLU activation
- **Hidden Layer 2**: Fully connected, 64 neurons, ReLU activation
- **Output Layer**: Fully connected, Number of neurons = Number of stocks (N) (no activation)

Input to the DNN

The input to the Deep Q-Network (DQN) is a tensor representing the continuous yield values of the stocks. Specifically:

- **State Representation**:
  - Each stock has a continuous yield value, which represents the observed return of the stock. This is a real number (continuous value).
  - For N stocks, the state is represented as a vector of length N, where each element is a continuous value corresponding to the yield of each stock.
  - This vector of continuous values is converted to a tensor and is the input to the DQN.

Output of the DNN

The output of the Deep Q-Network (DQN) is a vector of Q-values, where each Q-value corresponds to the expected future reward of taking a specific action in the given state. Specifically:

- **Q-Values**:
  - The DQN outputs a Q-value for each possible action. In this case, the possible actions are choosing which stock to invest in.

- For N stocks, the output is a vector of length N, where each element represents the Q-value of investing in the corresponding stock.
- These Q-values are used to make decisions by selecting the action (stock to invest in) with the highest Q-value, which represents the highest expected future reward.

Why This Input and Output?

- **Why This Input?**:
  - The input state vector represents the current market situation (continuous yield values for each stock), which is the crucial information needed to make informed investment decisions. By encoding this state as a vector of continuous values, the DQN can process and learn from the variations in stock yields.
- **Why This Output?**:
  - The output Q-values represent the expected future rewards for each possible action (investing in each stock). By providing a Q-value for each action, the DQN enables the agent to make decisions that maximize long-term profit based on the current state of the market.
  - This approach aligns with the principles of Q-learning, where the goal is to learn a policy that maximizes the cumulative discounted reward over time.

Detailed Code Explanation

Here's the relevant code segment with explanations:

```python
class DQN(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(DQN, self).__init__()
        self.fc1 = nn.Linear(input_dim, 64)
        self.fc2 = nn.Linear(64, 64)
        self.fc3 = nn.Linear(64, output_dim)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

- `input_dim`: Number of input features, which corresponds to the number of stocks.
- `output_dim`: Number of possible actions, which also corresponds to the number of stocks.

Example

- **Input State Vector** (for 3 stocks): `[0.05, -0.02, 0.03]`
  - Stock 1: Yield 5%
  - Stock 2: Yield -2%
  - Stock 3: Yield 3%
- **Output Q-Values**: `[Q(s, a1), Q(s, a2), Q(s, a3)]`
  - `Q(s, a1)`: Expected reward for investing in Stock 1
  - `Q(s, a2)`: Expected reward for investing in Stock 2
  - `Q(s, a3)`: Expected reward for investing in Stock 3

The DQN uses these Q-values to decide which stock to invest in to maximize expected future rewards.