1. When predicting values using sine wave data, is there a performance difference between the model that only contains Dense layers and one that includes an RNN layer? Which performs better?

Yes, and RNN layer includes Dense layers performs better. The RNN layer is specifically designed to handle sequential data. It processes data step by step and maintains hidden states, which enables it to learn temporal patterns over time, and thus capturing temporal dependencies in time-series data like sine waves.

2. Have you tried stacking two consecutive RNN layers in the model? How would you configure the parameters for the second RNN layer if the first RNN layer is defined as RNN(1, 16)? Briefly explain your reasoning.

Yes, the first RNN layer outputs a shape of (batch_size, 16), which means only the hidden state of the last time step is passed to the next RNN layer. In this case, the next RNN layer should be configured as RNN(16, 16) to ensure that the input and output dimensions match. This avoids errors such as operands could not be broadcast together with shapes (32,16) (32,x) caused by dimension mismatches.

Reason:

1) The first RNN(1, 16) outputs a hidden state with the shape (batch_size, 16) that no longer includes time step information.
2) The second RNN layer's input dimension must match the output dimension of the previous RNN, which is 16.
3) If the second RNN is defined as RNN(16, x) where $x \neq 16$, a dimension mismatch will occur during matrix operations, leading to broadcasting errors.

3. What would be the effects with the larger size of hidden units in RNN layer?

Increasing the number of hidden units in an RNN layer has the following effects:

1) **Improved Capacity**: A larger number of hidden units allows the RNN to capture more complex temporal relationships in the data, improving its ability to model intricate patterns.
2) **Higher Computational Cost**: Larger hidden units increase the number of parameters in the RNN, which requires more memory and computation during training and inference.
3) **Risk of Overfitting**: With more hidden units, the model may memorize the training data instead of generalizing well to unseen data, especially if the dataset is small.
4) **Longer Training Time**: More hidden units lead to slower training due to increased parameter updates and backpropagation through time.

While increasing the hidden units can improve performance on complex datasets, it is essential to balance model capacity, computational cost, and the risk of overfitting.