Reasons for Overfitting are as follows:

1. **High Variance Low Bias:** Complex models have high variance, which makes them prone to overfitting as they can learn intricate patterns in the training data that do not generalize.
2. **Complex Models:** Deep networks with many layers and parameters can easily fit the training data, including noise.
3. **Insufficient Training Data:** When the amount of training data is too small relative to the complexity of the model, the model may learn patterns specific to the training data.
4. **Noise in Training Data:** Irrelevant features or mislabeled data can lead to overfitting as the model tries to learn these noisy details.
5. **Training for Too Long:** Prolonged training can lead to the model fitting the training data very closely, including its noise.

Techniques to reduce overfitting:

1. **Increase training data.**
   Adding More Data: Increasing the size of the training dataset can help the model learn more general patterns.

2. **Reduce model complexity**
   Simplifying the Model: Reducing the number of hidden layers or Reducing number of neurons in hidden layers or parameters in the model to make it less complex.

3. **Regularization**: Adding a penalty to the loss function for large weights. Common techniques include:
   - **L1 Regularization**: Adds the sum of the absolute values of the weights.
   - **L2 Regularization**: Adds the sum of the squared values of the weights.

4. **Cross-Validation**: Using techniques like k-fold cross-validation to ensure the model performs well on different subsets of the data.

5. **Early Stopping**: Monitoring the model's performance on a validation set and stopping training when performance on the validation set starts to degrade.

6. **Dropout**: Randomly sets a fraction of the input units to 0 at each update during training time, which helps prevent the network from becoming too reliant on specific neurons.

7. **Batch Normalization**: Normalizing the inputs of each layer to have a mean of zero and a variance of one, which can help stabilize and accelerate the training process, reducing the tendency to overfit.

8. **Weight Sharing**: Using the same weights across different parts of the network, such as in convolutional layers, to reduce the number of parameters.
9. **Ensemble Methods**: Combining predictions from multiple models to improve generalization. Techniques include bagging, boosting, and stacking.
10. **Data Augmentation**: Generating more training data by applying random transformations such as rotations, translations, and flips to the existing data, which helps the model generalize better.