1. How can each of these parameters be fine-tuned?

• Number of hidden layers

• Network architecture (network depth)

• Each layer's number of neurons (layer width)

• Form of activation

• Optimization and learning

• Learning rate and decay schedule

• Mini batch size

• Algorithms for optimization

• The number of epochs (and early stopping criteria)

• Overfitting that be avoided by using regularization techniques.

• L2 normalization

• Drop out layers

• Data augmentation

A. Each of the parameters listed can be fine-tuned in the following ways:

* Number of hidden layers: The number of hidden layers can be increased or decreased to optimize the performance of the network. Generally, adding more layers can help the network learn more complex representations of the data, but this also increases the risk of overfitting.
* Network architecture: The network architecture, or the way the layers are connected, can be modified to improve performance. This includes changing the types of layers used, such as adding convolutional or pooling layers, or changing the order of the layers.
* Each layer's number of neurons: The number of neurons in each layer can be increased or decreased to optimize the performance of the network. This can affect the capacity of the network to learn more complex representations of the data, but also increases the risk of overfitting.
* Form of activation: The activation function used in each layer can be modified to optimize the performance of the network. This includes using functions like ReLU or sigmoid that are known to work well for different types of problems.
* Optimization and learning: The optimization algorithm used to train the network can be modified to improve performance. This includes using different optimization techniques like gradient descent or Adam, and tuning the learning rate and other hyperparameters.
* Learning rate and decay schedule: The learning rate can be adjusted to optimize the performance of the network, and a decay schedule can be used to decrease the learning rate over time to help the network converge.
* Mini batch size: The mini batch size can be modified to optimize the performance of the network. Generally, larger mini batch sizes can help the network converge faster, but smaller batch sizes can help prevent overfitting.
* Algorithms for optimization: Different optimization algorithms, such as stochastic gradient descent or Adam, can be used to improve performance.
* The number of epochs (and early stopping criteria): The number of epochs can be modified to improve the performance of the network. Early stopping criteria can also be used to prevent overfitting by stopping training when the validation loss starts to increase.
* Overfitting that can be avoided by using regularization techniques, such as L1 or L2 regularization or dropout layers, can be added to the network.
* Data augmentation: The training data can be augmented to create new examples, which can help the network learn more robust representations of the data. Techniques like rotation, flipping, or scaling can be used to augment the data.

Top of Form