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

**Number of Hidden Layers**:

* + You can experiment with different numbers of hidden layers to find the architecture that best suits your problem. Deeper networks may capture more complex features but can be harder to train.

**Network Architecture (Network Depth)**:

* + You can vary the depth of the network by adding or removing layers. Adjusting architecture can help balance model complexity and computational cost.

**Each Layer's Number of Neurons (Layer Width)**:

* + Changing the number of neurons in each layer affects the model's capacity. Increasing the width can allow the network to learn more features, but too many neurons can lead to overfitting.

**Form of Activation**:

* + You can experiment with different activation functions (e.g., ReLU, sigmoid, tanh) to find the one that works best for your data. The choice of activation can influence training speed and performance.

**Optimization and Learning**:

* + You can choose different optimization algorithms (e.g., SGD, Adam, RMSprop) to improve convergence speed and training stability.

**Learning Rate and Decay Schedule**:

* + Tuning the learning rate is critical. You can use learning rate schedules, such as learning rate annealing, to adapt the learning rate during training for better convergence.

**Mini Batch Size**:

* + The mini-batch size can affect training speed and generalization. Smaller batches offer a form of regularization, while larger batches may speed up training.

**Algorithms for Optimization**:

* + You can experiment with various optimization techniques, like gradient clipping, to prevent gradients from exploding. Different optimizers may work better for different tasks.

**The Number of Epochs (and Early Stopping Criteria)**:

* + The number of training epochs can be adjusted. Early stopping can be employed to halt training when performance on a validation set stops improving.

**Overfitting that can be Avoided by Using Regularization Techniques**:

* + Techniques like dropout, L1/L2 regularization, and weight decay can be used to mitigate overfitting. These techniques penalize large weights and encourage simplicity.

**L2 Normalization**:

* + Applying L2 normalization (weight decay) to the model's parameters can help prevent overfitting and improve generalization by encouraging smaller weights.

**Dropout Layers**:

* + Adding dropout layers can be effective in reducing overfitting. Dropout randomly deactivates a portion of neurons during training, forcing the network to learn more robust features.

**Data Augmentation**:

* + Data augmentation involves creating variations of the training data by applying transformations (e.g., rotation, scaling) to increase the diversity of the training set. This helps the model generalize better.

To fine-tune these parameters, you typically use techniques like grid search, random search, or automated hyperparameter optimization methods. Experimenting with different combinations and values of these parameters is essential for finding the best neural network configuration for a given task.