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

• Number of hidden layers: The number of hidden layers can be fine-tuned by adjusting the complexity of the model, such as the number of neurons in each layer and the number of connections between layers.

• Network architecture (network depth): Network architecture can be fine-tuned by changing the number of layers in the neural network and the number of neurons in each layer. This can be done by experimenting with different combinations of layers and neurons to find the best configuration for the specific neural network task.

• Each layer's number of neurons (layer width): To fine-tune the number of neurons in a given layer, we can use hyperparameter optimization techniques such as grid search, random search, or Bayesian optimization.

• Form of activation: The form of the activation function can be fine-tuned by changing the parameters used to define the function. For example, if the chosen activation function is a sigmoid function, the parameters to be fine-tuned include the slope of the curve and the intercept.

• Optimization and learning: Optimization and learning can be fine-tuned by adjusting the parameters of the optimization algorithm and the learning rate. This includes changing the batch size, the number of epochs, the type of optimizer, the initial learning rate, the decay rate, and the momentum rate. Additionally, hyperparameter tuning can be used to fine-tune the model by adjusting the values of the hyperparameters.

• Learning rate and decay schedule: The learning rate and decay schedule can be fine-tuned by adjusting the initial learning rate, the decay rate, and the decay step. The initial learning rate should be set to a high enough value to ensure that the model is making meaningful progress, while not so high that the model is unable to learn effectively, leading to instability. The decay rate should be set to a value that will sufficiently reduce the learning rate over time, while not reducing it so quickly that the model is unable to learn effectively. Finally, the decay step should be set to a value that will allow for a sufficient reduction in learning rate over time, while not reducing it too quickly.

• Mini batch size: Mini batch size can be fine-tuned by trial-and-error approach. Sometimes, a smaller batch size can help the model generalize better.

• Algorithms for optimization: Different algorithms for optimization, such as gradient descent, stochastic gradient descent, Adam and so on, can be used to fine-tune the parameters of a model.

• The number of epochs (and early stopping criteria): can be fine-tuned using cross-validation and monitoring the validation loss.

• Overfitting that be avoided by using regularization techniques. Regularization is a technique used to reduce the complexity of a model by limiting the magnitude of the model parameters. This can help prevent overfitting by penalizing large weights and providing a smoother fit to the data. Regularization techniques can be used to fine-tune a model’s parameters by adjusting the penalty applied to large weights, or by introducing a penalty for large weights. Regularization techniques can also be used to reduce the likelihood of overfitting by forcing the model to be more general.

• L2 normalization: L2 normalization can be fine-tuned by adjusting the regularization strength parameter, which controls the strength of the regularization term in the loss function.

• Drop out layers: Dropout layers are a type of regularization technique that helps reduce overfitting by randomly “dropping out” certain neurons, or nodes, from the network during training. This prevents the neurons from over-fitting the training data by forcing the network to learn multiple independent representations of the same data. Dropout rate and number of dropouts can be fine-tuned by adjusting the hyperparameters.

• Data augmentation: Data augmentation techniques, such as flipping, cropping, and adding noise, can be used to increase the size of the training dataset and help improve the performance of the model.