SUMMARY

The results of the scientific articles analysis and the experiments show that first of all for a parameter selection should be checks and chosen the loss parameter (taking also in consideration activation and optimizer functions, although they have less priority, even with a suitable loss the result may not be good). There is no generalization or rule how to choose these three parameters. Only that in the most of the cases RMRprop optimizer and ReLU activation is generally good enough. They can be used in order to tune them better later.

Rest of the parameters doesn't have such a big effect, and these three mentioned above. A number of hidden layers depends on how complex is the dataset, but in usual cases, up to 5 layers are enough.

A number of neurons should be between the size of an input layer and an output layer (not to exceed the number of features). Although, if there is need to increase the complexity of the model, the value of neurons should be increased together with the number of layers.

A number of epochs is the easiest parameter to control. It indicates how many loops dataset does through transformations, so the training can be stopped as soon as there is no improvement using early stopping callback.

Batch size should be a power of two for memory allocation. The value should be checked in the big ranges with the step of 30-60, as within small ranges there is almost no change present.

For adaptive optimizers, there is no need to change the default value for learning rate, but in case of, for example, SGD learning rate may have a huge impact on the performance. In addition, there can be implemented the learning rate decay function, which, again, may not be useful in cases when optimizer tunes learning rate itself.

If the training result is much higher than the validation one, then overfitting is present. To handle it the complexity may be reduced by using regularization methods. One of the most effective ones is Dropout, which randomly sets the output features to zero, which prevents the model from catching the noise of the data.