Loss Function 1：

No gradient for the Coarse Layer.
I can get two tensor from the model, which is the softmax(output):

1. TC (TensorCoarse): each value of this (TC_i) means the possibility of being coarse class like FR1 or FR2.
2. TF (TensorFine): each value of this (TF_i) means the possibility of being fine class like double-double.

Previous loss function is

$$Loss = w_1 * \mathrm{LogNLLLoss}\ (TC) + w_2 * \mathrm{LogNLLLoss}\ (TF)$$

$$\mathrm{LogNLLLoss}\ (\mathrm{x}) = \mathrm{NLLLoss}(\log_{\mathrm{e}} x) = \mathrm{NLLLoss}(ln(x))$$

Where w_i is the weight for i layer.
Now new loss function could be

$$Loss = \mathrm{LogNLLLoss}\ (softmax(\ TF'))$$

$$TF_i' = \begin{cases} TF_i * TC_{FR1}\ , FR1\ include\ i \\ TF_i * TC_{FR2}\ , FR2\ include\ i \end{cases}$$

Advantages:

1. Training network without setting hyper-parameters weights w_i,
2. If I have reliable classification of FR, the accuracy of detailed classification must be improved. Because the possibility of right TC_i is multiplied by a higher value of TF than wrong one. This condition can be achieved via pre-trained network like setting the weight to [1,0] using previous loss function.
3. For the backward progress, if the possibility of TC is higher which means network is more confident for this FR classification, the step of backward would be bigger. In this way, backward of more reliable training data contributes more for the training.