I implement the MCDropout method.

First, I set the dimension of the two hidden layers of the MINSTNet as 128.

In the Dropout trainer, I add two parameters: weight\_decay and prior\_length\_scale

And then, prepare for the training, using SGD as the optimizer, 100 epochs:

self.network = MNISTNet(in\_features=28 \* 28, out\_features=10)

self.train\_loader = torch.utils.data.DataLoader(

dataset\_train, batch\_size=self.batch\_size, shuffle=True, drop\_last=True

)

self.optimizer = torch.optim.SGD(self.network.parameters(), lr=self.learning\_rate, weight\_decay=self.weight\_decay)

In the training part, pass the training data **batch\_x** to the network and get output, then calculate the hyperparameter **tao** mentioned in the paper. Get the loss between output and label **batch\_y** using function **log\_softmax** and divide the loss by tau times batch\_size. Finally, loss.backward() and optimizer.step()

Calculate the accuracy using the code below:

probability\_samples = torch.stack([F.softmax(self.network(x), dim=1) for \_ in range(num\_sample)], dim=0)

estimated\_probability = torch.mean(probability\_samples, dim=0)