At first, I trained this neural network with variant hidden units, batch size 10, and learning rate 0.1. The results are in the following table:

Hidden nodes	Train Acc	Train f-score	Dev Acc	Dev f-score
10	97.350571	0.838364	97.177983	0.830821
11	97.246828	0.835185	97.066220	0.824708
12	97.222887	0.826516	97.094160	0.828383
13	97.206927	0.833966	97.066220	0.822335
14	97.206927	0.831731	97.122101	0.826891
15	97.302689	0.834151	97.345627	0.849445
16	97.246828	0.829292	96.954457	0.814310
17	97.055303	0.818182	96.898575	0.814691
18	97.318650	0.836520	97.150042	0.828283
19	97.119145	0.828829	97.066220	0.821732
20	97.214907	0.831482	97.038279	0.821549
30	97.294709	0.833087	97.233864	0.836364

Accuracy value and f-score value did not vary a lot. Thus, I picked the relatively higher dev f-score with number of hidden nodes 15.

Then I trained this neural network with hidden unit 15, variant batch size, and learning rate 0.1.

Batch Size	Train Acc	Train f-score	Dev Acc	Dev f-score
10	97.350571	0.838364	97.177983	0.830821
15	97.055303	0.812595	97.122101	0.826307
20	97.246828	0.831901	97.159942	0.828859
25	97.342590	0.838271	97.373568	0.843333
30	97.478254	0.847343	97.541213	0.855263
35	97.478254	0.847047	97.429450	0.849180
40	97.406432	0.845137	97.373568	0.846906

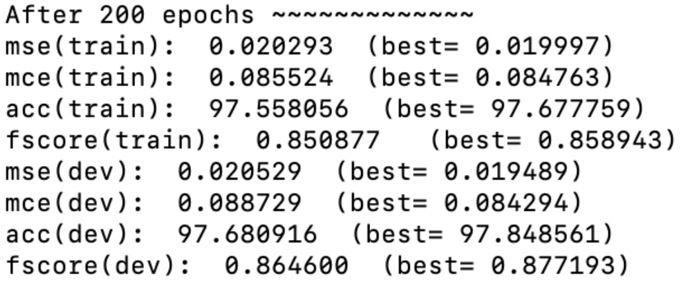
I chose batch size 30 for it gives the highest dev accuracy and dev f-score.

Then I trained this neural network with hidden unit 15, batch size 30, and variant learning rate.

Learning Rate	**Train Acc**	**Train f-score**	**Dev Acc**	**Dev f-score**
0.001	97.230867	0.826756	97.038279	0.830671
0.005	97.677759	0.858943	97.848561	0.877193
0.01	97.566036	0.855660	97.597094	0.862559
0.05	97.454313	0.846265	97.457390	0.851330
0.1	97.478254	0.847343	97.541213	0.855263
0.2	97.414412	0.844530	97.373568	0.848875
0.3	97.023382	0.815255	96.898575	0.814070
0.4	96.983481	0.821866	96.842693	0.816129
0.5	95.132072	0.655380	95.613300	0.703214

Learning Rate	Train Acc	**Train f-score**	Dev Acc	**Dev f-score**
0.5	95.132072	0.655380	95.613300	0.703214
0.4	96.983481	0.821866	96.842693	0.816129
0.3	97.023382	0.815255	96.898575	0.814070
0.2	97.414412	0.844530	97.373568	0.848875
0.1	97.478254	0.847343	97.541213	0.855263
0.05	97.454313	0.846265	97.457390	0.851330
0.01	97.566036	0.85566	97.597094	0.862559
0.005	97.677759	0.858943	97.848561	0.877193
0.001	97.230867	0.826756	97.038279	0.830671

Therefore, my recommendation to solve this problem is 15 hidden units, 30 batch size, and 0.005 learning rate.



```
After 200 epochs ~~~~~~~~~~~~~
mse(train):  0.020293  (best= 0.019997)
mce(train):  0.085524  (best= 0.084763)
acc(train):  97.558056  (best= 97.677759)
fscore(train):  0.850877   (best= 0.858943)
mse(dev):  0.020529  (best= 0.019489)
mce(dev):  0.088729  (best= 0.084294)
acc(dev):  97.680916  (best= 97.848561)
fscore(dev):  0.864600  (best= 0.877193)
```

An interesting observation that I discovered is that the f-score value increases as the accuracy increases, but suddenly drops to 0 when the accuracy decreases to roughly 91-92. The reason this phenomenon occurs is due to the unbalanced quality of this dataset described in the project requirement file. Therefore, the f-score has many sharp decreases to 0.0 in the plot. However, if I lower the learning rate of this neural network, the number of sharp decreases will also decrease. When I change the learning rate to 0.005, there is no 0.0 value in the f-score plot except in the first epoch. I think the reason is that the low learning rate updates the weight slower at each epoch, but in the right direction. Although there are still some up and down in the accuracy plot and the f-score plot, differences are within a small range of value. However, when I set the learning rate to be 0.001, the updates in each epoch will be too small. Thus, I picked 0.005 to be the best learning rate.