Feature selection is a very important issue in this project. There are many features in the dataset. Some of them are useful. They play a great role in predicting the missing data. Some of them are useless. They are meaningless data under the context of predicting missing data in the dataset. We first select some potential unimportant feature groups. Then, we set parts of them to zero, and watch if the test loss increases. If the test loss increases much, then these features are important. If the corresponding test loss does not increase much, then this group of features is not important.
We set the number of neurons in the second and third layers to be 180 and 100. We set dropout to be 0.1 because it is faster and more appropriate in this experiment.
There are 9 potential unimportant groups. They are 120:130, 130:150, 150:164, 164:175, 175:182, 182:189, 189:194, 194:205, and 205:223. We use a vector to denote whether to close or open a group of features. 0 represents closing a group, setting none of the features in this group to 0. 1 represents opening a group, which means setting all of the features in this group to 0. 000000000 represents close all groups, which means to keep the original value and set none of them to 0. 000000000 is the baseline case.

The experimental results are shown below.

Close-open combination	Test error	Train error	epoch
100000000	27.2508	25.2822	30
010000000	29.7388	26.3616	60
001000000	25.8937	22.2732	50
000100000	28.4796	26.0596	25
000010000	26.9821	23.3587	45
000001000	25.6469	23.2698	25
000000100	25.9523	23.8008	25
000000010	25.4880	22.3731	40
000000001	26.5137	22.9699	60

From the experiment, we can see that features 194:205 is the most unimportant features. Their test error is the smallest, which means opening them results in the smallest error increasing. So, they are the most useless features in predicting the missing data. We map this intermediate features into the original feature space. In the original feature space. Features 21- 25, 104, 117-121 are the most unimportant features. We can ignore them to decrease the computational complexity in real-world applications.