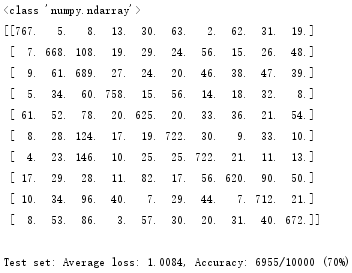
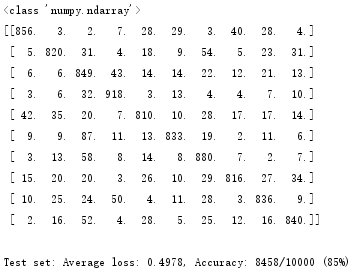
**Part 1: Japanese Character Recognition**

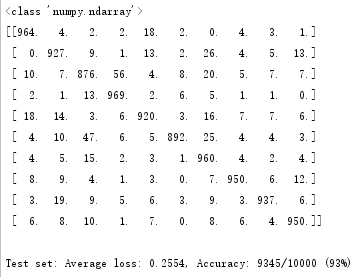
1. This is the final confusion matrix and final accuracy of the model *NetLin*.



1. I tried different values for the number of hidden nodes and got corresponding accuracy as follows.[20(75%), 50(82%), 80(83%), 100(84%), 130(84%), 150(84%), 180(85%), 200(84%), 230(85%), 250(85%), 300(85%), 350(85%), 400(85%), 500(85%),800(85%)]. We could see that the accuracy is stable at around when the number of hidden nodes is larger than 180. This is the final confusion matrix and final accuracy of the model *NetLin* when the number of hidden nodes is 180.



1. This is the final confusion matrix and final accuracy of the model *NetConv*. I also tried many different values of parameters and discuss them in 4(c) in details.



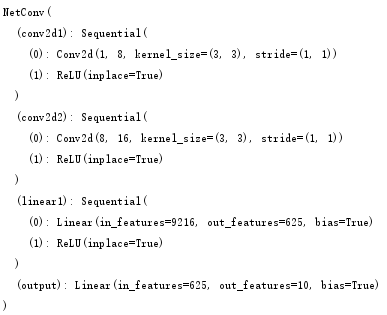
1. The accuracy of convolutional network is the highest. It could achieve 93% as above. The accuracy of a model which a linear function is low, which achieved only 70% accuracy. The accuracy of a fully connected 2-layer network is the middle number which is 85%. Therefore, the convolutional network should be more suitable than a simple linear function and a fully connected 2-layer network when processing the image task.
2. From the three confusion matrixes for each model as above, we could conclude that which characters are more likely to be mistaken for which other characters as follows.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| NetLin | 5 | 2 | 1 | 2 | 2 | 2 | 2 | 8 | 2 | 2 |
| NetFull | 7 | 6 | 3 | 2 | 0 | 2 | 2 | 9 | 3 | 2 |
| NetConv | 4 | 6 | 3 | 2 | 0 | 2 | 2 | 9 | 1 | 2 |

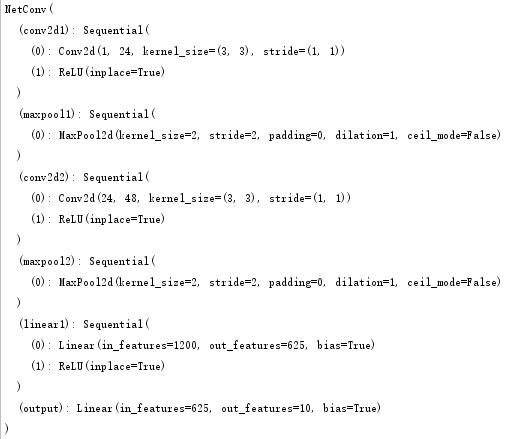
From the table as above, we could see that 1=”ki” is more likely to be mistaken for 6=”ma”, 2=”su” is more likely to be mistaken for 3=”tsu”, 3=”tsu” is more likely to be mistaken for 2=”su”, 4=”na” is more likely to be mistaken for 0=”o”, 5=”ha” is more likely to be mistaken for 2=”su”, 6=”ma” is more likely to be mistaken for 2=”su”, 7=”ya” is more likely to be mistaken for 9=”wo”, 9=”wo” is more likely to be mistaken for 2=”su”.

The reason is that these written characters are similar, so these networks could not distinguish them clearly.

1. This model doesn’t have any parameters to change except learning rate and momentum. Therefore, I tried different values for the learning rate in 0.01(default), 0.05, 0.1, 0.2. I found that when the learning rate equals to 0.01, the accuracy of this linear function could achieve 70%. However, other values achieved lower accuracy which is around 67%. I also tried different momentum. But it did not have huge impact. In conclusion, I want to say that NetLin is limited by the architecture of itself instead of parameters.
2. In this model, I also tried different value of the learning rate and the momentum except the number of hidden nodes which I said above. For the learning rate, I got that when the learning rate is 0.04 and the momentum is 0.05, the accuracy is 89%. When the momentum is 0.9 and learning rate is 0.01, the accuracy is 88%.
3. At the beginning, I used the model as follows. However, the accuracy is only 92%.

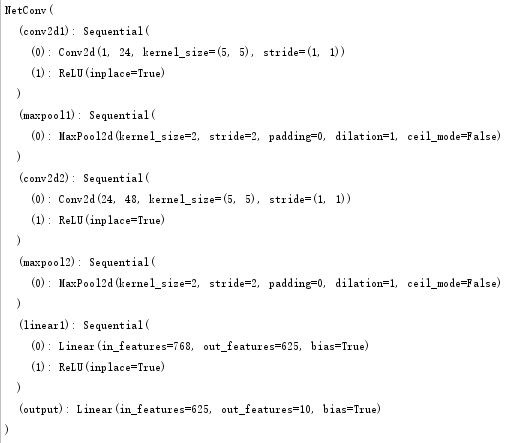


Then, I tried to increase the kernel size of the convolution operation, but the accuracy is still 92%. Therefore, I tried to change the architecture of the network by adding max pool layer as follows. The accuracy became 93%.

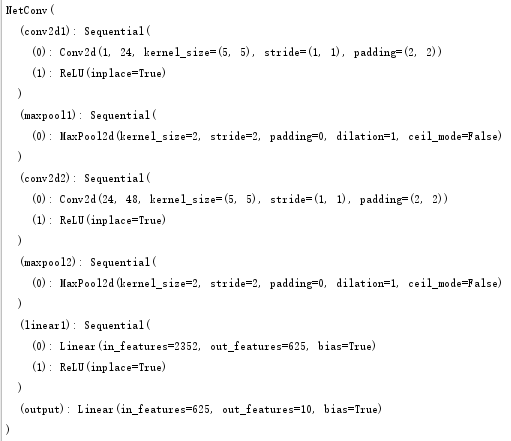


Next, I wanted to increase the accuracy by increasing the kernel size of the max pool layer from 2 to 3, but accuracy became 91%, which is less than ideal.

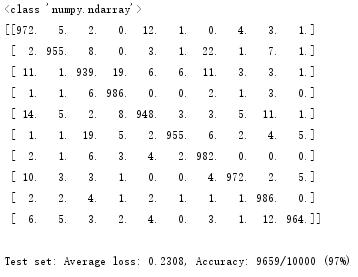
Then, I tried to increase the kernel size of the convolution layer from 3 to 5 and the model is as follows. The accuracy became 94%.



I also tried to use padding(2\*2). The accuracy increased to 95%. The architecture is as follows.



Except the parameters of the model, I also thought that the learning rate and momentum may could increase the result. So, I tired different values of learning rate in [0.02, 0.05, 0.1, 0.2, 0.3]. I found that when the learning rate is 0.1, the accuracy is 96%. Then, I tried to change the momentum from 0.1 to 0.99. When the momentum is 0.7, the accuracy is 97%. It is better than the previous cases. The details is as follows.



In conclusion, I want to say that changing architecture and the value of different parameters of the model is very import if we want to get better result. Some parameters are more important than the others, such as the kernel size of the convolution layer, the learning rate and the max pooling layer. The learning rate will have a huge impact on training, especially when the number of epoch is small. I think that when we try to fit a model, we need choose the important parameters firstly and change others slightly afterwards. But we still need to determine how to choose the best values of these parameters and the architecture basing the tasks.