COMP5434 - Big Data Computing

Hong Kong Polytechnic University

Department of Applied Mathematics

Bonus 1

Authors:
ZHANG Shimin (ID: 22049616G)
QIU Ruipeng (ID: 22068387G)
QUAN Hongli (ID: 22045145G)
HUANG Qiuyi (ID: 22057918G)
TANG Ka Shun (ID: 22028269G)

Date: March 29, 2023

1 What impairs the model performance?

Before employing any method or trick to improve the model performance, we need to figure out what truly has a negative influence on model.
Let us first look at the training and testing curves of the original ANN model on task2. For accuracy, within 200 epochs, the test accuracy reaches a maximum value of 80.47 around the epoch-100 and remains largely static. There is a consistent increase in training accuracy, but it does not exceed 90 percent. This illustrates that the model did not fit the training set during the 200 rounds of training.

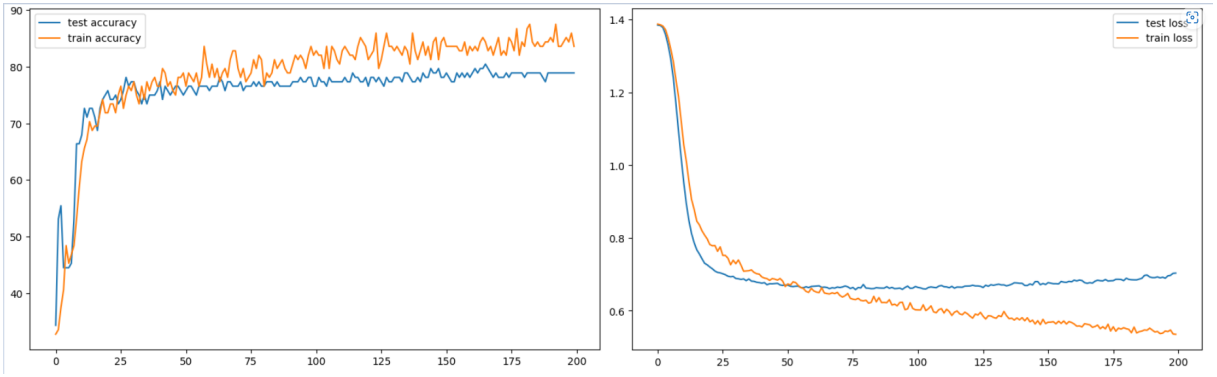

Figure 1: Training and Testing curves for the origin model.

Secondly, as for the loss curves, the loss of the model continued to decrease during the training period. However, the test loss started to increase after 100 rounds. This is clearly a phenomenon caused by overfitting. Overfitting is a classic topic and we believe that it is one of the most important reasons for the failure to improve the accuracy of the model tests.
Therefore, our approach starts from both improving training accuracy and overcoming over-fitting.

2 Solutions

2.1 Improving training accuracy

(a) Adding more hidden layers to the network.

To improve the training accuracy, it is natural to think of increasing the number of parameters of the model. Compared to lightweight models, large models have more parameters and therefore will be more capable of learning for the training set.
Here we did a first experiment to increase the number of hidden layers of the ANN model. In order to investigate the effect of expanding the model on its performance, we conducted experiments on networks with 2, 4, 6 and 8 hidden layers respectively.

Increasing the network parameters allows the model to achieve close to 100 percent training accuracy during training. So we have mainly plotted the test accuracy curve.

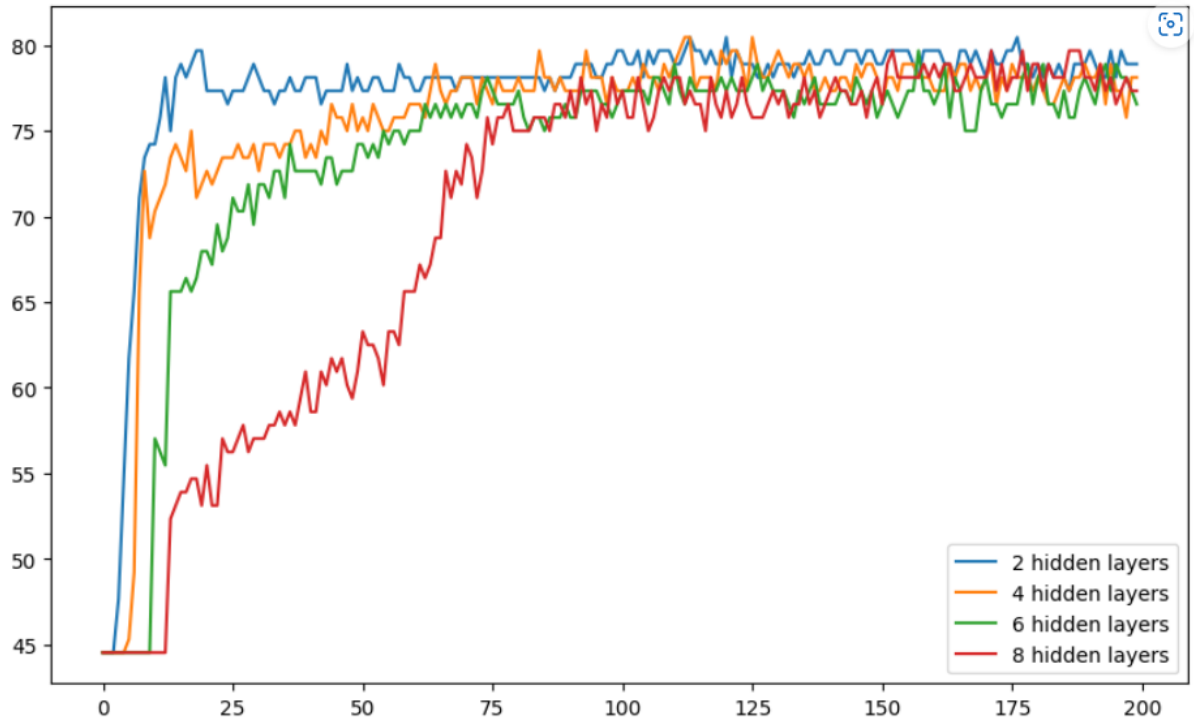

Figure 2: Testing accuracy for the ANN with different number of hidden layers.

From the above figure, we can see that the method of simply increasing the number of hidden layers of the network can improve the training accuracy of the model to make it fit on the training set. However, it does not solve the problem that the accuracy of the model testing cannot be improved, and the phenomenon of overfitting still exists. On the other hand, too many network parameters overload the neural network and increase the computational consumption which leads to a reduction in the convergence speed of the model. This approach has a side effect on the performance of the model.

(b) Adding batch normalization layers to the network.

Batch Normalization is a strategy to keep the input to each layer of a deep neural network equally distributed during training. BN improves the training speed and makes the convergence process much faster. At the same time, BN can be seen as a regularisation expression to prevent overfitting, similar to Dropout.
Applying the example of an ANN with 4 hidden layers, we add BNs in each hidden layer via torch.nn.BatchNorm1d. After adding BN, the model fitted the training set completely at around 50 epochs. Nevertheless, batch normalization does not provide

any further improvement in test accuracy on this dataset compared to Dropout. In other words, the overfitting remains unresolved.

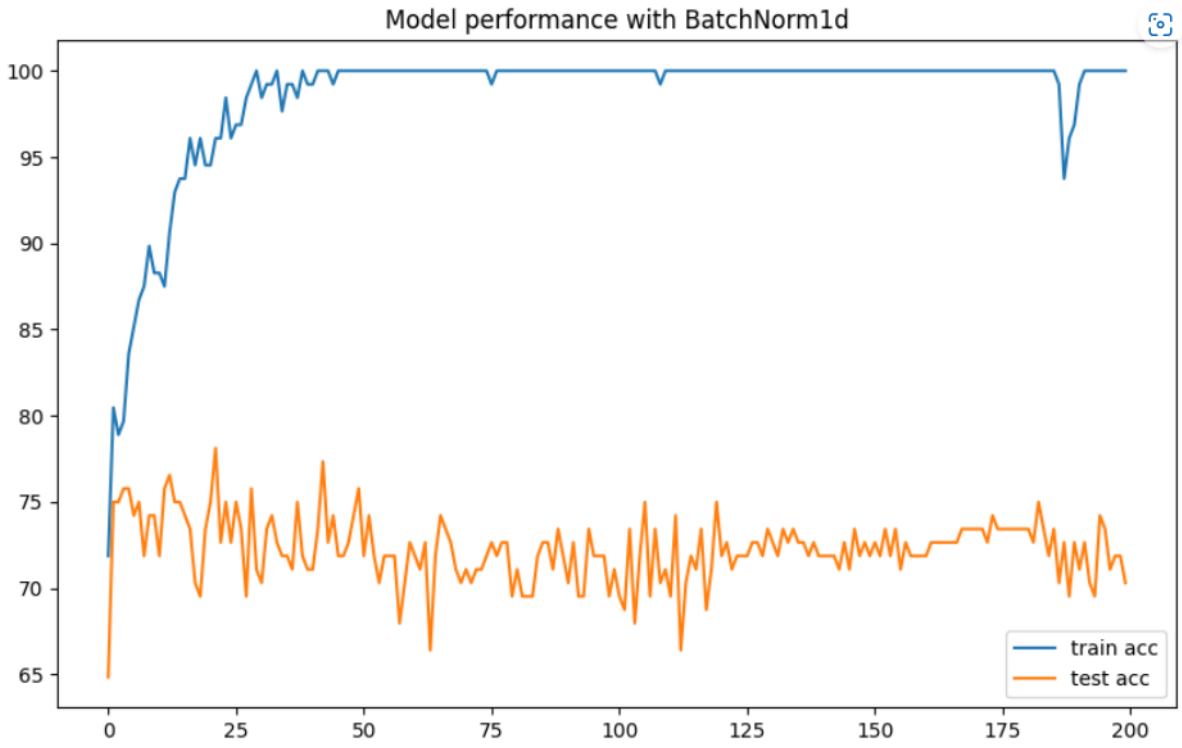

Figure 3: Testing and training curves for the ANN with BatchNorm1d layers.

2.2 Replacing artificial neurons by spiking neurons

Through the above experiments, although the two methods can improve the training accuracy of the model and help the network to fit the training set completely. But it cannot improve the performance of the model on the training set, which represents its low generalization ability. We argue that for the data given in the task, the training set has a total of 4000 samples. it is difficult to train a neural network model of this size with high generalisation ability without considering data augmentation. Therefore, we consider the use of spiking neural networks as an alternative to artificial neural networks to improve the model accuracy.

Spiking neurons in SNNs are biologically more plausible than linear neurons in ANNs. Spiking neurons generate membrane voltages by receiving inputs and continue to accumulate. When the voltage exceeds a threshold, the neuron delivers a spike signal and the membrane voltage decays back to baseline. Such a neuronal structure has helped SNNs to achieve performance close to or even exceeding that of DNNs for many tasks

in the field of artificial intelligence. However, few have used SNNs on non-image or non-speech datasets like the one in our task.

In our experiments, compared to the original ANN network, only the neurons in the hidden layers were replaced by spiking neurons. The following diagram shows the network structure of the SNN:

```
==========================================================================
Layer (type:depth-idx)                             Param #
==========================================================================
SNN                                                --
├─Sequential: 1-1                                    --
│    └─Linear: 2-1                                   13,056
│    └─Linear: 2-2                                   65,792
│    └─IFNode: 2-3                                   --
│    │    └─S2NN: 3-1                                --
│    └─Linear: 2-4                                   65,792
│    └─IFNode: 2-5                                   --
│    │    └─S2NN: 3-2                                --
│    └─Linear: 2-6                                   1,028
│    └─ReLU: 2-7                                     --
==========================================================================
Total params: 145,668
Trainable params: 145,668
Non-trainable params: 0
==========================================================================
```

Figure 4: SNN architecture.

We use SpikingJelly framework to implement the spiking neural network. The spiking neuron model is Integrate-Fire, its dynamic equation is shown below. We choose S2NN as surrogate gradient since it has the best performance.

$$V[t] = V[t-1] + Input[t] \tag{1}$$

Surprisingly, the SNN with only two layers of spiking neurons achieves a test accuracy of 83.6, which obtains an improvement of 3 percentage points compared with ANN's best result.

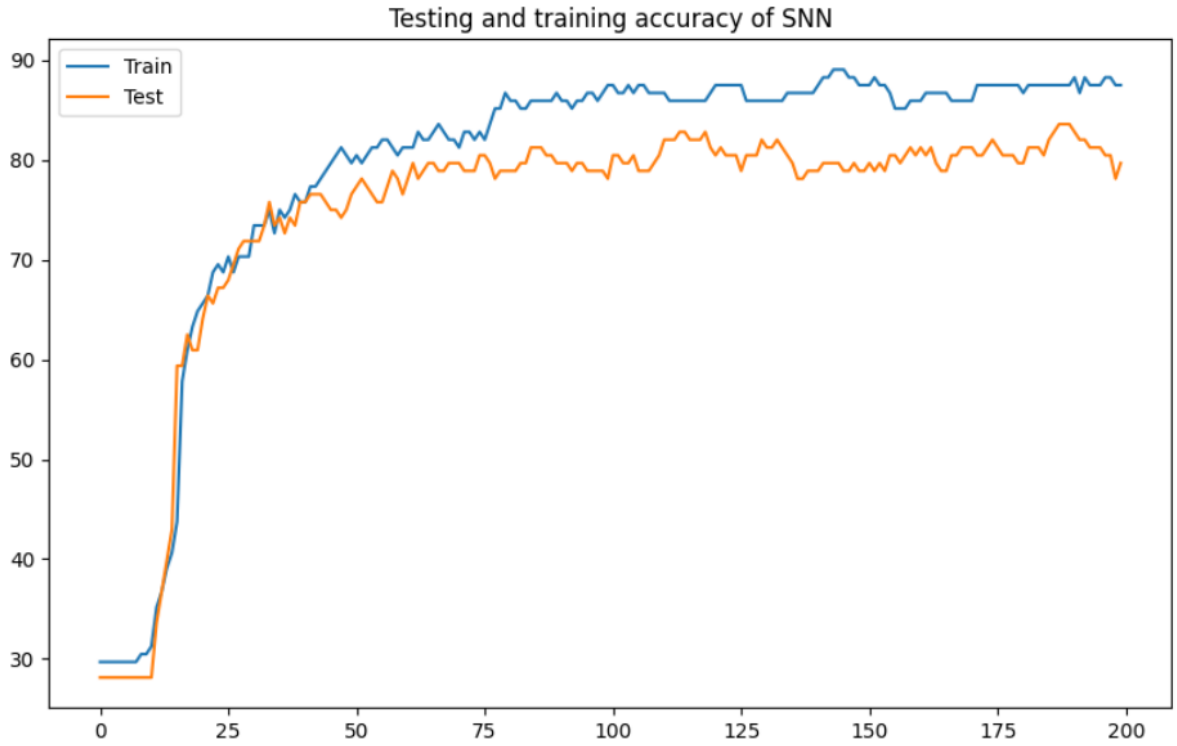

Figure 5: SNN result.

3 Conclusion

Here we use a total of three methods to improve our classification model. By increasing the number of layers in the network and introducing BNs helps the model to fit on the training set, but does not improve the generalization ability of the model. Incorporating spiking neurons to make the network a SNN gave us an accuracy of 83.6 on the test set, a three percentage point improvement over the best ANN model.
This report shows the process of our exploration on the bonus1 problem. Our dataset is not a typical deep learning task (e.g. CV or NLP) and therefore does not exactly match the learning capability of a deep neural network model. In this case, it is normal for the model to appear to be overfitted. By replacing ANNs with SNNs, we gained some improvement in the accuracy of the training set. SNNs are a new field of deep learning and there are not many cases of using them for non-typical deep learning tasks yet, so we believe that SNNs have some room for improvement and potential for development in this area (e.g., using different charging models or alternative gradients).

Python code for Bonus1 please refer to COMP5434_bonus1.ipynb .