ECE228_GPU_AE_STDUENT **Hangquan Zhao A53320603**

Problem 1 Overfitting

1. Explain the indication of overfitting and how this occurs

Overfitting is a modeling error that occurs when a function is too closely fit to a limited set of data points. Overfitting the model generally takes the form of making an overly complex model to explain idiosyncrasies in the data under study, which leads to the result that our model doesn't generalize well from our training data to unseen data.

Figure 1: the train/validation accuracy vs epoch

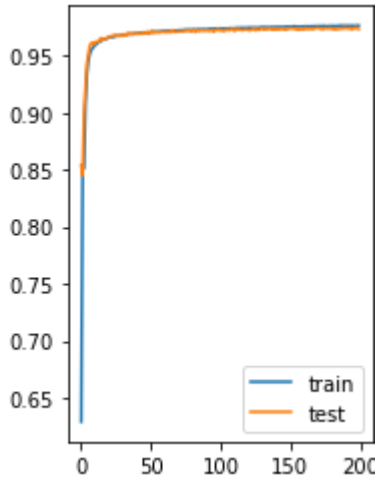
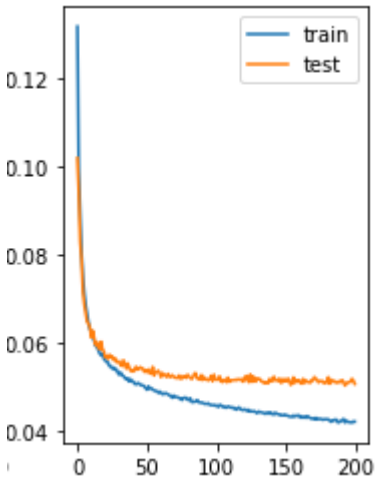


Figure 2: the train/validation loss vs epoch



The train/validation accuracy and loss vs epoch are shown in the figure 1 and figure 2, we can see the difference between loss and accuracy of training data and test data become larger and larger with the epoch numbers increase, which indicates the overfitting situation becomes worse. The final model cannot work as well as the training data and predicted value doesn't match very well with observation values.

2. Explain how overfit can hinder performance of a model when deployed

Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize.

3. Name two ways to avoid this.

Use a resampling technique to estimate model accuracy:

The most popular resampling technique is k-fold cross validation. It allows we to train and test your model k-times on different subsets of training data and build up an estimate of the performance of a machine learning model on unseen data.

Hold back a validation dataset.

A validation dataset is simply a subset of your training data that we hold back from our machine learning algorithms until the very end of your project. After we have selected and tuned our machine learning algorithms on our training dataset we can evaluate the learned models on the validation dataset to get a final objective idea of how the models might perform on unseen data.

1.2 Answer the following question:

1. Explain how dropout affected your loss (provide plot supporting your answer). [+5 answer, +5 plot]

Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel, During training, some number of layer outputs are randomly ignored or "dropped out." This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different "view" of the configured layer. In this section, we try dropping out the all the hidden layers except the input layer and output layer.

Figure 3: the train/validation accuracy vs epoch

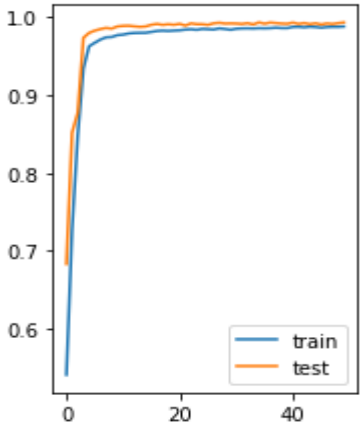
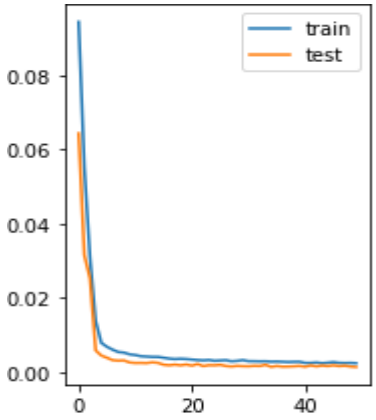


Figure 4: the train/validation loss vs epoch



The train/validation accuracy and loss with dropout vs epoch are shown in the Figure 3 and Figure 4. And we can calculate that the final accuracy of test data is 99.22%, which is larger than the original value(95.8%). And the final loss is 0.0014 compared to 0.0539 of original model. The reason why dropout can improve the performance of training model is because it has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs. What's more, when we add the input layers with dropout, whose result is 88.6%, we can know we'd better not add dropout in the input layers. That's because it will decrease the useful information and disturb the performance.

Bonus Answer the following question:

1. Considering that encoder and decoder can be constructed as separate components, trained as a single unit, and then separated for use . What uses can you brainstorm? [+5 bonus makeup points]

The encoder-decoder model is a way of using recurrent neural networks for sequence-to-sequence prediction problems. It was initially developed for machine translation problems, and also it has proven successful at related sequence-to-sequence prediction problems such as text summarization and question answering.

In Natural Language Processing, Google translate is very successful case. The input will undergo folds (encoding) to retain only important information which in-turn is passed to a DECODER. DECODER helps translate from English to other language with the help of input from ENCODER. What's more, Speech recognition is also another Google paper compares the existing seq2seq models on the speech recognition task. It can also be used in the information security and signal compression which decrease the burdens of signal transmitting.

Problem 2 AutoEncoder

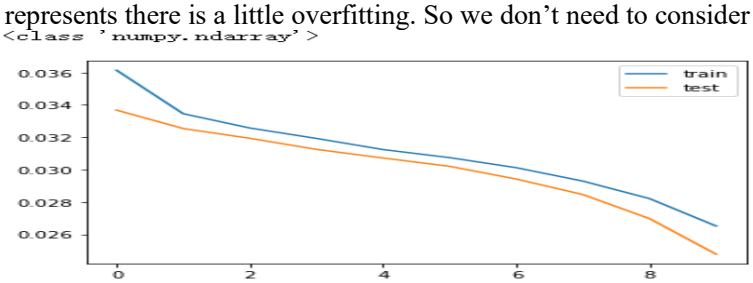
2.1 2.2 please see the ECE228_GPU_AE_STUDENT.pdf

2.3 Machine Anomaly Detection

Please see the histogram plot, mean and std. dev. of normal data, and confusion matrix in the ECE228_GPU_AE_STUDENT.pdf. The TP is 79, and loss is 0.0265, val_loss is 0.0248. The mean of normal data is 0.0094 and standard deviation is 0.0006.

The reason why I choose current model:

1. The first factor I consider is the parameter number, there are two different type models to detect the anomaly signal. The first is Linear Autoencoder and the second is Convolutional Autoencoder. The reason why I choose the convolutional one is because it has less parameters to train, which increase the efficiency and shorten the training timing.
2. Secondly, because at the beginning, we normalize the training data, therefore, the value of voice datasets is always 0-1, Therefore, I use the sigmoid layers in the output layers, which is very useful to filter the anomaly data and the better result.
3. Thirdly, another very important choice is the loss function choosing. Because this learning type is the regression, so we shouldn't choose the binary or probability loss function, but use the mean square error.
4. Fourthly, the data number control, because at first, I use the all 8 channels to train data, which leads to a dead core. I try using one channel as the result.
5. Fifthly, I use the Relu Function as the activation function for hidden layers to make it not linear. That is because the Relu does not activate all the neurons at the same time. So the ReLU function is far more computationally efficient when compared to the sigmoid and tanh function.
6. Sixthly, after running the model repeatedly with different initiating values, the value loss fluctuated slightly with 150 epochs, and the value loss basically stable.
7. Seventhly, the loss graph is as follow. In the following Figure, we can see the test loss is less than the training loss, which represents there is a little overfitting. So we don't need to consider about the overfitting problem.



8. Eighthly, in logic, the more filters, the more accuracy for the training data prediction, the more time for training, and the more probability for overfitting. So for tradeoff the performance and efficiency, I choose the input filter size as 16, kernel size as 3 which has 3,217 parameters in total.