ECS 189G-001
Deep Learning

Winter 2023

Course Project: Stage 2 Report

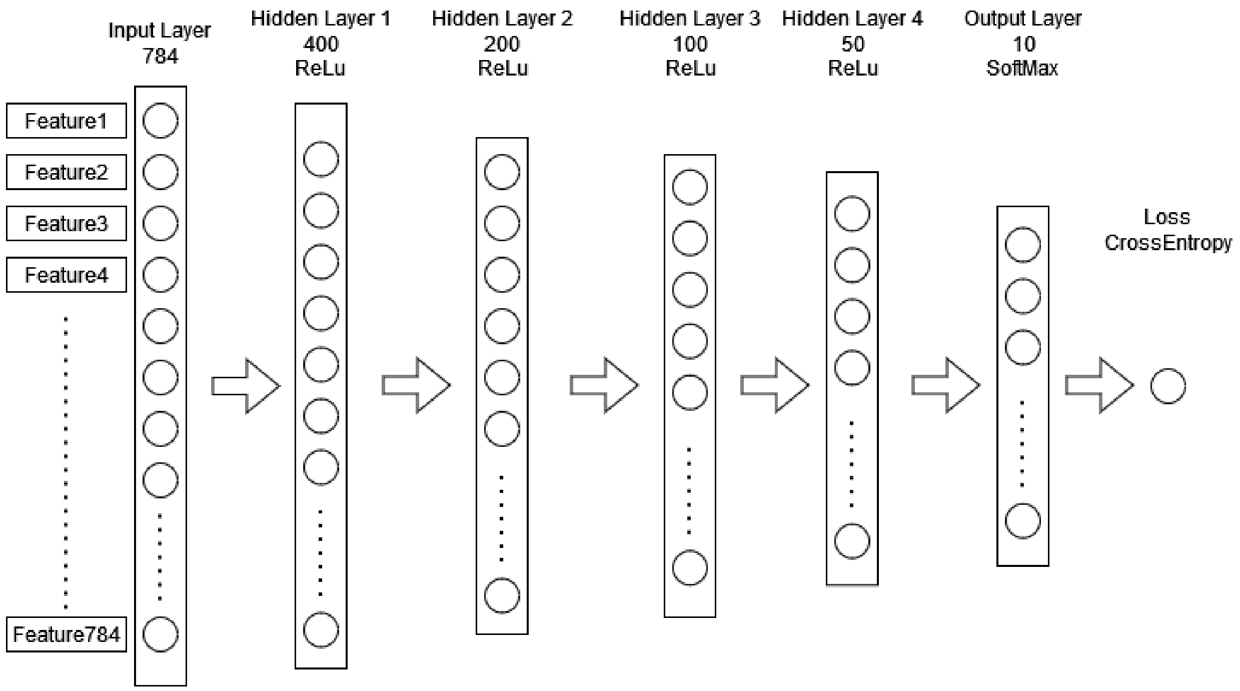
Team Information

Enter Your Team Name Here (delete the extra rows if your team has less than 4 students)		
Name : Chung Ying Hsu	ID: 920918764	Email: cyhsu@ucdavis.edu

Section 1: Task Description

In this task, we will use 784 features to train a model to predict the labels corresponding to these 784 features. This deep learning model will use the multi-layer perceptron method for modeling. We can learn various optimizers and different loss functions in the process of building a model. Adjust the model architecture to find a better model.

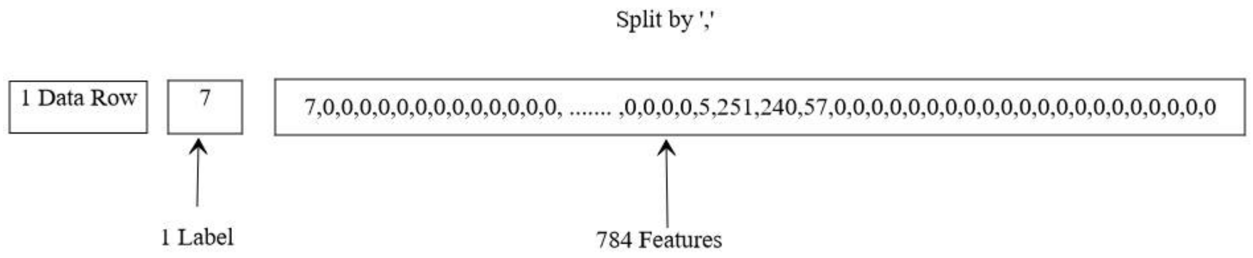
Section 2: Model Description



Section 3: Experiment Settings

3.1 Dataset Description

Each data row has a total of 785 integer values. Only the first value is the label we need to predict, and the value of the label is only 1~10, and the remaining 784 values are all Feature. Since the dataset of stage 2 has already divided it into training set and test set, we don't need to split it. For such a data set, the model to be trained is a bit like a classification model, which needs to be classified into 10 categories through 784 features.



(what are the datasets used in the project, how to partition the training/testing sets)

3.2 Detailed Experimental Setups

I finally chose to use KFold to divide my training set into 20.
Depth: 4 Hidden Layer
Layer dimensions: 784 -> 400 -> 200 -> 100 -> 50 -> 10 -> 1
Learning rate: 5e-4
Epoch: 2000
Loss function: Both CrossEntropy and NLLLoss are often used as multi-classification loss functions, so after trying both loss functions, I think CrossEntropy can bring me better results. Therefore I chose CrossEntropyLoss as my loss function.
Opimizer: The optimizer I use here is Adam. Choosing different optimizers will result in different local optimal solutions. If I choose the SGD optimizer, in this model, there will be no way to get a local optimal solution because the gradient descent speed is too slow. Although the learning rate and the number of epochs are increased, there may still be no way to get a better model than adam.

The above is the parameter setting of my model. This is the result of selection after many comparisons.

3.3 Evaluation Metrics

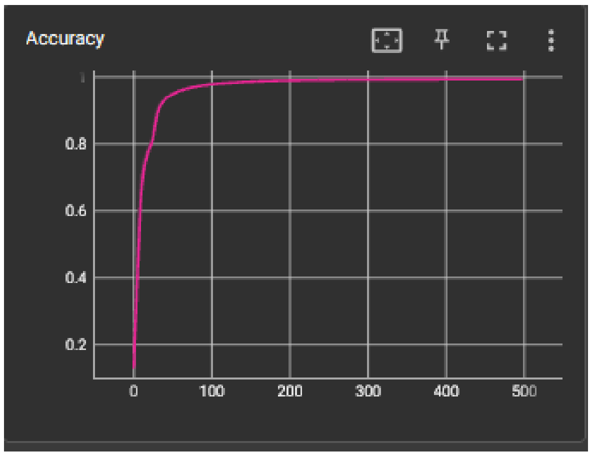
In the experiment, we will use Accuracy, Precision, Recall, F1 score as our evaluation metrics.

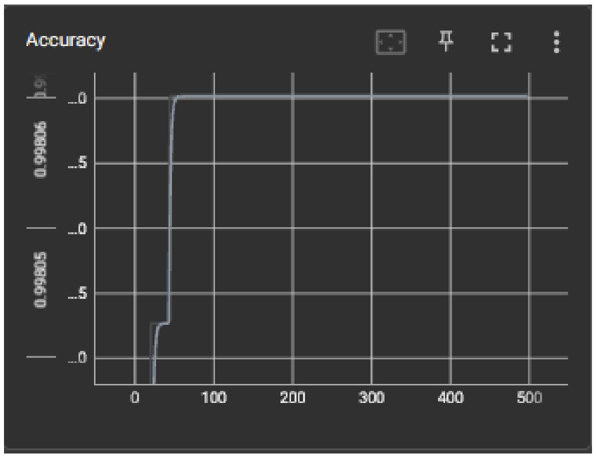
3.4 Source Code

https://drive.google.com/file/d/1CogA0Gzy3-wJrWpd2_JH7yoBj2AOgOrR/view?usp=sharing

3.5 Training Convergence Plot

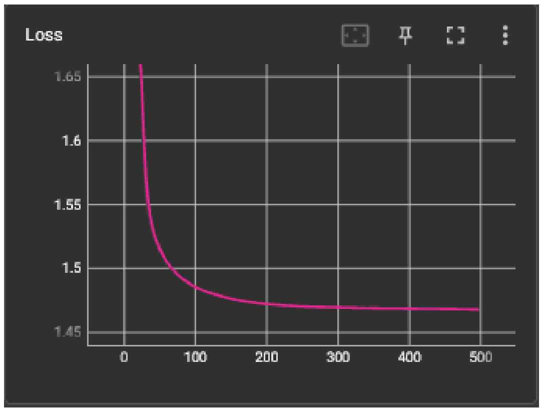
Here is the learning accuracy curve and loss function curve
The left picture shows the accuracy learning curve of the 1st Fold, right picture is last Fold

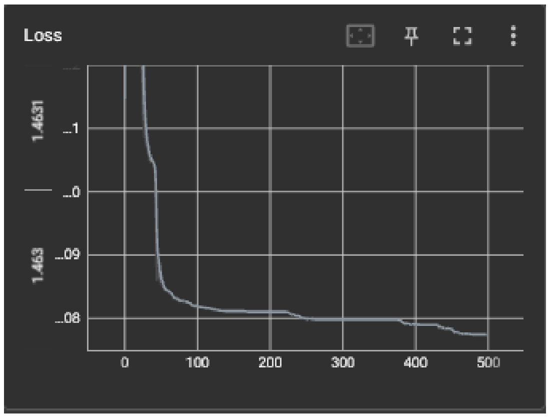




Loss Learning Curve

The left picture shows the loss learning curve of the 1st Fold, right picture is last Fold





It can be clearly seen here that the accuracy rate and loss are gradually converging.

3.6 Model Performance

Training Set

Accuracy-Score: 0.9981166666666667

Precision-Score: 0.9981172650833913

Recall-Score: 0.9981166666666667

F1-Score: 0.9981165180050526

Testing Set

Accuracy-Score: 0.982

Precision-Score: 0.9820023694475958

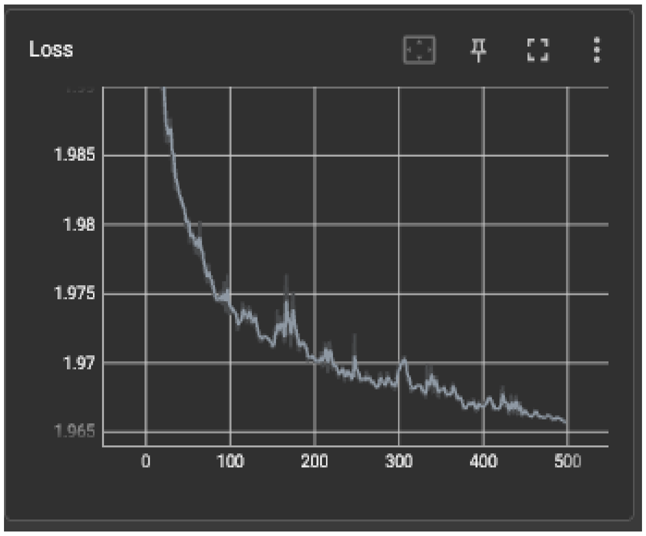
Recall-Score: 0.982

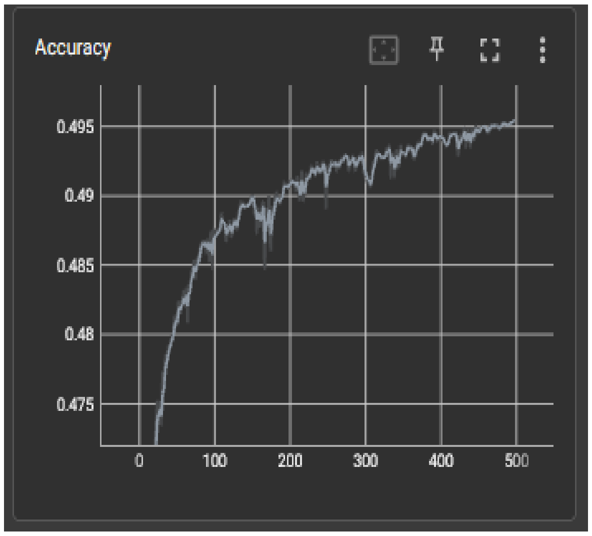
F1-Score: 0.9819935073167915

3.7 Ablation Studies

I did a lot of tests with different numbers of layers and different training sets. Below is my model evolution.

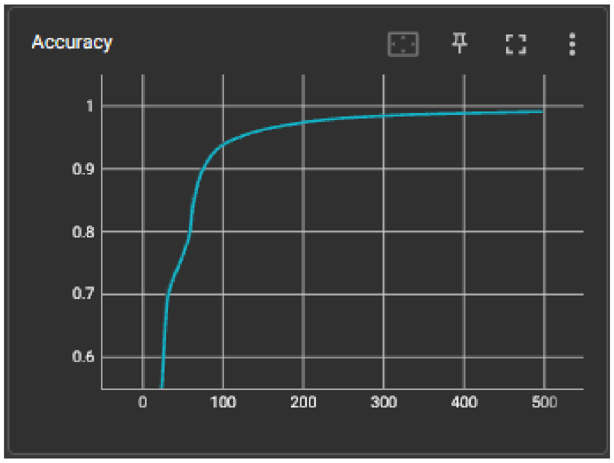
1. TrainData:Original Layers:3 Opt: Adam Learning Rate:5e-3 Epoch:500 Loss:CrossEntropy

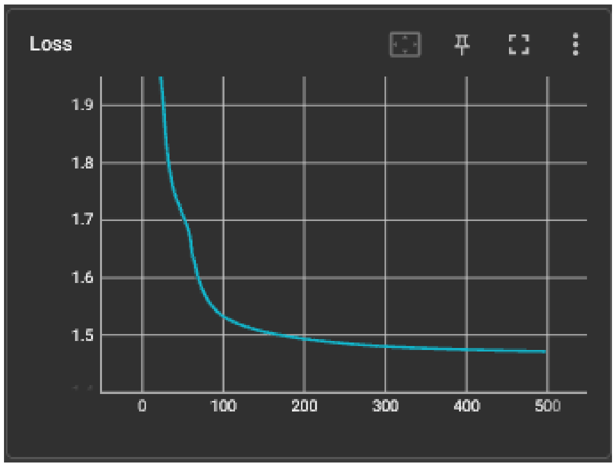




Since the accuracy rate was too low, I started to increase the number of layers.

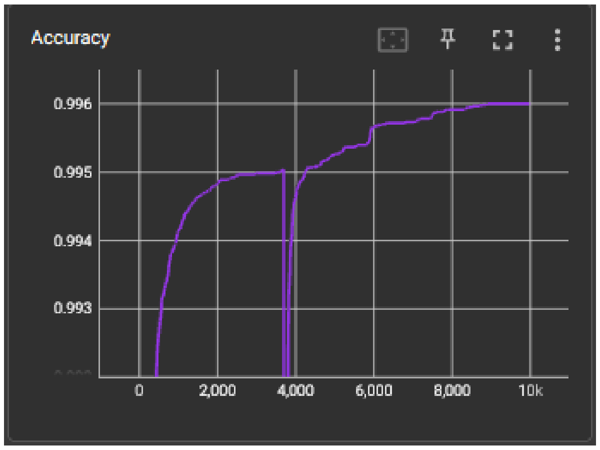
2. TrainData:Original Layers:6 Opt: Adam Learning Rate:1e-4 Epoch:500 Loss:CrossEntropy

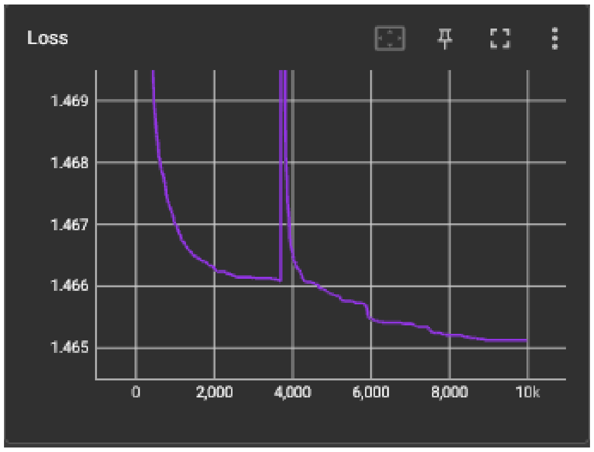




The accuracy of the **training set** here has reached **99.08%**, while the accuracy of the **test set** has reached **97.35%**.
It is already a very high accuracy rate, next I try to increase epoch.

3. TrainData:Original Layers:6 Opt: Adam Learning Rate:1e-4 Epoch:10000 Loss:CrossEntropy





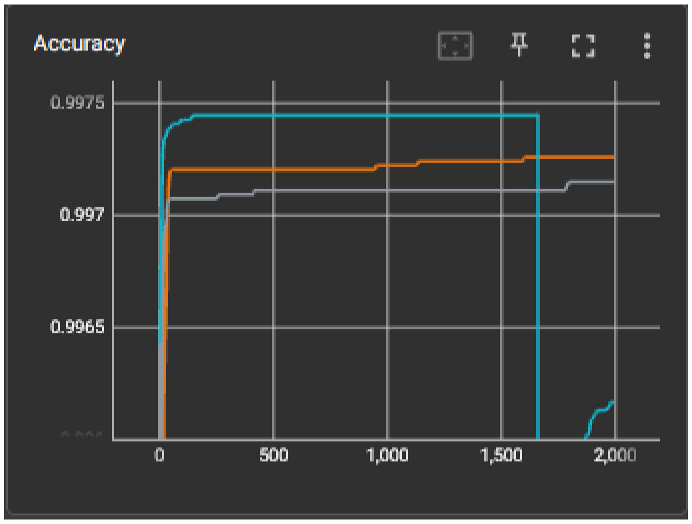
Training set accuracy: 99.6% Test set accuracy: 97.73%
The shortcomings of Adam are also reflected here. If the training stops when the epoch is close to 4000, we will miss the local optimal solution, which is something we don't want to see. Increasing the number of epochs can still find a better answer.

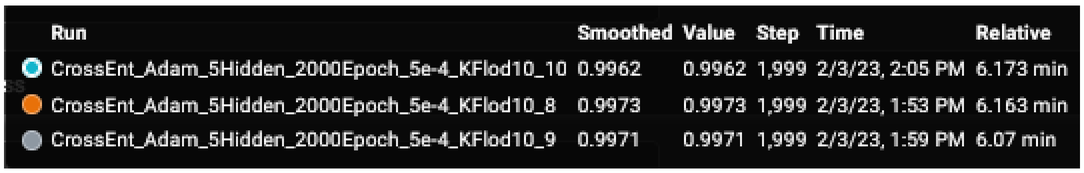
4. TrainData:Original Layers:6 Opt: AdamW LRate:5e-4 Epoch:2000 Loss:CrossEntropy
I started trying different Optimizers.
Training set **accuracy: 99.55% Test set accuracy: 97.75%**
So far, I think the limit is almost here. I also made a lot of other attempts, such as changing the loss function to NLLLoss. NLLLoss did not give me better model accuracy.
Until I made a whole new attempt.

5.**TrainData:KFold K=5 Layers:6 Opt: Adam LRate:5e-4 Epoch:2000 Loss:CrossEntropy**
Here I use KFold to separate the training set, and the test set also uses the test set provided by the project.
In such a method, **training set accuracy: 99.66% test set accuracy: 97.99%**
This made my accuracy even better, so I started to try this part.

6. **TrainData:KFold K=10 Layers:6 Opt: Adam LRate:5e-4 Epoch:2000 Loss:CrossEntropy**

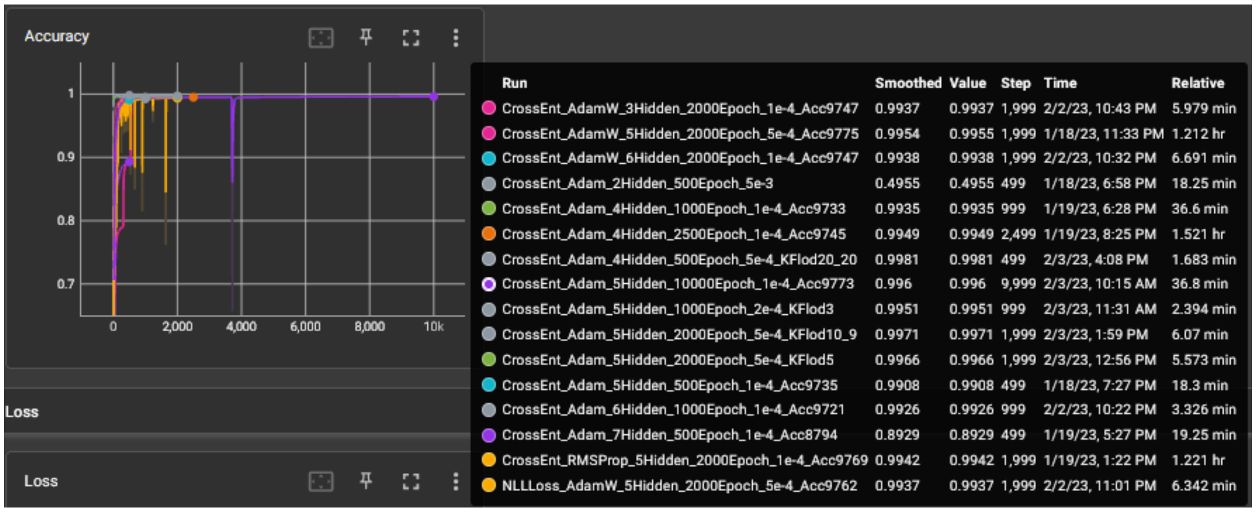
In this attempt, the last few epochs when fold=10 left the local optimal solution. But if you use the result of fold=9 to test, you will get the **training set accuracy rate: 99.71%** and the **test set accuracy rate: 98.11%.** This is a very good result.





	Run	Smoothed	Value	Step	Time	Relative
●	CrossEnt_Adam_5Hidden_2000Epoch_5e-4_KFlod10_10	0.9962	0.9962	1,999	2/3/23, 2:05 PM	6.173 min
●	CrossEnt_Adam_5Hidden_2000Epoch_5e-4_KFlod10_8	0.9973	0.9973	1,999	2/3/23, 1:53 PM	6.163 min
●	CrossEnt_Adam_5Hidden_2000Epoch_5e-4_KFlod10_9	0.9971	0.9971	1,999	2/3/23, 1:59 PM	6.07 min

Finally, I attach the learning rate of all my models. In addition, I found that if the dataset read by torch uses a list, it will cause a training burden, but nparray will not. I didn't go to understand the time complexity of it. But this problem can be clearly found from my training process. In the first few weeks, it took me 1.5 hours to do 2500epoch training. After I improved it, I found that it only took 37 minutes to train for 10,000 epochs. This speeds up my training process and gives me more time to find other ways to improve my model.
In**"script\stage_2_script"**, use the terminal to enter the command: "**tensorboard --logdir=runs**"
to see all my training processes.



	Run	Smoothed	Value	Step	Time	Relative
●	CrossEnt_AdamW_3Hidden_2000Epoch_1e-4_Acc9747	0.9937	0.9937	1,999	2/2/23, 10:43 PM	5.979 min
●	CrossEnt_AdamW_5Hidden_2000Epoch_5e-4_Acc9775	0.9954	0.9955	1,999	1/18/23, 11:33 PM	1.212 hr
●	CrossEnt_AdamW_6Hidden_2000Epoch_1e-4_Acc9747	0.9938	0.9938	1,999	2/2/23, 10:32 PM	6.691 min
●	CrossEnt_Adam_2Hidden_500Epoch_5e-3	0.4955	0.4955	499	1/18/23, 6:58 PM	18.25 min
●	CrossEnt_Adam_4Hidden_1000Epoch_1e-4_Acc9733	0.9935	0.9935	999	1/19/23, 6:28 PM	36.6 min
●	CrossEnt_Adam_4Hidden_2500Epoch_1e-4_Acc9745	0.9949	0.9949	2,499	1/19/23, 8:25 PM	1.521 hr
●	CrossEnt_Adam_4Hidden_500Epoch_5e-4_KFlod20_20	0.9981	0.9981	499	2/3/23, 4:08 PM	1.683 min
◉	CrossEnt_Adam_5Hidden_10000Epoch_1e-4_Acc9773	0.996	0.996	9,999	2/3/23, 10:15 AM	36.8 min
●	CrossEnt_Adam_5Hidden_1000Epoch_2e-4_KFlod3	0.9951	0.9951	999	2/3/23, 11:31 AM	2.394 min
●	CrossEnt_Adam_5Hidden_2000Epoch_5e-4_KFlod10_9	0.9971	0.9971	1,999	2/3/23, 1:59 PM	6.07 min
●	CrossEnt_Adam_5Hidden_2000Epoch_5e-4_KFlod5	0.9966	0.9966	1,999	2/3/23, 12:56 PM	5.573 min
●	CrossEnt_Adam_5Hidden_500Epoch_1e-4_Acc9735	0.9908	0.9908	499	1/18/23, 7:27 PM	18.3 min
●	CrossEnt_Adam_6Hidden_1000Epoch_1e-4_Acc9721	0.9926	0.9926	999	2/2/23, 10:22 PM	3.326 min
●	CrossEnt_Adam_7Hidden_500Epoch_1e-4_Acc8794	0.8929	0.8929	499	1/19/23, 5:27 PM	19.25 min
●	CrossEnt_RMSProp_5Hidden_2000Epoch_1e-4_Acc9769	0.9942	0.9942	1,999	1/19/23, 1:22 PM	1.221 hr
●	NLLLoss_AdamW_5Hidden_2000Epoch_5e-4_Acc9762	0.9937	0.9937	1,999	2/2/23, 11:01 PM	6.342 min