Homework 3

Spring 2020 CSCI 5525: Machine Learning

Due on March 20th 11:59 p.m.

Homework Policy. (1) You are encouraged to collaborate with your classmates on homework problems, but each person must write up the final solutions individually. You need to fill in above to specify which problems were a collaborative effort and with whom. (2) Regarding online resources, you should **not**:

- Google around for solutions to homework problems,
- Ask for help on online.
- Look up things/post on sites like Quora, StackExchange, etc.

Submission. Submit a PDF using this LaTeX template for written assignment part and submit .py files for all programming part. You should upload all the files on Canvas.

Written Assignment

Instruction. For each problem, you are required to write down a full mathematical proof to establish the claim.

Problem 1. VC Dimension.

(**5 points**) Prove this claim formally: If $\mathcal{F}$ is finite, then $\text{VCD}(\mathcal{F}) \leq \log(|\mathcal{F}|)$.

Your answer. As $\mathcal{F}$ is finite, we assume that the $\text{VCD}(\mathcal{F}) = n$, and that is to say, $\mathcal{F}$ can shatter n points, so there are at least 2^n options for the label of y in $\mathcal{F}$.

$$\text{VCD}(\mathcal{F}) = n = \log(2^n) \leq \log(|\mathcal{F}|)$$

Problem 2. VC Dimension.

(**5 points**) Suppose the space of instances $X = \mathbb{R}^2$.

A binary classifier H_t whose decision boundary is a triangle as Fig. 1 sketches, where points inside the triangle are classified as positive labels. What is the VC dimension of H_t? Please give a proof.

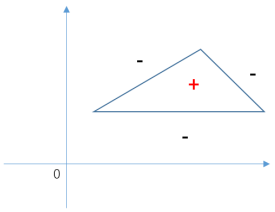

Figure 1: sketch of H_t

Your answer. The VC dimension is 7.
Assume we have 7 points on a circle, if there are only negative points on an arc between two positive points, we call this arc a negative block.
Apparently, with 7 points, we can find at most 3 negative blocks. If we can find 4 negative blocks, we should have at least 4 positive points(to separate four blocks) and 4 negative points(each block have at least one negative point), it's impossible.
So with at most 3 negative blocks, one edge of the triangle decision boundary can be used to separate one negative block from the positive points.
Assume there are 8 points, if one of the points is inside the convex hull of the rest, we cannot label that point negative while the rest are positive. Otherwise, if all points are on the convex hull, we cannot label these points with an alternating order (+,-,+,-,+,-,+,-).

Problem 3. Uniform convergence.

In this problem, we will prove a stronger generalization error bound for the binary classification that uses more information about the distribution. Let us say that P is a distribution over (X, Y) pairs where $X \in \mathcal{X}$ and $Y \in \{+1, -1\}$. Let $\mathcal{H} \subset \mathcal{X} \to \{+1, -1\}$ be a finite hypothesis class and let ℓ denote the zero-one loss $\ell(\hat{y}, y) = 1\{\hat{y} \neq y\}$. Let $R(h) = \mathbb{E}\,\ell(h(X), Y)$ denote the risk and let $h^* = \min_{h\in\mathcal{H}} R(h)$. Given n samples, let $\hat{h}_n$ denote the empirical risk minimizer. Here, we want to prove a sample complexity bound of the form:

$$R(\hat{h}_n) - R(h^*) \leq c_1\sqrt{\frac{R(h^*)\log(|\mathcal{H}|/\delta)}{n}} + c_2\frac{\log(|\mathcal{H}|/\delta)}{n} \tag{1}$$

for constants c_1, c_2. If $R(h^*)$ is small, this can be a much better bound than the usual excess risk bound. In particular, if $R(h^*) = 0$, this bound recovers the $1/n$-rate.

To prove the result, we will use Bernstein's inequality, which is a sharper concentration result.

Theorem 1 *(Bernstein's inequality). Let $X_1, ..., X_n$ be i.i.d real-valued random variables with mean zero, and such that $|X_i| \leq M$ for all i. Then, for all $t > 0$*

$$\mathbb{P}[\sum_{i=1}^{n} X_i \geq t] \leq \exp\left(-\frac{t^2/2}{\sum_{i=1}^{n}\mathbb{E}[X_i^2] + Mt/3}\right) \tag{2}$$

(a) (**6 points**) Using this inequality, show that with probability at least $1 - \delta$

$$|\bar{X}| \leq \sqrt{\left(\frac{2\,\mathbb{E}[X^2]\log(2/\delta)}{n}\right)} + \frac{2M\log(2/\delta)}{3n} \tag{3}$$

where $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and X_i's satisfy the conditions of Bernstein's inequality.

(b)(**Extra Credits 4 points**) Then, we will use Eq. (3) and the union bound, show Eq. (1)

At first, you will use the idea of Bernstein's inequality to show that with probability at least $1-\delta$, $\forall h \in \mathcal{H}$

$$|\hat{R}(h) - R(h)| \leq \sqrt{\left(\frac{R(h)2\log(2|\mathcal{H}|/\delta)}{n}\right)} + \frac{2M\log(2|\mathcal{H}|/\delta)}{3n} \tag{4}$$

where $\hat{R}(h)$ is the empirical risk and $R(h)$ is the true risk.

(Hint: consider random variables X as the value of loss function. Pay attention to that X has zero mean.)

(c) (**Extra Credits 4 points**) Finally, we will use Eq. (4) to find (1)

Your answer. (a)Let $t = nt$, we have:

$$\mathbb{P}[\frac{1}{n}\sum_{i=1}^{n} X_i \geq t] \leq \exp\left(-\frac{n^2t^2/2}{\sum_{i=1}^{n}\mathbb{E}[X_i^2] + Mnt/3}\right) \tag{5}$$

Let $X_i = -X_i$, $t = nt$, we have:

$$\mathbb{P}[\frac{1}{n}\sum_{i=1}^{n} X_i \leq -t] \leq \exp\left(-\frac{n^2t^2/2}{\sum_{i=1}^{n}\mathbb{E}[X_i^2] + Mnt/3}\right) \tag{6}$$

So the equation with the (5) + (6), we have:

$$\mathbb{P}[|\frac{1}{n}\sum_{i=1}^{n} X_i| \geq t] = \mathbb{P}[|\bar{X}| \geq t] \leq 2\exp\left(-\frac{n^2t^2/2}{\sum_{i=1}^{n}\mathbb{E}[X_i^2] + Mnt/3}\right) \tag{7}$$

Let $2\exp\left(-\frac{n^2t^2/2}{\sum_{i=1}^{n}\mathbb{E}[X_i^2] + Mnt/3}\right) = \delta$, we have:

$$\frac{n^2t^2/2}{\sum_{i=1}^{n}\mathbb{E}[X_i^2] + Mnt/3} = \ln(2/\delta)$$

$$n^2t^2 - \frac{2}{3}\ln(2/\delta)Mnt - 2\ln(2/\delta)\sum_{i=1}^{n}\mathbb{E}[X_i^2] = 0$$

With the quadratic formula, and $t \geq 0$, we keep the positive solution. Besides, we know that $\sqrt{x^2+y^2} \leq x+y$ When $x, y \geq 0$:

$$
\begin{aligned}
t &= \frac{\frac{2}{3}\ln(2/\delta)Mn + \sqrt{(\frac{2}{3}\ln(2/\delta)Mn)^2 - 4*(n^2)*(-2\ln(2/\delta)\sum_{i=1}^{n}\mathbb{E}[X_i^2]}}{2n^2}) \\
&= \frac{\ln(2/\delta)M}{3n} + \sqrt{(\frac{\ln(2/\delta)M}{3n})^2 + \frac{2\ln(2/\delta)\sum_{i=1}^{n}\mathbb{E}[X_i^2]}{n^2}} \\
&\leq \frac{\ln(2/\delta)M}{3n} + \frac{\ln(2/\delta)M}{3n} + \sqrt{\frac{2\ln(2/\delta)\sum_{i=1}^{n}\mathbb{E}[X_i^2]}{n^2}} \\
&= \frac{2\ln(2/\delta)M}{3n} + \sqrt{\frac{2\ln(2/\delta)\sum_{i=1}^{n}\mathbb{E}[X_i^2]}{n^2}} \\
&= \frac{2M\ln(2/\delta)}{3n} + \sqrt{\frac{2\,\mathbb{E}[X^2]\ln(2/\delta)}{n}}
\end{aligned}
$$

Then, we have:

$$\mathbb{P}[|\bar{X}| \geq t] \leq \delta$$

Or:

$$\mathbb{P}[|\bar{X}| \leq t] \geq 1-\delta$$

And so that:

$$|\bar{X}| \leq t \leq \frac{2M\ln(2/\delta)}{3n} + \sqrt{\frac{2\,\mathbb{E}[X^2]\ln(2/\delta)}{n}}$$

Problem 4. Bias-Variance Trade-off.

(**Total 5 points**) In the lecture, **expected test error using a machine learning algorithm, $\boldsymbol{A}$**, can be given as (with respect to squared loss):

$$E_{x,y,D}\left[\left(f_D(x)-y\right)^2\right]$$

where D represents set of training points and (x, y) pairs are test points

We can write:

$$
\begin{aligned}
&E_{x,y,D}\left[\left(f_D(x)-y\right)^2\right] \\
&= E_{x,y,D}\left[\left(\left(f_D(x)-\bar{f}(x)\right)+\left(\bar{f}(x)-y\right)\right)^2\right] \\
&= E_{x,D}\left[\left(f_D(x)-\bar{f}(x)\right)^2\right] + 2\,E_{x,y,D}\left[\left(f_D(x)-\bar{f}(x)\right)\left(\bar{f}(x)-y\right)\right] + E_{x,y}\left[\left(\bar{f}(x)-y\right)^2\right]
\end{aligned}
$$

a) Prove $E_{x,y,D}\left[\left(f_D(x)-\bar{f}(x)\right)\left(\bar{f}(x)-y\right)\right]=0$

Proving above gives $E_{x,y,D}\left[\left(f_D(x)-y\right)^2\right]=\underbrace{E_{x,D}\left[\left(f_D(x)-\bar{f}(x)\right)^2\right]}_{\text{Variance}}+E_{x,y}\left[\left(\bar{f}(x)-y\right)^2\right]$

where 1^{st} term is **variance**. The 2^{nd} term can further be decomposed as follows:

$$E_{x,y}\left[\left(\bar{f}(x)-y\right)^2\right]=E_{x,y}\left[\left(\bar{f}(x)-\bar{y}(x)\right)+\left(\bar{y}(x)-y\right)^2\right]$$
$$=\underbrace{E_{x,y}\left[\left(\bar{y}(x)-y\right)^2\right]}_{\text{Noise}}+\underbrace{E_x\left[\left(\bar{f}(x)-\bar{y}(x)\right)^2\right]}_{\text{Bias}^2}+2\,E_{x,y}\left[\left(\bar{f}(x)-\bar{y}(x)\right)\left(\bar{y}(x)-y\right)\right]$$

where 1^{st} term is **Noise** and 2^{nd} term is **Bias²**

b) Prove $E_{x,y}\left[\left(\bar{f}(x)-\bar{y}(x)\right)\left(\bar{y}(x)-y\right)\right]=0$

Proving above gives the magic of **Bias-Variance Decomposition**,

$$\underbrace{E_{x,y,D}\left[\left(f_D(x)-y\right)^2\right]}_{\text{Expected Test Error}}=\underbrace{E_{x,D}\left[\left(f_D(x)-\bar{f}(x)\right)^2\right]}_{\text{Variance}}+\underbrace{E_{x,y}\left[\left(\bar{y}(x)-y\right)^2\right]}_{\text{Noise}}+\underbrace{E_x\left[\left(\bar{f}(x)-\bar{y}(x)\right)^2\right]}_{\text{Bias}^2}$$

Your answer. a)

$$\begin{aligned}E_{\mathbf{x},y,D}\left[\left(f_D(\mathbf{x})-\bar{f}(\mathbf{x})\right)\left(\bar{f}(\mathbf{x})-y\right)\right]&=E_{\mathbf{x},y}\left[\left(\bar{f}(\mathbf{x})-y\right)E_D\left[f_D(\mathbf{x})-\bar{f}(\mathbf{x})\right]\right]\\&=E_{\mathbf{x},y}\left[\left(\bar{f}(\mathbf{x})-y\right)\left(E_D\left[f_D(\mathbf{x})\right]-\bar{f}(\mathbf{x})\right)\right]\\&=E_{\mathbf{x},y}\left[\left(\bar{f}(\mathbf{x})-y\right)\left(\bar{f}(\mathbf{x})-\bar{f}(\mathbf{x})\right)\right]\\&=0\end{aligned}$$

b)

$$\begin{aligned}E_{\mathbf{x},y}\left[\left(\bar{f}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\left(\bar{y}(\mathbf{x})-y\right)\right]&=E_{\mathbf{x}}\left[\left(\bar{f}(\mathbf{x})-\bar{y}(\mathbf{x})\right)E_{y|\mathbf{x}}\left(\bar{y}(\mathbf{x})-y\right)\right]\\&=E_{\mathbf{x}}\left[\left(\bar{f}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\left(\bar{y}(\mathbf{x})-E_{y|\mathbf{x}}\left(y\right)\right)\right]\\&=E_{\mathbf{x}}\left[\left(\bar{f}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\left(\bar{y}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\right]\\&=0\end{aligned}$$

Problem 5. Neural network.

5.1 (**6 points**) Consider a neural neural network with input $x \in R^{d_i,1}$ and the number of classes being d_o. The network can be given by $\hat{y} = \text{softmax}(W_2(ReLU(W_1 x + b_1)) + b_2)$ where $W_1 \in R^{d_1,d_i}$, $b_1 \in R^{d_1,1}$, $W_2 \in R^{d_o,d_1}$, $b_2 \in R^{d_o,1}$ and Loss $L = \text{cross\_entropy}(y, \hat{y})$

Cross entropy loss between $y, \hat{y}$ is given by $-\sum_i^m y_i \log(\hat{y}_i)$ and softmax clamps x to [0,1] range using $\frac{e^{x_i}}{\sum_i e^{x_i}}$.

Derive expressions for partial derivatives (be precise and clear) $\frac{\partial L}{\partial W_2}, \frac{\partial L}{\partial b_2}, \frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial b_1}$

Your answer. The network is equivalent to:

$$\begin{aligned} y' &= W_1 x + b_1 \\ a &= ReLU(y') \\ z &= W_2 a + b_2 \\ \hat{y} &= \text{softmax}(z) \\ L &= -\sum_i^m y_i \log(\hat{y}_i) \end{aligned}$$

The $\frac{\partial L}{\partial \hat{y}_i}$ is:

$$\frac{\partial L}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i}$$

The $\frac{\partial \hat{y}_i}{\partial z_i}$ is:

$$\frac{\partial \hat{y}_i}{\partial z_i} = \begin{cases} -\frac{e^{z_i} e^{z_j}}{(\sum_i e^{z_i})^2} = -\hat{y}_i \hat{y}_j & (i \neq j) \\ \frac{e^{z_i} \sum_i e^{z_i} - (e^{z_i})^2}{(\sum_i e^{z_i})^2} = \hat{y}_i(1 - \hat{y}_i) & (i = j) \end{cases}$$

So the $\frac{\partial L}{\partial z_i}$ is:

$$\begin{aligned} \frac{\partial L}{\partial z_i} &= \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i} \\ &= \sum_{i=j} \frac{\partial L_j}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial z_i} + \sum_{i \neq j} \frac{\partial L_j}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial z_i} \\ &= \sum_{i=j} (-\frac{y_i}{\hat{y}_i})(\hat{y}_i(1 - \hat{y}_i)) + \sum_{i \neq j} (-\frac{y_j}{\hat{y}_j})(-\hat{y}_i \hat{y}_j)) \\ &= -y_i(1 - \hat{y}_i) + \sum_{i \neq j} y_j \hat{y}_i \\ &= -y_i + \sum_j y_j \hat{y}_i \\ &= \hat{y}_i - y_i \end{aligned}$$

The $\frac{\partial L}{\partial W_2}$ is:

$$\begin{aligned} \frac{\partial L}{\partial W_2} &= \frac{\partial L}{\partial z} \frac{\partial z}{\partial W_2} \\ &= (\hat{y} - y) a^T \end{aligned}$$

The $\frac{\partial L}{\partial b_2}$ is:

$$\begin{aligned}\frac{\partial L}{\partial b_2} &= \frac{\partial L}{\partial z}\frac{\partial z}{\partial b_2} \\ &= \hat{y} - y\end{aligned}$$

The $\frac{\partial L}{\partial W_1}$ is:

$$\begin{aligned}\frac{\partial L}{\partial W_1} &= \frac{\partial L}{\partial z}\frac{\partial z}{\partial a}\frac{\partial a}{\partial y'}\frac{\partial y'}{\partial W_1} \\ &= \begin{cases} 0, y' \leq 0 \\ W_2^T(\hat{y} - y)x^T, y' > 0 \end{cases}\end{aligned}$$

The $\frac{\partial L}{\partial b_1}$ is:

$$\begin{aligned}\frac{\partial L}{\partial b_1} &= \frac{\partial L}{\partial z}\frac{\partial z}{\partial a}\frac{\partial a}{\partial y'}\frac{\partial y'}{\partial b_1} \\ &= \begin{cases} 0, y' \leq 0 \\ W_2^T(\hat{y} - y), y' > 0 \end{cases}\end{aligned}$$

5.2 (**6 points**) Consider a convolutional neural network architecture given in figure 2 for MNIST to classify digits (10 digits). Given that the input images are reduced to size 10×10 with only 1 channel (represented as a matrix in $\mathbb{R}^{10\times 10}$). The covolutional layer ues a 3×3 filter, $\mathbf{W}_{conv}$, with stride 2 and use zero padding to make it fit. The dimensions of the outputs of each layer are shown.

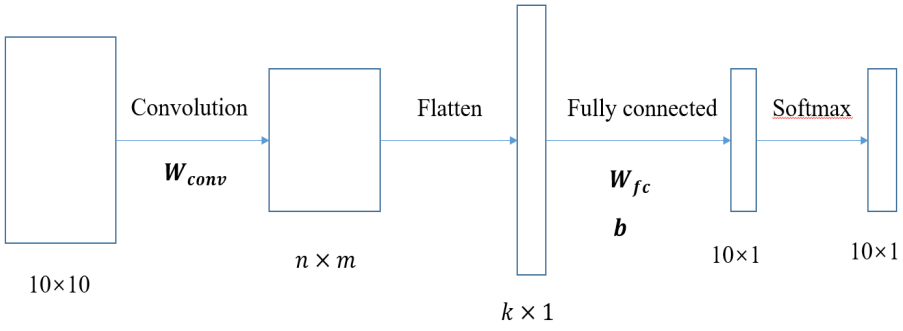

Figure 2: A toy CNN

(a) What is n, m, k in the graph?
(b) What is the size of $\mathbf{W}_{conv}$, $\mathbf{W}_{fc}$, and $\mathbf{b}$?

Your answer. (a) $m = n = 5$, $k = 25$
(b) the size of $\mathbf{W}_{conv}$ is $(1, 1, 3, 3)$, $\mathbf{W}_{fc}$ is $(25, 10)$, and $\mathbf{b}$ is $(10, 1)$

Programming Assignment

Instruction. For each problem, you are required to report descriptions and results in the PDF and submit code as python file (.py) (as per the question).

- **Python** version: Python 3.
- Please follow PEP 8 style of writing your Python code for better readability in case you are wondering how to name functions & variables, put comments and indent your code
- **Packages allowed**: pytorch, numpy, pandas, matplotlib
- **Submission**: Submit report, description and explanation of results in the main PDF and code in .py files.
- Please PROPERLY COMMENT your code in order to have utmost readability
- Please provide the required functions for each problem.
- There shall be NO runtime errors involved.
- There would be PENALTY if any of the above is not followed

Programming Common

This programming assignment focuses on implementing neural network for handwritten digits. This problem is about multi class classification and you will use MNIST dataset. You already have an idea of the dataset by now.

You will use pytorch to implement neural network. The implementation has to be for the CPU version only - No GPU's or MPI parallel programming is required for this assignment. Follow the installation instructions at https://pytorch.org in case you want to use your local machine, we recommend using conda environment.

Here is the pytorch tutorial. If you are new to pytorch, we recommend you to go through the tutorial here. https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

Problem 6. Neural Network Implementation.

(**27 Points**)

1. (**No Points**) You can directly download MNIST in pytorch using torchvision.datasets.MNIST. It will allow you to

   ```
   trainset = torchvision.datasets.MNIST(root='./data', train=True, download=True,
   transform=transform)
   testset = torchvision.datasets.MNIST(root='./data', train=False, download=True,
   transform=transform)
   ```

2. (**9 Points**) First, implement a multi-layer fully connected network:
   - Input: 1-channel input, size 28x28
   - Keep batch size as 32.
   - Fully connected layer 1: Input with bias; output - 128

- ReLu Layer
- Fully connected layer 2: input - 128; output - 10
- Softmax layer
- Use cross entropy as loss function
- Use SGD as optimizer.

Train using mini-batches of the given batch size. Plot loss and training accuracy for every epoch. At the end of the training, save your model with the name "mnist-fc". Load the saved model and report testing accuracy on the reloaded model. We should be able to reload your trained model and test.

epoch: An epoch tells you the number of times the learning algorithm sees the whole training data set. When it has seen all the training samples, you say 1 epoch is over and you start iterating in the next epoch.

3. (**10 Points**) Implement a convolutional neural network with the following specifications.
   - Input: 1-channel input, size 28x28
   - Keep batch size as 32.
   - Convolution layer: Convolution kernel size is (3, 3) with stride as 1. Input channels - 1; Output channels - 20
   - Max-pool: 2x2 max pool
   - ReLu Layer
   - Flatten input for feed to fully connected layers
   - Fully connected layer 1: flattened input with bias; output - 128
   - ReLu Layer
   - Fully connected layer 2: input - 128; output - 10
   - Softmax layer as above
   - Use cross entropy as loss function
   - Use SGD as optimizer.

   Train using mini-batches of the given batch size. Plot loss and training accuracy for every epoch. At the end of the training, save your model with the name "mnist-cnn". Load the saved model and report testing accuracy on the reloaded model. We should be able to reload your trained model and test.

4. (**4 Points**) For the convolutional network implemented above, vary the batch sizes as [32, 64, 96, 128] and plot the convergence run time vs batch sizes.

5. (**4 Points**) "torch optim" provides many optimizers. Try the following optimizers in your last implementation - "SGD", "ADAM", "ADAGRAD". (SGD is already covered in the class, for ADAM and ADAGRAD refer to https://arxiv.org/abs/1412.6980and http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf respectively). Plot loss vs epochs for each optimizer. Briefly describe.

Submission: Submit all plots requested and explanation in latex PDF. Submit your program in a file named (hw3_mnistfc.py and hw3_mnistcnn.py).

2. The loss and the accuracy are shown below, the testing accuracy is 95.36%:

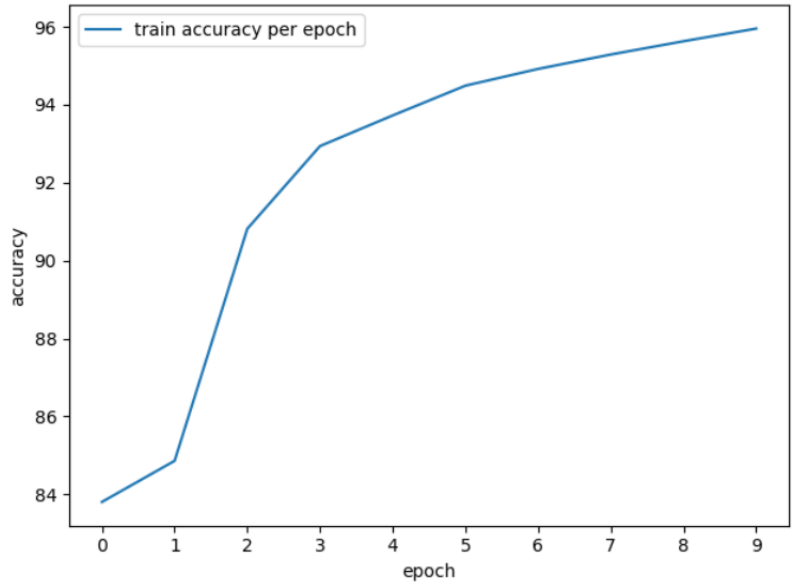

Figure 3: full connect training accuracy

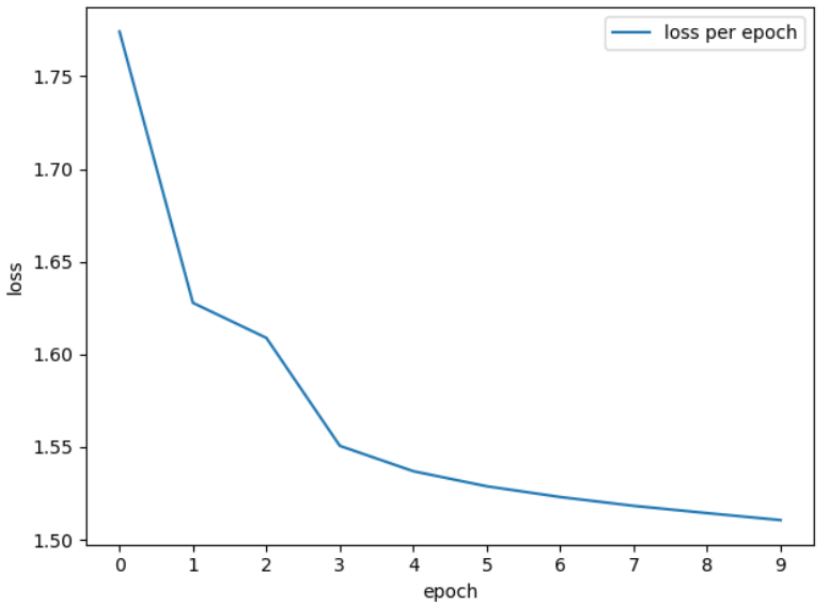

Figure 4: full connect training loss

3. The loss and the accuracy are shown below, the testing accuracy is 97.57%.

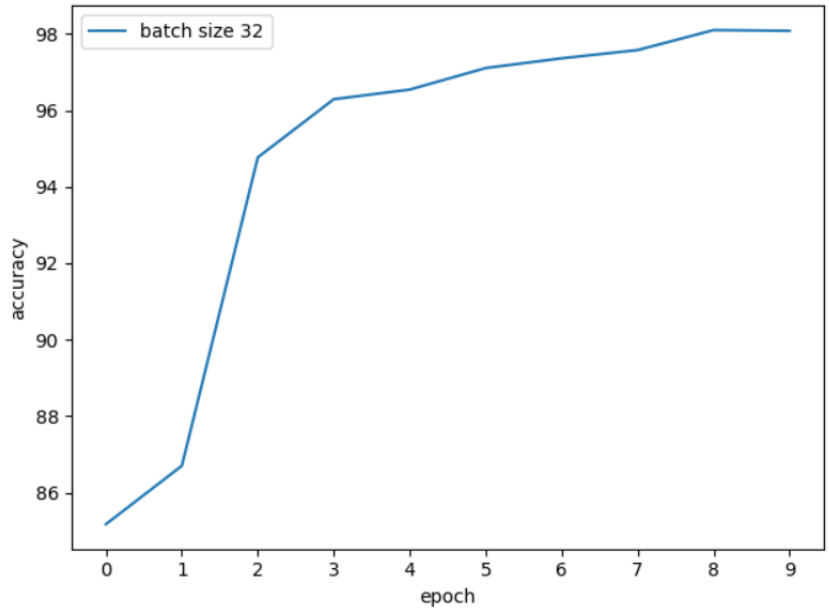

Figure 5: CNN batch 32 training accuracy

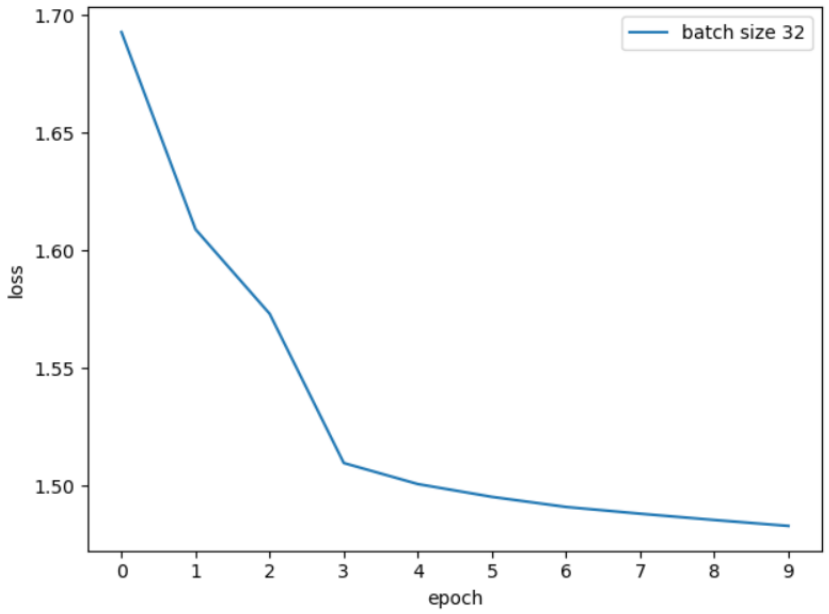

Figure 6: CNN batch 32 training loss

4. The comparisons with different batches are shown below:

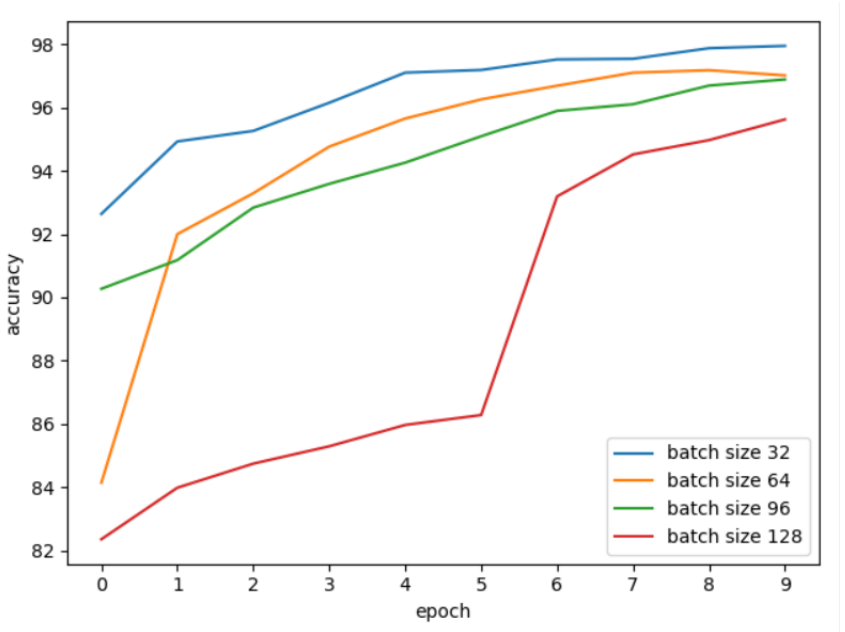

Figure 7: Different batches training accuracy

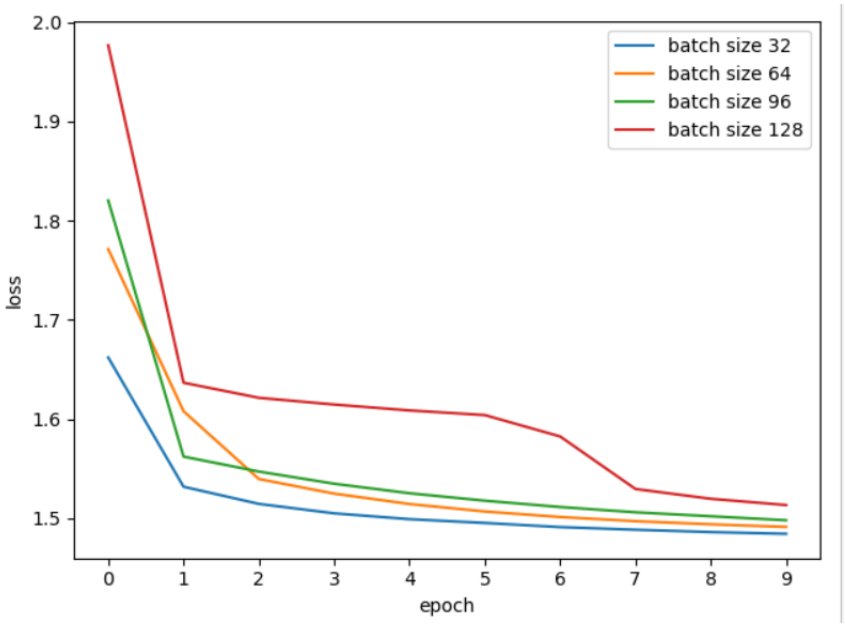

Figure 8: Different batches training loss

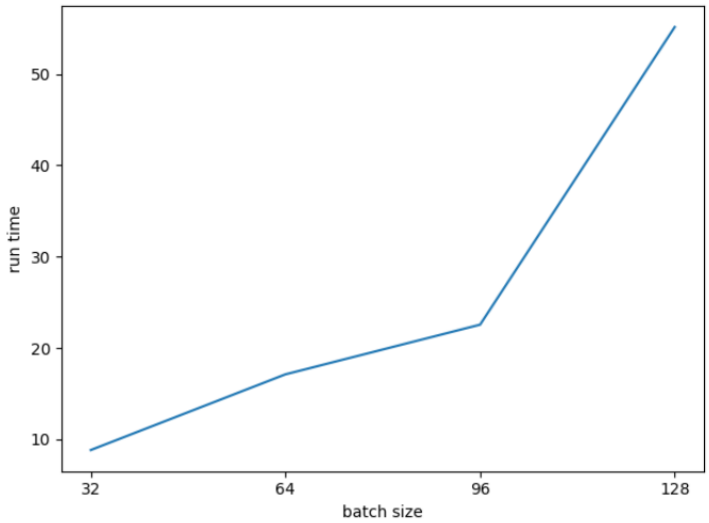

Figure 9: Run time compare when avgloss=1.53

5. The comparisons with different optimizer are shown below, the learning rates for SGD and Adagrad are 0.01, the learning rate for Adam is 0.001, all other parameters are default:

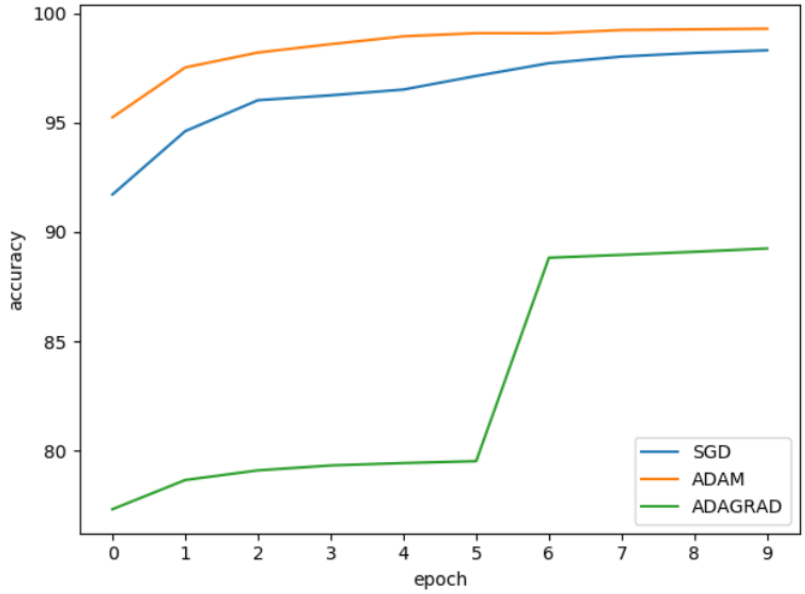

Figure 10: Different optimizer training accuracy

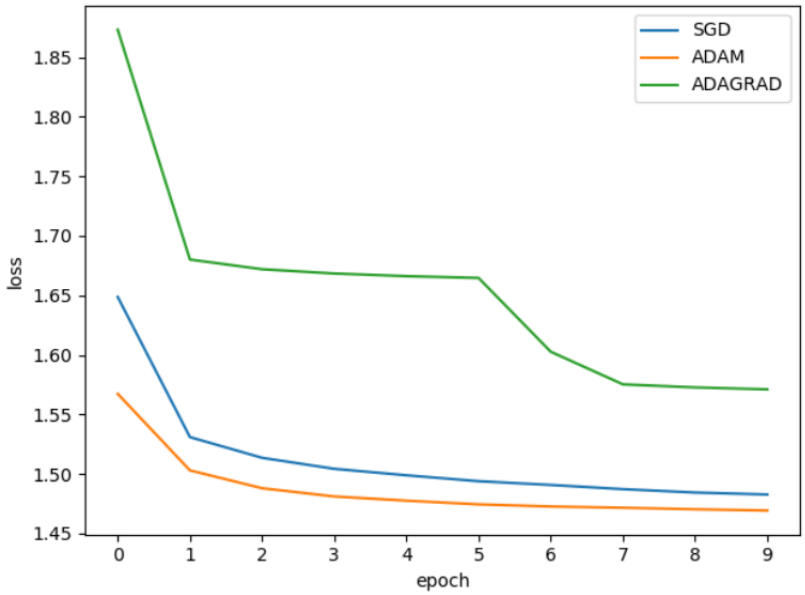

Figure 11: Different optimizer training loss