HW 2

NetID: ziyet3

1.Expectation Maximization

(a) Mixture of Bernoullis

i.

$$P(x|q) = \prod_{d=1}^{D} P(x_d|q_d) = \prod_{d=1}^{D} q_d^{x_d}(1-q_d)^{1-x_d}$$

ii.

$$P(x^{(i)}|p,\pi) = \sum_{k=1}^{K} \pi_k P(x^{(i)}|p^{(k)})$$

iii.

$$\log P(D|\pi,p) = \log \prod_{i=1}^{n} P(x^{(i)}|\pi,p) = \sum_{i=1}^{n}\sum_{k=1}^{K} \pi_k P(x^{(i)}|p^{(k)})$$

(b) Expectation step

i.

$$P(z_k^{(i)} = 1) = \pi_k$$

$$P(z^{(i)}|\pi) = \prod_{k=1}^{K} \pi_k^{z_k^{(i)}}$$

ii.

$$P(x^{(i)}|z^{(i)},p,\pi) = \prod_{k=1}^{K} P(x^{(i)}|p^{(k)})^{z_k^{(i)}}$$

iii.

$$P(Z,D|p,\pi) = P(D|Z,p,\pi) * P(Z|\pi)$$

$$P(D|Z,p,\pi) = \prod_{i=1}^{n} P(x^{(i)}|z^{(i)},p,\pi) = \prod_{i=1}^{n}\prod_{k=1}^{K} P(x^{(i)}|p^{(k)})^{z_k^{(i)}}$$

$$P(Z|\pi) = \prod_{i=1}^{n} P(z^{(i)}|\pi) = \prod_{i=1}^{n}\prod_{k=1}^{K} \pi_k^{z_k^{(i)}}$$

$$P(Z,D|p,\pi) = P(D|Z,p,\pi) * P(Z|\pi) = \prod_{i=1}^{n}\prod_{k=1}^{K} P(x^{(i)}|p^{(k)})^{z_k^{(i)}} \pi_k^{z_k^{(i)}}$$

iV.
Need to show that: $\eta(z_k^{(i)}) = RHS = \frac{\pi_k P(x^{(i)}|p^{(k)})}{\sum_{j=1}^{K} \pi_j P(x^{(i)}|p^{(j)})} = \frac{\pi_k P(x^{(i)}|p^{(k)})}{P(x^{(i)}|p,\pi)}$

By definition:
$\eta(z_k^{(i)}) = E[z_k^{(i)}|x^{(i)}, \pi, p]$

$= 1 * P(z_k^{(i)} = 1|x^{(i)}, \pi, p) + 0 * P(z_k^{(i)} = 0|x^{(i)}, \pi, p)$

$= P(z_k^{(i)} = 1|x^{(i)}, \pi, p)$

$= \frac{P(z_k^{(i)}=1, x^{(i)}|\pi,p)}{P(x^{(i)}|\pi,p)}$

$= \frac{P(x^{(i)}|p^{(k)})\pi_k}{P(x^{(i)}|\pi,p)} = RHS$

v.
$E[\log P(Z, \mathcal{D}|\tilde{p}, \tilde{\pi})|\mathcal{D}, p, \pi]$

$= \sum_{i=1}^{n}\sum_{k=1}^{K} P(z_k^{(i)}|x^{(i)}, p, \pi) * \log P(z_k^{(i)}, x^{(i)}|\tilde{p}, \tilde{\pi})$

$= \sum_{i=1}^{n}\sum_{k=1}^{K} P(z_k^{(i)}|x^{(i)}, p, \pi) * \log(P(x^{(i)}|\tilde{p}^{(k)})^{z_k^{(i)}} \tilde{\pi}_k^{z_k^{(i)}})$

$= \sum_{i=1}^{n}\sum_{k=1}^{K} (P(z_k^{(i)}|x^{(i)}, p, \pi) * z_k^{(i)}) * \log(P(x^{(i)}|\tilde{p}^{(k)})\tilde{\pi}_k)$

$= \sum_{i=1}^{n}\sum_{k=1}^{K} \eta(z_k^{(i)}) * \log(P(x^{(i)}|\tilde{p}^{(k)})\tilde{\pi}_k)$

$= \sum_{i=1}^{n}\sum_{k=1}^{K} \eta(z_k^{(i)}) * \log(\tilde{\pi}_k \prod_{d=1}^{D} \tilde{p}_d^{(k)^{x_d^{(i)}}} (1 - \tilde{p}_d^{(k)})^{1-x_d^{(i)}})$

$= \sum_{i=1}^{n}\sum_{k=1}^{K} \eta(z_k^{(i)}) * [\log \tilde{\pi}_k + \sum_{d=1}^{D} x_d^{(i)} * \log \tilde{p}_d^{(k)} + (1 - x_d^{(i)}) * \log(1 - \tilde{p}_d^{(k)})]$

(c) Maximization step
i.
$0 = \frac{\delta}{\delta \tilde{p}_d^{(k)}} \{\sum_{i=1}^{n}\sum_{k=1}^{K} \eta(z_k^{(i)}) * \sum_{d=1}^{D} x_d^{(i)} * \log \tilde{p}_d^{(k)} + (1 - x_d^{(i)}) * \log(1 - \tilde{p}_d^{(k)})\}$

$= \sum_{i=1}^{n} \frac{\eta(z_k^{(i)}) * x_d^{(i)}}{\tilde{p}_d^{(k)}} - \sum_{i=1}^{n} \frac{\eta(z_k^{(i)}) * (1 - * x_d^{(i)})}{(1 - \tilde{p}_d^{(k)})}$

$= \sum_{i=1}^{n} \frac{\eta(z_k^{(i)})(x_d^{(i)}(1 - \tilde{p}_d^{(k)}) - (1 - x_d^{(i)})\tilde{p}_d^{(k)})}{\tilde{p}_d^{(k)}(1 - \tilde{p}_d^{(k)})}$

$= \sum_{i=1}^{n} \eta(z_k^{(i)})(x_d^{(i)} - \tilde{p}_d^{(k)})$

Then, $\tilde{p}_d^{(k)} = \frac{\sum_{i=1}^{n} \eta(z_k^{(i)}) x_d^{(i)}}{\sum_{i=1}^{n} \eta(z_k^{(i)})} = \frac{\sum_{i=1}^{n} \eta(z_k^{(i)}) x_d^{(i)}}{N_k}$

Therefore, $\tilde{p}^{(k)} = \frac{1}{N_k}[\sum_{i=1}^{n}\eta(z_k^{(i)})x_1^{(i)}, \sum_{i=1}^{n}\eta(z_k^{(i)})x_2^{(i)}, \ldots, \sum_{i=1}^{n}\eta(z_k^{(i)})x_D^{(i)}] = \frac{\sum_{i=1}^{n}\eta(z_k^{(i)})x^{(i)}}{N_k}$

ii.

$$
\begin{aligned}
0 &= \frac{\delta}{\delta\tilde{\pi}_k}\{[\sum_{i=1}^{n}\sum_{k=1}^{K}\eta(z_k^{(i)}) * \log\tilde{\pi}_k] + \lambda(\sum_{k=1}^{K}\tilde{\pi}_k - 1)\} \\
&= \sum_{i=1}^{n}\frac{\eta(z_k^{(i)})}{\tilde{\pi}_k} + \lambda \\
&= \sum_{i=1}^{n}\eta(z_k^{(i)}) + \lambda\tilde{\pi}_k
\end{aligned}
$$

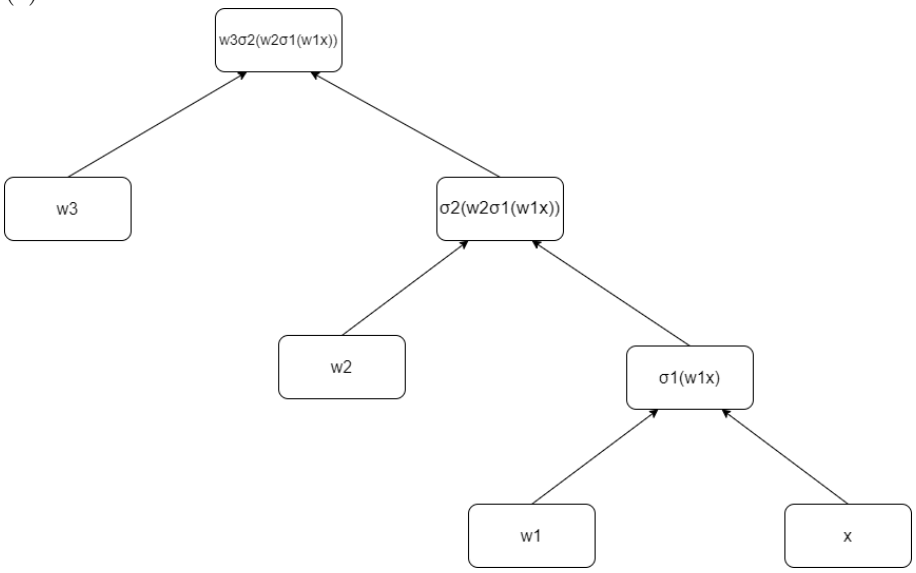
Since $\sum_{k'=1}^{K}\sum_{i=1}^{n}\eta(z_{k'}^{(i)}) + \sum_{k'=1}^{K}\lambda\tilde{\pi}_{k'} = \sum_{k'=1}^{K}\sum_{i=1}^{n}\eta(z_{k'}^{(i)}) + \lambda = 0,$

We have $\lambda = -\sum_{k'=1}^{K}\sum_{i=1}^{n}\eta(z_{k'}^{(i)})$

Therefore, $\tilde{\pi}_k = \frac{\sum_{i=1}^{n}\eta(z_k^{(i)})}{\sum_{k'=1}^{K}\sum_{i=1}^{n}\eta(z_{k'}^{(i)})} = \frac{N_k}{\sum_{k'=1}^{K}N_{k'}}$

2. Deep Net

(a)



(b)
(1)

$$\frac{\delta\sigma_1}{\delta u} = \frac{e^{-u}}{(1+e^{-u})^2}$$

(2)

$$\frac{\delta\sigma_1}{\delta u} = \frac{e^{-u}}{(1+e^{-u})^2} = \frac{e^{-u}}{1+e^{-u}} * \frac{1}{1+e^{-u}} = \frac{1}{e^{u}+1} * \frac{1}{1+e^{-u}}$$

$$= (1 - \frac{e^{u}}{e^{u}+1}) * \frac{1}{1+e^{-u}} = (1 - \frac{1}{1+e^{-u}}) * \frac{1}{1+e^{-u}}$$

$$= (1 - \sigma_1(u))\sigma_1(u)$$

(c)
A forward pass is the process of applying the input data to the neural network, propagating it through the layers of the network, and producing an output. During the forward pass, each layer of the network performs a series of computations that transform the input data into a form that is more useful for the subsequent layers. These computations typically involve a set of learnable parameters (weights and biases) that are initialized randomly and updated during the training process.

A backward pass, also known as backpropagation, is the process of computing the gradients of the loss function with respect to the learnable parameters in the network, by propagating the errors from the output layer back through the

layers of the network. During the backward pass, the gradients are computed using the chain rule of differentiation, which allows the errors to be attributed to each layer of the network in proportion to their contribution to the overall error. The gradients are then used to update the learnable parameters of the network, using an optimization algorithm such as stochastic gradient descent, in order to reduce the loss function and improve the accuracy of the network.

(d)
$\frac{\delta f}{\delta w_3} = \sigma_2(w_2\sigma_1(w_1x))$

So, we should retain $\sigma_2(w_2\sigma_1(w_1x))$ from the forward pass.

(e)
let $y = w_2\sigma_1(w_1x)$, $z = \sigma_2(y)$

$$\frac{\delta f}{\delta w_2} = \frac{\delta f}{\delta z} * \frac{\delta z}{\delta y} * \frac{\delta y}{\delta w_2}$$

$$= w_3 * (1 - \sigma_2(y)) * \sigma_2(y) * \sigma_1(w_1x)$$

$$= w_3 * (1 - \sigma_2(w_2\sigma_1(w_1x))) * \sigma_2(w_2\sigma_1(w_1x)) * \sigma_1(w_1x)$$

So, we should retain $\sigma_1(w_1x)$, $\sigma_2(w_2\sigma_1(w_1x))$, and $w_3\sigma_2(w_2\sigma_1(w_1x))$ from the forward pass.

(f)
let $a = w_1x$, $b = \sigma_1(a)$, $c = w2b$, d $= \sigma_2(c)$.

$$\frac{\delta f}{\delta w_1} = \frac{\delta f}{\delta d} * \frac{\delta d}{\delta c} * \frac{\delta c}{\delta b} * \frac{\delta b}{\delta a} * \frac{\delta a}{\delta w1}$$

$$= w_3 * (1 - \sigma_2(c)) * \sigma_2(c) * w_2 * (1 - \sigma_1(a)) * \sigma_1(a) * x$$

$$= w_3 * (1 - \sigma_2(w_2\sigma_1(w_1x))) * \sigma_2(w_2\sigma_1(w_1x)) * w_2 * (1 - \sigma_1(w_1x)) * \sigma_1(w_1x) * x$$

So, we should retain $\sigma_1(w_1x)$, $w_2\sigma_1(w_1x)$, $\sigma_2(w_2\sigma_1(w_1x))$, and $w_3\sigma_2(w_2\sigma_1(w_1x))$ from the forward pass.

(g)
The output dimension after the first layer is $20 \times 24 \times 24$

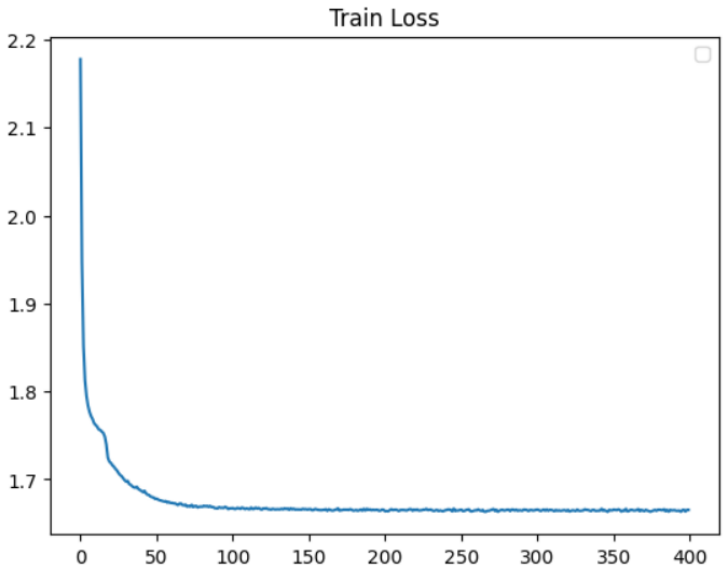
After applying max-pooling, the output dimension is $20 \times 12 \times 12$

(h)
since the final output has dimension $50 \times 4 \times 4$, the dimension right before the second max-pooling is $50 \times 8 \times 8$, which is also the desired output after the second convolution operation.

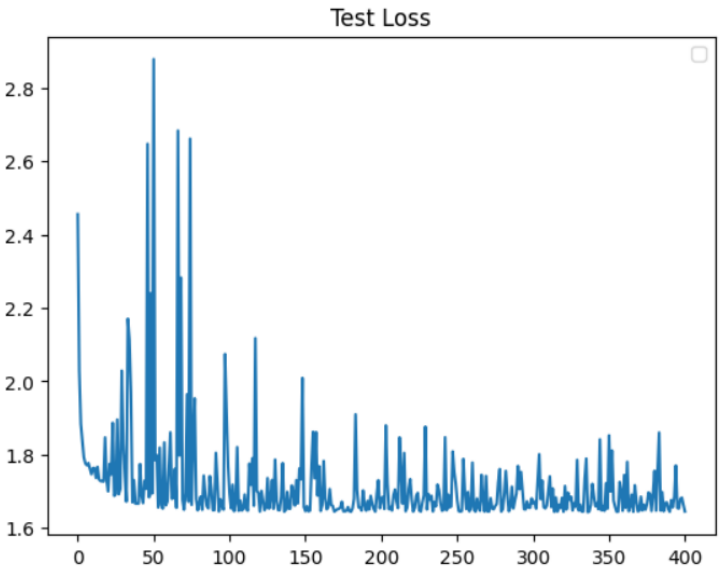
The filter dimension should be $20 \times 5 \times 5$, the stride is 1, and we need 50 of such filters. That means, in_channel=20, out_channel=50, and size=5×5

3. ResNet

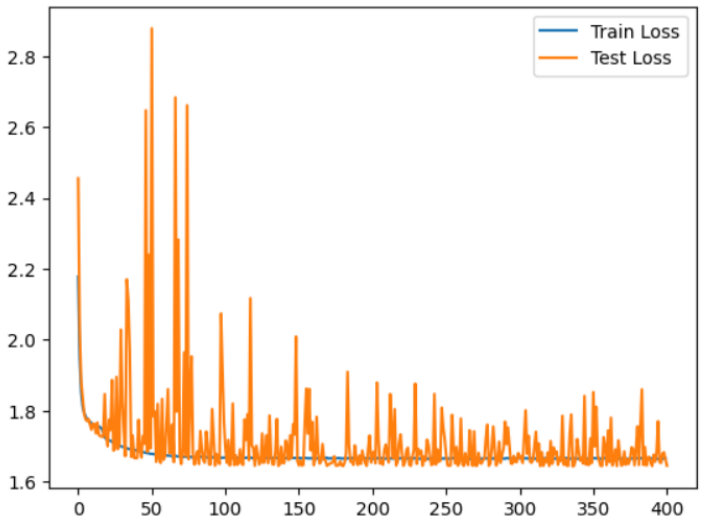
(c)
C = 1:
Train Loss:



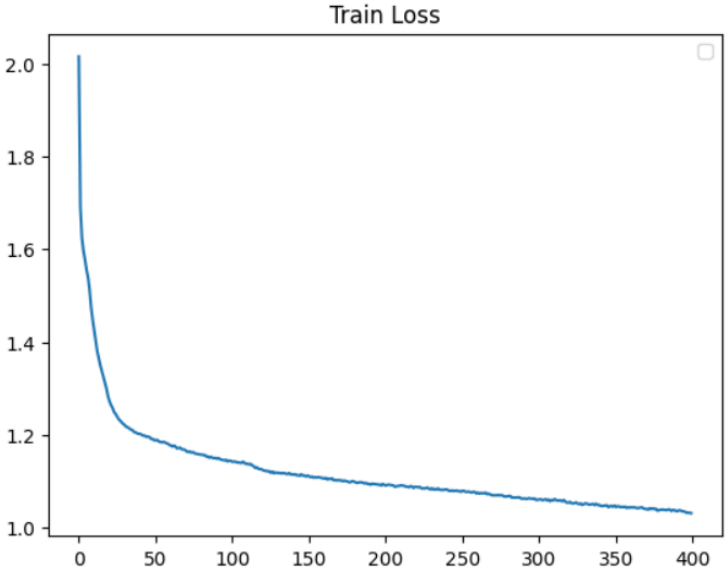
Test Loss:



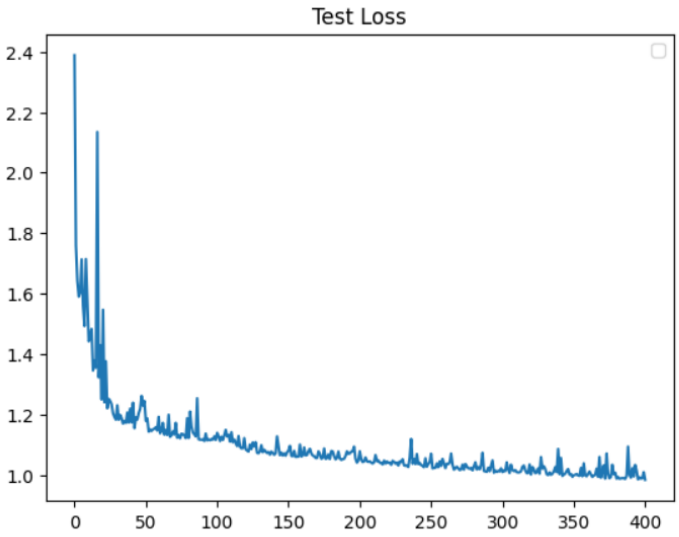
Train and Test Loss together:



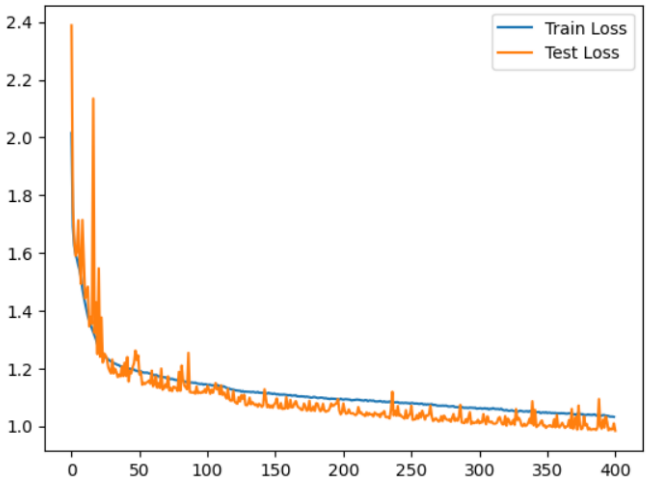
C = 2:
Train Loss:



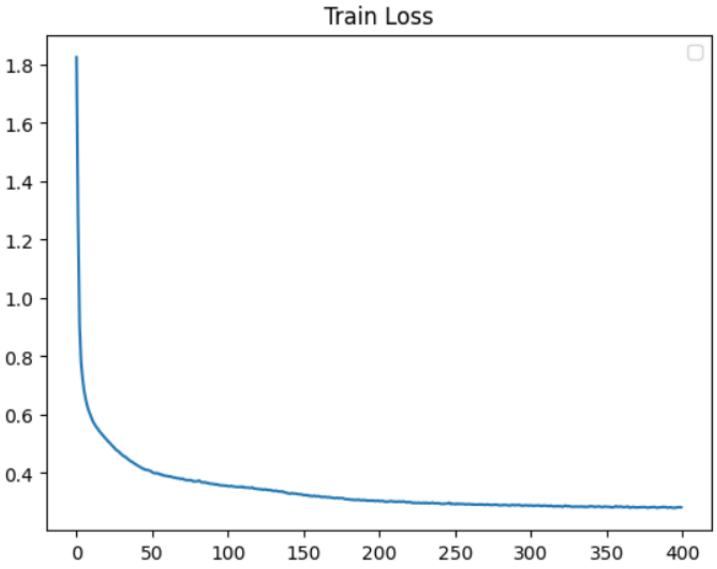
Test Loss:



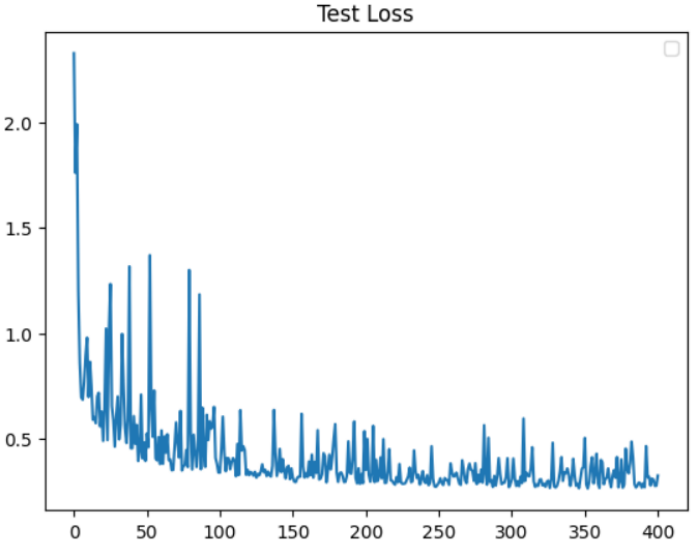
Train and Test Loss together:



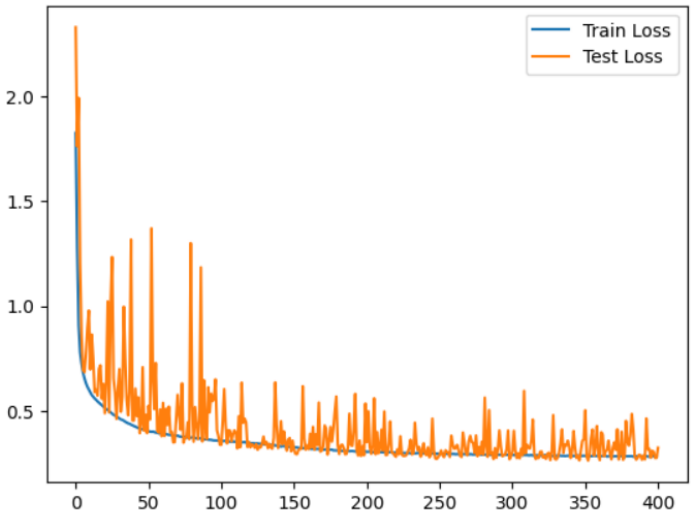
C = 4:
Train Loss:



Test Loss:

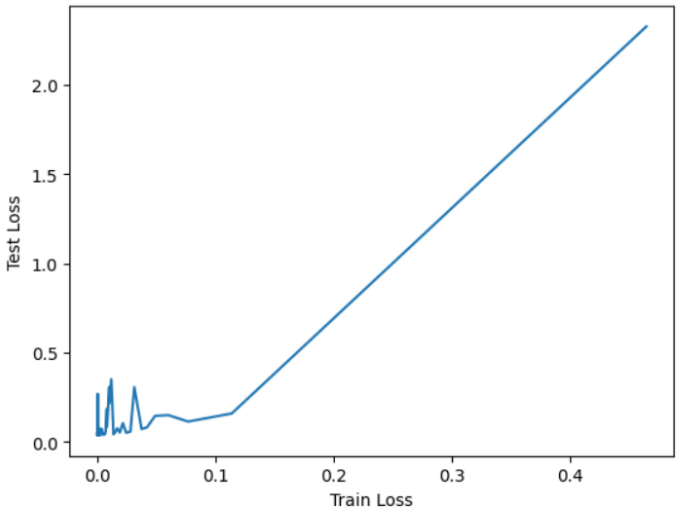


Train and Test Loss together:



As C increases, the train and test errors decrease faster, and the final train and test errors get smaller. Also, the gap between test error and train error gets smaller as C increases.

(d)
Train Error V.S. Test Error:



First, both the final train error and test error are much smaller than those in (c).
Also, both errors decreases much faster than than those in (c).
The fluctuation of the test error is much smaller and the gap between the train error and test error become much smaller.