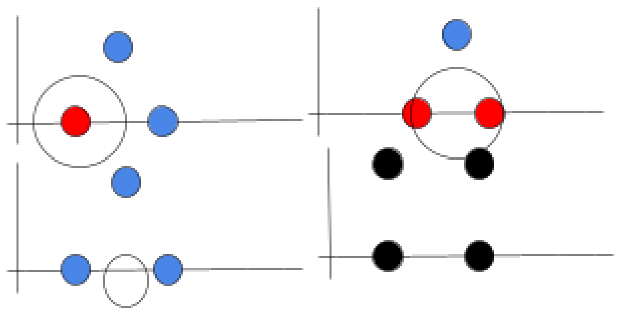
PS3 CS 146

1.

A. VC(H)=3.



Similar to a line- can't shatter 4 points.

B.

i. Let $H_1 \subseteq H_2$. Consider the following cases:

1. $H_1 = H_2$. This means that $VC(H_1)=VC(H_2)$
2. $H_1 \subset H_2$. This means that $VC(H_1) < VC(H_2)$ as H_1 cannot shatter the same N as H_2

Therefore, $VC(H_1) \leq VC(H_2)$

ii. Suppose H_2 is just a rectangle that covers the whole domain and range, and so all points on the plane are 1, and H_3 is the same but all points are 0.

Then, $VC(H_2)=VC(H_3)=0$ as it can't even shatter one point.

However, $VC(H_1) \cup VC(H_2)$ would be able to shatter one point since it can label it 0 or 1.

Therefore, $VC(H_1)=1> VC(H_2)+VC(H_3)$, which disproves the statement.

2.

a. Yes, this is a kernel. Consider k(x,z)= $\phi(x)^T\phi(z)$. Let $\phi(x)$ be the vector of 0/1 representing words that appear x (1 means it appears, 0 means it doesn't). Let $\phi(z)$ be the vector of 0/1 representing words that appear z. Then $\phi(x)^T\phi(z)$ would be the intersection and give the number of unique words in both.

b. Using the rules:

1. Scaling: $x * z\,(\frac{1}{||x||})(\frac{1}{||z||}) = \frac{x}{||x||} * \frac{z}{||z||}$
2. Sum: $1 + \frac{x}{||x||} * \frac{z}{||z||}$
3. Product: $(1 + \frac{x}{||x||} * \frac{z}{||z||})(1 + \frac{x}{||x||} * \frac{z}{||z||}) = (1 + \frac{x}{||x||} * \frac{z}{||z||})^2$

   $(1 + \frac{x}{||x||} * \frac{z}{||z||})^2(1 + \frac{x}{||x||} * \frac{z}{||z||}) = (1 + \frac{x}{||x||} * \frac{z}{||z||})^3$

Therefore it is also a kernel.

c. $(1 + \beta x * z)^3 = B^3 x^3 z^3 + 3 B^2 x^2 z^2 + 3 B x z + 1$

$\phi_\beta(x) = [\sqrt{B^3}x^3, \sqrt{3B^2}x^2, \sqrt{3B}x, 1]$

It is similar to the kernel $(1 + x * z)^3$ as to when $\beta = 1$. The difference is that it changes the weight of higher order terms depending on the value of β.

3.

a. Constraint: $y_n \theta^T x_n \geq 1$

Suppose y= -1 => $-\theta^* x \geq 1$,

To minimize, $\theta^* x = -1$.

=> $\theta^* = -\frac{x}{||x||^2}$

b. $y_n \theta^T x_n = 1$

$y_1 \theta^T x_1 = 1(\theta)(1, 1) = 1$

$y_2 \theta^T x_2 = -1(\theta)(1, 0) = 1$

θ*=(-1,2)^T

$\gamma = \sqrt{\frac{1}{5}}$

c. $y_n \theta^T x_n + b = 1$

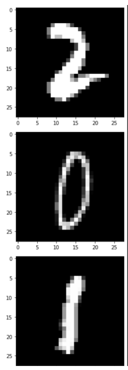
$y_1 \theta^T x_1 + b = 1(\theta)(1, 1) = 1$

$y_2 \theta^T x_2 + b = -1(\theta)(1, 0) = 1$

θ*=(-1-b,2)^T,b=-1, $\gamma = 1/2$

4.

a.



```
plot_img(X_train[10])
plot_img(X_train[20])
plot_img(X_train[35])
```

b.

```
X_train = torch.from_numpy(X_train)
y_train = torch.from_numpy(y_train)

X_valid = torch.from_numpy(X_valid)
y_valid = torch.from_numpy(y_valid)

X_test = torch.from_numpy(X_test)
y_test = torch.from_numpy(y_test)
```

c.

```
train_loader = DataLoader(TensorDataset(X_train, y_train), batch_size=10)
valid_loader = DataLoader(TensorDataset(X_valid, y_valid), batch_size=10)
test_loader = DataLoader(TensorDataset(X_test, y_test), batch_size=10)
```

d.

```
######################################################################
# OneLayerNetwork
######################################################################

class OneLayerNetwork(torch.nn.Module):
    def __init__(self):
        super(OneLayerNetwork, self).__init__()

        ### ========== TODO : START ========== ###
        ### part d: implement OneLayerNetwork with torch.nn.Linear

        self.oneLayerNetwork = torch.nn.Linear(784,3)

        ### ========== TODO : END ========== ###

    def forward(self, x):
        # x.shape = (n_batch, n_features)

        ### ========== TODO : START ========== ###
        ### part d: implement the foward function

        outputs = self.oneLayerNetwork(x)

        ### ========== TODO : END ========== ###
        return outputs
```

e.

```
model_one = OneLayerNetwork()
criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model_one.parameters(), lr=0.0005)
```

f.

```
y_pred = model.forward(batch_X)
model.zero_grad()
loss = criterion(y_pred, batch_y)
loss.backwards()
optimizer.step()
```

```
Start training OneLayerNetwork...
| epoch  1 | train loss 1.075398 | train acc 0.453333 | valid loss 1.084938 | valid acc 0.453333 |
| epoch  2 | train loss 1.021364 | train acc 0.566667 | valid loss 1.031102 | valid acc 0.553333 |
| epoch  3 | train loss 0.972648 | train acc 0.630000 | valid loss 0.982742 | valid acc 0.593333 |
| epoch  4 | train loss 0.928398 | train acc 0.710000 | valid loss 0.938953 | valid acc 0.640000 |
| epoch  5 | train loss 0.887963 | train acc 0.783333 | valid loss 0.899045 | valid acc 0.700000 |
| epoch  6 | train loss 0.850839 | train acc 0.826667 | valid loss 0.862485 | valid acc 0.753333 |
| epoch  7 | train loss 0.816627 | train acc 0.850000 | valid loss 0.828852 | valid acc 0.793333 |
| epoch  8 | train loss 0.785000 | train acc 0.886667 | valid loss 0.797807 | valid acc 0.846667 |
| epoch  9 | train loss 0.755688 | train acc 0.900000 | valid loss 0.769067 | valid acc 0.866667 |
| epoch 10 | train loss 0.728461 | train acc 0.903333 | valid loss 0.742397 | valid acc 0.873333 |
| epoch 11 | train loss 0.703122 | train acc 0.913333 | valid loss 0.717596 | valid acc 0.880000 |
| epoch 12 | train loss 0.679499 | train acc 0.920000 | valid loss 0.694488 | valid acc 0.886667 |
| epoch 13 | train loss 0.657439 | train acc 0.933333 | valid loss 0.672921 | valid acc 0.886667 |
| epoch 14 | train loss 0.636807 | train acc 0.943333 | valid loss 0.652760 | valid acc 0.886667 |
| epoch 15 | train loss 0.617482 | train acc 0.943333 | valid loss 0.633883 | valid acc 0.886667 |
| epoch 16 | train loss 0.599356 | train acc 0.943333 | valid loss 0.616184 | valid acc 0.886667 |
| epoch 17 | train loss 0.582330 | train acc 0.943333 | valid loss 0.599565 | valid acc 0.893333 |
| epoch 18 | train loss 0.566316 | train acc 0.943333 | valid loss 0.583938 | valid acc 0.900000 |
| epoch 19 | train loss 0.551234 | train acc 0.943333 | valid loss 0.569225 | valid acc 0.906667 |
| epoch 20 | train loss 0.537010 | train acc 0.943333 | valid loss 0.555355 | valid acc 0.906667 |
| epoch 21 | train loss 0.523580 | train acc 0.943333 | valid loss 0.542262 | valid acc 0.906667 |
| epoch 22 | train loss 0.510882 | train acc 0.943333 | valid loss 0.529888 | valid acc 0.906667 |
| epoch 23 | train loss 0.498862 | train acc 0.950000 | valid loss 0.518179 | valid acc 0.906667 |
| epoch 24 | train loss 0.487470 | train acc 0.950000 | valid loss 0.507086 | valid acc 0.906667 |
| epoch 25 | train loss 0.476660 | train acc 0.950000 | valid loss 0.496564 | valid acc 0.906667 |
| epoch 26 | train loss 0.466391 | train acc 0.953333 | valid loss 0.486573 | valid acc 0.926667 |
| epoch 27 | train loss 0.456625 | train acc 0.953333 | valid loss 0.477076 | valid acc 0.926667 |
| epoch 28 | train loss 0.447328 | train acc 0.953333 | valid loss 0.468038 | valid acc 0.926667 |
| epoch 29 | train loss 0.438467 | train acc 0.956667 | valid loss 0.459429 | valid acc 0.933333 |
| epoch 30 | train loss 0.430013 | train acc 0.956667 | valid loss 0.451220 | valid acc 0.940000 |
```

g.

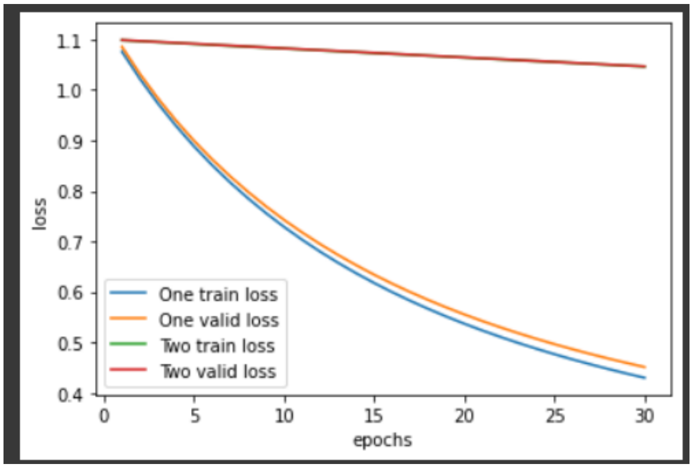
```python
class TwoLayerNetwork(torch.nn.Module):
    def __init__(self):
        super(TwoLayerNetwork, self).__init__()
        ### ========== TODO : START ========== ###
        ### part g: implement TwoLayerNetwork with tor
        self.tLN1 = torch.nn.Linear(784, 400)
        self.tLN2 = torch.nn.Linear(400, 3)

        ### ========== TODO : END ========== ###

    def forward(self, x):
        # x.shape = (n_batch, n_features)

        ### ========== TODO : START ========== ###
        ### part g: implement the foward function

        sigmoid= torch.nn.Sigmoid()
        outputs= self.tLN2(sigmoid(self.trN1))


        ### ========== TODO : END ========== ###
        return outputs
```

h.

```python
model_two = TwoLayerNetwork()
criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model_two.parameters(), lr=0.0005)
### ========== TODO : END ========== ###
```

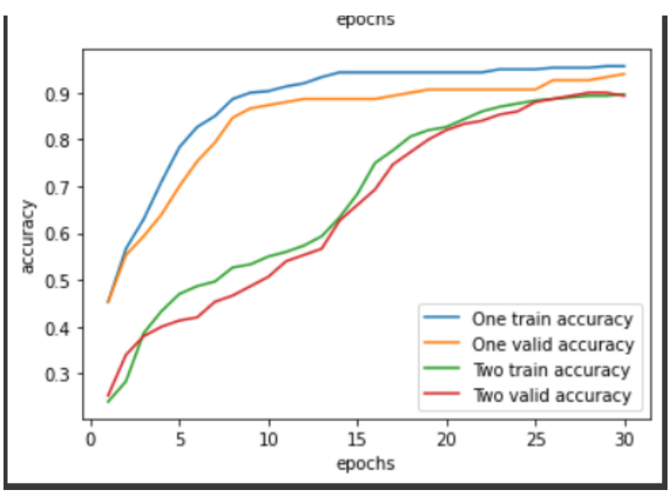
```
Start training TwoLayerNetwork...
| epoch  1 | train loss 1.098020 | train acc 0.240000 | valid loss 1.098498 | valid acc 0.253333 |
| epoch  2 | train loss 1.096157 | train acc 0.283333 | valid loss 1.096622 | valid acc 0.340000 |
| epoch  3 | train loss 1.094329 | train acc 0.386667 | valid loss 1.094783 | valid acc 0.380000 |
| epoch  4 | train loss 1.092512 | train acc 0.433333 | valid loss 1.092956 | valid acc 0.400000 |
| epoch  5 | train loss 1.090700 | train acc 0.470000 | valid loss 1.091135 | valid acc 0.413333 |
| epoch  6 | train loss 1.088891 | train acc 0.486667 | valid loss 1.089318 | valid acc 0.420000 |
| epoch  7 | train loss 1.087085 | train acc 0.496667 | valid loss 1.087503 | valid acc 0.453333 |
| epoch  8 | train loss 1.085281 | train acc 0.526667 | valid loss 1.085691 | valid acc 0.466667 |
| epoch  9 | train loss 1.083480 | train acc 0.533333 | valid loss 1.083882 | valid acc 0.486667 |
| epoch 10 | train loss 1.081682 | train acc 0.550000 | valid loss 1.082076 | valid acc 0.506667 |
| epoch 11 | train loss 1.079886 | train acc 0.560000 | valid loss 1.080273 | valid acc 0.540000 |
| epoch 12 | train loss 1.078093 | train acc 0.573333 | valid loss 1.078472 | valid acc 0.553333 |
| epoch 13 | train loss 1.076302 | train acc 0.593333 | valid loss 1.076674 | valid acc 0.566667 |
| epoch 14 | train loss 1.074514 | train acc 0.633333 | valid loss 1.074878 | valid acc 0.626667 |
| epoch 15 | train loss 1.072727 | train acc 0.683333 | valid loss 1.073084 | valid acc 0.660000 |
| epoch 16 | train loss 1.070942 | train acc 0.750000 | valid loss 1.071292 | valid acc 0.693333 |
| epoch 17 | train loss 1.069159 | train acc 0.776667 | valid loss 1.069502 | valid acc 0.746667 |
| epoch 18 | train loss 1.067377 | train acc 0.806667 | valid loss 1.067713 | valid acc 0.773333 |
| epoch 19 | train loss 1.065597 | train acc 0.820000 | valid loss 1.065926 | valid acc 0.800000 |
| epoch 20 | train loss 1.063817 | train acc 0.826667 | valid loss 1.064139 | valid acc 0.820000 |
| epoch 21 | train loss 1.062038 | train acc 0.843333 | valid loss 1.062354 | valid acc 0.833333 |
| epoch 22 | train loss 1.060260 | train acc 0.860000 | valid loss 1.060569 | valid acc 0.840000 |
| epoch 23 | train loss 1.058483 | train acc 0.870000 | valid loss 1.058785 | valid acc 0.853333 |
| epoch 24 | train loss 1.056706 | train acc 0.876667 | valid loss 1.057001 | valid acc 0.860000 |
| epoch 25 | train loss 1.054928 | train acc 0.883333 | valid loss 1.055217 | valid acc 0.880000 |
| epoch 26 | train loss 1.053151 | train acc 0.886667 | valid loss 1.053433 | valid acc 0.886667 |
| epoch 27 | train loss 1.051374 | train acc 0.890000 | valid loss 1.051650 | valid acc 0.893333 |
| epoch 28 | train loss 1.049596 | train acc 0.893333 | valid loss 1.049865 | valid acc 0.900000 |
| epoch 29 | train loss 1.047818 | train acc 0.893333 | valid loss 1.048081 | valid acc 0.900000 |
| epoch 30 | train loss 1.046038 | train acc 0.896667 | valid loss 1.046295 | valid acc 0.893333 |
```

i.



We see that the loss from one layer decreases at a faster rate than the two layer loss. Moreover, we see that the same layer graphs are close to each other, which implies there isn't overfitting or underfitting happening.

j.



We can see that the accuracies of one layer increased faster than the two layer accuracies. However, as the epochs increased, the accuracies of all graphs converged to about 0.9.

k.

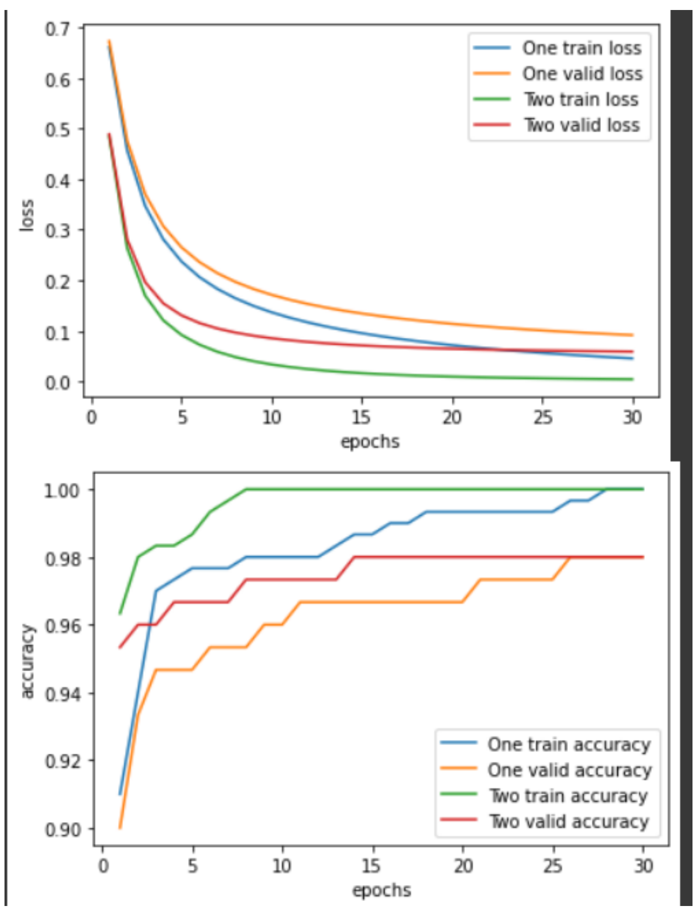
```
1LN:  tensor(0.9600)
2LN:  tensor(0.9000)
```

The accuracy of the one layer is greater than the accuracy of the two layers. This makes sense as the 2 layer has more parameters, so more epochs would be needed. Thus to increase the accuracy, increase the epochs.

l.

```
Start training OneLayerNetwork...
| epoch  1 | train loss 0.661372 | train acc 0.910000 | valid loss 0.672927 | valid acc 0.900000 |
| epoch  2 | train loss 0.456800 | train acc 0.940000 | valid loss 0.476012 | valid acc 0.933333 |
| epoch  3 | train loss 0.346736 | train acc 0.970000 | valid loss 0.370065 | valid acc 0.946667 |
| epoch  4 | train loss 0.280993 | train acc 0.973333 | valid loss 0.307033 | valid acc 0.946667 |
| epoch  5 | train loss 0.237471 | train acc 0.976667 | valid loss 0.265548 | valid acc 0.946667 |
| epoch  6 | train loss 0.206357 | train acc 0.976667 | valid loss 0.236089 | valid acc 0.953333 |
| epoch  7 | train loss 0.182834 | train acc 0.976667 | valid loss 0.213983 | valid acc 0.953333 |
| epoch  8 | train loss 0.164295 | train acc 0.980000 | valid loss 0.196703 | valid acc 0.953333 |
| epoch  9 | train loss 0.149216 | train acc 0.980000 | valid loss 0.182770 | valid acc 0.960000 |
| epoch 10 | train loss 0.136649 | train acc 0.980000 | valid loss 0.171261 | valid acc 0.960000 |
| epoch 11 | train loss 0.125971 | train acc 0.980000 | valid loss 0.161569 | valid acc 0.966667 |
| epoch 12 | train loss 0.116756 | train acc 0.980000 | valid loss 0.153280 | valid acc 0.966667 |
| epoch 13 | train loss 0.108704 | train acc 0.983333 | valid loss 0.146099 | valid acc 0.966667 |
| epoch 14 | train loss 0.101594 | train acc 0.986667 | valid loss 0.139809 | valid acc 0.966667 |
| epoch 15 | train loss 0.095260 | train acc 0.986667 | valid loss 0.134248 | valid acc 0.966667 |
| epoch 16 | train loss 0.089578 | train acc 0.990000 | valid loss 0.129293 | valid acc 0.966667 |
| epoch 17 | train loss 0.084448 | train acc 0.990000 | valid loss 0.124846 | valid acc 0.966667 |
| epoch 18 | train loss 0.079790 | train acc 0.993333 | valid loss 0.120830 | valid acc 0.966667 |
| epoch 19 | train loss 0.075543 | train acc 0.993333 | valid loss 0.117184 | valid acc 0.966667 |
| epoch 20 | train loss 0.071652 | train acc 0.993333 | valid loss 0.113856 | valid acc 0.966667 |
| epoch 21 | train loss 0.068076 | train acc 0.993333 | valid loss 0.110807 | valid acc 0.973333 |
| epoch 22 | train loss 0.064778 | train acc 0.993333 | valid loss 0.108000 | valid acc 0.973333 |
| epoch 23 | train loss 0.061726 | train acc 0.993333 | valid loss 0.105407 | valid acc 0.973333 |
| epoch 24 | train loss 0.058896 | train acc 0.993333 | valid loss 0.103004 | valid acc 0.973333 |
| epoch 25 | train loss 0.056265 | train acc 0.993333 | valid loss 0.100770 | valid acc 0.973333 |
| epoch 26 | train loss 0.053812 | train acc 0.996667 | valid loss 0.098686 | valid acc 0.980000 |
| epoch 27 | train loss 0.051522 | train acc 0.996667 | valid loss 0.096739 | valid acc 0.980000 |
| epoch 28 | train loss 0.049378 | train acc 1.000000 | valid loss 0.094914 | valid acc 0.980000 |
| epoch 29 | train loss 0.047369 | train acc 1.000000 | valid loss 0.093200 | valid acc 0.980000 |
| epoch 30 | train loss 0.045482 | train acc 1.000000 | valid loss 0.091587 | valid acc 0.980000 |
```

```
Start training TwoLayerNetwork...
| epoch  1 | train loss 0.484446 | train acc 0.963333 | valid loss 0.488961 | valid acc 0.953333 |
| epoch  2 | train loss 0.263492 | train acc 0.980000 | valid loss 0.280668 | valid acc 0.960000 |
| epoch  3 | train loss 0.169251 | train acc 0.983333 | valid loss 0.196060 | valid acc 0.960000 |
| epoch  4 | train loss 0.120736 | train acc 0.983333 | valid loss 0.154269 | valid acc 0.966667 |
| epoch  5 | train loss 0.092051 | train acc 0.986667 | valid loss 0.130751 | valid acc 0.966667 |
| epoch  6 | train loss 0.072639 | train acc 0.993333 | valid loss 0.115402 | valid acc 0.966667 |
| epoch  7 | train loss 0.058536 | train acc 0.996667 | valid loss 0.104522 | valid acc 0.966667 |
| epoch  8 | train loss 0.047935 | train acc 1.000000 | valid loss 0.096434 | valid acc 0.973333 |
| epoch  9 | train loss 0.039789 | train acc 1.000000 | valid loss 0.090208 | valid acc 0.973333 |
| epoch 10 | train loss 0.033423 | train acc 1.000000 | valid loss 0.085282 | valid acc 0.973333 |
| epoch 11 | train loss 0.028379 | train acc 1.000000 | valid loss 0.081300 | valid acc 0.973333 |
| epoch 12 | train loss 0.024333 | train acc 1.000000 | valid loss 0.078024 | valid acc 0.973333 |
| epoch 13 | train loss 0.021052 | train acc 1.000000 | valid loss 0.075293 | valid acc 0.973333 |
| epoch 14 | train loss 0.018364 | train acc 1.000000 | valid loss 0.072989 | valid acc 0.980000 |
| epoch 15 | train loss 0.016140 | train acc 1.000000 | valid loss 0.071028 | valid acc 0.980000 |
| epoch 16 | train loss 0.014283 | train acc 1.000000 | valid loss 0.069343 | valid acc 0.980000 |
| epoch 17 | train loss 0.012720 | train acc 1.000000 | valid loss 0.067884 | valid acc 0.980000 |
| epoch 18 | train loss 0.011393 | train acc 1.000000 | valid loss 0.066614 | valid acc 0.980000 |
| epoch 19 | train loss 0.010258 | train acc 1.000000 | valid loss 0.065500 | valid acc 0.980000 |
| epoch 20 | train loss 0.009281 | train acc 1.000000 | valid loss 0.064517 | valid acc 0.980000 |
| epoch 21 | train loss 0.008434 | train acc 1.000000 | valid loss 0.063646 | valid acc 0.980000 |
| epoch 22 | train loss 0.007696 | train acc 1.000000 | valid loss 0.062871 | valid acc 0.980000 |
| epoch 23 | train loss 0.007048 | train acc 1.000000 | valid loss 0.062177 | valid acc 0.980000 |
| epoch 24 | train loss 0.006478 | train acc 1.000000 | valid loss 0.061554 | valid acc 0.980000 |
| epoch 25 | train loss 0.005973 | train acc 1.000000 | valid loss 0.060992 | valid acc 0.980000 |
| epoch 26 | train loss 0.005523 | train acc 1.000000 | valid loss 0.060484 | valid acc 0.980000 |
| epoch 27 | train loss 0.005121 | train acc 1.000000 | valid loss 0.060023 | valid acc 0.980000 |
| epoch 28 | train loss 0.004761 | train acc 1.000000 | valid loss 0.059603 | valid acc 0.980000 |
| epoch 29 | train loss 0.004437 | train acc 1.000000 | valid loss 0.059220 | valid acc 0.980000 |
| epoch 30 | train loss 0.004144 | train acc 1.000000 | valid loss 0.058869 | valid acc 0.980000 |
```



```
1LN:  tensor(0.9733)
2LN:  tensor(0.9667)
```

We can see that the Adam optimizer decreases the loss and increases accuracy greatly compared to SGD for both layers. The new accuracies for the 1LN and 2LN are 0.9733 and 0.9667 respectively, which is a notable increase in the 2LN. The loss graph shows the loss for all data converging to around 0.1 much faster than with SGD, showing that high epochs aren't necessary with Adam. Likewise, the accuracy graph shows fast growth to high accuracy, especially for the 2 layer.