1. Show your model architecture and testing accuracy.

2-layer NN, 500 hidden units, loss function use CrossEncropy.
Weight initial all random small value use normal distribution.
Bias initial all zero.
1 layer: activation function use ReLU
2 layer: activation function use Softmax

```
epoch: 0, train_loss: 0.251503, validation_loss: 0.260554, val_acc: 92.6889
epoch: 1, train_loss: 0.180763, validation_loss: 0.196883, val_acc: 94.5056
epoch: 2, train_loss: 0.140018, validation_loss: 0.161661, val_acc: 95.4778
epoch: 3, train_loss: 0.113483, validation_loss: 0.139584, val_acc: 96.0556
epoch: 4, train_loss: 0.094883, validation_loss: 0.124898, val_acc: 96.4444
epoch: 5, train_loss: 0.080989, validation_loss: 0.114401, val_acc: 96.6833
epoch: 6, train_loss: 0.070271, validation_loss: 0.106960, val_acc: 96.8889
epoch: 7, train_loss: 0.061667, validation_loss: 0.101497, val_acc: 96.9944
epoch: 8, train_loss: 0.054579, validation_loss: 0.097283, val_acc: 97.0833
epoch: 9, train_loss: 0.048798, validation_loss: 0.094185, val_acc: 97.1944
epoch: 10, train_loss: 0.043958, validation_loss: 0.091835, val_acc: 97.2278
epoch: 11, train_loss: 0.039766, validation_loss: 0.089931, val_acc: 97.2722
epoch: 12, train_loss: 0.036047, validation_loss: 0.088304, val_acc: 97.3389
epoch: 13, train_loss: 0.032726, validation_loss: 0.086993, val_acc: 97.3500
epoch: 14, train_loss: 0.029842, validation_loss: 0.085940, val_acc: 97.3722
test acc: 97.6300%
```

Testing accuracy: 97.63%

2. How do you implement feed forward and backward propagation? A briefexplanation is fine.

Forward:
each layer output is F(W*X + B).
-- W is weight matrix.
-- B is bias matrix.
-- F is activation function.
-- X is input dataset

Backward:
From end to begin layer. Calculate derivative.

Main idea is $\frac{dL}{dw} = \frac{dL}{da} \times \frac{da}{dz} \times \frac{dz}{dw}$

Output layer $\frac{dL}{da} \times \frac{da}{dz}$ = model output – target. because cross-Encropy + Softmax .

$\frac{dz}{dw}$ = hidden layer output.

So, output layer $\frac{dL}{dw}$ = (model output – target) * hidden layer output

Hidden layer, use chain rule.

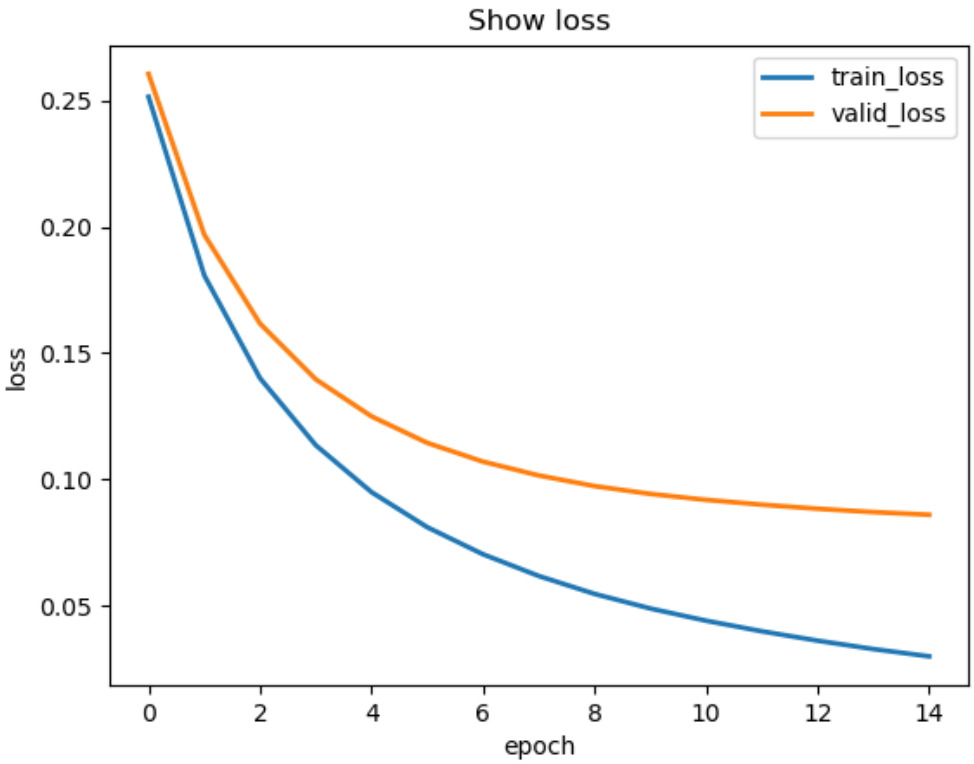
$\frac{dL}{da}$ = $\sum$ weight $*$ (output layer's $\frac{dL}{da} x \frac{da}{dz}$)

$\frac{da}{dz}$ = ReLU derivative.

$\frac{dz}{dw}$ = input dataset.

So, hidden layer $\frac{dL}{dw}$ = $\frac{dL}{da}$ * input dataset

3. Plot training loss and validation loss. (loss vs. epochs figure)



4. If we use a very deep NN with a large number of neurons, will the accuracy increase? Why or why not?

May not increase accuracy only use a deep NN. Because it is same if we use a lots of hidden units. Need other additional strategies to increase

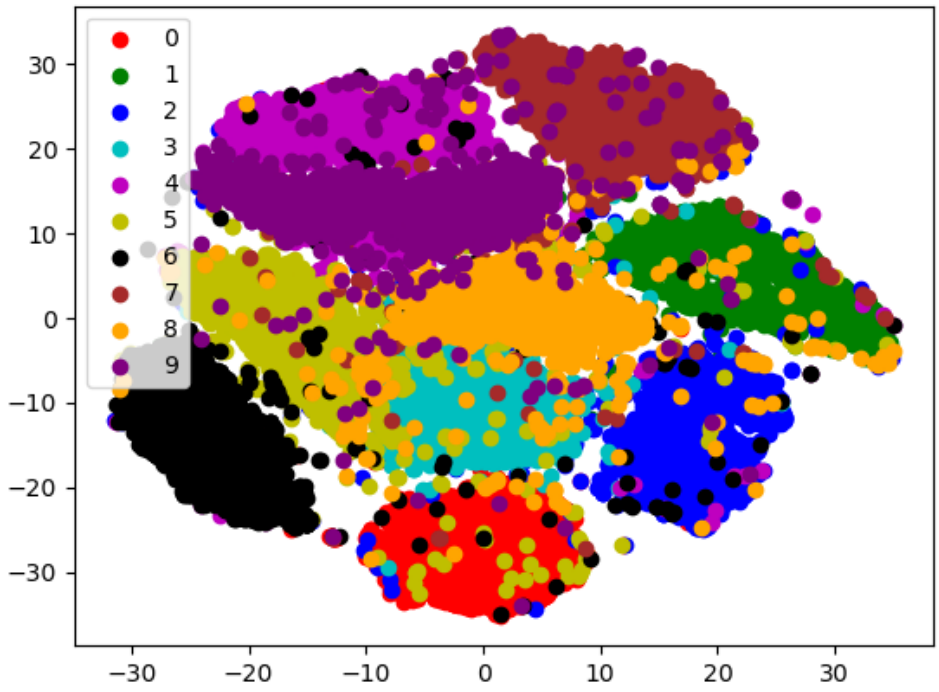
accuracy.

5. Why do we need to validate our model?

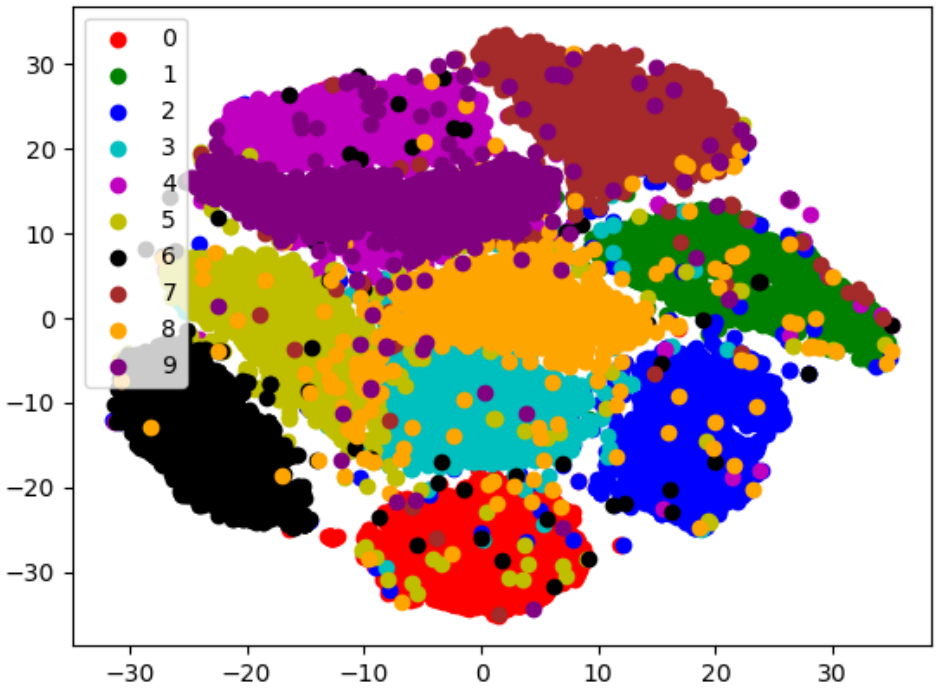
Because it is a method to make sure our model is not training bad, and use for early stopping method to avoid overfitting.

6. t-SNE results (optional, not included in 20%)

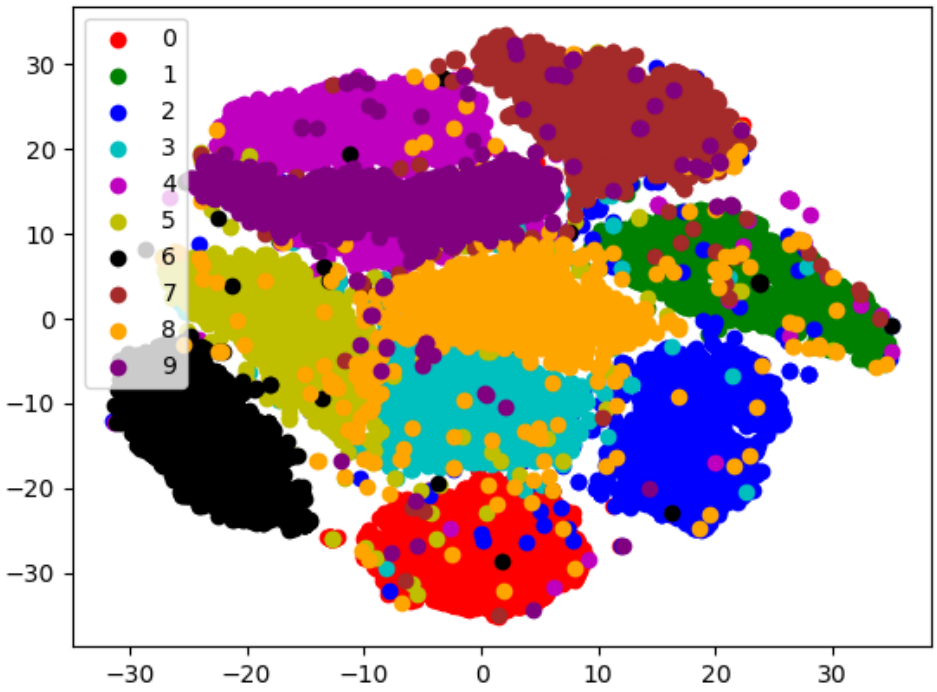
0 epoch validation data visualization



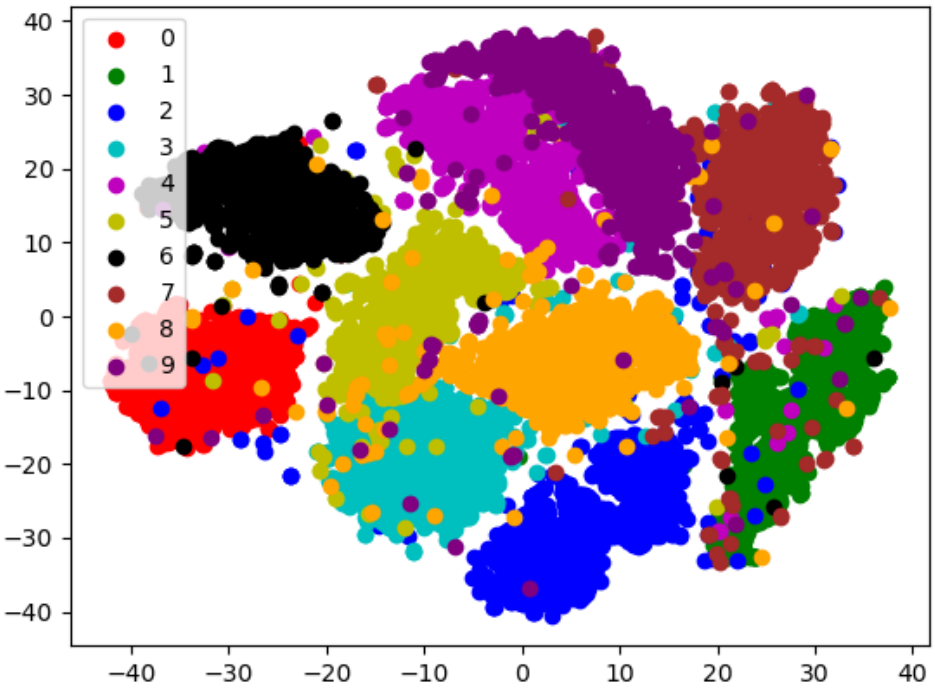
2 epoch validation data visualization



14 epoch validation data visualization



Test data


0
1
2
3
4
5
6
7
8
9
40
30
20
10
0
−10
−20
−30
−40
−40
−30
−20
−10
0
10
20
30
40