Section A - (Question 1)

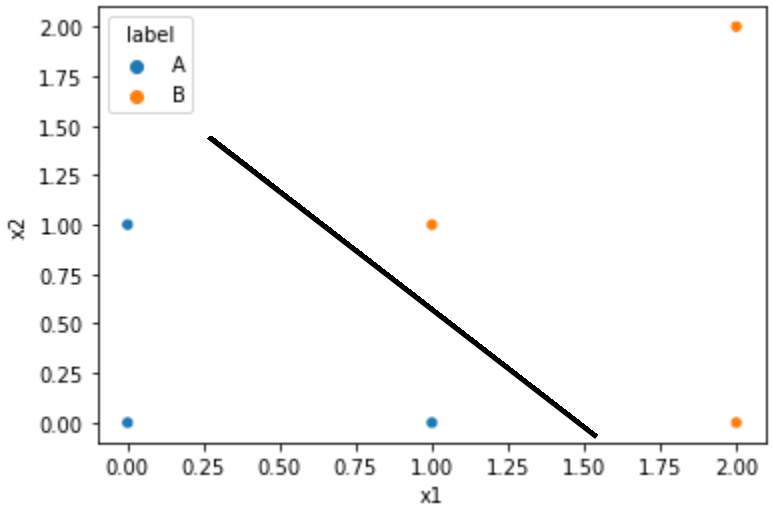
1. (10 points) John has the following dataset:

Class	x1	x2
Ä	0	0
A	1	0
A	0	1
B	1	1
B	2	2
B	2	0

He is studying machine learning and encountered some questions which he is unable to solve. Help him solve the following questions.

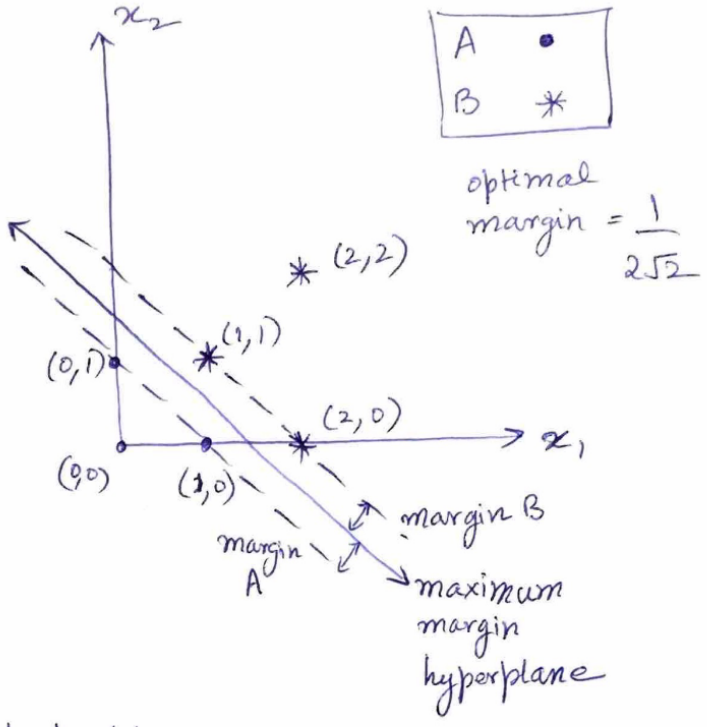
1. (2 points) Are the points linearly separable? Support your answer by plotting the points.
2. (3 points) Find out the weight vector corresponding to the maximum margin hyperplane. Also find the support vectors present.
3. (2 points) What is the effect on the optimal margin if we remove any one of the support vectors in this question?
4. (3 points) In general, for any dataset, what can we say about the effect on optimal margin, if we remove any of the support vectors?

(1) Yes, the data is linearly separable; a line can be plotted like the one I have drawn below.



1)

Class	x_1	x_2
A	0	1
A	0	0
A	1	0
B	1	1
B	2	0
B	2	2



Yes, the classes are linearly separable.

2) Slope of maximum margin hyperplane will be equal to that of margin A & B, because all of them are parallel. ∴ slope = −1

Maximum margin hyperplane is equidistant from both margins. Hence, $(\frac{1}{2}, \frac{1}{2})$ being a point on margin A and (1, 1) being a point on margin B, their mid point will lie on max margin hyperplane.

∴ $\left(\frac{\frac{1}{2}+1}{2}, \frac{\frac{1}{2}+1}{2}\right) = \left(\frac{3}{4}, \frac{3}{4}\right)$ is a point on this hyperplane.

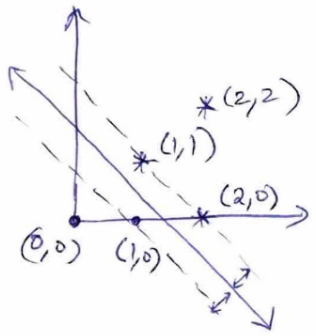
Eqn of max margin hyperplane :- $\frac{x_2 - 3/4}{x_1 - 3/4} = -1$

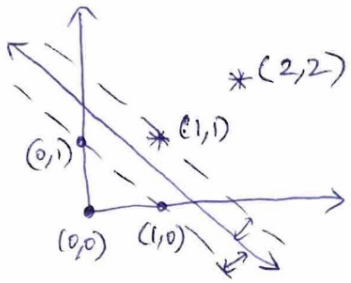
$$\Rightarrow \boxed{x_1 + x_2 = 3/2}$$

Weight vector is : (1, 1)

3) The support vectors corresponding to margin A & B are → (1,0) (0,1) & (1,1) (2,0)

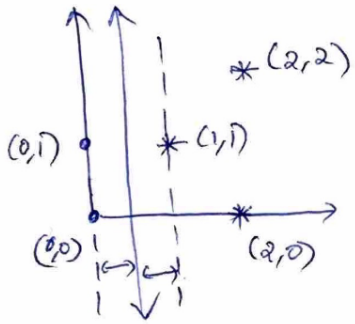
If we remove (0,1), (2,0) → then there will be no effect on the optimal margin.

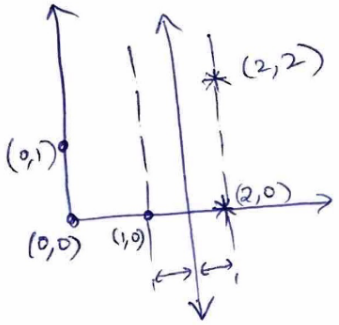




optimal margin $= \frac{1}{2\sqrt{2}}$

But if we remove (1,0) or (1,1) then the optimal margin will increase





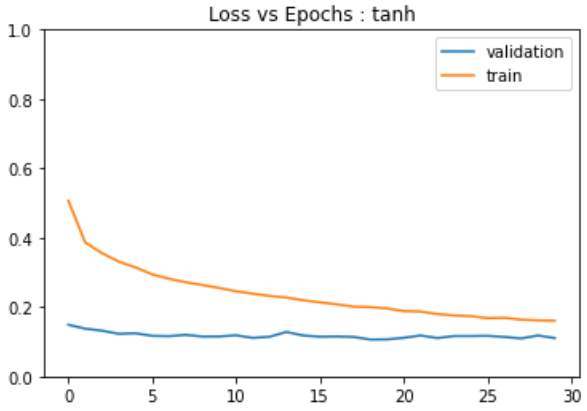
optimal margin $= \frac{1}{2}$

4) Like the above part we can have multiple results that depend completely on the dataset. Removing one of the support vectors might change the optimal margin. Our aim is to maximize the margin hyperplane.

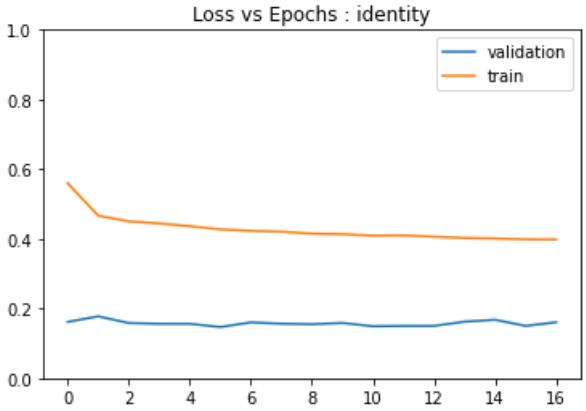
Section C

1. (3.5 points) Plot training loss v/s epochs and validation loss v/s epochs for activations sigmoid, ReLU, tanh and linear (default learning rate). Which is the best activation function? Give analysis and comparison for each.

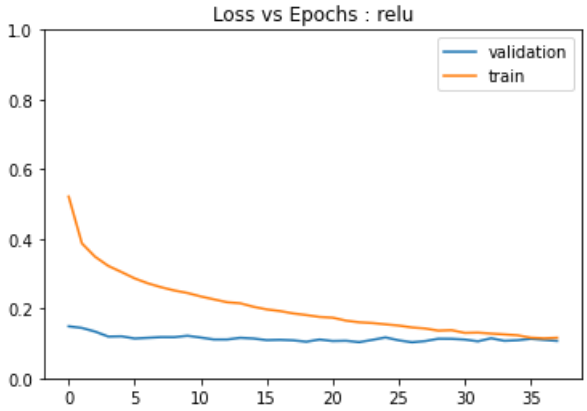
First, I preprocessed the data by dividing pixel values by 255



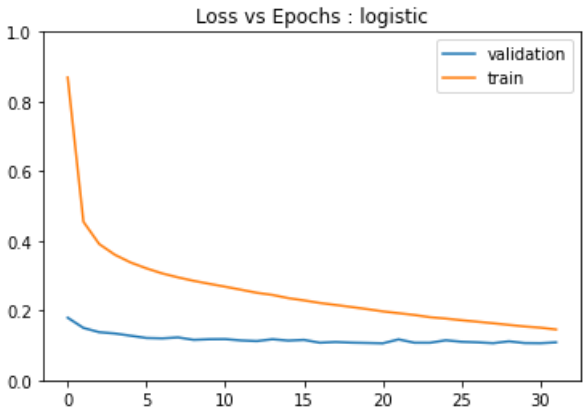
Train Accuracy: 0.9289
Test Accuracy : 0.8945



Train Accuracy: 0.8611
Test Accuracy : 0.8567



Train Accuracy: 0.9468
Test Accuracy : 0.8956



Train Accuracy: 0.9323
Test Accuracy : 0.8964

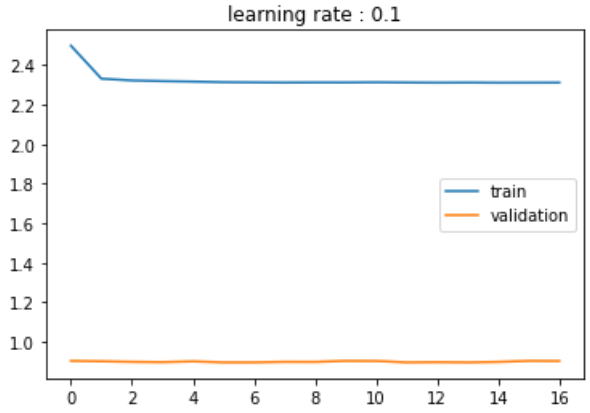
I had the following common parameters -

- **Layers-** (256, 32)
- **max_epochs-** 400
- **batch_size-** 64

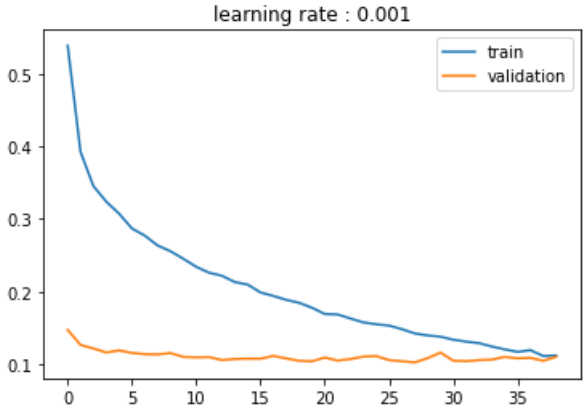
- I experimented with batch sizes 32, 64, 100, and 200 and got the best results in batch size 64.
- As you can see from the figure, there needs to be more learning done in activation function identity, which shows it's not the appropriate activation for this data.

- For the rest of the activation functions, I am getting good learning curves that aren't overfitting or underfitting.
- Relu has the best training accuracy.
- Logistic has the best testing accuracy.

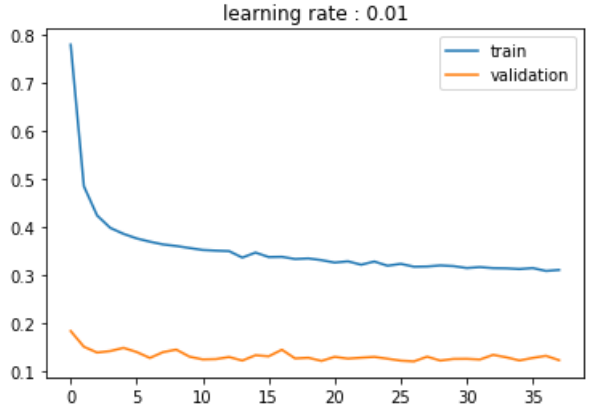
2. (3.5 points) Using the best activation function obtained above, train models using learning rates [0.1, 0.01, 0.001]. Plot training loss v/s epochs and validation loss v/s epochs. Which is the best learning rate? Give explanations of the results obtained for each learning rate.



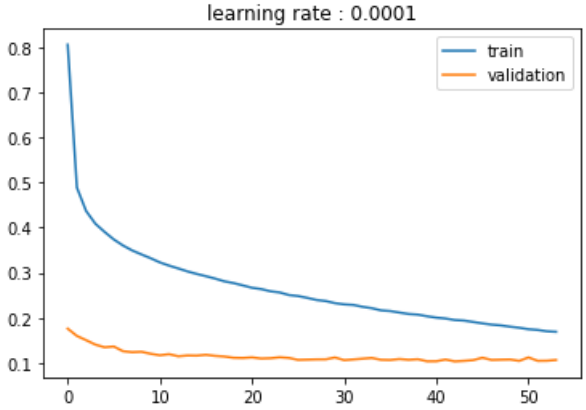
Accuracy- 0.1



Accuracy- 0.93436666



Accuracy- 0.900116666



Accuracy- 0.931083333

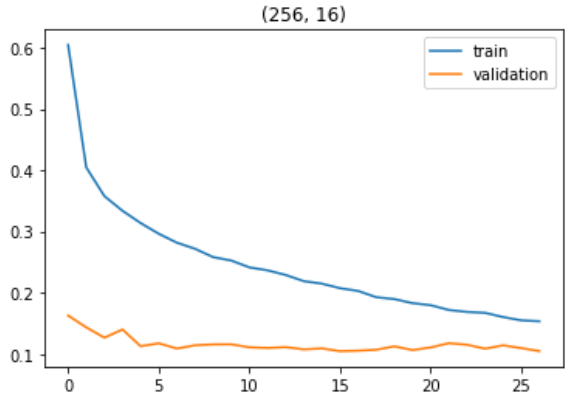
I had the following common parameters -

- Layers- (256, 32)
- max_epochs- 400
- batch_size- 64
- activation- relu

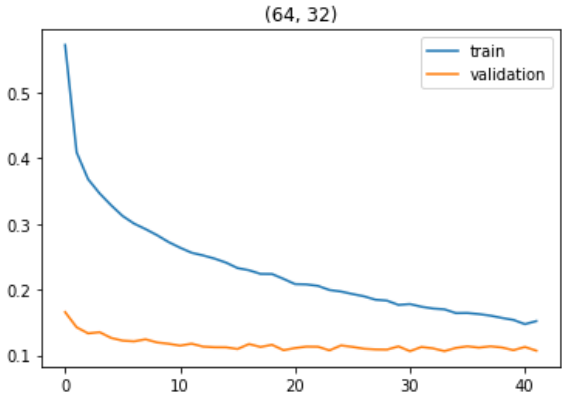
Observations -

- **0.1** - too big learning rate and either stuck at local minima or not jumping over the minima again and again.
- **0.01** - this is good but is not giving us enough precision therefore, the accuracy is lesser than 0.001 and 0.0001. Too much gap between the train and val losses
- **0.001** - the curve is smooth and gradual, and this is the optimal learning rate
- **0.0001** - the curve is even more smooth, but the rate is too small, and the accuracy is also less than 0.001, which might be because of slight overfitting.

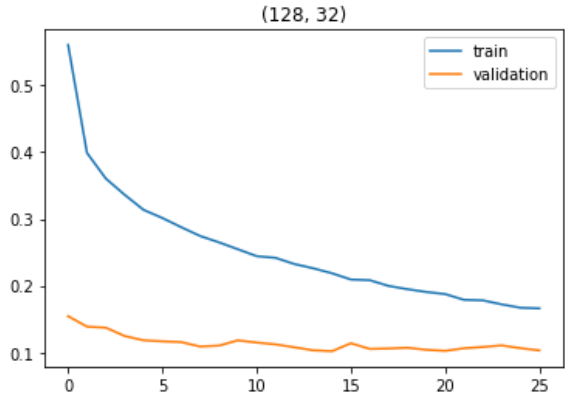
3. (3 points) Decrease the number of neurons in each layer to various values. What do you observe? Plot training loss v/s epochs. Justify your answer.



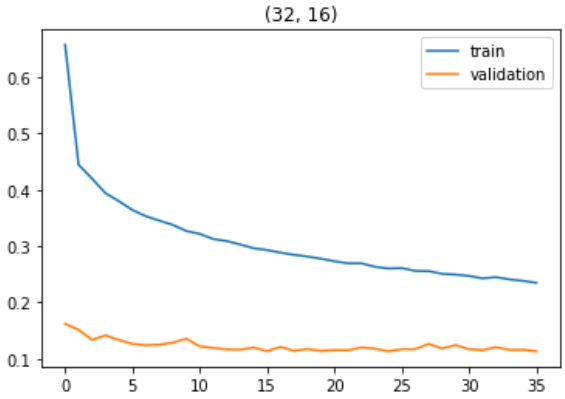
Accuracy- 0.9268833333333333



Accuracy- 0.9307



Accuracy- 0.9232333333333334



Accuracy- 0.9064833333333333

If we reduce the number of neurons, then there is a chance that underfitting might occur. We can also observe from the graphs that the gap between training and validation loss is increasing with decreasing layer sizes. The overall loss is also increasing, and the model's accuracy is decreasing.

4. (5 points) Perform grid search on appropriate parameters of MLPClassifier. Choose the best parameters. Give an analysis of why you might have got those parameters.

```
parameter_space = {
    'hidden_layer_sizes': [(256,64) , (256,32) , (256 , 128)],
    'solver': ['sgd', 'adam' , 'lbfgs'],
    'learning_rate_init': [0.001 , 0.0001 , 0.01],
    'alpha': [0.0001 , 0.001 , 0.01],
    'learning_rate': ['constant','adaptive'],
}
```

```
from sklearn.model_selection import GridSearchCV
mlp = MLPClassifier(random_state=33 , max_iter = 5000 , batch_size=64 , activation='relu')
clf = GridSearchCV(mlp, parameter_space, n_jobs=-1, cv=2)
clf.fit(X_np, y_np)
```

The best parameters were -

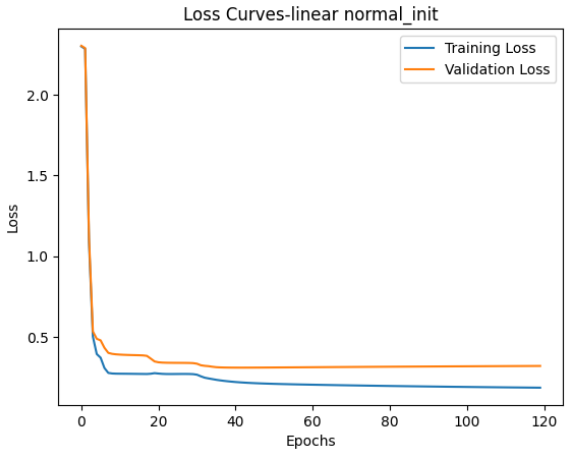
- **alpha**: 0.0001
- **Hidden_layer_sizes:** (256, 64)
- **Learning_rate:** constant
- **learning_rate_init:** 0.001
- **solver:** adam
- **Batch_size:** 64
- **Activation:** relu
- **Max_iter:** 5000.

Lbfgs is generally a very fast and good solver however adam is better for vast datasets, and could also be proved because my lbfgs was not converging even after increasing max_iters however it worked fine when i experimented with only a fraction of the data. The default alpha is 0.001, which is too much as we are already getting a pretty smooth curve, and this data doesn't need all that much regularisation. Learning rate and batch size were something that I was able to predict due to my experimentations and previous plotting results.

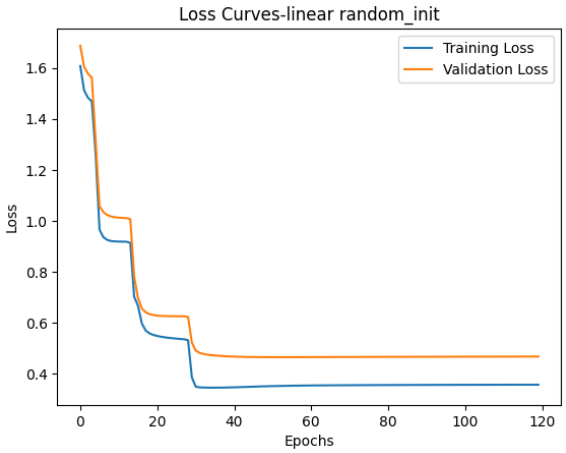
Section B

Choose the remaining parameters appropriately. Plot training loss v/s epochs and validation loss v/s epochs for each activation function. Also, save all the trained models, as you might be asked to run them during the demo (TAs won't wait for the model to train). Note that it is possible that your training is stuck at a local minimum. Try fiddling around with the batch size.

Linear

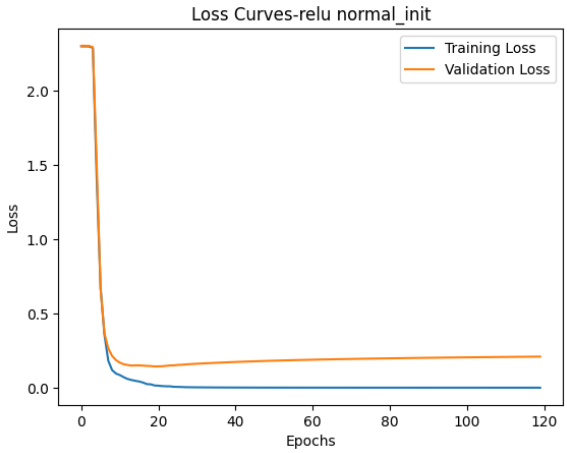



```
Training Accuracy: 0.9297083333333334
Validation Accuracy: 0.9099166666666667
Test Accuracy: 0.919
```

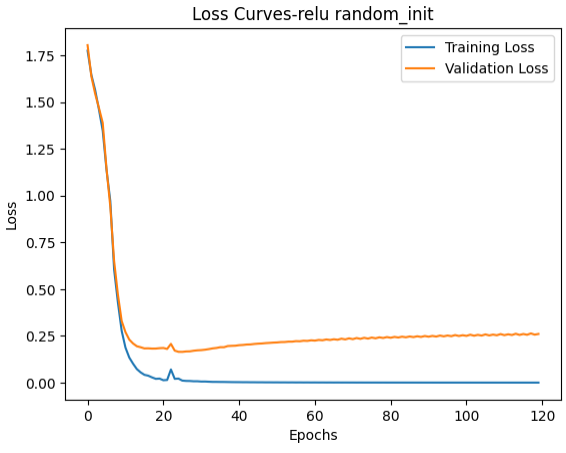



```
Training Accuracy: 0.880625
Validation Accuracy: 0.8678333333333333
Test Accuracy : 0.8717
```

Relu

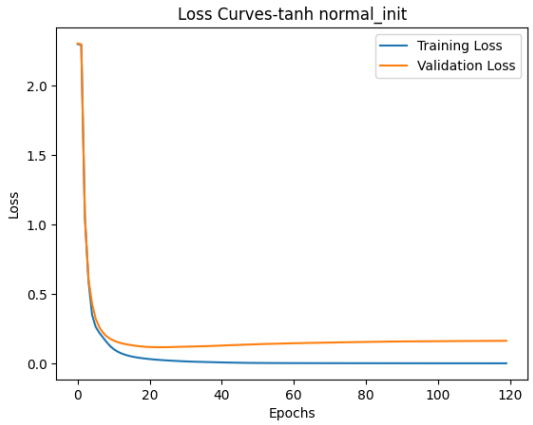



```
Training Accuracy : 1.0
Validation Accuracy: 0.97075
Test Accuracy : 0.9712
```

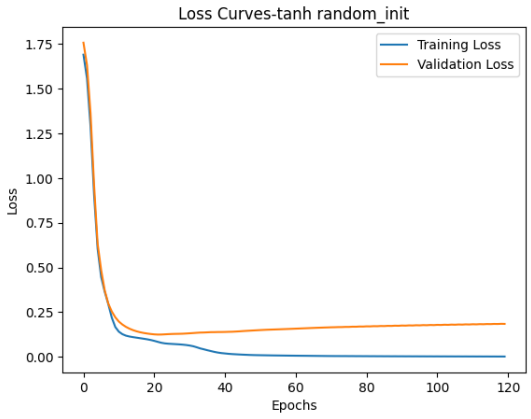



```
Training Accuracy: 0.9999375
Validation Accuracy: 0.96525
Test Accuracy : 0.96
```

Tanh

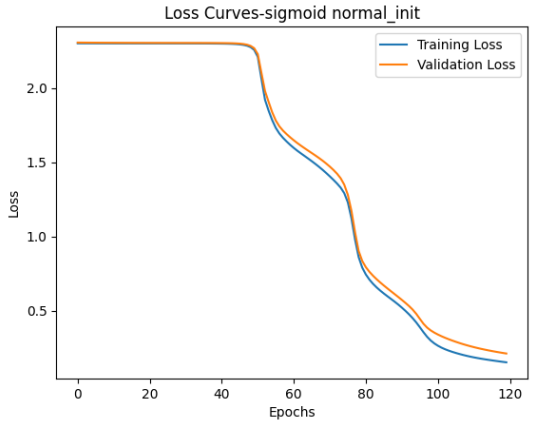


Training Accuracy: 0.9999791666666666
Validation Accuracy: 0.97
Test Accuracy: 0.9714

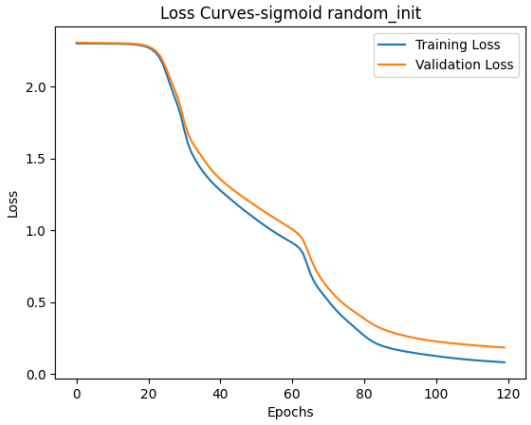


Training Accuracy: 0.9999791666666666
Validation Accuracy: 0.96775
Test Accuracy : 0.9703

Sigmoid

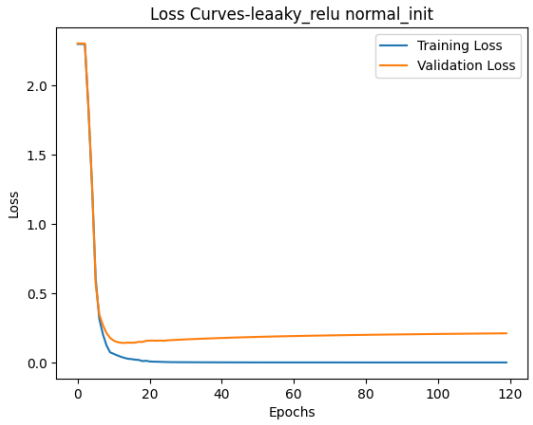


Training Accuracy: 0.9534583333333333
Validation Accuracy: 0.94075
Test Accuracy : 0.9449

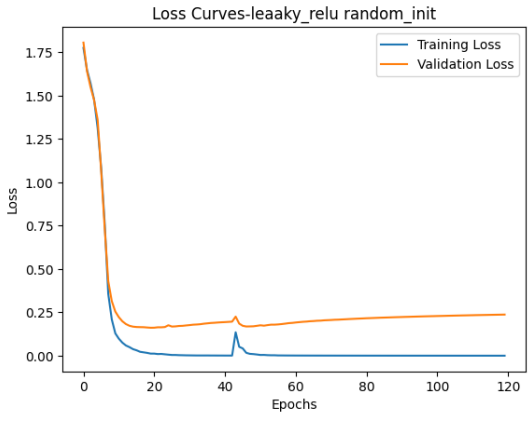


Training Accuracy: 0.9700416666666667
Validation Accuracy: 0.94925
Test Accuracy : 0.9533

Leaky Relu



Training Accuracy: 1.0
Validation Accuracy: 0.9725833333333334
Test Accuracy: 0.9723



Training Accuracy: 0.9999791666666666
Validation Accuracy: 0.9679166666666666
Test Accuracy: 0.968

First, I preprocessed the data by dividing pixel values by 255 and did one-hot encoding of the labels.