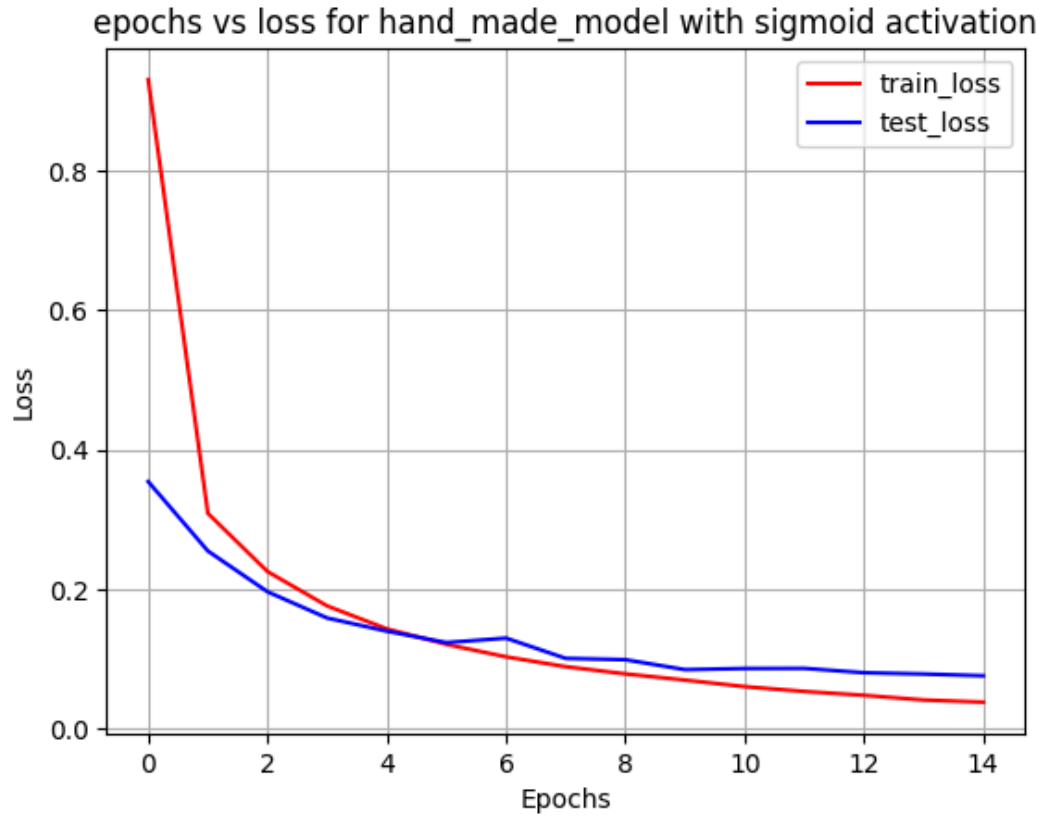
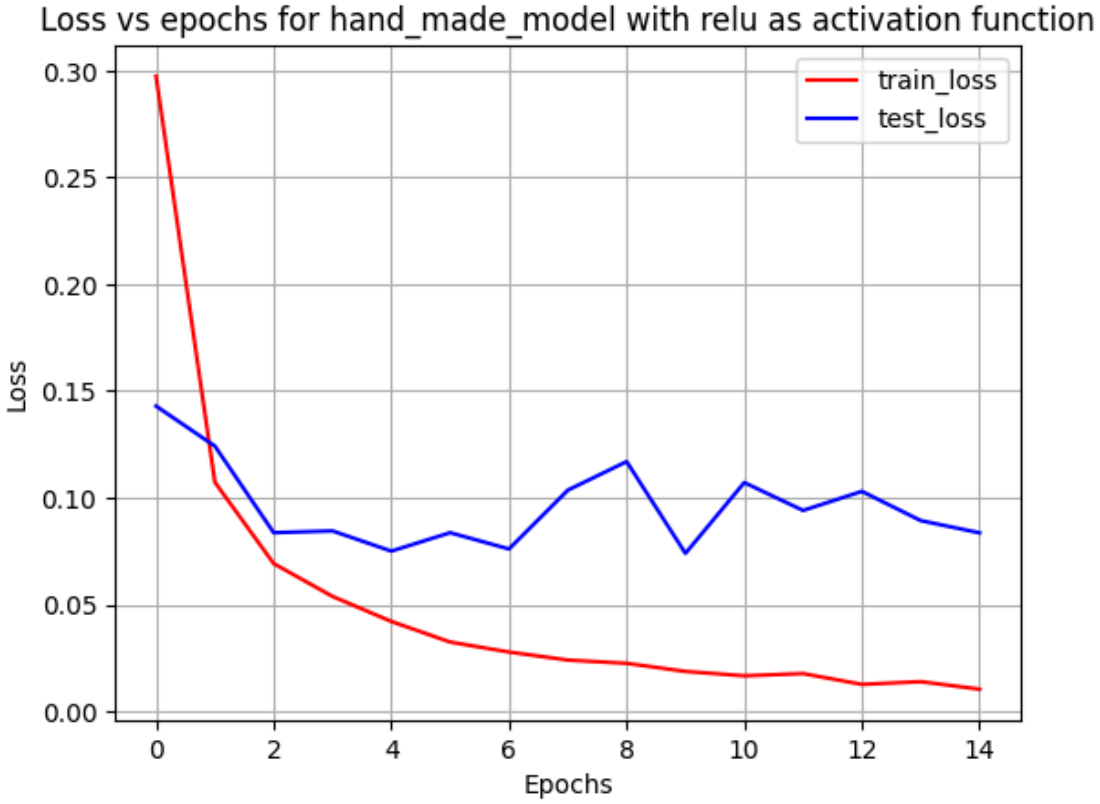
1.ReLU activation function performed best, when compared to Tanh and sigmoid.
2.Results observed:

S.no	Model	Activation func	Train accuracy	Test accuracy
1	Without lib	sigmoid	99.15	97.66
2	Without lib	relu	99.78	98.29
3	Without lib	tanh	99.85	97.8
4	With lib and without regularizer	sigmoid	98.51	97.04
5	With lib and with regularizer	sigmoid	97.65	96.87
6	With lib and without regularizer	relu	97.925	96.58
7	With lib and with regularizer	relu	97.48	96.26

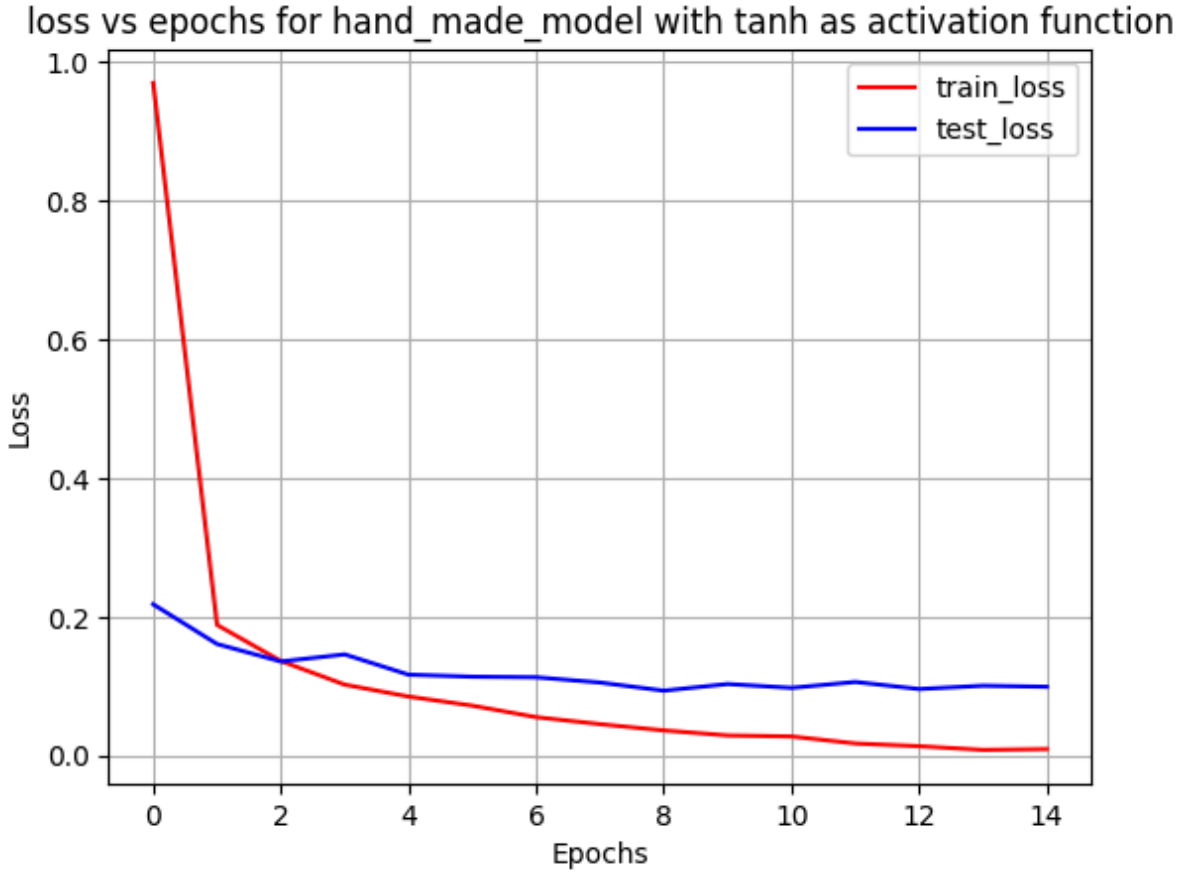
Graph 1:



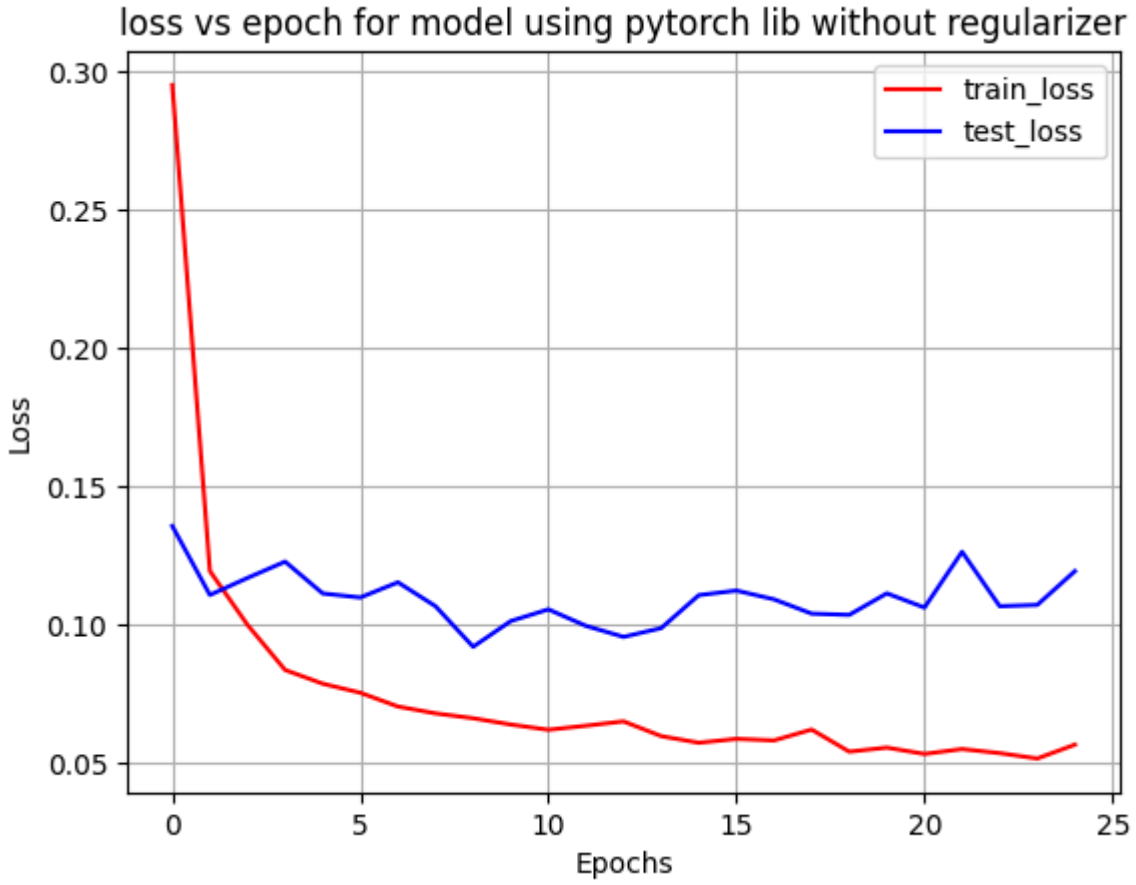
Graph 2:


Loss vs epochs for hand_made_model with relu as activation function
train_loss
test_loss
Loss
Epochs
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
10
12
14

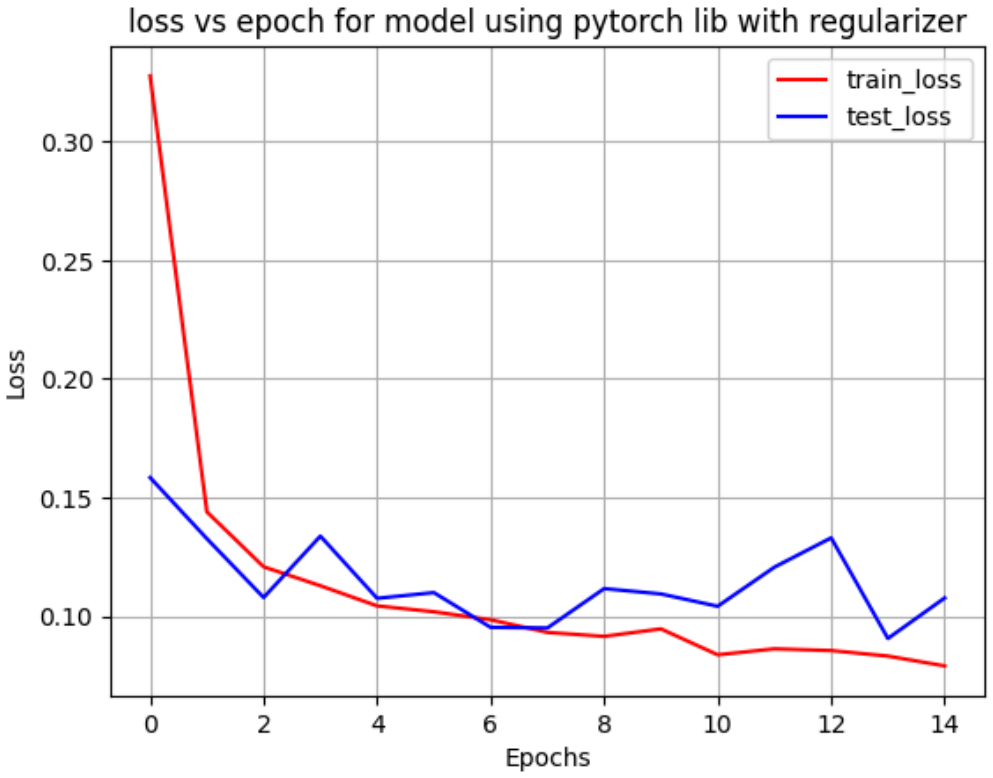
Graph 3:


loss vs epochs for hand_made_model with tanh as activation function
train_loss
test_loss
Loss
Epochs
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14

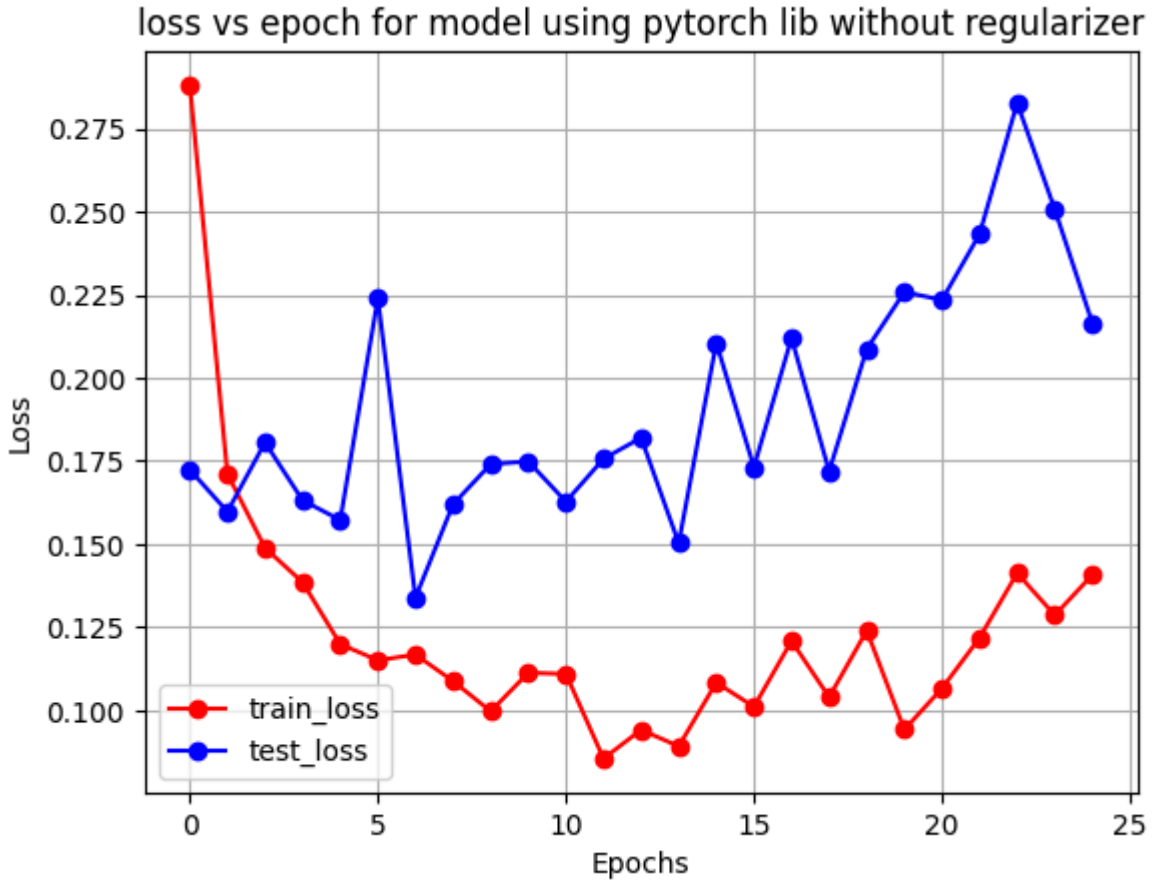
Graph 4



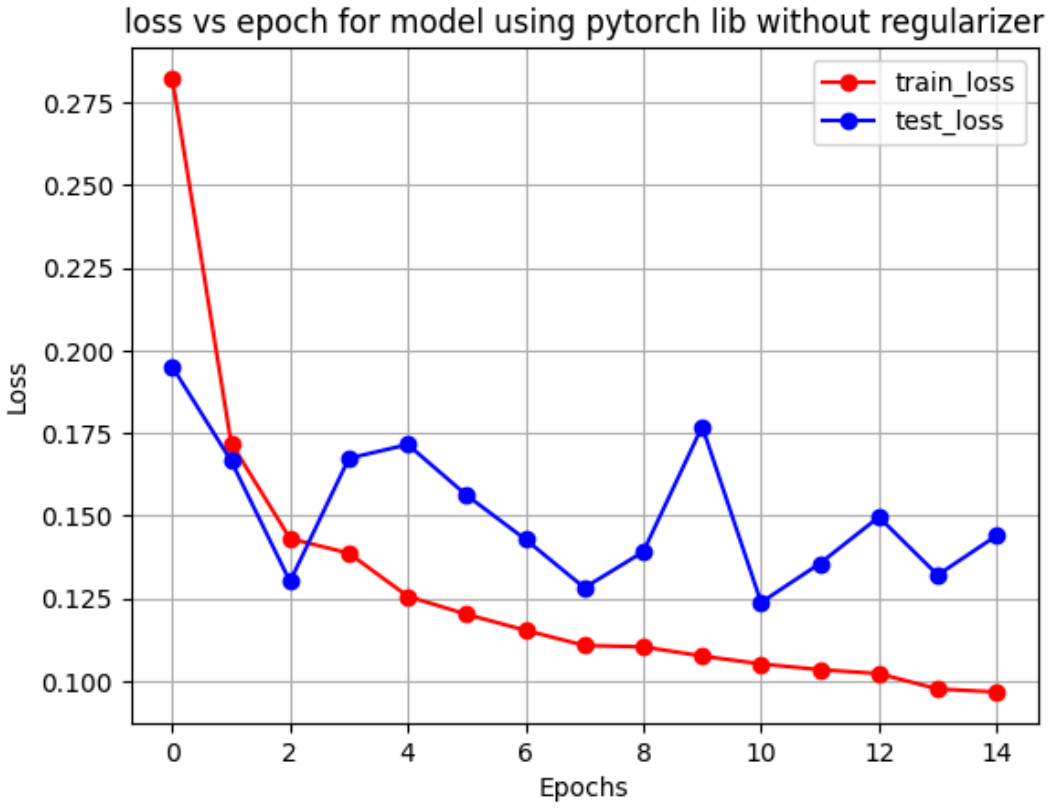
Graph 5



Graph 6:



Graph 7:



Observation:
The network coded from scratch performed slightly better than the model created using libraries, but overall the performance was almost similar.

When L2 regularization was applied, the models did not perform as well compared to the ones without regularization. One possible reason is that the training dataset may contain very little noise. In such cases, adding a regularization penalty can reduce the magnitude of weights unnecessarily, which may lead to the model losing some essential learning ability. As a result, both training and testing accuracy decreased.

Another possible explanation is that the chosen regularization strength (λ) might have been too high relative to the dataset size and complexity, causing underfitting. With further tuning of λ, it is possible that the regularized models could achieve a better balance between generalization and accuracy.