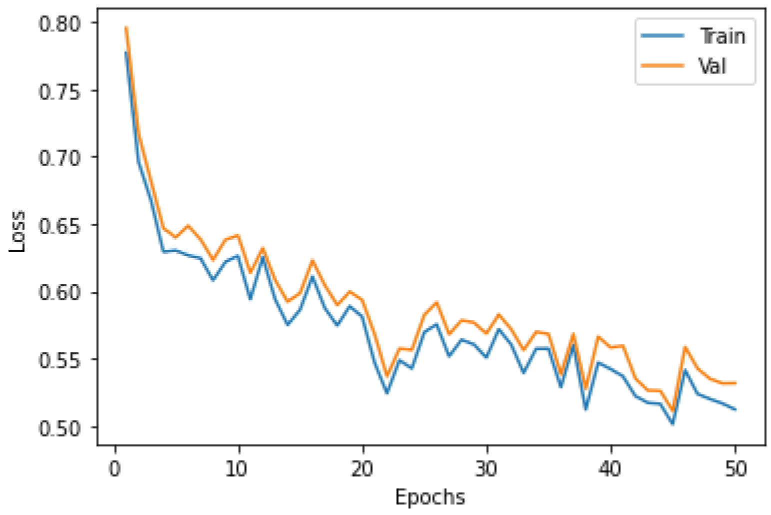
ML Assignment -3

Name-Ishit Bajpai
2020380

Section C

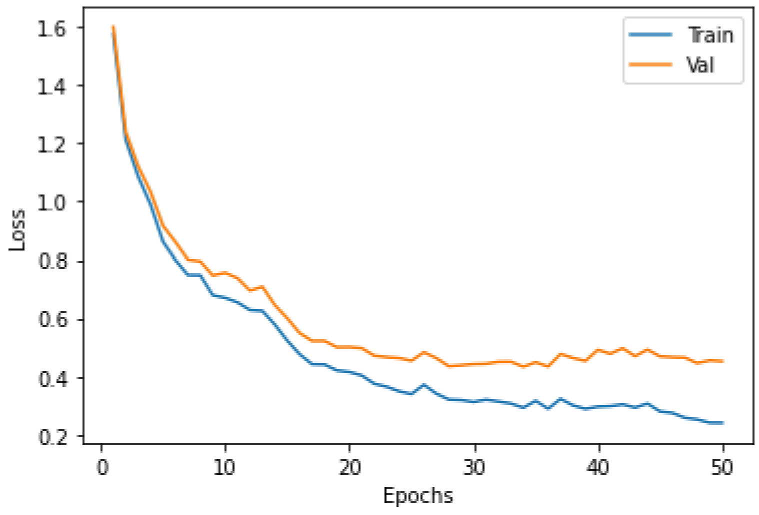
Q1

1.Sigmoid



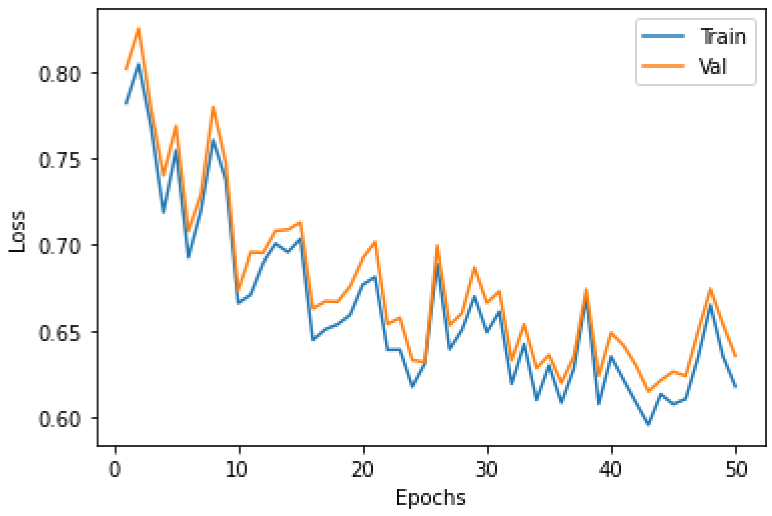
Training Loss 0.5121656243356365
Validation Loss 0.5316081109064739

2.Relu



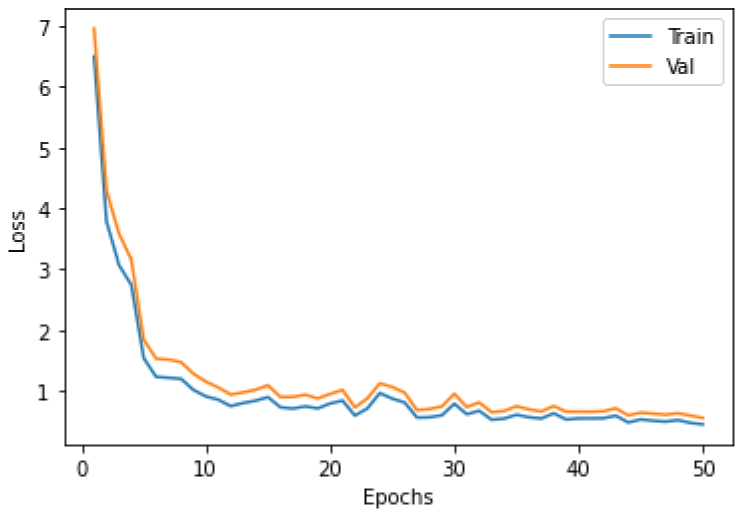
Training Loss 0.2403317582242233
Validation Loss 0.4522314115506646

3.tanh



Training Loss 0.6174670197379032
Validation Loss 0.6352474545632485

4.Identity



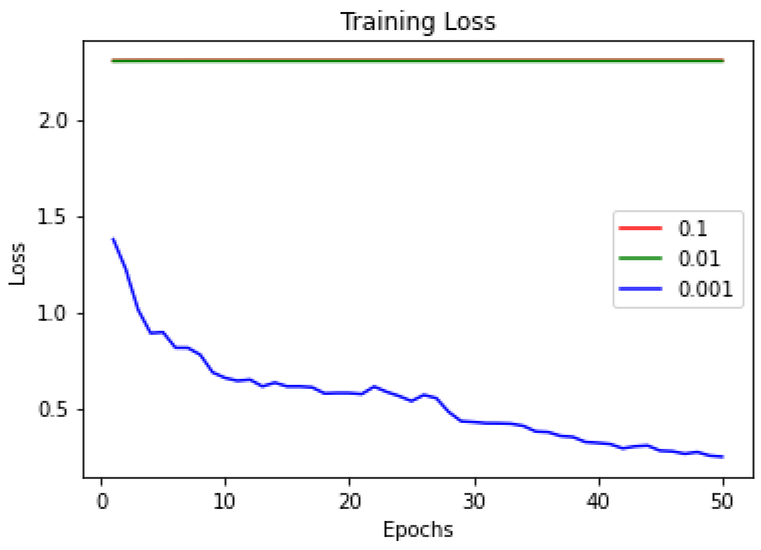
Training Loss 0.4487138803855511
Validation Loss 0.5531913168560768

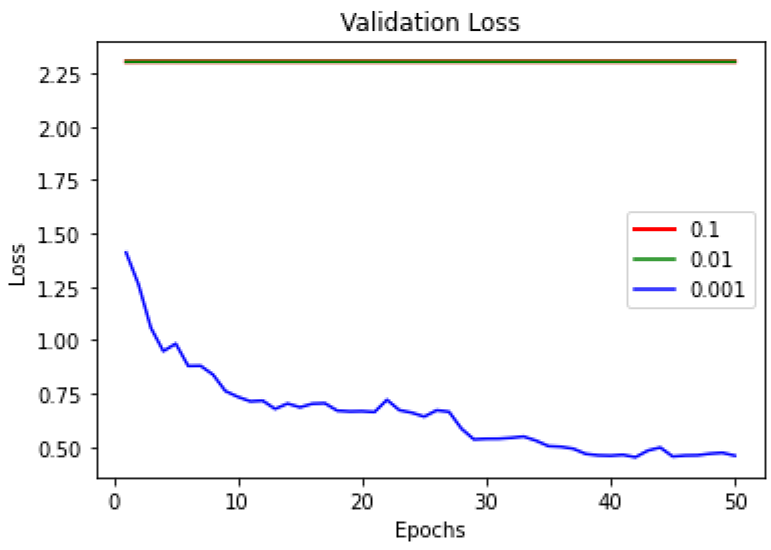
Activation Function	Training Loss	Validation Loss
Sigmoid	0.5121656243356365	0.5316081109064739
ReLu	**0.2403317582242233**	**0.4522314115506646**
tanh	0.6174670197379032	0.6352474545632485
Identity	0.4487138803855511	0.5531913168560768

ReLu is the best activation function.
Training Loss : ReLu < Identity < Sigmoid < tanh
Validation Loss: ReLu < Sigmoid < Identity < tanh

Relu is most efficient because gradient descent run faster than tanh and sigmoid as it is flat at only one place (x<0) while tanh and sigmoid tends to flatten out at two places as |x| increases.
it is less computationally expensive and provides better results than tanh and sigmoid in the same number of iterations.
Identity sigmoid function does not allow neural networks to learn patterns as it is just a linear combination of input and back-propagation is not possible(as opposed to relu).

Q2





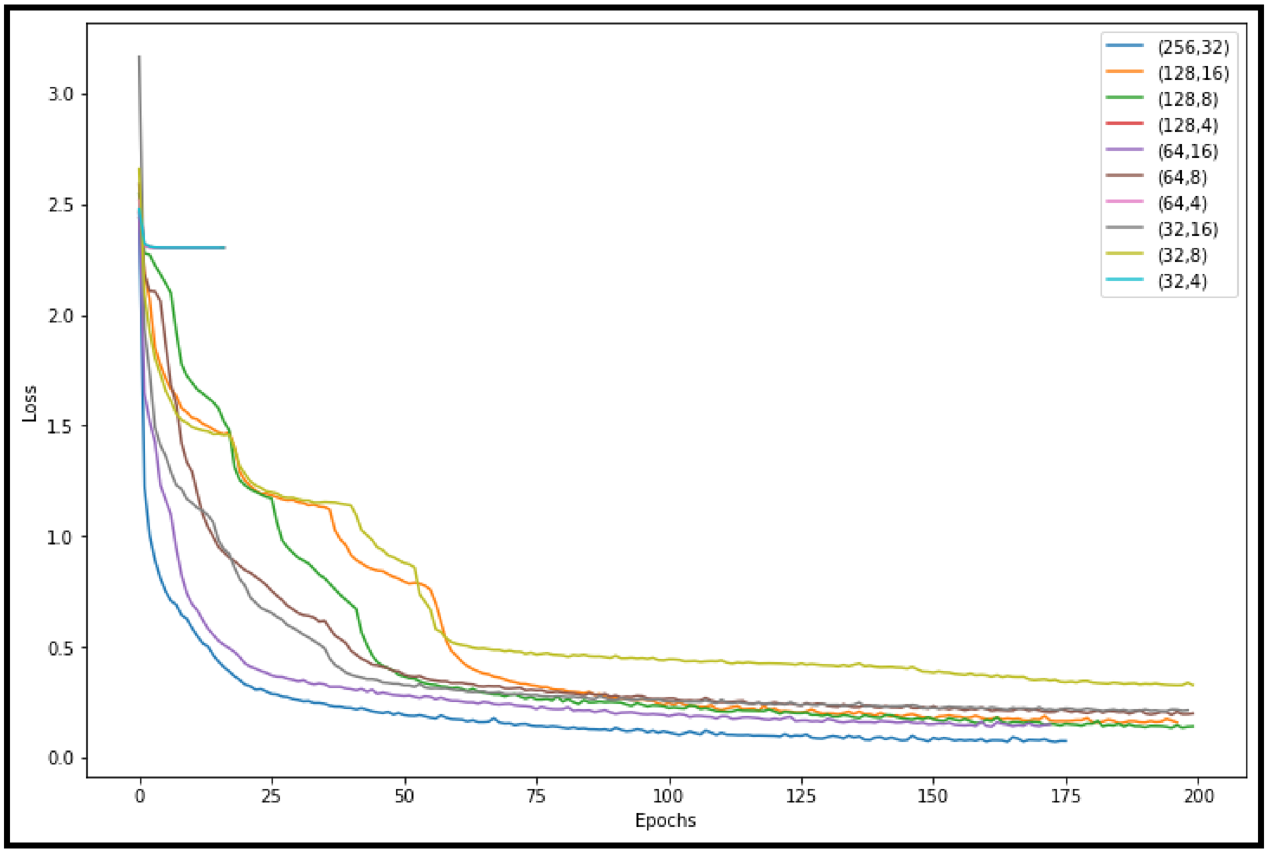
Note:These

Alpha	Training loss	Validation loss
0.1	2.3035165687069568	2.3053621605483565
0.01	2.3026300518004685	2.3028437166388773
0.001	**0.24730510849988496**	**0.45997101014721165**

Best learning rate = 0.001
Observation:
As alpha’s value becomes small, training and validation loss decreases.
As alpha decreases, the neural network converges very slowly (gradient descent runs slower) and therefore passes through more points and is able to capture more information about data as compared to larger values of alpha (and smaller alpha has less chances of missing the global minima as larger values of alpha might overshoot)

Thus for a smaller learning rate algorithm and sufficient iterations model tries to overfit data and therefore loss is least in the smallest alpha .
For alpha = 0.001 training loss is much smaller than Validation loss as compared to other alpha indicating possible overfitting.

Q3.

- 

256,32
0.07524217305256606

(128,16)
0.14807865784083069

(128,8)
0.12354102513370128

(128,4)
2.302583070867583

(64,16)
0.137025639612997

(64,8)
0.18788120057668095

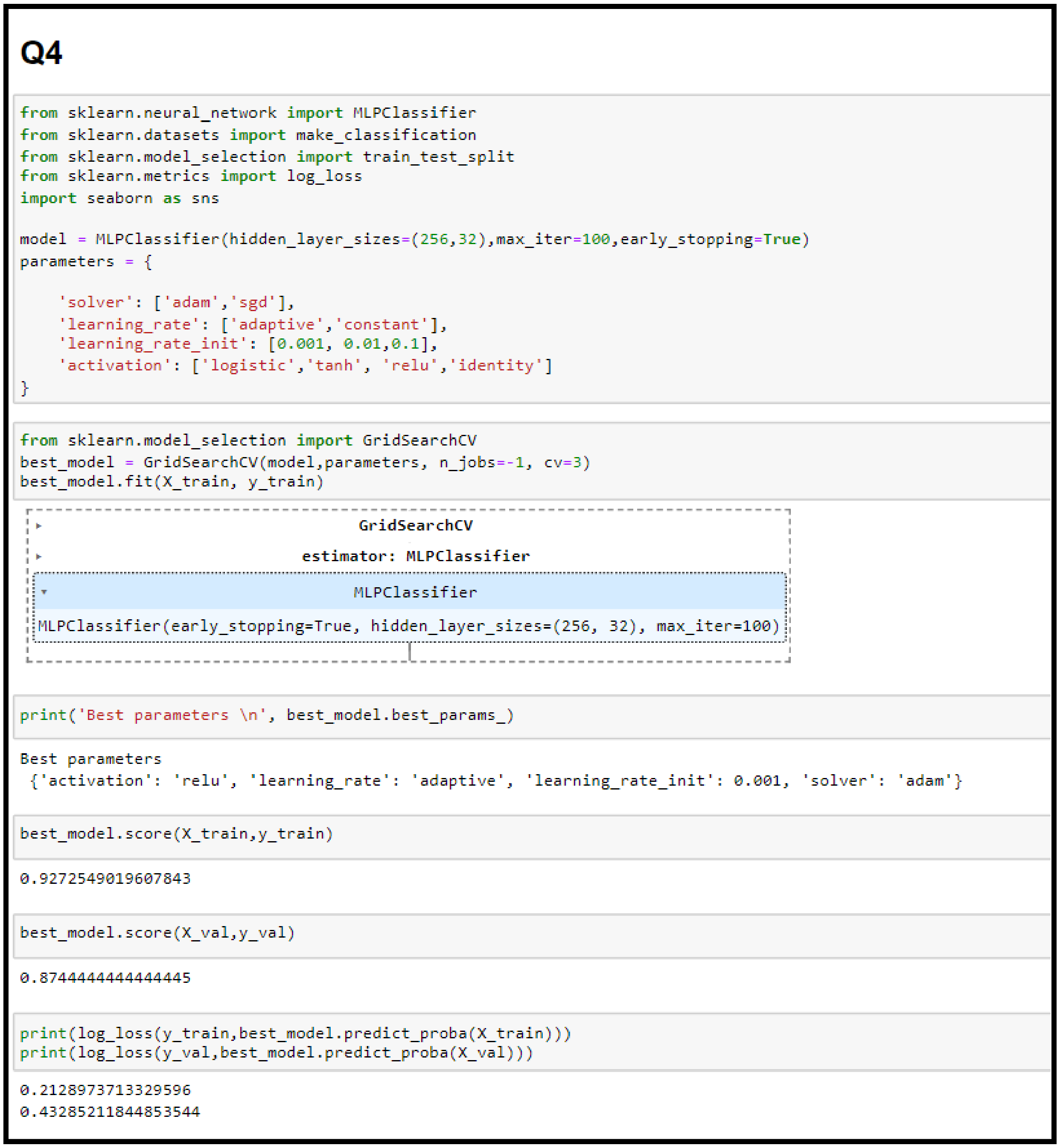
(64,4)
2.3025801646120128

(32,16)
0.20568937613144989

(32,8)
0.3193959502308099

(32,4)
2.3025776960363626
Observation:
Best learning rate for layer size = (256,32) with a loss of (0.07)
1)Using fewer neurons in hidden layers makes it difficult for Neural networks to learn from training data and can result in underfitting.
2)Using too many neurons allows neural networks to learn about data more thoroughly and can result in overfitting.
Therefore as the number of neurons decreases the training loss increases.
Using more neurons helps neural networks capture(learn) more information about training data thus reducing the loss.

Q4.
Best parameters
```
 {'activation': 'relu', 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'solver': 'adam'}
```



Q4

```
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import log_loss
import seaborn as sns

model = MLPClassifier(hidden_layer_sizes=(256,32),max_iter=100,early_stopping=True)
parameters = {

    'solver': ['adam','sgd'],
    'learning_rate': ['adaptive','constant'],
    'learning_rate_init': [0.001, 0.01,0.1],
    'activation': ['logistic','tanh', 'relu','identity']
}
```

```
from sklearn.model_selection import GridSearchCV
best_model = GridSearchCV(model,parameters, n_jobs=-1, cv=3)
best_model.fit(X_train, y_train)
```

```
GridSearchCV
estimator: MLPClassifier
MLPClassifier
MLPClassifier(early_stopping=True, hidden_layer_sizes=(256, 32), max_iter=100)
```

```
print('Best parameters \n', best_model.best_params_)
```

```
Best parameters
 {'activation': 'relu', 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'solver': 'adam'}
```

```
best_model.score(X_train,y_train)
```

```
0.9272549019607843
```

```
best_model.score(X_val,y_val)
```

```
0.8744444444444445
```

```
print(log_loss(y_train,best_model.predict_proba(X_train)))
print(log_loss(y_val,best_model.predict_proba(X_val)))
```

```
0.2128973713329596
0.43285211844853544
```

Best Activation function was **ReLu** as explained before(non-linear activation function and gradient descent runs faster than sigmoid and tanh i.e converges quickly to minima).

Best layer was (256,32) as compared to layers with fewer neurons .As the number of neurons decreases the training loss increases.Using more neurons help neural networks capture(learn) more information about training data thus reducing the loss

Solver Adam is chosen because it automatically adjusts alpha. Adam updates the alpha so as to converge to global minima faster than sgd.

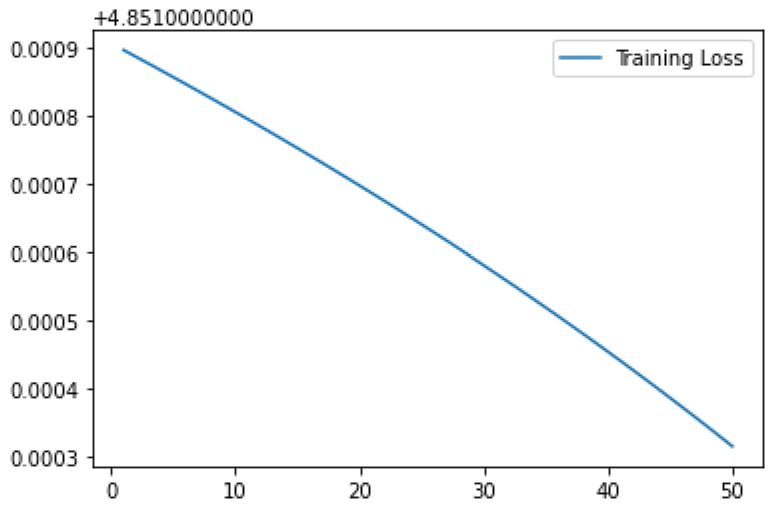
Adaptive learning rate is better than constant learning rate as adaptive learning allows model to update alpha so as to give better results.

Learning rate = 0.001
As alpha's value becomes small, training and validation loss decreases(seen before)
As alpha decreases, the neural network converges very slowly (gradient descent runs slower) and therefore passes through more points and is able to capture more information about data as compared to larger values of alpha (and smaller alpha has less chances of missing the global minima as larger values of alpha might overshoot)

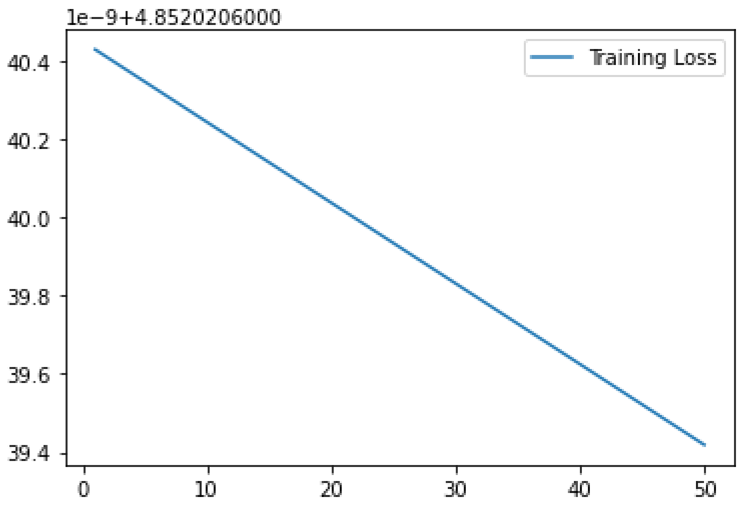
SectionB

Training Loss

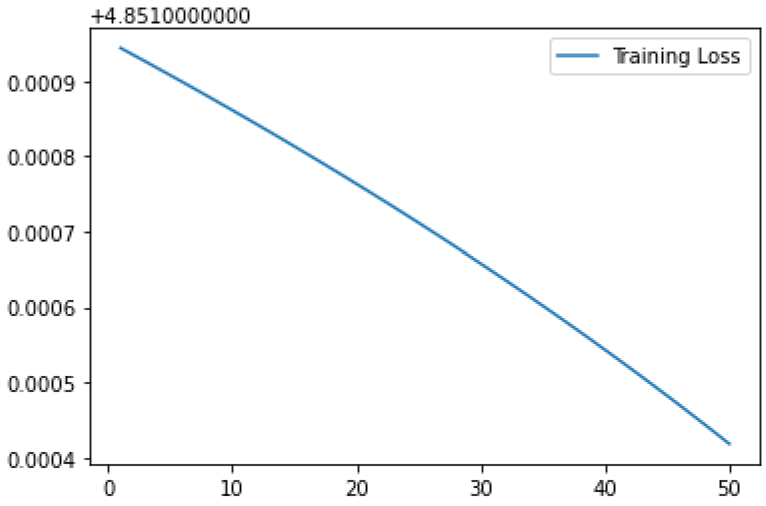
ReLu



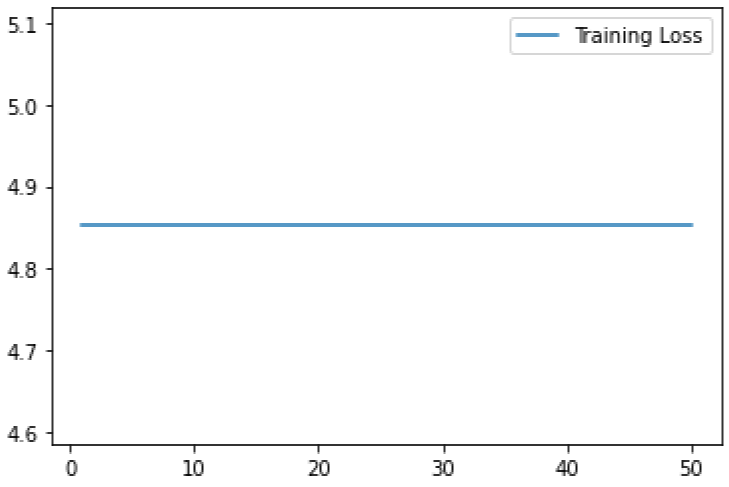
tanh



LeakyReLu

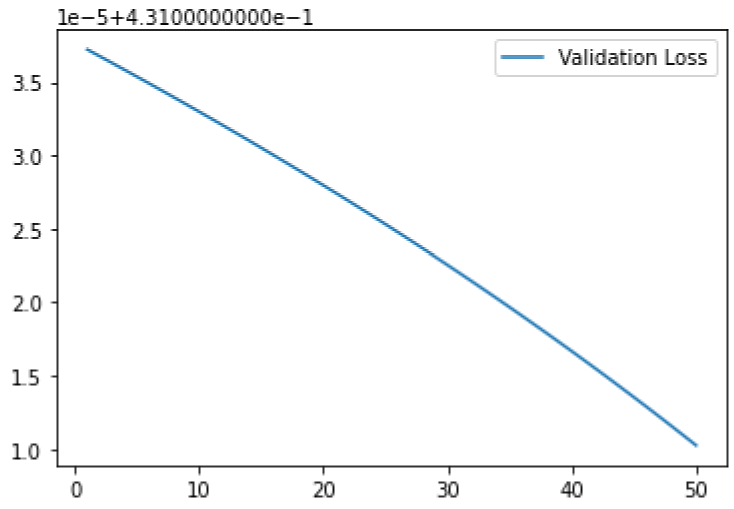


Sigmoid

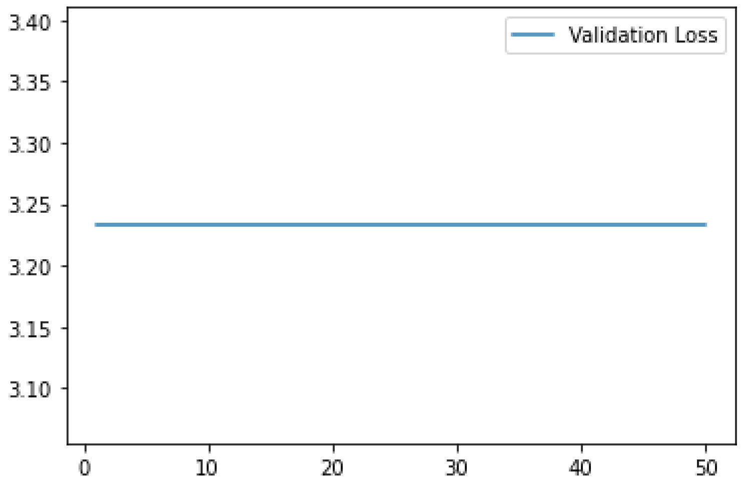


Validation Losses

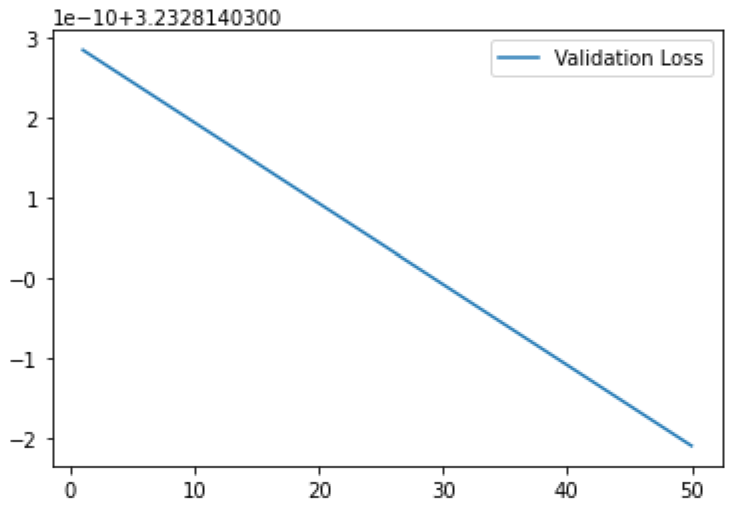
ReLu



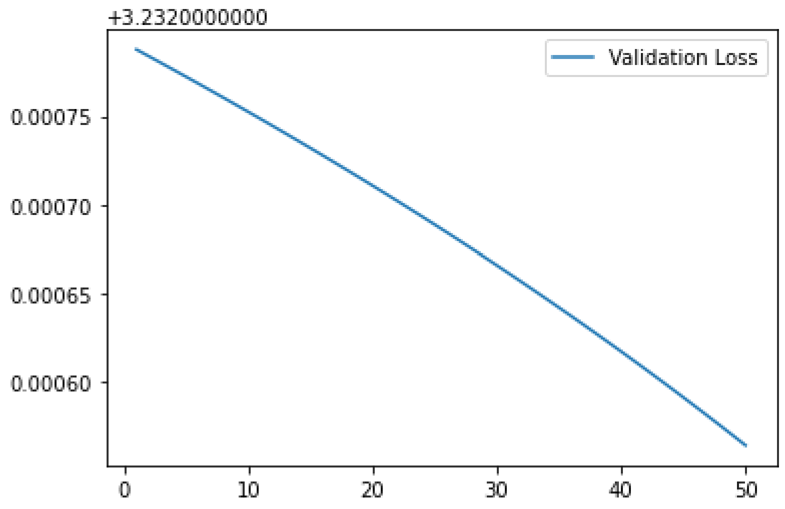
Sigmoid



Tanh



LeakyReLu


+3.2320000000
Validation Loss
0.00075
0.00070
0.00065
0.00060
0
10
20
30
40
50