SENTIMENT ANALYSIS ON AMAZON BOOK REVIEW

<u>NAME</u>	<u>ROLL</u>
Kingshuk Basak	1611CS06
Garima Sahu	1721EE04

PAPERS READ

- Transfer Learning for Cross-Lingual Sentiment Classification with Weakly Shared Deep Neural Networks. By: Guangyou Zhou, Zhao Zeng, Jimmy Xiangji Huang, and Tingting He.
- Neural Network Based Context Sentiment Analysis. By: S.Suruthi, M.Pradeeba, A.Sumaiya.
- Convolution Neural Networks for Sentiment Classification. By: Yoon Kim.
- An Approach to Sentiment Analysis using Artificial Neural Network with Comparative Analysis of Different Techniques. By: Pranali Borele, DIlipkumar A. Borikar
- Convolutional Neural networks for Sentiment Classification. By: Yoon Kun.
- Neural Network for Sentiment Analysis on Twitter. By: Brett Duncan and Yanqing Zhang.

Neural Network for Sentiment Analysis on Twitter.

By: Brett Duncan and Yanqing Zhang.

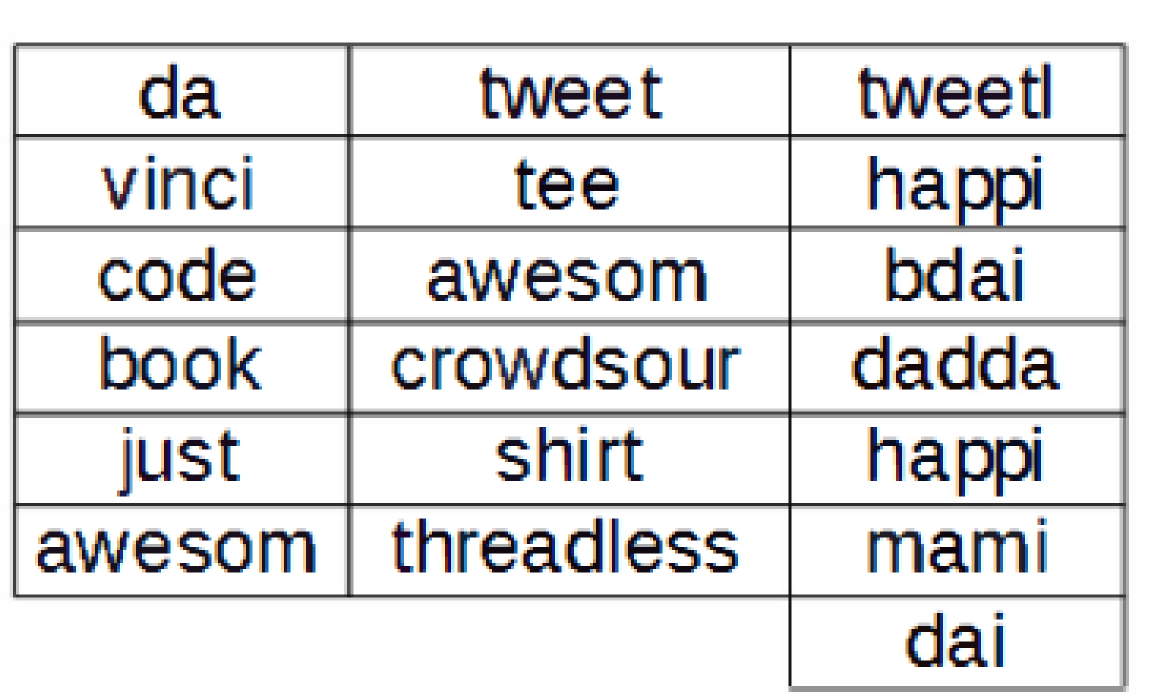
DESCRIPTION

- Data Set: Twitter.
- Methods:
  - Tokenization.
  - Stemming.
  - Preprocessing.
  - Creating Vocabulary list.
  - Loading Training Labels.
  - Map Variable.
  - Numerical Training Vector.
  - Training the Neural Network.
  - Collecting Test Tweets.
  - Numerical Test Vector.

PRE-PROCESSING

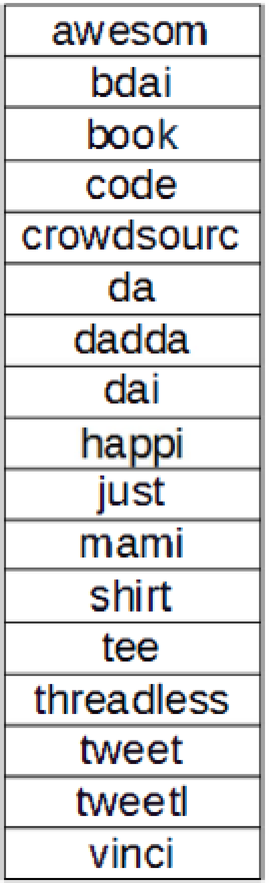
- Removal of Punctuation and symbols.
- Removal of @ words.
- Removal of single character.
- Removal of stop words.

AFTER PRE-PROCESSING



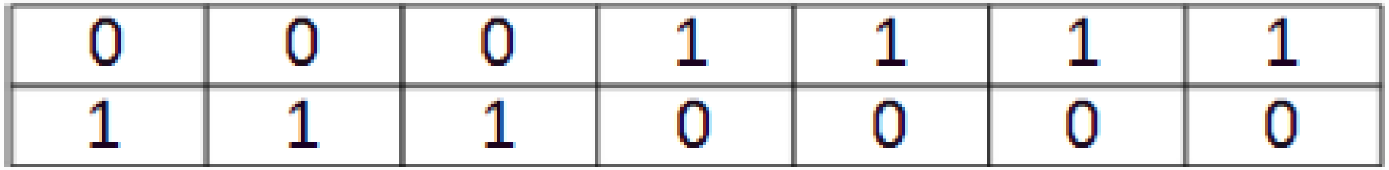
da	tweet	tweetl
vinci	tee	happi
code	awesom	bdai
book	crowdsour	dadda
just	shirt	happi
awesom	threadless	mami
		dai

CREATING VOCABULARY LIST



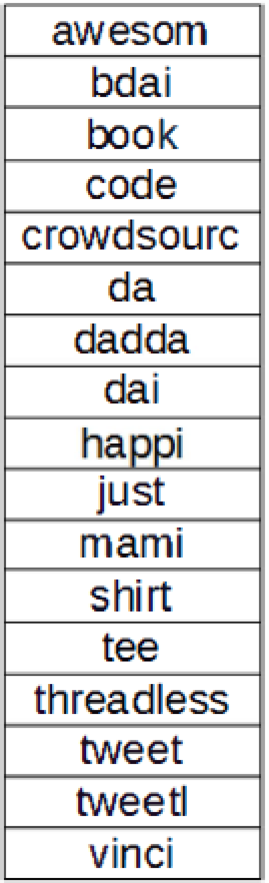
awesom
bdai
book
code
crowdsourc
da
dadda
dai
happi
just
mami
shirt
tee
threadless
tweet
tweetl
vinci

LOADING TRAINING LABELS



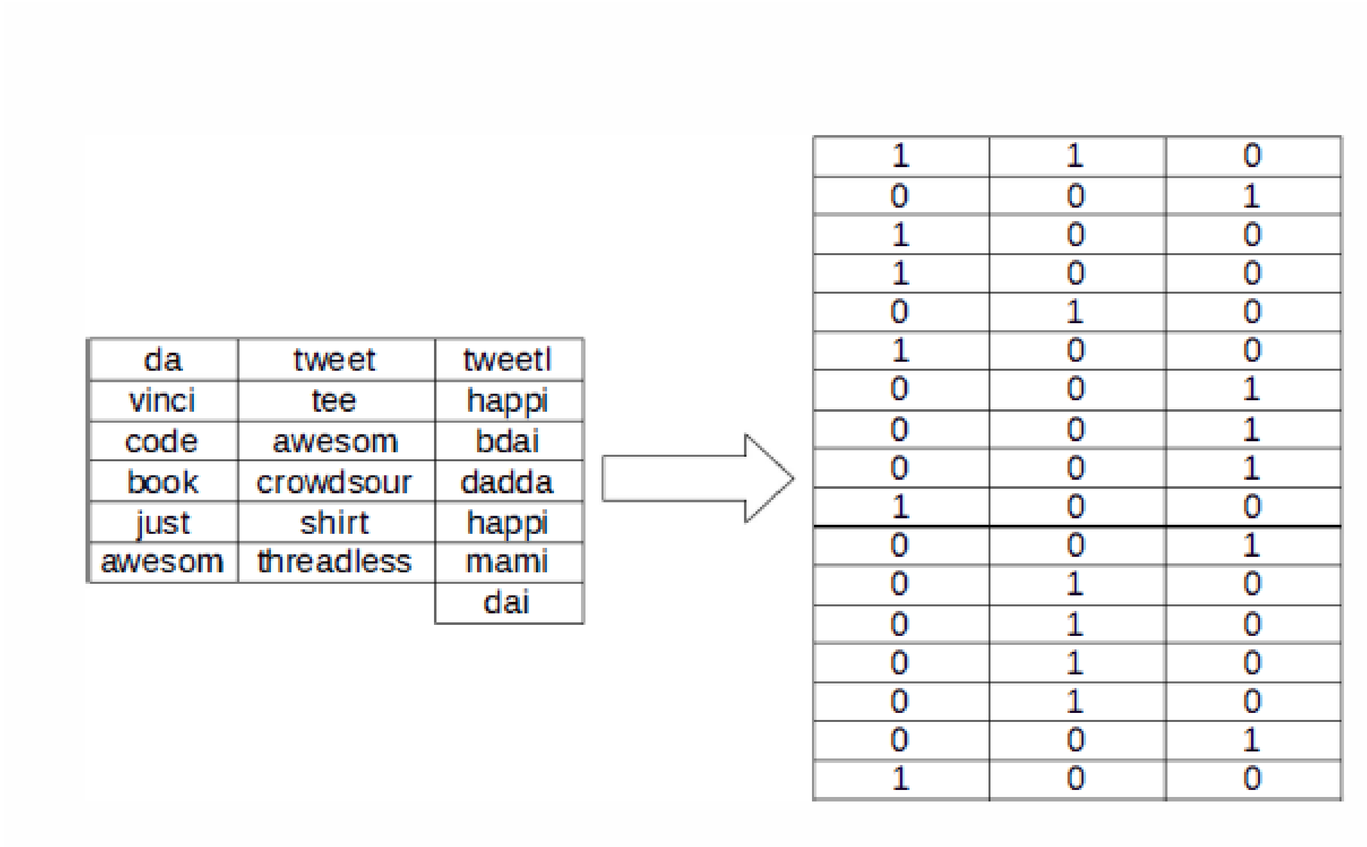
0	0	0	1	1	1	1
1	1	1	0	0	0	0

MAP VARIABLE



awesom
bdai
book
code
crowdsourc
da
dadda
dai
happi
just
mami
shirt
tee
threadless
tweet
tweetl
vinci

NUMERICAL TRAINING VECTOR



da	tweet	tweetl
vinci	tee	happi
code	awesom	bdai
book	crowdsour	dadda
just	shirt	happi
awesom	threadless	mami
		dai

1	1	0
0	0	1
1	0	0
1	0	0
0	1	0
1	0	0
0	0	1
0	0	1
0	0	1
1	0	0
0	0	1
0	1	0
0	1	0
0	1	0
0	1	0
0	0	1
1	0	0

DATA SET

- Amazon Book Review Data
  - Andy-Weir-The-Martian
  - Donna-Tartt-The-Goldfinch
  - EL-James-Fifty-Shades-of-Grey
  - Fillian_Flynn-Gone_Girl
  - John-Green-The-Fault-in-our-Stars
  - Laura-Hillenbrand-Unbroken

USED LIBRARY

- NUMPY
- SKLEARN
- TENSORFLOW
- MATPLOTLIB
- HTMLPARSER
- ARGPARSE
- GENSIM
- DOC2VEC

PRE-PROCESSING

- Pre-processing is done using Java.
- Extracted sentences on basis of review.
- Removal of all HTML tags.
- Removal of Punctuations.
- File divided into 4 parts.
  - train.txt
  - train_target.txt
  - test.txt
  - text_target.txt

PROCESS DONE

- **DOC2VEC:**
  - Vocabulary list is created for all reviews.
  - Loading of Training as well as text data done.
  - Mapping of variables for training and text data with vocabulary.
  - As a output we are getting vector form of test and training data.

CREATING NEURAL NETWORK

We created neural network with two hidden layers with weights and biases depended on the vector size of the data.

- The input to the network is in vector form generated by Doc2vec.
- Tanh and Relu is being used as activation function in layer 1 and layer 2 respectively.
- Dropout is used for regularization to overcome over-fitting.
- We have used Feed Forward Neural Network.

TRAINING NEURAL NETWORK

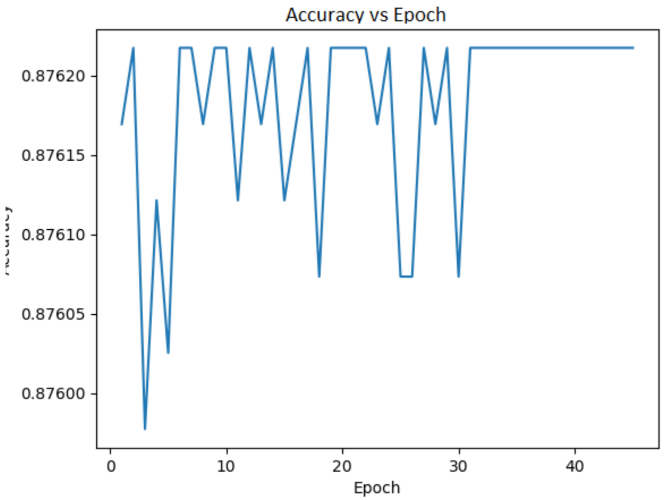
- Sparse softmax cross entropy is used for calculating the loss in the network.
- Rmsprop optimizer for optimizing the loss function.
- Epoch iteration being done with batch size 100.
- Calculation of loss and accuracy being done for each epoch.
- We have tested outcome by varying number of total epochs. Like – 45 , 100, 250, 500

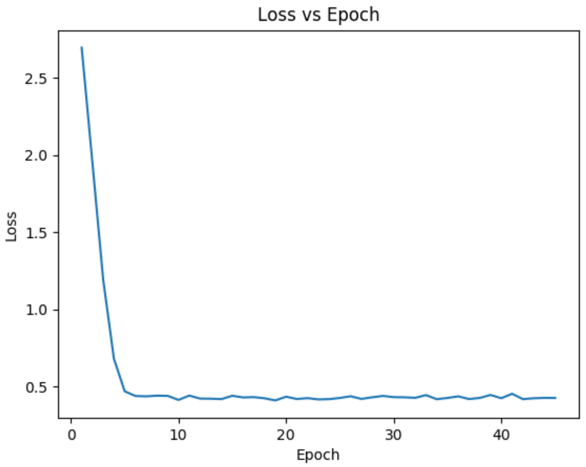
RESULTS

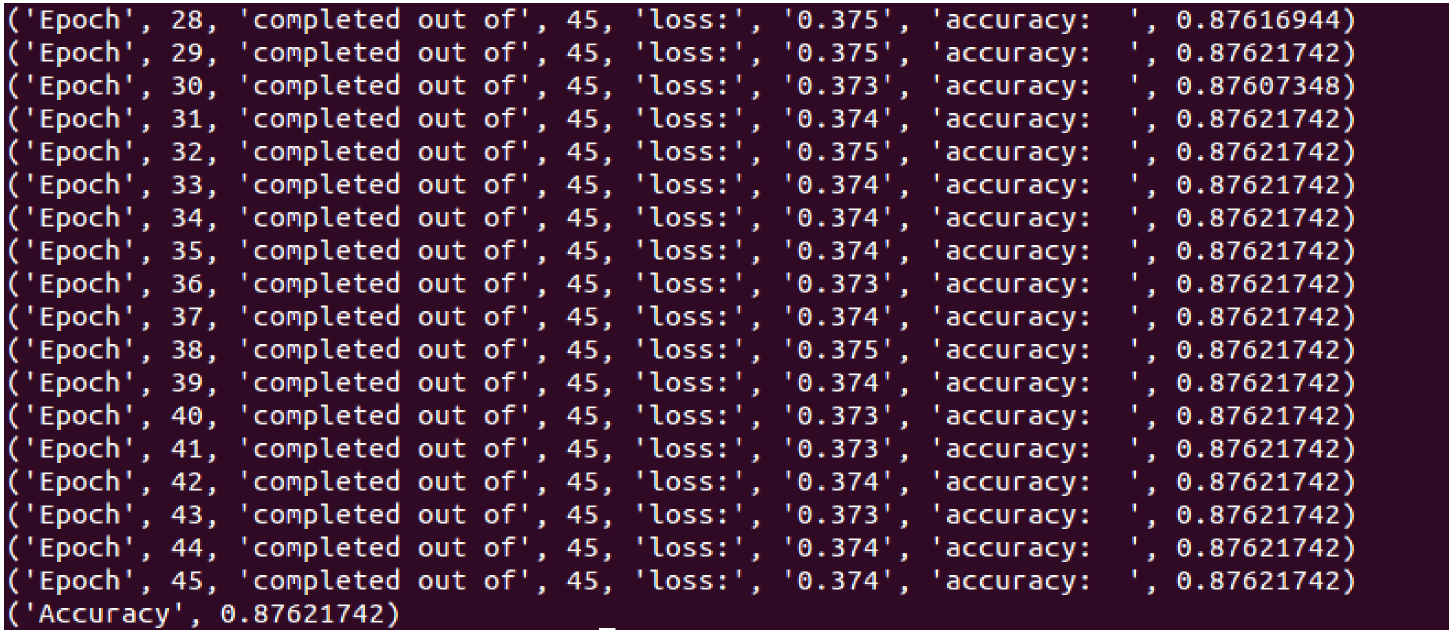
- Accuracy
- Epoch VS Loss graph.
- Epoch VS Accuracy graph.

- Epoch lists:
  - 45
  - 100
  - 250
  - 500

EPOCH 45

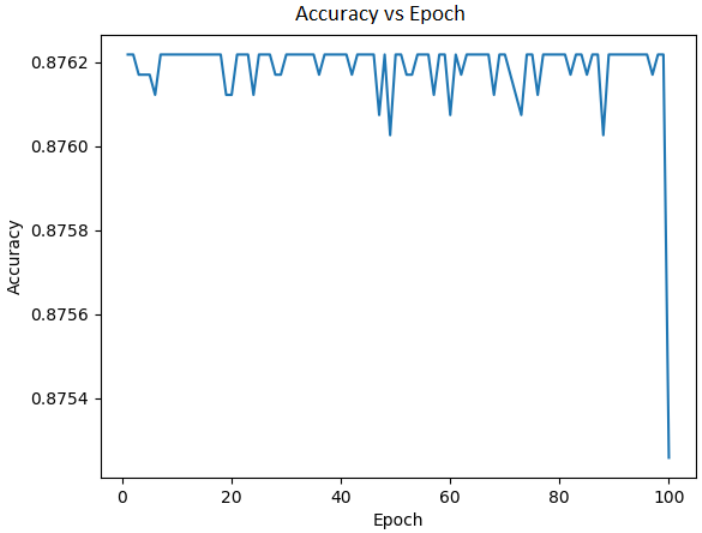


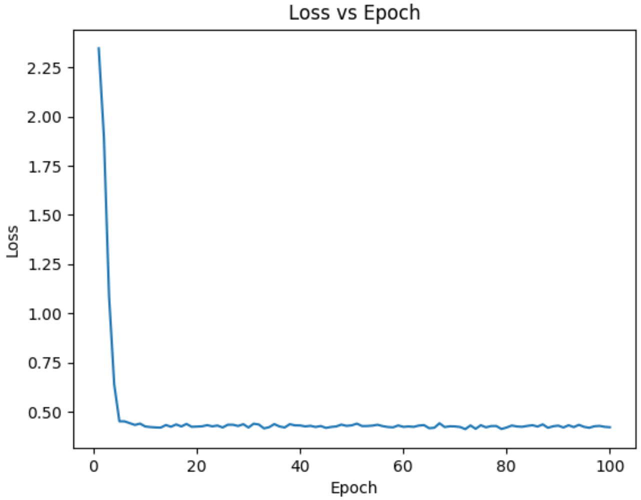


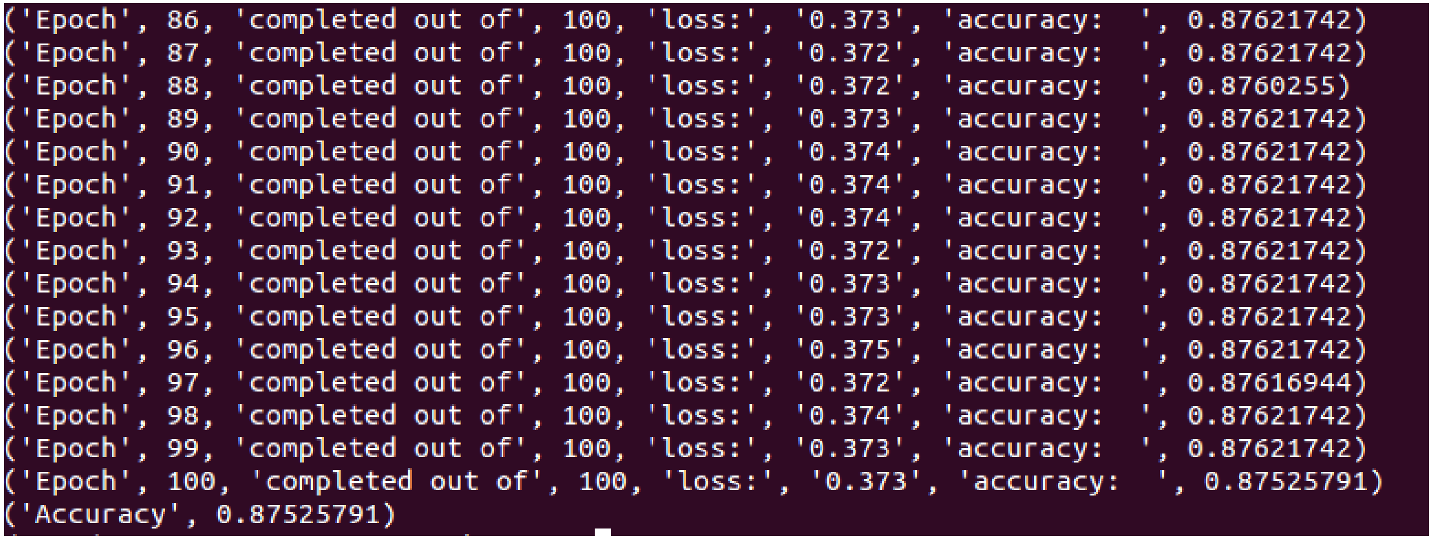


```
('Epoch', 28, 'completed out of', 45, 'loss:', '0.375', 'accuracy:  ', 0.87616944)
('Epoch', 29, 'completed out of', 45, 'loss:', '0.375', 'accuracy:  ', 0.87621742)
('Epoch', 30, 'completed out of', 45, 'loss:', '0.373', 'accuracy:  ', 0.87607348)
('Epoch', 31, 'completed out of', 45, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 32, 'completed out of', 45, 'loss:', '0.375', 'accuracy:  ', 0.87621742)
('Epoch', 33, 'completed out of', 45, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 34, 'completed out of', 45, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 35, 'completed out of', 45, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 36, 'completed out of', 45, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 37, 'completed out of', 45, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 38, 'completed out of', 45, 'loss:', '0.375', 'accuracy:  ', 0.87621742)
('Epoch', 39, 'completed out of', 45, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 40, 'completed out of', 45, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 41, 'completed out of', 45, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 42, 'completed out of', 45, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 43, 'completed out of', 45, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 44, 'completed out of', 45, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 45, 'completed out of', 45, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Accuracy', 0.87621742)
```

EPOCH 100

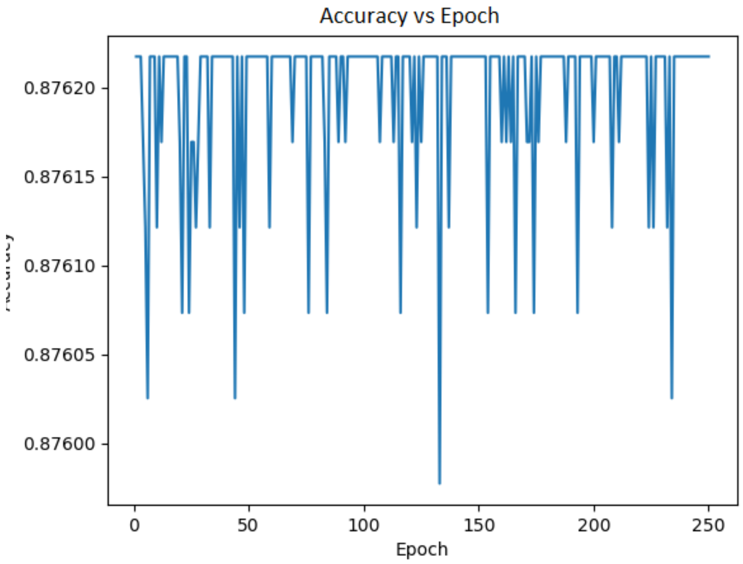


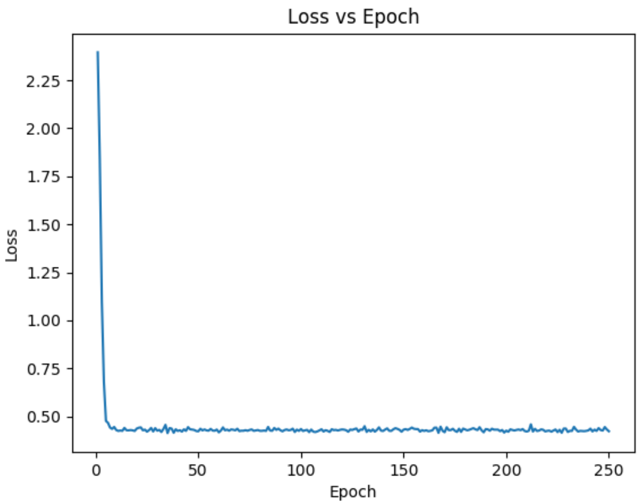


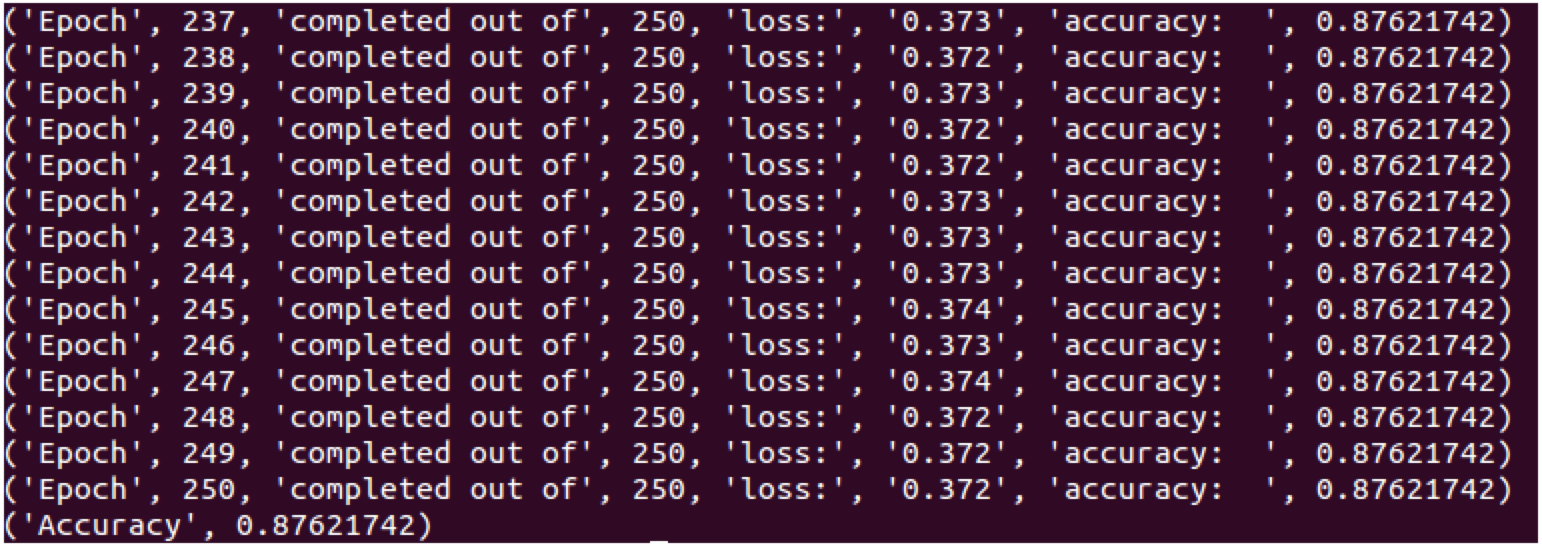


```
('Epoch', 86, 'completed out of', 100, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 87, 'completed out of', 100, 'loss:', '0.372', 'accuracy:  ', 0.87621742)
('Epoch', 88, 'completed out of', 100, 'loss:', '0.372', 'accuracy:  ', 0.8760255)
('Epoch', 89, 'completed out of', 100, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 90, 'completed out of', 100, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 91, 'completed out of', 100, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 92, 'completed out of', 100, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 93, 'completed out of', 100, 'loss:', '0.372', 'accuracy:  ', 0.87621742)
('Epoch', 94, 'completed out of', 100, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 95, 'completed out of', 100, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 96, 'completed out of', 100, 'loss:', '0.375', 'accuracy:  ', 0.87621742)
('Epoch', 97, 'completed out of', 100, 'loss:', '0.372', 'accuracy:  ', 0.87616944)
('Epoch', 98, 'completed out of', 100, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 99, 'completed out of', 100, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 100, 'completed out of', 100, 'loss:', '0.373', 'accuracy:  ', 0.87525791)
('Accuracy', 0.87525791)
```

EPOCH 250

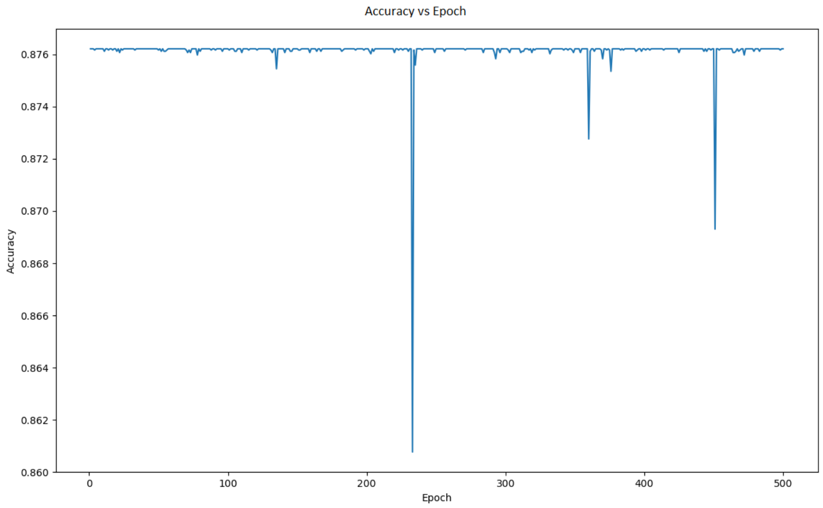


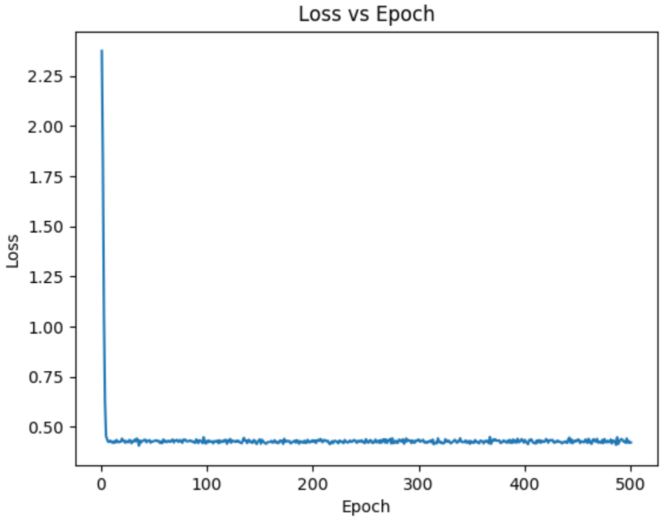


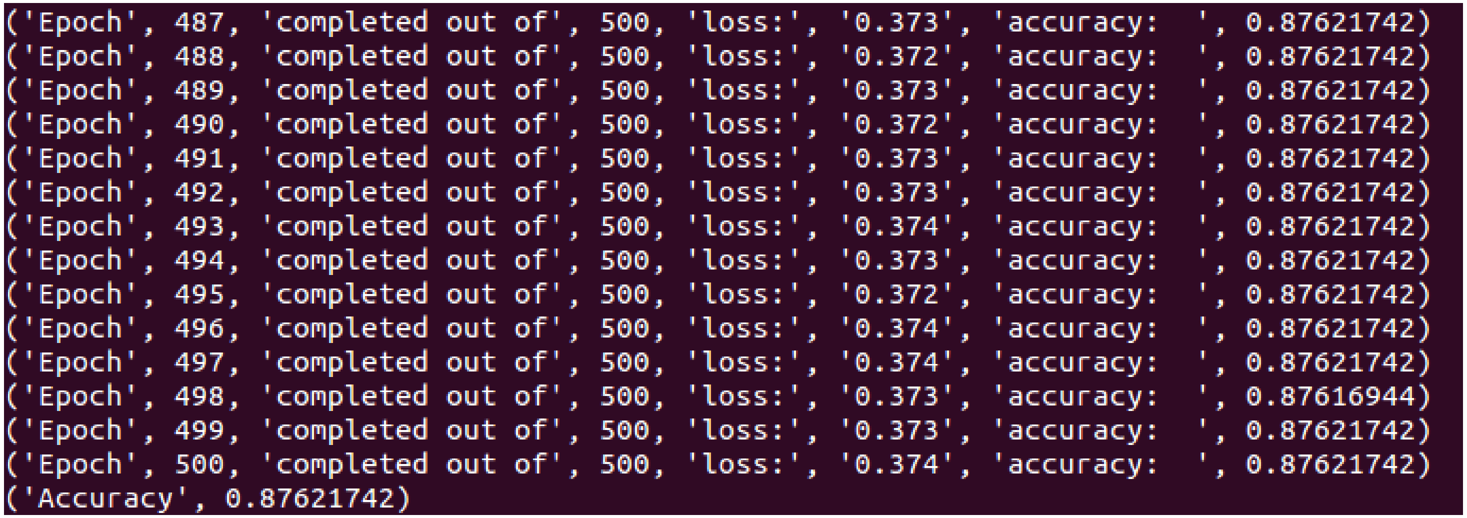


```
('Epoch', 237, 'completed out of', 250, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 238, 'completed out of', 250, 'loss:', '0.372', 'accuracy:  ', 0.87621742)
('Epoch', 239, 'completed out of', 250, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 240, 'completed out of', 250, 'loss:', '0.372', 'accuracy:  ', 0.87621742)
('Epoch', 241, 'completed out of', 250, 'loss:', '0.372', 'accuracy:  ', 0.87621742)
('Epoch', 242, 'completed out of', 250, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 243, 'completed out of', 250, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 244, 'completed out of', 250, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 245, 'completed out of', 250, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 246, 'completed out of', 250, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 247, 'completed out of', 250, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 248, 'completed out of', 250, 'loss:', '0.372', 'accuracy:  ', 0.87621742)
('Epoch', 249, 'completed out of', 250, 'loss:', '0.372', 'accuracy:  ', 0.87621742)
('Epoch', 250, 'completed out of', 250, 'loss:', '0.372', 'accuracy:  ', 0.87621742)
('Accuracy', 0.87621742)
```

EPOCH 500







```
('Epoch', 487, 'completed out of', 500, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 488, 'completed out of', 500, 'loss:', '0.372', 'accuracy:  ', 0.87621742)
('Epoch', 489, 'completed out of', 500, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 490, 'completed out of', 500, 'loss:', '0.372', 'accuracy:  ', 0.87621742)
('Epoch', 491, 'completed out of', 500, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 492, 'completed out of', 500, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 493, 'completed out of', 500, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 494, 'completed out of', 500, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 495, 'completed out of', 500, 'loss:', '0.372', 'accuracy:  ', 0.87621742)
('Epoch', 496, 'completed out of', 500, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 497, 'completed out of', 500, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Epoch', 498, 'completed out of', 500, 'loss:', '0.373', 'accuracy:  ', 0.87616944)
('Epoch', 499, 'completed out of', 500, 'loss:', '0.373', 'accuracy:  ', 0.87621742)
('Epoch', 500, 'completed out of', 500, 'loss:', '0.374', 'accuracy:  ', 0.87621742)
('Accuracy', 0.87621742)
```

FUTURE WORK

- Implementation using Convolutional Neural Network.

THANK YOU