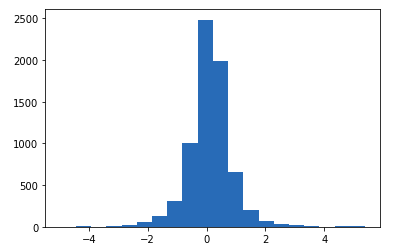
**Part I – Sentiment Analysis**

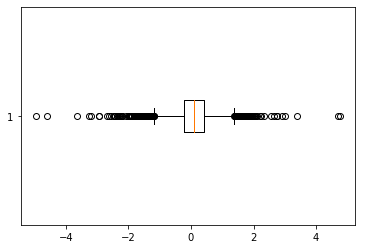
**Approach 1 – Word Count Vectorization**

**Report 1: Describe your word features and the classification model and report the accuracy of your model on the test data.**

* Describe your word features and the classification model and report the accuracy of your model on the test data.
* We have captured positive word list and negative word list from two files reviews.txt and labels.txt which correspond to features and labels respectively using the collection Counter library.
* We picked out words from review through split function and verified that they are in our vocabulary list and ignored others (i.e. ‘ ’, ’.’, ‘’)
* Using the word count greater than 100 we calculated the positive to negative ratio of each word.
* To get into the specified range we converted into the log values.
* To understand the data more clearly, we plotted the histogram and boxplot.
* Histogram



* Boxplot



* Through the plots, we understood that the deciding factor of the review was the use of strong (higher positive to negative ratio) positive words and strong (lower positive to negative ratio) negative words respectively.
* After verifying different percentiles range we came across the word feature which gave an accuracy of 77.1% to be in a range greater than 80 percentile and less than 10 percentiles as a threshold of positive and negative words respectively.

**Approach 2 - Neural Network-Based Sentiment Classification**

**Report 2: Report the accuracy and running time of the vanilla neural network. Is it better or worse than your rule-based algorithm implemented in Approach 1?**

* We have implemented ignore words function to compare the rule-based and machine learning approach.
* We have considered the same threshold (i.e. greater than 80 percentile and less than 10 percentile) to ignore words.
* We can observe that machine learning gives better accuracy than a rule-based approach. This is because machine learning can generalize the new data better, in case of a rule-based approach we need to add rules for every new data we see.
* Also, a machine learning-based approach is robust to outliers and anomalies. For instance, a statement like *“this is not a bad movie”* will be incorrectly classified as negative by a rule-based approach, whereas machine learning models can accurately classify these kinds of sentences due to its adaptability properties.

**Report 3: Report the performance of the model (both in terms of accuracy and time using different values for the hidden layer width, number of hidden layers, number of epochs and a set of ignored words. What has the most impact on performance?**

**With CPU as background**

Epochs:50, Weight=10, layer=1

Train acc: 0.904375, Test\_acc: 0.841250

Time elapsed - 22.347902297973633 seconds.

**With GPU as background**

**50 Epochs with different weight values**

Epochs=50, Weight=5, layer=1

Train acc: 0.920000, Test\_acc: 0.855000

Time elapsed - 13.598512172698975 seconds.

Epochs=50, Weight=10, layer=1

Train acc: 0.905042, Test\_acc: 0.851250

Time elapsed - 13.796868562698364 seconds.

Epochs=50, Weight=50, layer=1

Train acc: 0.907167, Test\_acc: 0.848750

Time elapsed - 14.32064151763916 seconds.

Epochs=50, Weight=100, layer=1

Train acc: 0.904333, Test\_acc: 0.838750

Time elapsed - 14.629113674163818 seconds.

**Performance impact with a different number of weights:** We can observe that with increasing weights the accuracy of the neural network decreases.

**100 Epochs with different weight values**

Epochs=100, Weight=5, layer=1

Train acc: 0.923625, Test\_acc: 0.852500

Time elapsed - 28.065927028656006 seconds.

Epochs=100, Weight=10, layer=1

Train acc: 0.906042, Test\_acc: 0.845000

Time elapsed - 28.596012353897095 seconds.

Epochs=100, Weight=50, layer=1

Train acc: 0.911333, Test\_acc: 0.841250

Time elapsed - 29.032610177993774 seconds.

Epochs=100, Weight=100, layer=1

Train acc: 0.909375, Test\_acc: 0.840000

Time elapsed - 29.917771577835083 seconds.

**Performance impact with a higher number of epochs:** We can observe that with an increased number of epochs the accuracy of the neural network decreases compared to the previous weights.

**50 Epochs with different hidden layers**

Epochs=50, Weight=10(h1->10->h2->10->out->2), layer=2

Train acc: 0.909292, Test\_acc: 0.853750

Time elapsed - 14.945465087890625 seconds.

Epochs=50, Weight=10(h1->10->h2->10->h3->10->out->2), layer=3

Train acc: 0.887417, Test\_acc: 0.817500

Time elapsed - 15.128944396972656 seconds.

Epochs=50, Weight(h1->10->h2->5->out->2), layer=2

Train acc: 0.910292, Test\_acc: 0.855000

Time elapsed - 14.639018297195435 seconds.

Epochs=50, Weight(h1->10->h2->10->h3->5->out->2), layer=3

Train acc: 0.899792, Test\_acc: 0.852500

Time elapsed - 15.199421882629395 seconds.

**Performance impact with a different number of hidden layers:** We can observe that with an increased number of hidden layers the accuracy of the neural network increases.

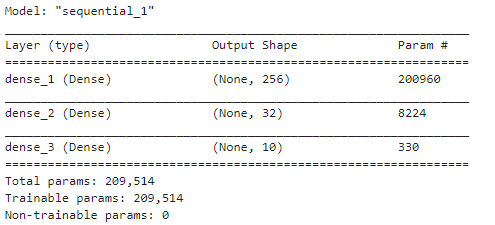
We tested with different intervals for ignore\_words function and picked the most optimum interval between 10 and 80 to ignore words and observed performance of 84%.

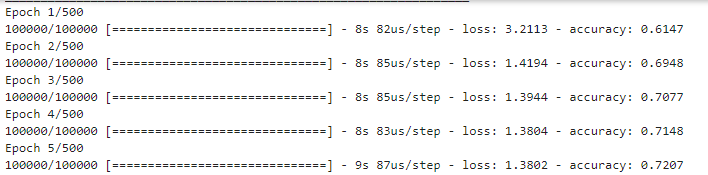
**Out of all the combinations, the maximum performance impact is observed with an increased number of hidden layers.**

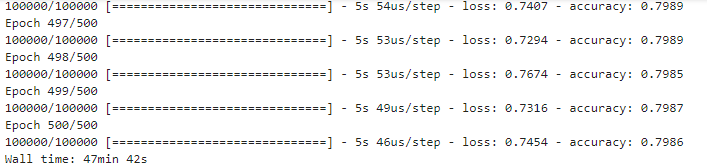
**Part II - Image Classification on the AI Quick Draw Dataset**

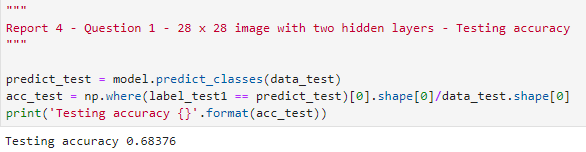
1. **Run the evaluation of the 2 hidden layers neural network in the notebook - PA2-Part2.ipynb and report the test accuracy and the run time.**

Model summary for 28 x 28 image size with 500 epochs and 2 hidden layers are listed below.





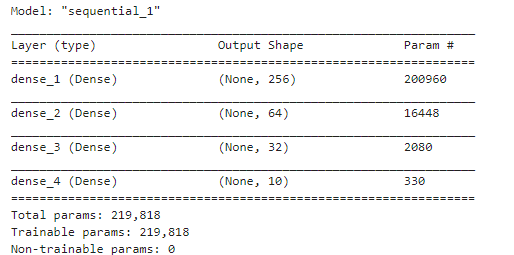


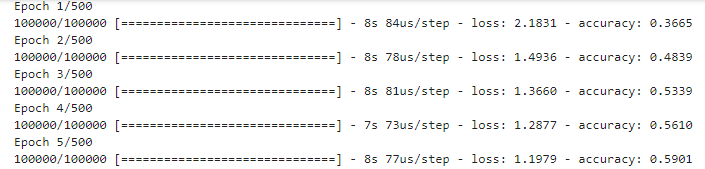
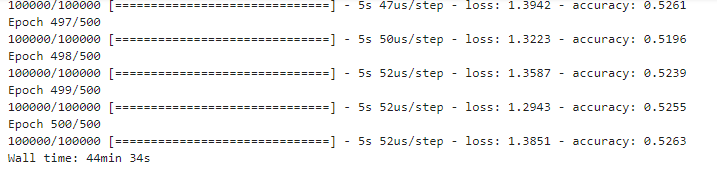


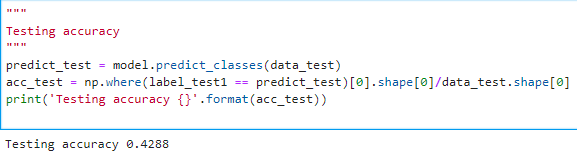
Training accuracy for the given model comes around *70% to 79%* as shown in the epochs figure above. Testing accuracy for the given model comes around *69%*. All the 500 epochs along with the accuracies, loss values are listed in part 2 of the Jupyter notebook.

Runtime for training the model is around *47 minutes 22 seconds* as shown in the figure above.

1. **Compare the performance when the number of hidden layers is increased to 3 and 5.**
2. Model summary for 28 x 28 image size with 500 epochs and 3 hidden layers are listed below.

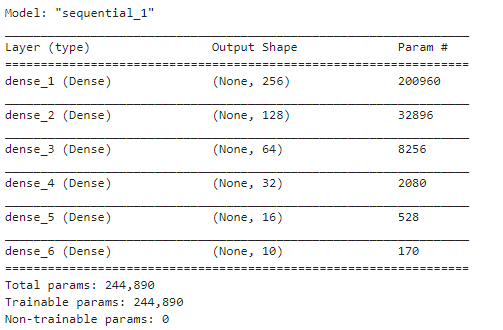


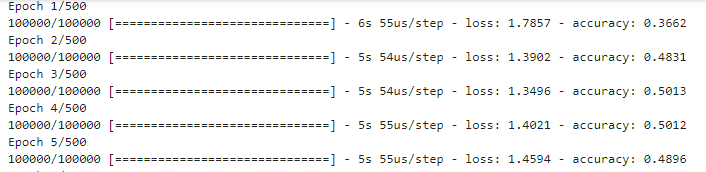


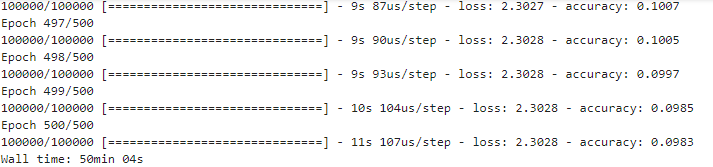
****

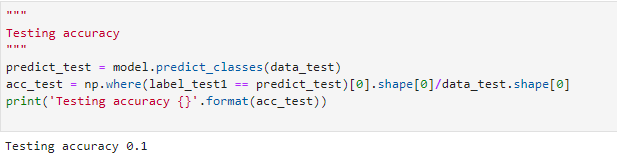
Training accuracy for the given model comes around *45% to 55%* as shown in the epochs figure above. Testing accuracy for the given model comes around *43%*. Runtime for training the model is around *44 minutes 34 seconds* as shown in the figure above.

1. Model summary for 28 x 28 image size with 500 epochs and 5 hidden layers are listed below.





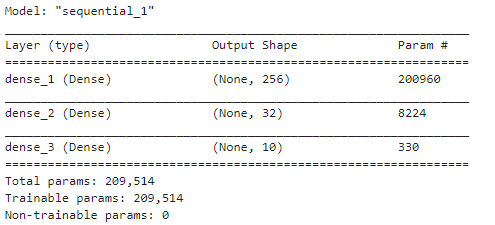


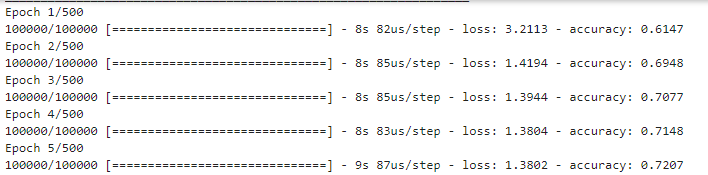


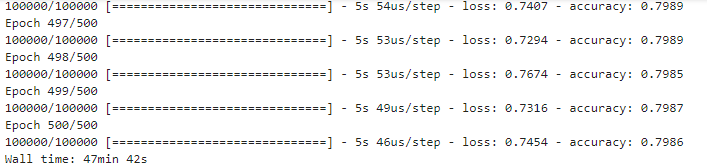
Training accuracy for the given model comes around *9% to 10%* as shown in the epochs figure above. Testing accuracy for the given model comes around *10%*. Runtime for training the model is around *50 minutes 04 seconds* as shown in the figure above.

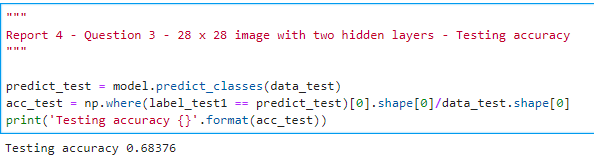
**Comparison between 3 and 5 hidden layers:** Usually, the performance of a neural network with a higher number of hidden gives accuracy. But here we can observe that the performance of the model with 3 hidden is higher than 5 hidden layers. This is because there is not enough training data on which the model with 5 hidden layers can generalize, which results in the underfitting of the model with lower accuracy.

1. **Use the resize images() function to reduce the resolution of the images to (20 x 20), (15 x 15), (10 x 10) and (5 x 5). Using the 2 hidden layer architecture, compare the performance at different resolutions, including the original (28 x 28) resolution, both in terms of test accuracy and time.**
2. Model summary for 28 x 28 image size with 500 epochs and 2 hidden layers are listed below.





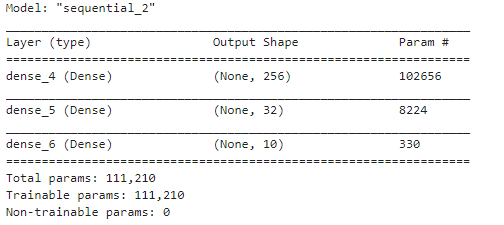


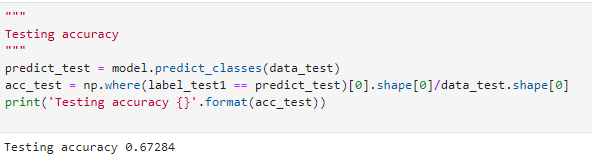
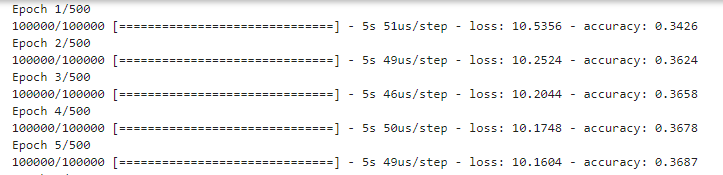
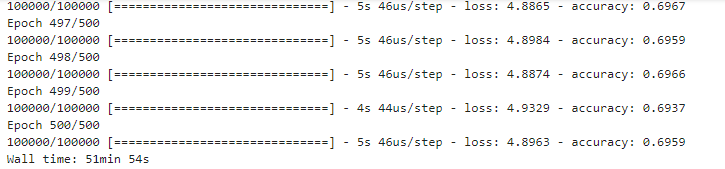


Training accuracy for the given model comes around *70% to 79%* as shown in the epochs figure above. Testing accuracy for the given model comes around *69%*. All the 500 epochs along with the accuracies, loss values are listed in part 2 of the Jupyter notebook.

Runtime for training the model is around *47 minutes 22 seconds* as shown in the figure above.

1. Model summary for 20 x 20 image size with 500 epochs and 2 hidden layers are listed below.

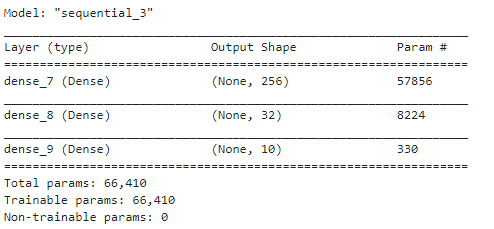


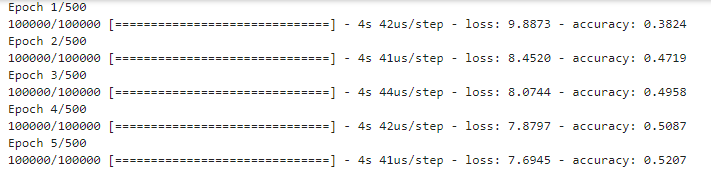
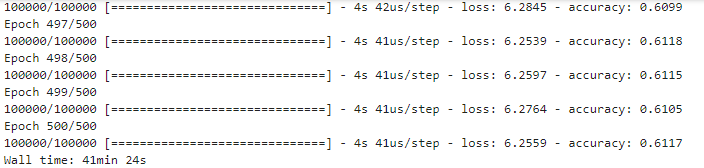
****

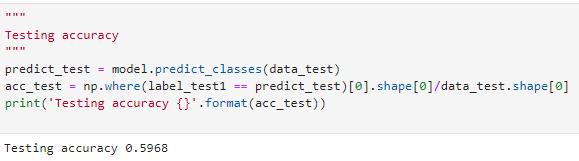
Training accuracy for the given model comes around 65*% to 70%* as shown in the epochs figure above. Testing accuracy for the given model comes around *67%*. Runtime for training the model is around *51 minutes 54 seconds* as shown in the figure above.

1. Model summary for 15 x 15 image size with 500 epochs and 2 hidden layers are listed below.

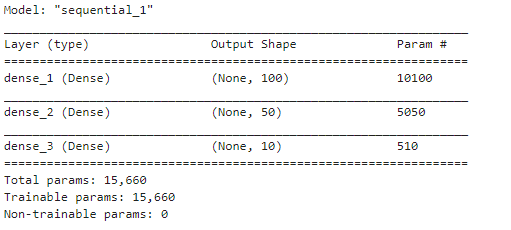
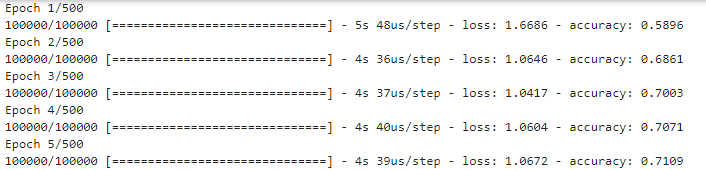
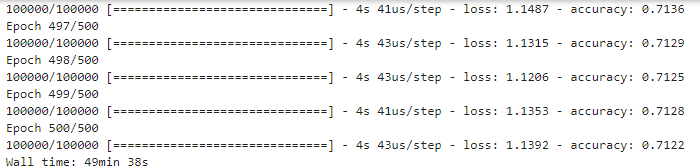
Training accuracy for the given model comes around 57*% to 62%* as shown in the epochs figure below. Testing accuracy for the given model comes around *60%*. Runtime for training the model is around *41 minutes 24 seconds* as shown in the figure above.

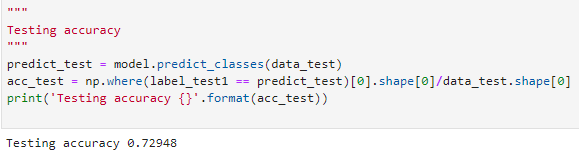
****

****

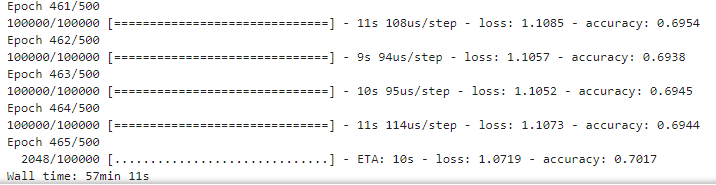
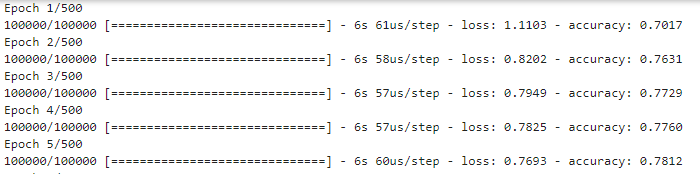
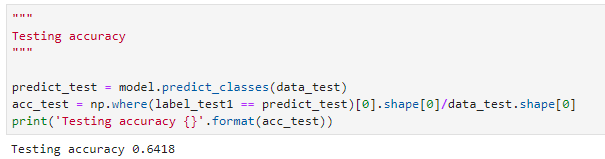
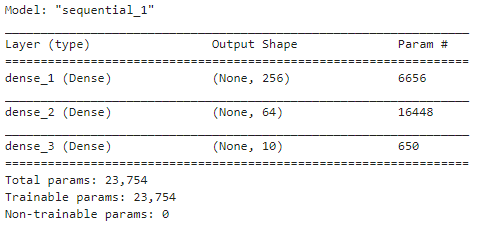
****

1. Model summary for 10 x 10 image size with 500 epochs and 2 hidden layers are listed below.

****

****

Training accuracy for the given model comes around *71% to 76%* as shown in the epochs figure above. Testing accuracy for the given model comes around *73%*. Runtime for training the model is around *49 minutes 38 seconds* as shown in the figure above.

1. Model summary for 5 x 5 image size with 500 epochs and 2 hidden layers are listed below.

Training accuracy for the given model comes around *63% to 69%* as shown in the epochs figure above. Testing accuracy for the given model comes around *64%*. Runtime for training the model is around *57 minutes 11 seconds* as shown in the figure above.

**Comparison between different image resolutions:** We can observe that there is no much difference between the image resolutions (28 x 28), (20 x 20), (15 x 15) size images. This is because the model doesn’t change much with minimal resolution changes. But as we further decrease the resolution of the images, we see a steep decrease in performance because the image properties fade away with a decrease in size.