

Deep Learning: Dog Vs Cat Classification using CNN

Objective: Implement a three layer CNN network, for classification tasks for the Dogs vs. Cats dataset.

- a. Compare the accuracy on the test dataset (split into train and test [70:30]) for the following optimization techniques:
 - i. Vanilla SGD
 - ii. Mini Batch SGD
 - iii. Mini Batch with momentum
 - iv. Mini Batch with Adam

Link of Google Colab Code File:

<https://colab.research.google.com/drive/1znn95DC4D4ModCw8oje8uEd2UPZgZnGZ?usp=sharing>

Steps Involved:

Due to computation constraints, I have used only 6252 pet images for training and testing my model, even though the original image data folder consists of a total of 25006 images.

1. Dataset Used: PetImages dataset.

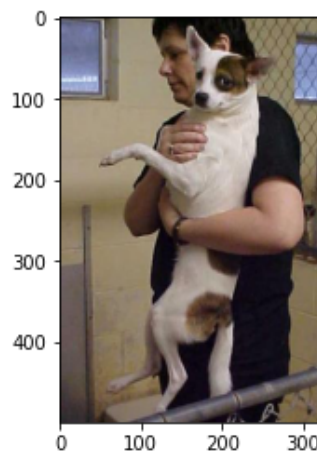
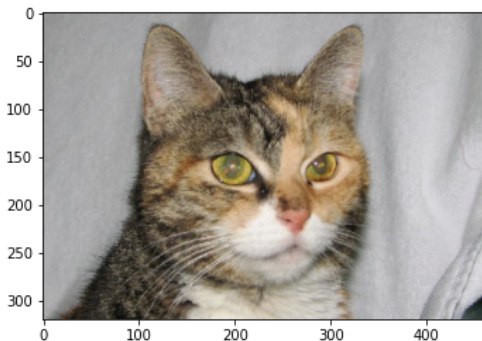
The modified dataset (due to computation constraints) contains

- 6252 images
- 2 classes (Cat and Dog).
- Images are of variable size and both coloured and grayscale.
- Class Cat consists of 3125 Cat Images and 3127 Dog Images.

2. Data Preprocessing:

2.1. Due to computation restraints I have used the first 6252 data images for our analysis.

2.2. Data Visualization



2.3. Train Test Split in the ratio 70:30

Total Data Size	Train data size	Test Data Size
6252	4376	1876

2.4. Removed Grayscale Images from the dataset.

```

Train Dataset Size:  4376
Length of trainImageLoader 4362
Removed grayscale image count in train data:  14

Test Dataset Size:  1876
Length of testImageLoader 1873
Removed grayscale image count in test data:  3

```

2.5. Transformed the data images with following operations

- `Resize((400, 400))`
- `Convert to Tensor`
- `Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5])`

2.6. Visualizing a batch of data

2.7. Moved data and labels to GPU if available.

3. Created a three layer CNN Network

```

-----
Layer (type)              Output Shape          Param #
-----
Conv2d-1                  [-1, 64, 222, 222]    4,864
BatchNorm2d-2             [-1, 64, 222, 222]    128
ReLU-3                    [-1, 64, 222, 222]     0
MaxPool2d-4               [-1, 64, 44, 44]       0
Conv2d-5                  [-1, 128, 44, 44]     73,856
ReLU-6                    [-1, 128, 44, 44]       0
Conv2d-7                  [-1, 256, 42, 42]    819,456
BatchNorm2d-8             [-1, 256, 42, 42]     512
ReLU-9                    [-1, 256, 42, 42]       0
AdaptiveAvgPool1d-10      [-1, 2, 256]           0
Linear-11                  [-1, 128]              32,896
Linear-12                  [-1, 64]               8,256
=====
Total params: 939,968
Trainable params: 939,968
Non-trainable params: 0
-----
Input size (MB): 0.57
Forward/backward pass size (MB): 87.26
Params size (MB): 3.59
Estimated Total Size (MB): 91.42
-----

```

Trained the model with different optimization techniques by varying the batch sizes and reported the observation as below

4. Observation:

i. Vanilla SGD:

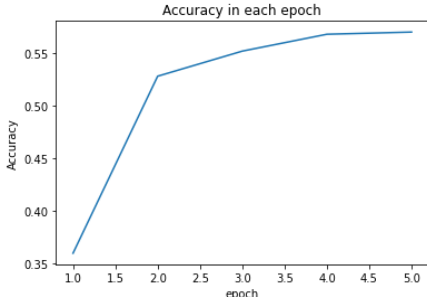
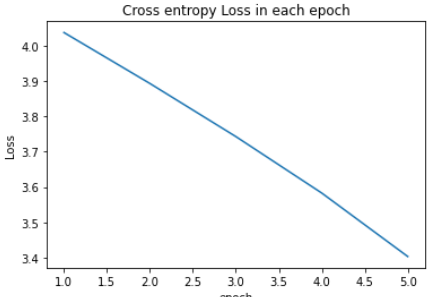
Out of Memory memory error was encountered while implementing Vanilla SGD due to computational constraints.

ii. Mini Batch SGD:

a) Batch Size =16

Train Dataset Size	Batch Size	Size of Train Data Loader	Number of Epochs	Learning Rate	Loss Function
4362	16	273	5	0.0001	CrossEntropy

```
Epoch: 0 Loss :4.038207085956385 Accuracy :36.0146252285192%
Epoch: 1 Loss :3.893679199750506 Accuracy :52.78793418647167%
Epoch: 2 Loss :3.7430718009389077 Accuracy :55.16453382084096%
Epoch: 3 Loss :3.582787113499162 Accuracy :56.764168190127975%
Epoch: 4 Loss :3.4028411572986372 Accuracy :56.96983546617916%
Finished Training
```

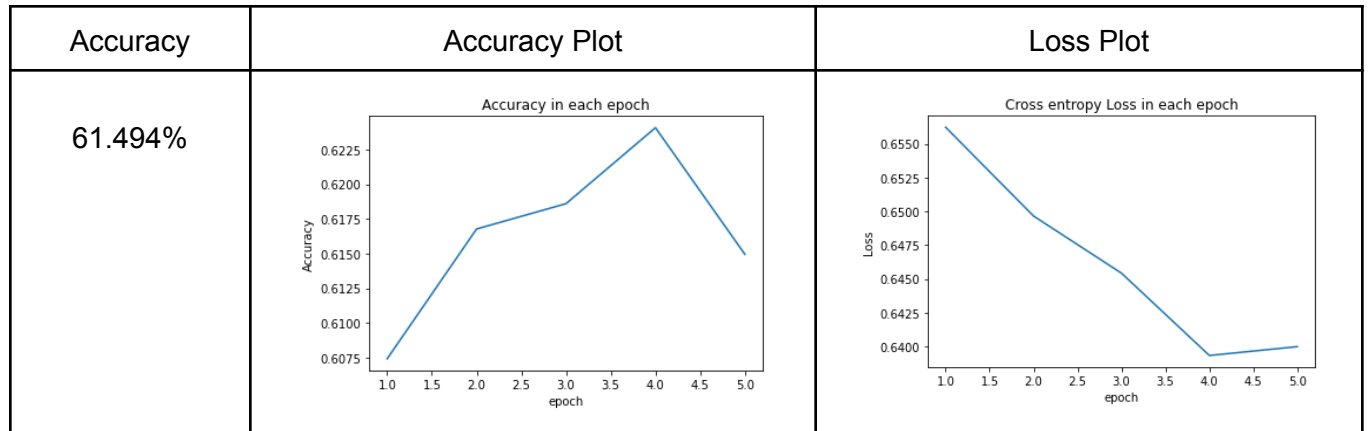
Accuracy	Accuracy Plot	Loss Plot
56.969%	 <p>Accuracy in each epoch</p>	 <p>Cross entropy Loss in each epoch</p>

Test Dataset Size	Test Accuracy	Accuracy of Class Cat	Accuracy of Class Dog
1873	57.0%	90.85%	22.91%

b) Batch Size =4

Train Dataset Size	Batch Size	Size of Train Data Loader	Number of Epochs	Learning Rate	Loss Function
4362	4	1091	5	0.001	CrossEntropy

Epoch: 0 Loss :0.6562260785421026 Accuracy :60.740402193784284%
 Epoch: 1 Loss :0.6496540060917246 Accuracy :61.677330895795244%
 Epoch: 2 Loss :0.6454252779102413 Accuracy :61.8601462522852%
 Epoch: 3 Loss :0.6393315321893535 Accuracy :62.40859232175503%
 Epoch: 4 Loss :0.6399846150790097 Accuracy :61.494515539305304%
 Finished Training



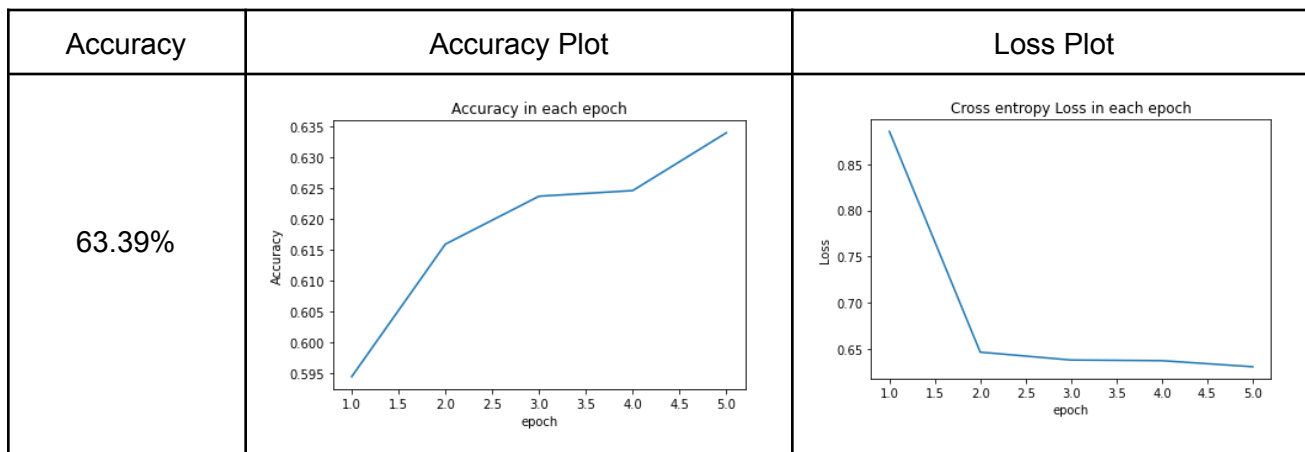
Test Dataset Size	Test Accuracy	Accuracy of class Cat	Accuracy of class Dog
1873	61.0%	88.95%	34.06%

iii) Mini Batch with Momentum

a) Batch Size =16

Train Dataset Size	Batch Size	Size of Train Data Loader	Number of Epochs	Learning Rate	Momentum	Loss Function
4362	16	273	5	0.001	0.9	CrossEntropy

Epoch: 0 Loss :0.8855152101250846 Accuracy :59.43784277879342%
 Epoch: 1 Loss :0.6462729209846727 Accuracy :61.585923217550274%
 Epoch: 2 Loss :0.6377317527312465 Accuracy :62.36288848263254%
 Epoch: 3 Loss :0.6370411593964157 Accuracy :62.45429616087751%
 Epoch: 4 Loss :0.6304357312279166 Accuracy :63.391224862888485%
 Finished Training



Test Dataset Size	Test Accuracy	Accuracy of Class Cat	Accuracy of Class Dog
1873	61.0%	91.16%	31.24%

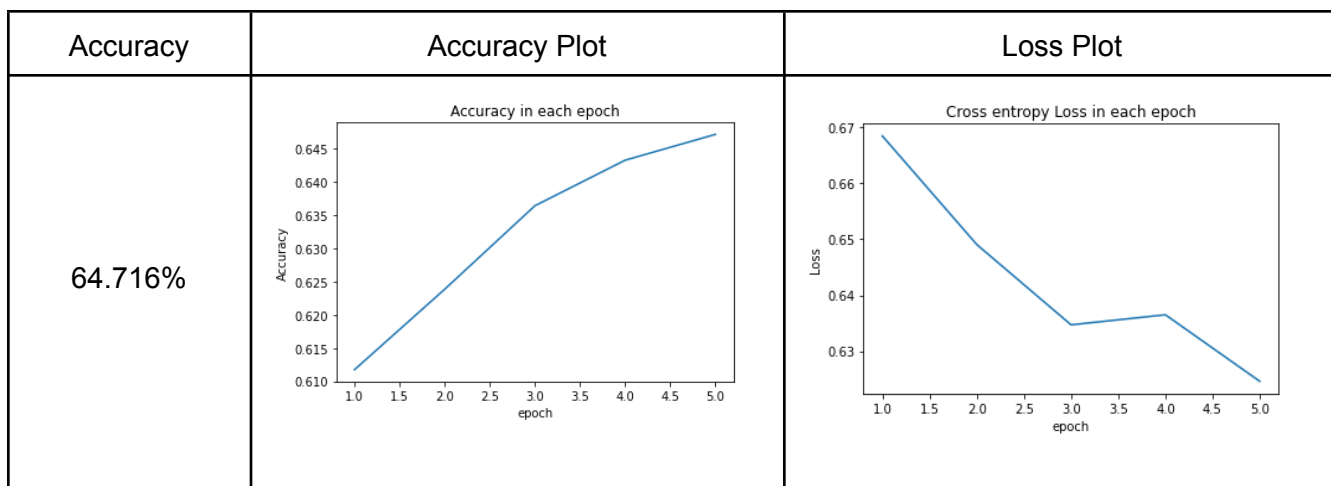
a) Batch Size =4

Train Dataset Size	Batch Size	Size of Train Data Loader	Number of Epochs	Learning Rate	Momentum	Loss Function
4362	4	1091	5	0.001	0.9	CrossEntropy

```

Epoch: 0 Loss :0.6684068574738676 Accuracy :61.17458866544789%
Epoch: 1 Loss :0.6490452413258213 Accuracy :62.385740402193775%
Epoch: 2 Loss :0.634709231623566 Accuracy :63.642595978062154%
Epoch: 3 Loss :0.6365009682351754 Accuracy :64.32815356489945%
Epoch: 4 Loss :0.6246256322158974 Accuracy :64.71663619744058%
Finished Training

```

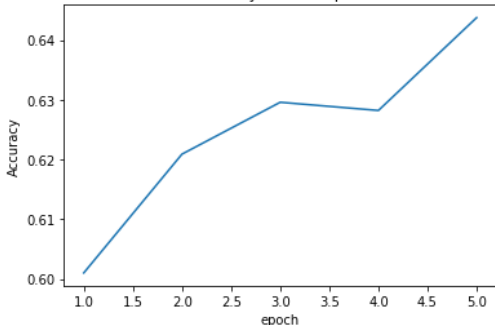
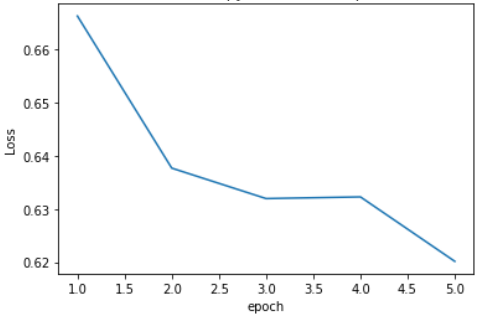


Test Dataset Size	Test Accuracy	Accuracy of class Cat	Accuracy of class Dog
1873	68.0%	85.74%	49.78%

iv) Mini Batch with Adam

a) Batch Size =16

Train Dataset Size	Batch Size	Size of Train Data Loader	Number of Epochs	Learning Rate	Loss Function
4362	16	273	5	0.001	CrossEntropy

Accuracy	Accuracy Plot	Loss Plot																								
64.373%	<p>Accuracy in each epoch</p>  <table><thead><tr><th>epoch</th><th>Accuracy</th></tr></thead><tbody><tr><td>1.0</td><td>0.601</td></tr><tr><td>2.0</td><td>0.621</td></tr><tr><td>3.0</td><td>0.629</td></tr><tr><td>4.0</td><td>0.628</td></tr><tr><td>5.0</td><td>0.644</td></tr></tbody></table>	epoch	Accuracy	1.0	0.601	2.0	0.621	3.0	0.629	4.0	0.628	5.0	0.644	<p>Cross entropy Loss in each epoch</p>  <table><thead><tr><th>epoch</th><th>Loss</th></tr></thead><tbody><tr><td>1.0</td><td>0.666</td></tr><tr><td>2.0</td><td>0.638</td></tr><tr><td>3.0</td><td>0.632</td></tr><tr><td>4.0</td><td>0.632</td></tr><tr><td>5.0</td><td>0.620</td></tr></tbody></table>	epoch	Loss	1.0	0.666	2.0	0.638	3.0	0.632	4.0	0.632	5.0	0.620
epoch	Accuracy																									
1.0	0.601																									
2.0	0.621																									
3.0	0.629																									
4.0	0.628																									
5.0	0.644																									
epoch	Loss																									
1.0	0.666																									
2.0	0.638																									
3.0	0.632																									
4.0	0.632																									
5.0	0.620																									

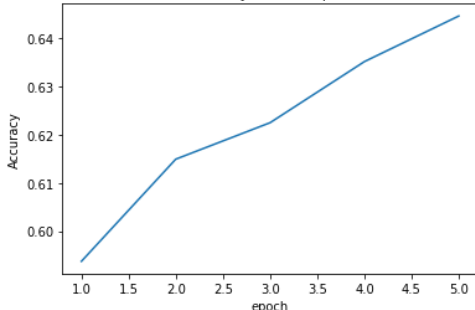
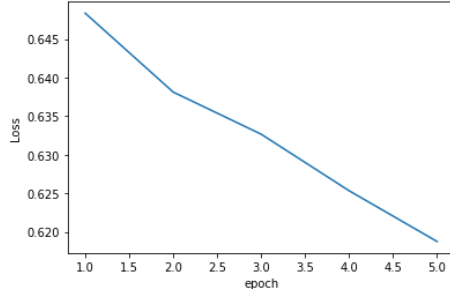
```
Epoch: 0 Loss :0.6661846342154352 Accuracy :60.10054844606947%
Epoch: 1 Loss :0.6376538970516195 Accuracy :62.088665447897625%
Epoch: 2 Loss :0.6319416199131466 Accuracy :62.95703839122486%
Epoch: 3 Loss :0.6322567548732217 Accuracy :62.8199268738574%
Epoch: 4 Loss :0.6201707428998877 Accuracy :64.37385740402193%
Finished Training
```

Test Dataset Size	Test Accuracy	Accuracy of Class Cat	Accuracy of Class Dog
1873	65.0%	90.2%	41.7%

a) Batch Size =4

Train Dataset Size	Batch Size	Size of Train Data Loader	Number of Epochs	Learning Rate	Loss Function
4362	4	1091	5	0.0001	CrossEntropy

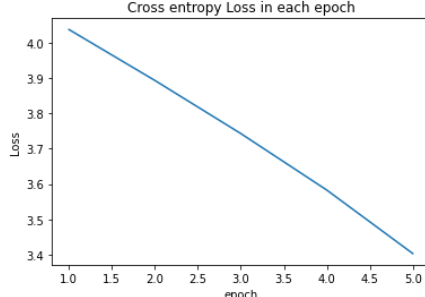
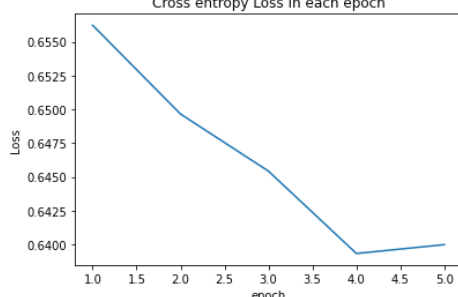
```
Epoch: 0 Loss :0.6483499389357112 Accuracy :59.37085465226455%
Epoch: 1 Loss :0.6381288223562988 Accuracy :61.49327269281789%
Epoch: 2 Loss :0.6326670775432489 Accuracy :62.25127913587265%
Epoch: 3 Loss :0.6253379553991306 Accuracy :63.52093992798938%
Epoch: 4 Loss :0.6187587383345986 Accuracy :64.46844798180784%
Finished Training
```

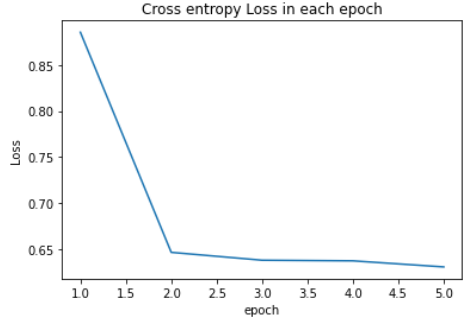
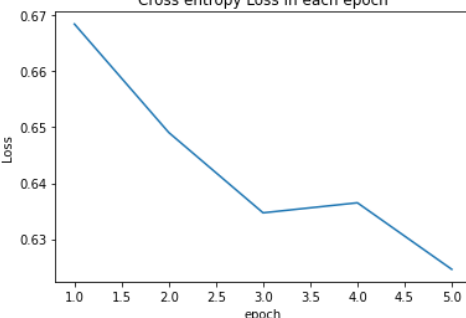
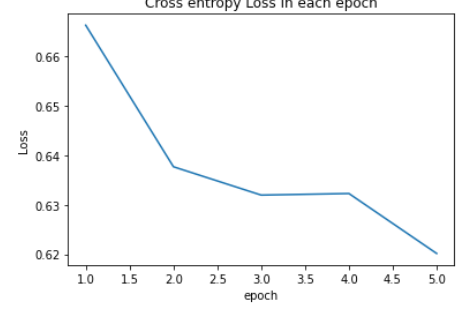
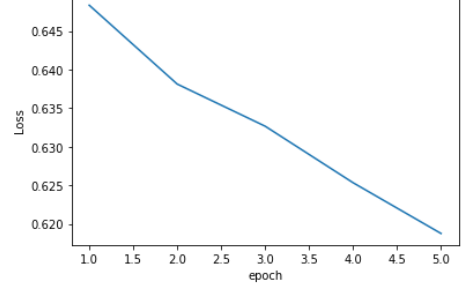
Accuracy	Accuracy Plot	Loss Plot																								
64.468%	<p>Accuracy in each epoch</p>  <table border="1"><thead><tr><th>epoch</th><th>Accuracy</th></tr></thead><tbody><tr><td>1.0</td><td>0.600</td></tr><tr><td>2.0</td><td>0.615</td></tr><tr><td>3.0</td><td>0.622</td></tr><tr><td>4.0</td><td>0.635</td></tr><tr><td>5.0</td><td>0.645</td></tr></tbody></table>	epoch	Accuracy	1.0	0.600	2.0	0.615	3.0	0.622	4.0	0.635	5.0	0.645	<p>Cross entropy Loss in each epoch</p>  <table border="1"><thead><tr><th>epoch</th><th>Loss</th></tr></thead><tbody><tr><td>1.0</td><td>0.645</td></tr><tr><td>2.0</td><td>0.638</td></tr><tr><td>3.0</td><td>0.632</td></tr><tr><td>4.0</td><td>0.625</td></tr><tr><td>5.0</td><td>0.620</td></tr></tbody></table>	epoch	Loss	1.0	0.645	2.0	0.638	3.0	0.632	4.0	0.625	5.0	0.620
epoch	Accuracy																									
1.0	0.600																									
2.0	0.615																									
3.0	0.622																									
4.0	0.635																									
5.0	0.645																									
epoch	Loss																									
1.0	0.645																									
2.0	0.638																									
3.0	0.632																									
4.0	0.625																									
5.0	0.620																									

Test Dataset Size	Test Accuracy	Accuracy of Class Cat	Accuracy of Class Dog
1873	65.0%	56.06%	73.27%

b. What are your preferred mini batch sizes? Explain your choice with proper gradient update plots.

The preferred mini batch sizes used were 4 and 16. Since I have reduced my image dataset from 25006 to 6252 due to computation and GPU allocation issues. When I tried to increase batch size to 64 it was throwing CUDA out of memory error and the reason for reducing the dataset was because it was taking a lot of time in image preprocessing.

Optimization Technique	Batch Size	Plot												
Mini Batch SGD	16	 <p>Cross entropy Loss in each epoch</p> <table><thead><tr><th>epoch</th><th>Loss</th></tr></thead><tbody><tr><td>1.0</td><td>4.00</td></tr><tr><td>2.0</td><td>3.85</td></tr><tr><td>3.0</td><td>3.70</td></tr><tr><td>4.0</td><td>3.55</td></tr><tr><td>5.0</td><td>3.40</td></tr></tbody></table>	epoch	Loss	1.0	4.00	2.0	3.85	3.0	3.70	4.0	3.55	5.0	3.40
epoch	Loss													
1.0	4.00													
2.0	3.85													
3.0	3.70													
4.0	3.55													
5.0	3.40													
Mini Batch SGD	4	 <p>Cross entropy Loss in each epoch</p> <table><thead><tr><th>epoch</th><th>Loss</th></tr></thead><tbody><tr><td>1.0</td><td>0.655</td></tr><tr><td>2.0</td><td>0.650</td></tr><tr><td>3.0</td><td>0.645</td></tr><tr><td>4.0</td><td>0.640</td></tr><tr><td>5.0</td><td>0.640</td></tr></tbody></table>	epoch	Loss	1.0	0.655	2.0	0.650	3.0	0.645	4.0	0.640	5.0	0.640
epoch	Loss													
1.0	0.655													
2.0	0.650													
3.0	0.645													
4.0	0.640													
5.0	0.640													

Mini Batch SGD with Momentum	16	 <table border="1"><thead><tr><th>epoch</th><th>Loss</th></tr></thead><tbody><tr><td>1.0</td><td>0.88</td></tr><tr><td>2.0</td><td>0.65</td></tr><tr><td>3.0</td><td>0.645</td></tr><tr><td>4.0</td><td>0.645</td></tr><tr><td>5.0</td><td>0.64</td></tr></tbody></table>	epoch	Loss	1.0	0.88	2.0	0.65	3.0	0.645	4.0	0.645	5.0	0.64
epoch	Loss													
1.0	0.88													
2.0	0.65													
3.0	0.645													
4.0	0.645													
5.0	0.64													
Mini Batch SGD with Momentum	4	 <table border="1"><thead><tr><th>epoch</th><th>Loss</th></tr></thead><tbody><tr><td>1.0</td><td>0.668</td></tr><tr><td>2.0</td><td>0.648</td></tr><tr><td>3.0</td><td>0.635</td></tr><tr><td>4.0</td><td>0.637</td></tr><tr><td>5.0</td><td>0.625</td></tr></tbody></table>	epoch	Loss	1.0	0.668	2.0	0.648	3.0	0.635	4.0	0.637	5.0	0.625
epoch	Loss													
1.0	0.668													
2.0	0.648													
3.0	0.635													
4.0	0.637													
5.0	0.625													
Mini Batch Adam	16	 <table border="1"><thead><tr><th>epoch</th><th>Loss</th></tr></thead><tbody><tr><td>1.0</td><td>0.665</td></tr><tr><td>2.0</td><td>0.638</td></tr><tr><td>3.0</td><td>0.632</td></tr><tr><td>4.0</td><td>0.632</td></tr><tr><td>5.0</td><td>0.62</td></tr></tbody></table>	epoch	Loss	1.0	0.665	2.0	0.638	3.0	0.632	4.0	0.632	5.0	0.62
epoch	Loss													
1.0	0.665													
2.0	0.638													
3.0	0.632													
4.0	0.632													
5.0	0.62													
Mini Batch Adam	4	 <table border="1"><thead><tr><th>epoch</th><th>Loss</th></tr></thead><tbody><tr><td>1.0</td><td>0.648</td></tr><tr><td>2.0</td><td>0.638</td></tr><tr><td>3.0</td><td>0.632</td></tr><tr><td>4.0</td><td>0.625</td></tr><tr><td>5.0</td><td>0.618</td></tr></tbody></table>	epoch	Loss	1.0	0.648	2.0	0.638	3.0	0.632	4.0	0.625	5.0	0.618
epoch	Loss													
1.0	0.648													
2.0	0.638													
3.0	0.632													
4.0	0.625													
5.0	0.618													

c. What are the advantages of shuffling and partitioning the mini batches?

- It helps the training to converge fast.
- Shuffling reduces variance and making sure that models remain general and overfit less.
- It prevents any bias during the training.
- It prevents the model from learning the order of the training.
- It is computationally efficient.
- Average of the training samples produces stable error gradients and convergence using mini batches.

d. Are there any advantages of using Adam over the other optimization methods?

- Straightforward to implement.
- Computationally efficient.
- Little memory requirements.
- Invariant to diagonal rescale of the gradients.
- Adam can take advantage of sparse features and obtain a faster convergence rate than normal SGD with momentum.