CSCE 636: Deep Learning Project Report

Anmol Anand

1 Proposed model

For this project, I have used a variant of ResNet that leads to a precise and a time efficient training model. The model achieves 93.56% accuracy in just 90 epochs. The major insight from this implementation is that when the loss starts to plateau, increasing the mini-batch size can lead to considerable reduction in the loss without even changing the learning rate.

2 Implementation details

Network specifications: A ResNet variant with the following specifications was used:

Layer	*Specifications*	*Output dimensions*
start layer	*Conv: kernel* = 3×3, *stride* = 1, *filters* = 64	*Spatial dim* = 32×32 *Channels* = 64
first stack layer	$6\times$ *standard ResNet blocks with shortcut connections*	*Spatial dim* = 32×32 *Channels* = 64
second stack layer	$6\times$ *standard ResNet blocks with shortcut connections*	*Spatial dim* = 16×16 *Channels* = 128
third stack layer	$6\times$ *standard ResNet blocks with shortcut connections*	*Spatial dim* = 8×8 *Channels* = 256
output layer	*Average pool: kernel* = 2×2, *stride* = 2 *Fully connected: out channels* = 2048 *Sigmoid* *Fully connected: out channels* = 10	*Linear dim* = 10

Every Convolutional layer is followed by *BatchNormalization (epsilon* = 10^{-5}, *momentum* = 0.997*)* and *LeakyReLU (negative_slope* = 0.01*)*.

Training hyperparameters: Cross entropy loss was optimized using stochastic gradient descent with the following hyperparameters:

Hyperparameter	*Value*
num_epochs	90
$weight\_decay(L2\_regularization)$	$2 \cdot 10^{-4}$
$mini\_batch\_size$	125 (epochs 1-60) 250 (epochs 61-70) 500 (epochs 71-80) 1000 (epochs 81-90)
$learning\_rate$	0.1 (throughout the 90 epochs)

Training was conducted using the entire training set of 50,000 samples.
Validation/evaluation of this model was conducted using the public test set of 10,000 samples.

Data preprocessing and augmentation:

In every epoch, a random augmentation of the original sample image is generated by applying the following operations independently. This augmentation is then used in this training epoch.

- Add a 4 pixel wide white border on all four sides of the original sample and then crop a random 32×32 image.
- With half probability, flip the sample horizontally.

Apart from the augmentations, we normalize each color channel of each pixel based on the mean and variance of that color channel over all pixels.

3 Observations

Increasing the mini batch size is far more effective than decreasing the learning rate whenever training loss starts to plateau:

While training this ResNet variant, I observed that the loss plateaus while approaching the end of 60 epochs with a prediction accuracy of 88%. Decreasing the learning rate did not help. But, doubling the batch size from 125 to 250 considerably decreased the loss further and consequently increased the prediction accuracy. I followed this approach of doubling the batch size whenever the loss started to plateau and it resulted in a 93.56% prediction accuracy at the end of 90 epochs. In the entire training process, the learning rate was 0.1 and decreasing the learning rate when loss started to plateau was not as effective as increasing the batch size.

Reason: It is beneficial to use a smaller batch size for the initial phase of training because the loss converges faster because of more iterations. In the later phase of training it is beneficial to use larger batch sizes because it leads to more accurate convergence. The latency of an epoch is the same irrespective of the batch size as we process the same number of samples. However, the runtime memory required is higher for larger batch sizes. This is the reason why I could not double the batch size after the 90^{th} epoch as my machine did not have enough runtime memory to train on mini batches with more than 1000 samples. If it were possible to train on larger batch sizes, I expect that the prediction accuracy could be increased even more.

Other deviations from the standard ResNet model that I have adopted in my model that helped in faster and more accurate convergence are:

- 64 filters are used in the first convolutional layer which increses to 128 and then to 256 in subsequent stack layers.
- ResNet size (number of blocks in a stack layer) is set to 6.
- The activation function used in the entire network is LeakyReLU. Compared to ReLU and ELU, LeakyReLU seemed to work best for this configuration of the model.
- The output layer has two fully connected layers.

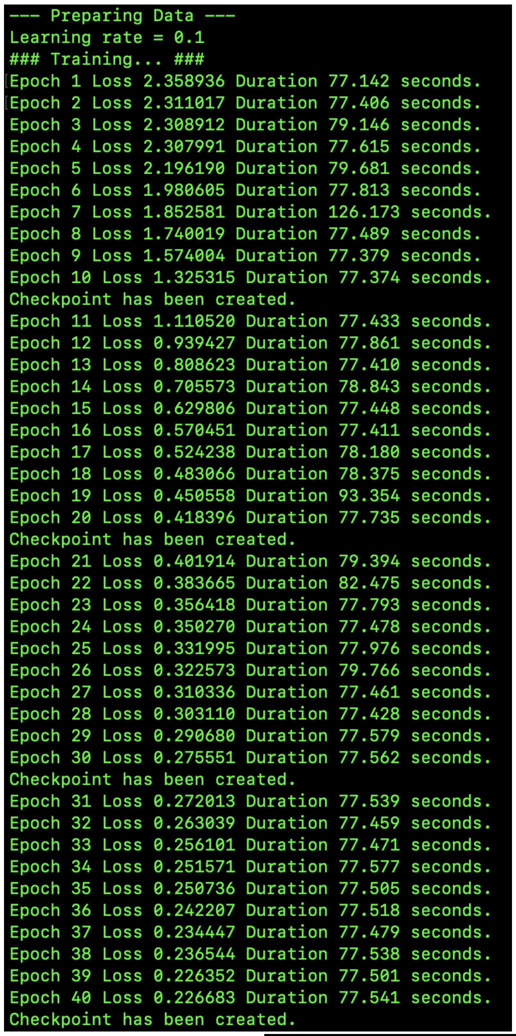
4 Results

For epochs 1 to 90, the chosen mini-batch size and the accuracy (on public test set) are shown in the table below:

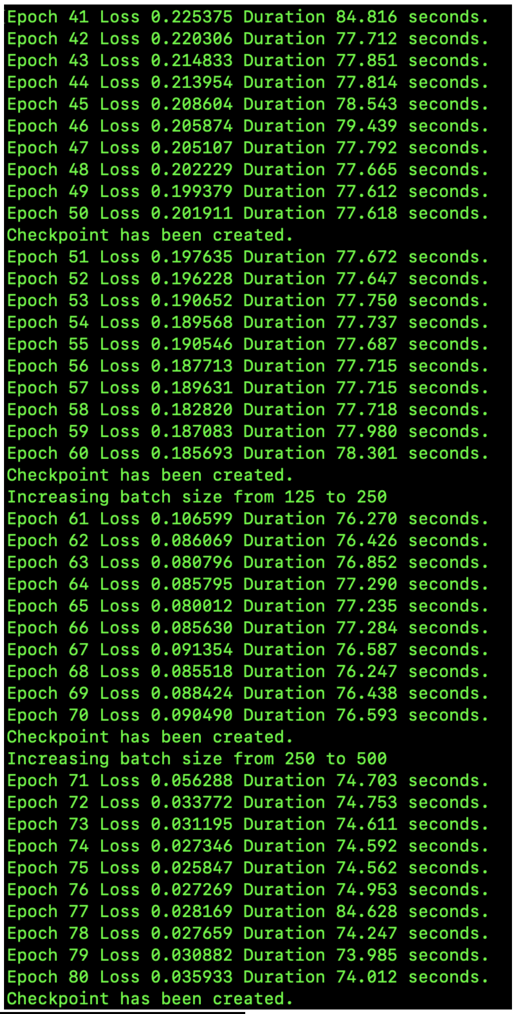
Epoch range ('l' to 'r')	*Mini batch size*	*Test set accuracy (at the end of 'r' epochs)*
1 *to* 30	125	86.41%
31 *to* 60	125	88.31%
61 *to* 70	250	90.48%
71 *to* 80	500	91.78%
81 *to* 90	1000	93.56%

The final accuracy on the public test set after 90 epochs came out to be 93.56%.

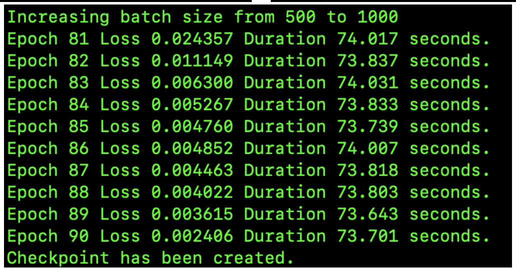
5 Logs



```
--- Preparing Data ---
Learning rate = 0.1
### Training... ###
Epoch 1 Loss 2.358936 Duration 77.142 seconds.
Epoch 2 Loss 2.311017 Duration 77.406 seconds.
Epoch 3 Loss 2.308912 Duration 79.146 seconds.
Epoch 4 Loss 2.307991 Duration 77.615 seconds.
Epoch 5 Loss 2.196190 Duration 79.681 seconds.
Epoch 6 Loss 1.980605 Duration 77.813 seconds.
Epoch 7 Loss 1.852581 Duration 126.173 seconds.
Epoch 8 Loss 1.740019 Duration 77.489 seconds.
Epoch 9 Loss 1.574004 Duration 77.379 seconds.
Epoch 10 Loss 1.325315 Duration 77.374 seconds.
Checkpoint has been created.
Epoch 11 Loss 1.110520 Duration 77.433 seconds.
Epoch 12 Loss 0.939427 Duration 77.861 seconds.
Epoch 13 Loss 0.808623 Duration 77.410 seconds.
Epoch 14 Loss 0.705573 Duration 78.843 seconds.
Epoch 15 Loss 0.629806 Duration 77.448 seconds.
Epoch 16 Loss 0.570451 Duration 77.411 seconds.
Epoch 17 Loss 0.524238 Duration 78.180 seconds.
Epoch 18 Loss 0.483066 Duration 78.375 seconds.
Epoch 19 Loss 0.450558 Duration 93.354 seconds.
Epoch 20 Loss 0.418396 Duration 77.735 seconds.
Checkpoint has been created.
Epoch 21 Loss 0.401914 Duration 79.394 seconds.
Epoch 22 Loss 0.383665 Duration 82.475 seconds.
Epoch 23 Loss 0.356418 Duration 77.793 seconds.
Epoch 24 Loss 0.350270 Duration 77.478 seconds.
Epoch 25 Loss 0.331995 Duration 77.976 seconds.
Epoch 26 Loss 0.322573 Duration 79.766 seconds.
Epoch 27 Loss 0.310336 Duration 77.461 seconds.
Epoch 28 Loss 0.303110 Duration 77.428 seconds.
Epoch 29 Loss 0.290680 Duration 77.579 seconds.
Epoch 30 Loss 0.275551 Duration 77.562 seconds.
Checkpoint has been created.
Epoch 31 Loss 0.272013 Duration 77.539 seconds.
Epoch 32 Loss 0.263039 Duration 77.459 seconds.
Epoch 33 Loss 0.256101 Duration 77.471 seconds.
Epoch 34 Loss 0.251571 Duration 77.577 seconds.
Epoch 35 Loss 0.250736 Duration 77.505 seconds.
Epoch 36 Loss 0.242207 Duration 77.518 seconds.
Epoch 37 Loss 0.234447 Duration 77.479 seconds.
Epoch 38 Loss 0.236544 Duration 77.538 seconds.
Epoch 39 Loss 0.226352 Duration 77.501 seconds.
Epoch 40 Loss 0.226683 Duration 77.541 seconds.
Checkpoint has been created.
```

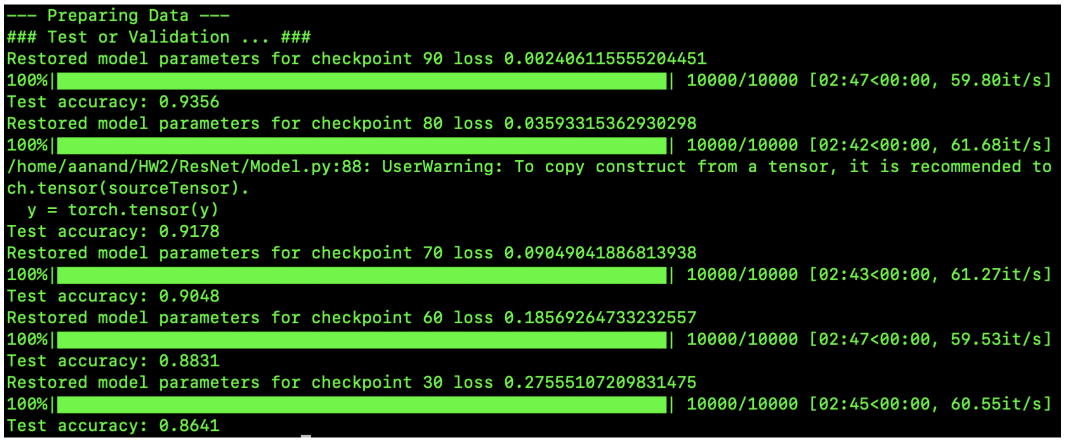


```
Epoch 41 Loss 0.225375 Duration 84.816 seconds.
Epoch 42 Loss 0.220306 Duration 77.712 seconds.
Epoch 43 Loss 0.214833 Duration 77.851 seconds.
Epoch 44 Loss 0.213954 Duration 77.814 seconds.
Epoch 45 Loss 0.208604 Duration 78.543 seconds.
Epoch 46 Loss 0.205874 Duration 79.439 seconds.
Epoch 47 Loss 0.205107 Duration 77.792 seconds.
Epoch 48 Loss 0.202229 Duration 77.665 seconds.
Epoch 49 Loss 0.199379 Duration 77.612 seconds.
Epoch 50 Loss 0.201911 Duration 77.618 seconds.
Checkpoint has been created.
Epoch 51 Loss 0.197635 Duration 77.672 seconds.
Epoch 52 Loss 0.196228 Duration 77.647 seconds.
Epoch 53 Loss 0.190652 Duration 77.750 seconds.
Epoch 54 Loss 0.189568 Duration 77.737 seconds.
Epoch 55 Loss 0.190546 Duration 77.687 seconds.
Epoch 56 Loss 0.187713 Duration 77.715 seconds.
Epoch 57 Loss 0.189631 Duration 77.715 seconds.
Epoch 58 Loss 0.182820 Duration 77.718 seconds.
Epoch 59 Loss 0.187083 Duration 77.980 seconds.
Epoch 60 Loss 0.185693 Duration 78.301 seconds.
Checkpoint has been created.
Increasing batch size from 125 to 250
Epoch 61 Loss 0.106599 Duration 76.270 seconds.
Epoch 62 Loss 0.086069 Duration 76.426 seconds.
Epoch 63 Loss 0.080796 Duration 76.852 seconds.
Epoch 64 Loss 0.085795 Duration 77.290 seconds.
Epoch 65 Loss 0.080012 Duration 77.235 seconds.
Epoch 66 Loss 0.085630 Duration 77.284 seconds.
Epoch 67 Loss 0.091354 Duration 76.587 seconds.
Epoch 68 Loss 0.085518 Duration 76.247 seconds.
Epoch 69 Loss 0.088424 Duration 76.438 seconds.
Epoch 70 Loss 0.090490 Duration 76.593 seconds.
Checkpoint has been created.
Increasing batch size from 250 to 500
Epoch 71 Loss 0.056288 Duration 74.703 seconds.
Epoch 72 Loss 0.033772 Duration 74.753 seconds.
Epoch 73 Loss 0.031195 Duration 74.611 seconds.
Epoch 74 Loss 0.027346 Duration 74.592 seconds.
Epoch 75 Loss 0.025847 Duration 74.562 seconds.
Epoch 76 Loss 0.027269 Duration 74.953 seconds.
Epoch 77 Loss 0.028169 Duration 84.628 seconds.
Epoch 78 Loss 0.027659 Duration 74.247 seconds.
Epoch 79 Loss 0.030882 Duration 73.985 seconds.
Epoch 80 Loss 0.035933 Duration 74.012 seconds.
Checkpoint has been created.
```



```
Increasing batch size from 500 to 1000
Epoch 81 Loss 0.024357 Duration 74.017 seconds.
Epoch 82 Loss 0.011149 Duration 73.837 seconds.
Epoch 83 Loss 0.006300 Duration 74.031 seconds.
Epoch 84 Loss 0.005267 Duration 73.833 seconds.
Epoch 85 Loss 0.004760 Duration 73.739 seconds.
Epoch 86 Loss 0.004852 Duration 74.007 seconds.
Epoch 87 Loss 0.004463 Duration 73.818 seconds.
Epoch 88 Loss 0.004022 Duration 73.803 seconds.
Epoch 89 Loss 0.003615 Duration 73.643 seconds.
Epoch 90 Loss 0.002406 Duration 73.701 seconds.
Checkpoint has been created.
```

Training logs
(Note that the epoch training time is same even after changing the mini batch size.)



```
--- Preparing Data ---
### Test or Validation ... ###
Restored model parameters for checkpoint 90 loss 0.002406115555204451
100%|████████████████████████████████████████████████████████████| 10000/10000 [02:47<00:00, 59.80it/s]
Test accuracy: 0.9356
Restored model parameters for checkpoint 80 loss 0.03593315362930298
100%|████████████████████████████████████████████████████████████| 10000/10000 [02:42<00:00, 61.68it/s]
/home/aanand/HW2/ResNet/Model.py:88: UserWarning: To copy construct from a tensor, it is recommended to
ch.tensor(sourceTensor).
  y = torch.tensor(y)
Test accuracy: 0.9178
Restored model parameters for checkpoint 70 loss 0.09049041886813938
100%|████████████████████████████████████████████████████████████| 10000/10000 [02:43<00:00, 61.27it/s]
Test accuracy: 0.9048
Restored model parameters for checkpoint 60 loss 0.18569264733232557
100%|████████████████████████████████████████████████████████████| 10000/10000 [02:47<00:00, 59.53it/s]
Test accuracy: 0.8831
Restored model parameters for checkpoint 30 loss 0.27555107209831475
100%|████████████████████████████████████████████████████████████| 10000/10000 [02:45<00:00, 60.55it/s]
Test accuracy: 0.8641
```

Evaluation logs for checkpoints {90, 80, 70, 60, 30}

Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.