ELF Implementation

1. Summary

Initial experiments using a standard Cross-Entropy based loss function resulted in stable training but yielded lower accuracy. I implemented LDAM along with Delayed Reweighting training schedule which yielded me better accuracy close to the paper's accuracy level.

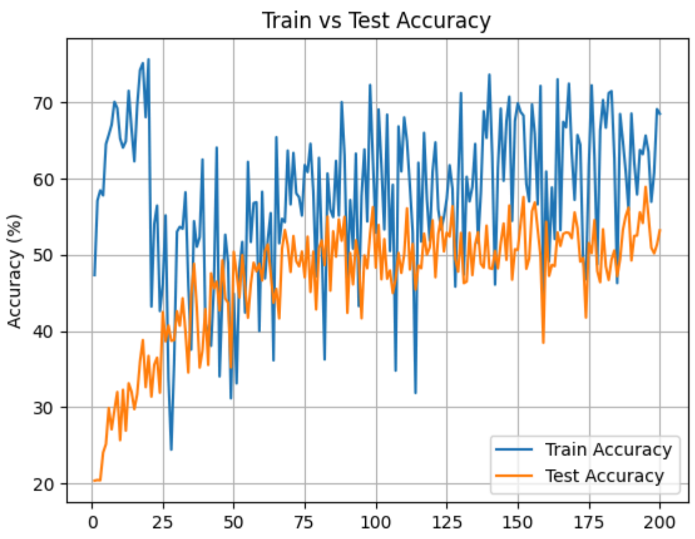
Transitioning to the more advanced LDAM loss, as specified by the paper, initially introduced severe training instability, causing the **loss to explode** and preventing the model from learning. However, after stabilizing the training process through learning rate adjustments and gradient clipping, the model trained with the full LDAM and Delayed Reweighting schedule is now performing significantly better,

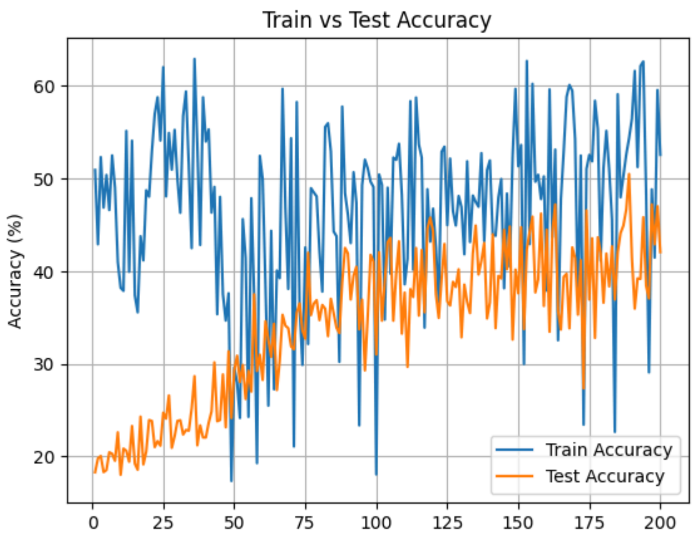
The current implementation is comparatively stable and produces meaningful results. The final step of post-hoc inference threshold searching was also implemented, revealing the model's true peak performance.

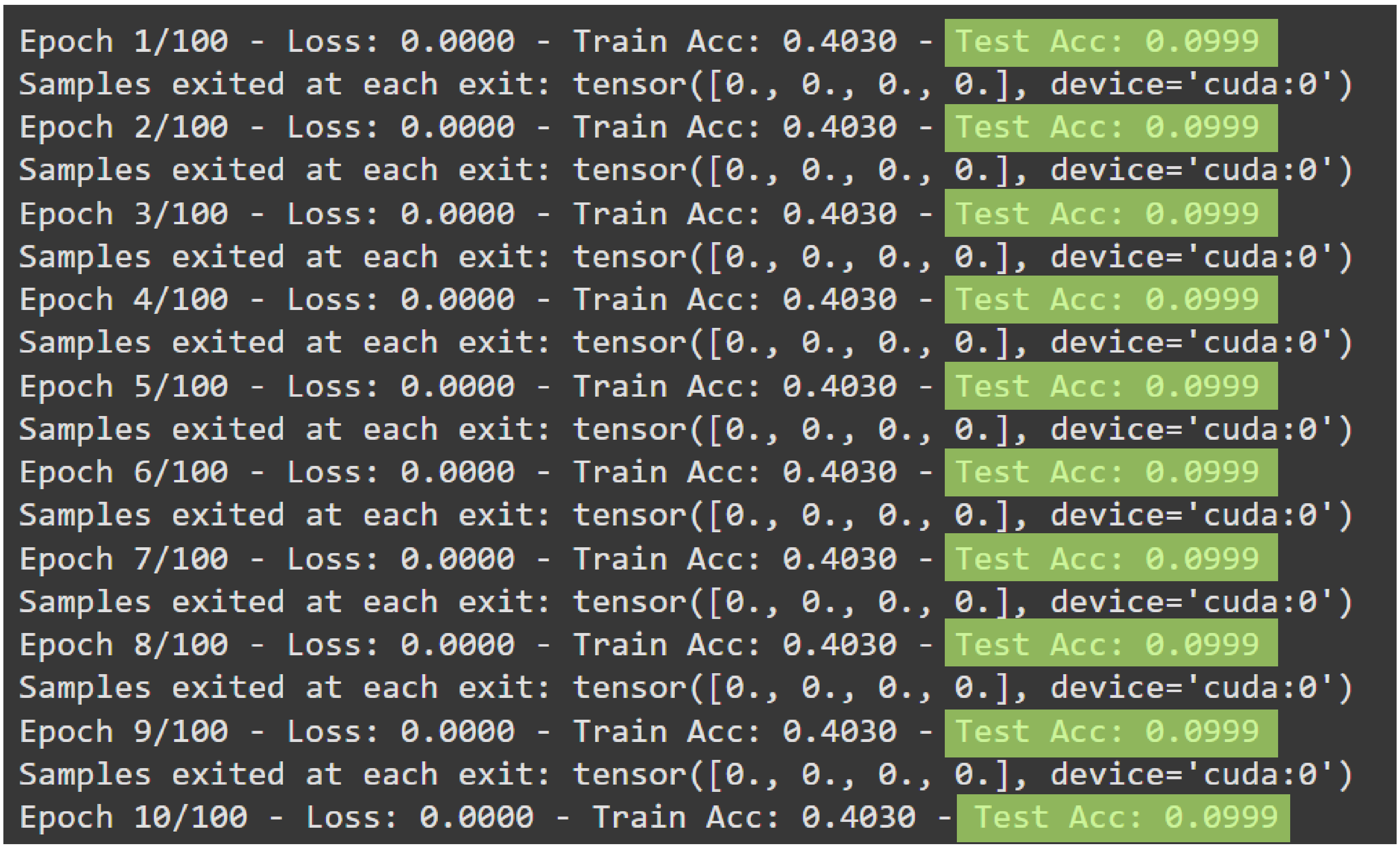
2. Training Instability

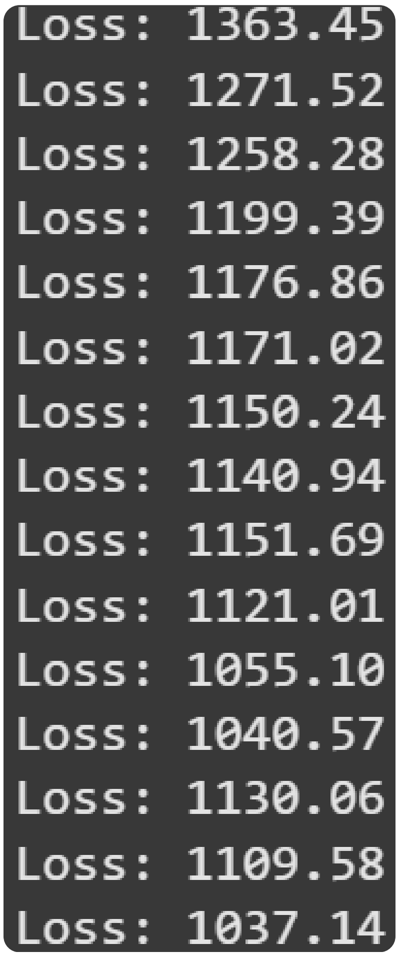
The initial phase of the project was marked by significant challenges in achieving a stable training process. Early implementations of the training loop, while structurally correct, suffered from **exploding gradients**.

- During the first few epochs, the training loss would not decrease but would instead rapidly climb to extremely large values (like >1200).
- This numerical instability prevented the model from learning entirely. The test accuracy remained stagnant at approximately 10% and did not neither increase nor decrease.






```
Epoch 1/100 - Loss: 0.0000 - Train Acc: 0.4030 - Test Acc: 0.0999
Samples exited at each exit: tensor([0., 0., 0., 0.], device='cuda:0')
Epoch 2/100 - Loss: 0.0000 - Train Acc: 0.4030 - Test Acc: 0.0999
Samples exited at each exit: tensor([0., 0., 0., 0.], device='cuda:0')
Epoch 3/100 - Loss: 0.0000 - Train Acc: 0.4030 - Test Acc: 0.0999
Samples exited at each exit: tensor([0., 0., 0., 0.], device='cuda:0')
Epoch 4/100 - Loss: 0.0000 - Train Acc: 0.4030 - Test Acc: 0.0999
Samples exited at each exit: tensor([0., 0., 0., 0.], device='cuda:0')
Epoch 5/100 - Loss: 0.0000 - Train Acc: 0.4030 - Test Acc: 0.0999
Samples exited at each exit: tensor([0., 0., 0., 0.], device='cuda:0')
Epoch 6/100 - Loss: 0.0000 - Train Acc: 0.4030 - Test Acc: 0.0999
Samples exited at each exit: tensor([0., 0., 0., 0.], device='cuda:0')
Epoch 7/100 - Loss: 0.0000 - Train Acc: 0.4030 - Test Acc: 0.0999
Samples exited at each exit: tensor([0., 0., 0., 0.], device='cuda:0')
Epoch 8/100 - Loss: 0.0000 - Train Acc: 0.4030 - Test Acc: 0.0999
Samples exited at each exit: tensor([0., 0., 0., 0.], device='cuda:0')
Epoch 9/100 - Loss: 0.0000 - Train Acc: 0.4030 - Test Acc: 0.0999
Samples exited at each exit: tensor([0., 0., 0., 0.], device='cuda:0')
Epoch 10/100 - Loss: 0.0000 - Train Acc: 0.4030 - Test Acc: 0.0999
```

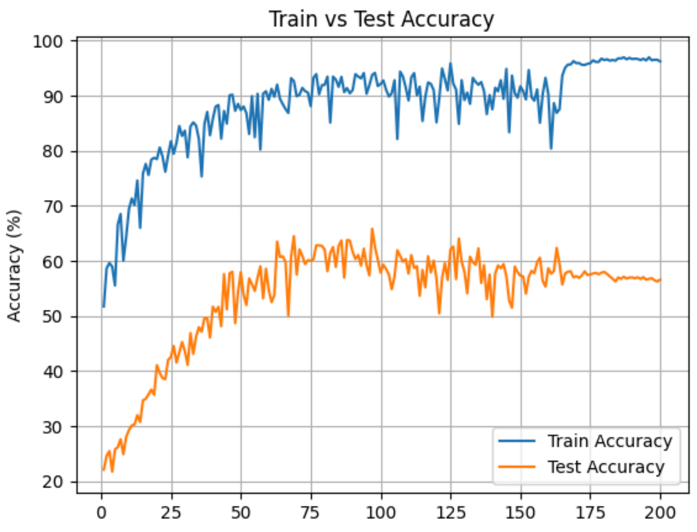

```
Loss: 1363.45
Loss: 1271.52
Loss: 1258.28
Loss: 1199.39
Loss: 1176.86
Loss: 1171.02
Loss: 1150.24
Loss: 1140.94
Loss: 1151.69
Loss: 1121.01
Loss: 1055.10
Loss: 1040.57
Loss: 1130.06
Loss: 1109.58
Loss: 1037.14
```

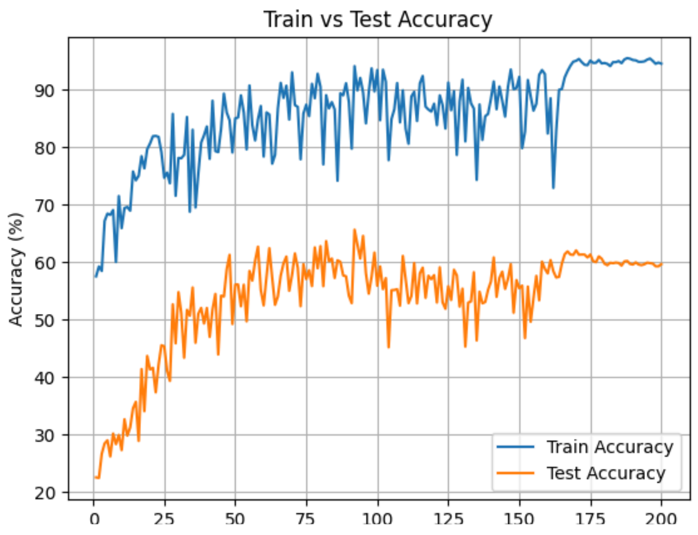
3. Current Status: Stable and Effective Training

The current model's training process is **more stable**, by a consistently decreasing loss and steadily increasing train/test accuracy throughout the 200 epochs.

This stability was achieved by implementing two critical changes:

1. **Learning Rate Adjustment:** The initial learning rate for the SGD optimizer was reduced from 0.1 to a more conservative 0.01. The original, higher learning rate was too aggressive when combined with the logit scaling in the LDAM loss, causing the optimizer to take excessively large steps and diverge.
2. **Gradient Clipping:** torch.nn.utils.clip_grad_norm_ was introduced into the training loop. This function monitors the magnitude (norm) of the gradients after backpropagation. If the norm exceeds a predefined threshold, the gradients are scaled down preventing exploding gradient issue.





4. Implemented Techniques

The current stable code is a detailed implementation of the methodologies described in the paper.

A. Loss Function I: LDAM Loss (Label-Distribution-Aware Margin)

- **Logit Scaling:** The logits are scaled by a factor s=30 before the loss calculation, which sharpens the probability distribution and aids in the margin-based optimization.

B. ELF Loss Aggregation

The total loss for a sample is not calculated from a single exit but is accumulated across multiple exits.

- **Per-Sample Logic:** For each sample in a batch, the LDAM loss is calculated at the first exit and added to that sample's total loss.
- **Training Exit Criterion:** The sample is then checked against the exit criterion: (prediction is correct) AND (confidence > training_threshold). The **training threshold was fixed at 0.2** as per the paper's recommendation for CIFAR-10.

C. Training Schedule: Two-Phase Delayed Reweighting (DRW)

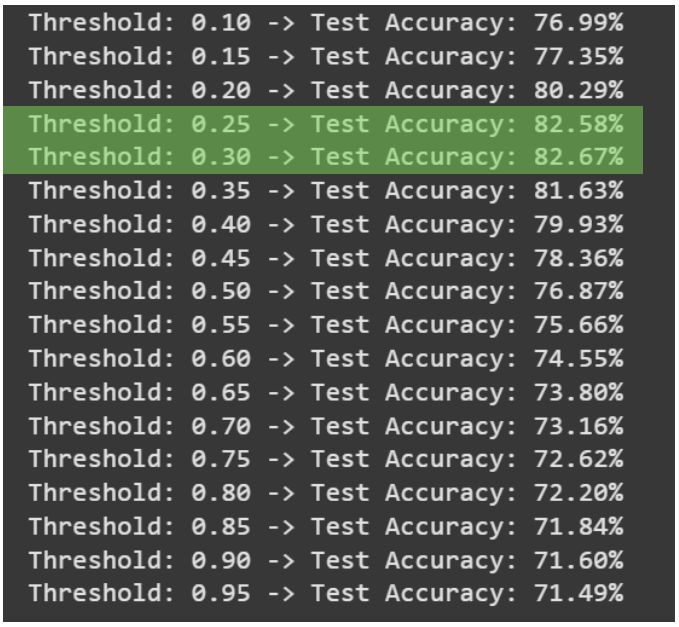
A two-phase training schedule was implemented to maximize performance, as prescribed by the paper.

- **Phase 1: Representation Learning (Epochs 1-159):** The model is trained using the LDAM loss but *without* any explicit class reweighting. The goal of this phase is for the model to learn general, robust visual features from the entire dataset. During this phase, the model **naturally overfits** to the high-frequency majority classes, which is expected.
- **Phase 2: Fine-tuning (Epochs 160-200):** At epoch 160, two things happen: the learning rate is reduced, and the DRW schedule activates. The LDAM loss is now weighted based on the "effective number of samples" per class. This re-weighting scheme up-weights the loss for minority classes, forcing the model to fine-tune its learned representations and improve its performance on the tail of the distribution.

5. Post-Hoc Inference Threshold Search

A critical step from the paper's methodology was implemented to find the model's true peak performance.

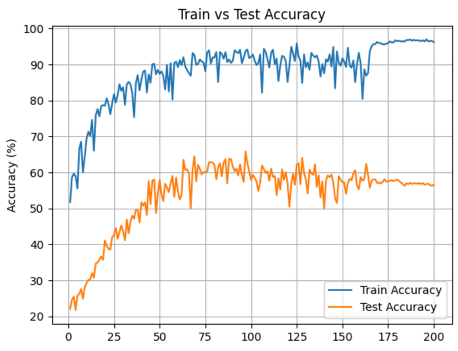
- After the 200-epoch training was complete, the final saved model was loaded. It was then evaluated on the test set multiple times, each time using a different inference threshold (from 0.1 to 0.95).
- This search revealed that the model's peak accuracy was achieved at a lower threshold. For the 100x, 50x, 10x imbalance ratio models, the best accuracy was found at an inference threshold of **0.20, 0.25** and **0.30**.


```
Threshold: 0.10 -> Test Accuracy: 76.99%
Threshold: 0.15 -> Test Accuracy: 77.35%
Threshold: 0.20 -> Test Accuracy: 80.29%
Threshold: 0.25 -> Test Accuracy: 82.58%
Threshold: 0.30 -> Test Accuracy: 82.67%
Threshold: 0.35 -> Test Accuracy: 81.63%
Threshold: 0.40 -> Test Accuracy: 79.93%
Threshold: 0.45 -> Test Accuracy: 78.36%
Threshold: 0.50 -> Test Accuracy: 76.87%
Threshold: 0.55 -> Test Accuracy: 75.66%
Threshold: 0.60 -> Test Accuracy: 74.55%
Threshold: 0.65 -> Test Accuracy: 73.80%
Threshold: 0.70 -> Test Accuracy: 73.16%
Threshold: 0.75 -> Test Accuracy: 72.62%
Threshold: 0.80 -> Test Accuracy: 72.20%
Threshold: 0.85 -> Test Accuracy: 71.84%
Threshold: 0.90 -> Test Accuracy: 71.60%
Threshold: 0.95 -> Test Accuracy: 71.49%
```

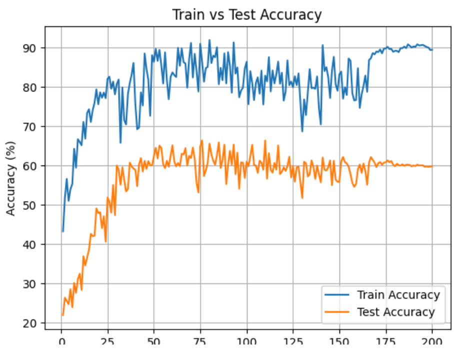
0.01 Learning rate Results:

These models were trained with a *0.01 learning rate* and gradient clipping, and using LDAM loss with a two-phase Delayed Reweighting schedule to handle long-tailed data distributions.

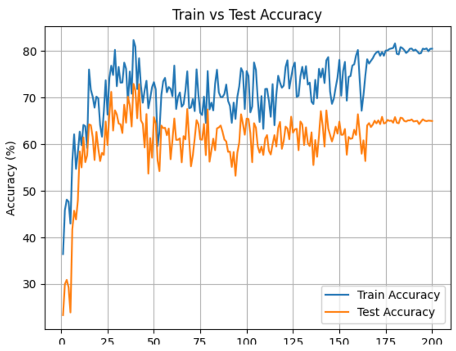
ELF: 0.01LR, LDAM, DRW, 200epoch	100x	50x	10x
Train Accuracy	90.31	89.4	80.5
Test Accuracy	69.19	74.02	80.82



100x



50x

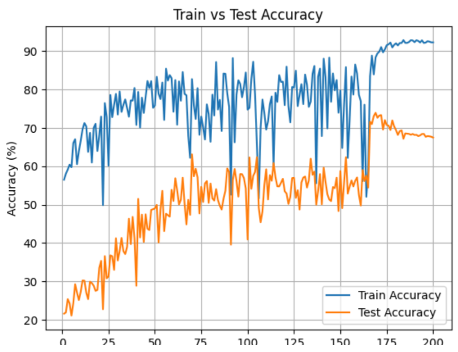


10x

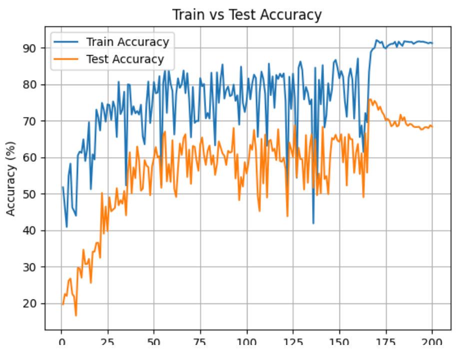
0.1 Learning rate Results:

These models were trained with a *0.1 learning rate* and gradient clipping, and using LDAM loss with a two-phase Delayed Reweighting schedule to handle long-tailed data distributions.

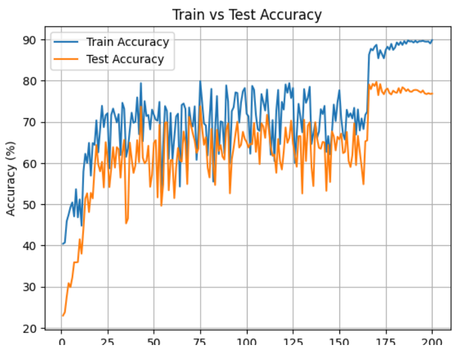
ELF: 0.1LR, LDAM, DRW, 200epoch	100x	50x	10x
Train Accuracy	92.23	91.27	89.96
Test Accuracy	72.29	76.45	82.67



100x



50x



10x

Comparison with the paper's accuracy:

	Paper's Accuracy	0.01 LR accuracy	0.1 LR accuracy	Approx Diff
100x	78.1	69.19	72.29	**5.8**
50x	82.4	74.02	76.74	**5.6**
10x	88	80.82	82.67	**5.3**