Deep Learning [CSL4020]
Intermediate Project Report 2

Submitted By:

Shalin Jain (B21CS070) Shashwat Roy (B21CS071)
Shubh Goyal (B21CS073) Sukriti Goyal (B21CS075)

Problem Statement

Finding the perfect model parameters giving the minimum loss is like searching for a needle in the loss landscape. However, optimizers automate and fasten the searching process through various techniques like gradient descent, analyzing gradients (partial derivatives of the loss w.r.t. parameters) to steer updates towards lower loss regions. Popular optimizers such as SGD, RMSProp and Adam further introduce momentum and gradient accumulation techniques for better learning rate adaptation, and parameter adjustments. But still, there remains room for making them faster and more efficient. We focus on formulating, developing and analyzing a new optimizer, by introducing further adaptive techniques in the existing optimizers.

Advancements After Adam

- **AdamW**[1]: It decouples weight decay from learning rate, thus improving Adam's generalization ability, enabling performance competitive with SGD momentum for image classification tasks where vanilla Adam underperforms.
- **Nadam**[2]: It replaces the regular momentum in Adam by the Nesterov momentum, which speeds up convergence and improves model performance.
- **RAdam**[3]: Rectified Adam (RAdam) introduces a new term on the lines of learning rate warmup, rectifying the variance of the adaptive learning rate.
- **Lookahead**[4]: In this algorithm, two sets of weights are updated iteratively. Intuitively, the algorithm chooses a search direction by looking ahead at the sequence of “fast weights" generated by another optimizer. It helps improve the learning stability and lowers the variance of its inner optimizer.
- **AdaBelief**[5]: It introduces an exponentially moving average based ‘belief’ factor based on the magnitude and sign of the current gradient instead of directly averaging gradients for the updation of weights and biases. It provides increased stability and quality in GAN samples.
- **Dynamical Learning Rate Scheduling**: Adaptively adjusts the learning rate during training, especially across different stages. YellowFin[6], an automatic tuner for momentum and learning rate in SGD, is one such example. YellowFin

optionally uses a negative-feedback loop to compensate for the momentum dynamics in asynchronous settings on the fly.

Shortcomings of these Optimizers

- Adaptive learning rate methods often use heuristics like moving averages or squared gradients, which might not accurately capture the true variance of the loss landscape.
- Techniques like momentum that sometimes help in escaping the local minima rely on specific parameter choices and might not be effective in all scenarios.
- Due to the adaptive nature of Adam, it becomes susceptible to the outliers in the data, thus giving poor performance on noisy sparse data.

Datasets and Architectures for Testing Improvements

As we gathered from different research papers on optimizers, the following datasets were used for the benchmarking of the optimisers so we also plan to use the following datasets along with the corresponding performant model architectures for evaluating our approaches.

- ResNet-18/ResNet-20 (for supervised image classification task) and GANs (for image generation task) training using *CIFAR-10*[7] image dataset and *MNIST*[8] digit dataset.
- ResNet-18/ResNet-20 training on *ImageNet*[9] dataset.
- LSTM training on *One Billion Word Dataset*[10] and *Penn Treebank Dataset*[11] to check performance for language modeling purposes.

Creating a Custom Optimizer *(Report 2 starts here)*

To ease experimentation, we programmed our own Adam by taking references from the actual PyTorch implementation of Adam. The code for the same can be found in our GitHub repository. Link to the repo: *https://github.com/Shubh-Goyal-07/DLProject*

Optimization algorithms like Adam have tunable hyperparameters (coefficients in moments updates). We tried various scheduling algorithms for β_1 , β_2 and compared the performance of a fully-connected neural network model on MNIST and CIFAR-10 datasets against the pytorch Adam optimizer. The performance was comparable to Vanilla Adam, RMSProp, but better than AdaGrad.

We tried to schedule β1 and β_2 by replacing β_1 with $\mathbf{(\beta_1)e^{(-t)}}$, and β_2 with $\mathbf{(\beta_2)e^{(-t)}}$. We did this because as epochs keep on progressing, we have to reduce the weightage of past gradients and increase the weightage of current gradients. This was explicitly suggested in the Adam research paper.

We obtained the following loss curves with our ***custom Adam*** with a fully-connected neural network architecture trained on the MNIST dataset.
Here **Adam_custom** refers to our implementation of Adam with the scheduled betas. **Adam_torch** refers to the Vanilla Adam. **AdaGrad_torch** is the Vanilla AdaGrad and **RMS_torch** refers to the Vanilla RMSProp.

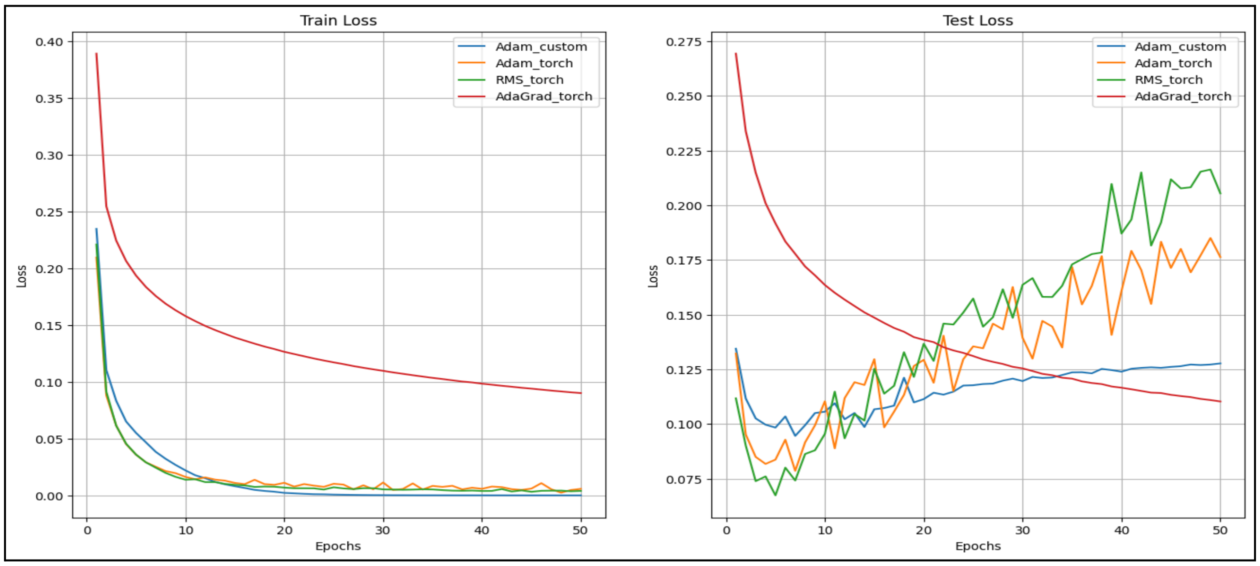

Fig.1: Loss Curves showing performance of different optimizers on MNIST

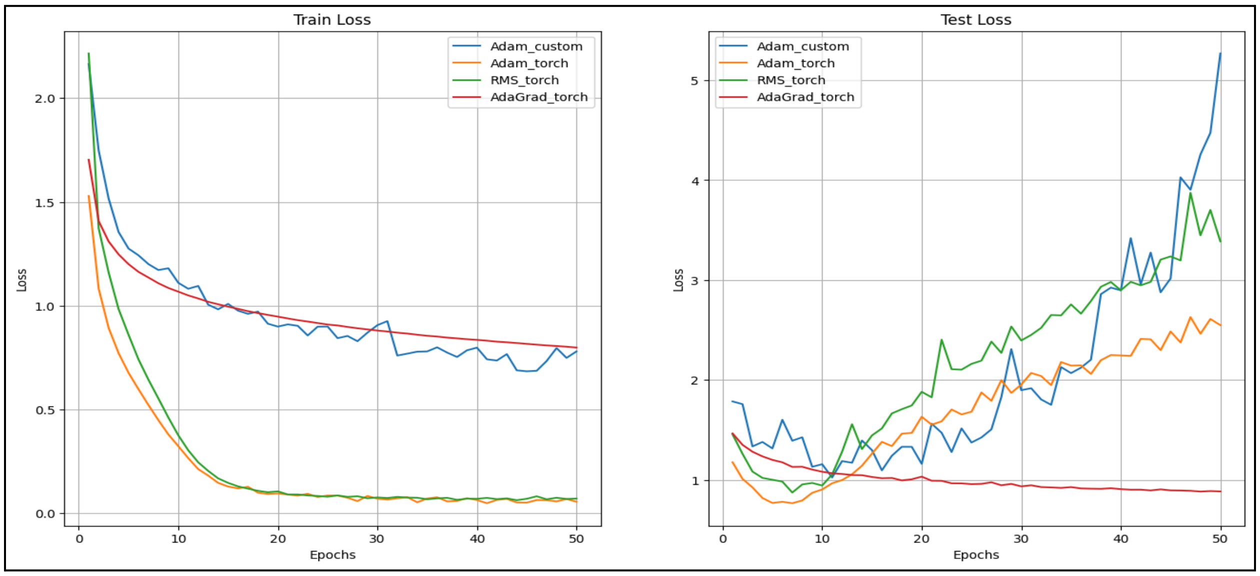

Fig.2: Loss Curves showing performance of different optimizers on CIFAR-10

Future work

- **Considering higher order moments**[12]: The Adam optimizer incorporates the accumulated first and second order moments. We can consider replacing a lower order moment with a higher order moment or incorporate the higher order moments in the original equation.
  A concept of skewness can be adopted in this context to help characterize the analysis. The skewness value of stochastic gradients can be negative, positive, or undefined, and based on the nature of distribution for each condition, we can incorporate skewness into Adam as the third raw moment. Skewness would help us with the non-normal distributions, enabling the machine learning models to perform better on datasets with a non-normal distribution.
- One of the ideas proposed is to incorporate the concept of long term and short term memory for gradients separately and use it to improve the algorithm. The motivation is that improving on the ability of the optimizer to selectively retain what past gradient values are useful should improve the performance of the optimizer.

References

1. https://arxiv.org/pdf/1711.05101v3.pdf
2. https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ
3. https://arxiv.org/pdf/1908.03265v4.pdf
4. https://proceedings.neurips.cc/paper_files/paper/2019/file/90fd4f88f588ae64038134f1eeaa023f-Paper.pdf
5. https://arxiv.org/pdf/2010.07468.pdf
6. https://arxiv.org/pdf/1706.03471.pdf
7. https://www.cs.toronto.edu/%7Ekriz/cifar.html
8. https://www.kaggle.com/datasets/hojjatk/mnist-dataset
9. https://paperswithcode.com/dataset/imagenet
10. https://www.statmt.org/lm-benchmark/
11. https://catalog.ldc.upenn.edu/docs/LDC95T7/cl93.html
12. https://www.researchgate.net/publication/336577530_On_Higher-order_Moments_in_Adam

---