Homework 5 - optimizers

This project focuses on comparing multiple optimizers and performing hyper-parameter tuning on a minimized CIFAR-10 dataset. The goal is to find configurations achieving over 40%, 50%, 60%, and 70% validation accuracy.

Data Preprocessing

The CIFAR-10 dataset was loaded and transformed to grayscale images of size 28x28. The transformations included resizing, grayscaling, and flattening the images. The mean and standard deviation of the dataset were calculated for normalization purposes.

For the Training Dataset the augmentation techniques I used are RandomHorizontalFlip & RandomRotation by up to 15 degrees to introduce variability in the dataset, helping the model generalize better by learning from slightly altered versions of the same image.

TransformDataset class is designed to apply transformations to the CIFAR-10 dataset.

CachedDataset class is designed to enhance the efficiency of data loading for the evaluation dataset.

Model Architecture

The project uses a Multilayer Perceptron (MLP) model. The model architecture consists of fully connected layers with batch normalization applied to each hidden layer, ReLU activations, and dropout of 30% applied after the activation function of each hidden layer to prevent overfiting.

Input Dimension: The MLP expects an input dimension of 784, corresponding to the flattened 28x28 grayscale images.

Hidden Layers: The model consists of two hidden layers. The first hidden layer has 1024 neurons, and the second has 512 neurons.

Output Layer: The final layer of the model is a linear layer that maps the output of the last hidden layer to 10 classes, corresponding to the CIFAR-10 dataset's 10 different image categories.

Optimizer and Hyperparameter Tuning

A range of optimizers including SGD, Adam, RMSProp, AdaGrad, and SGD with SAM were evaluated. Hyperparameter tuning was performed using weights & biases with parameters such as learning rate, batch size, weight decay, and momentum.

Training process: involved a custom loop with support for early stopping. The process included both first and second steps for SAM optimizer. Metrics like epoch training loss, accuracy, and

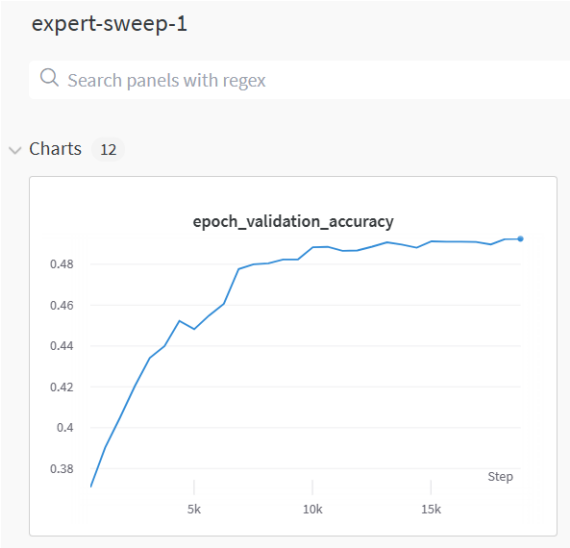
batch training loss were logged using wandb. The key focus was on observing how different optimization algorithms impact the model's learning and validation accuracies.

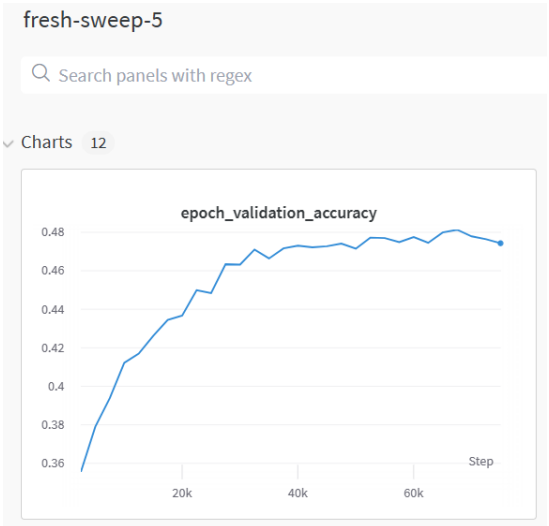
Results, observations, plots illustrating the validation accuracies of different configurations

1. Adam Optimizer (Runs: xlxbawi0, 3zot21y3) - showed the highest validation accuracy among the tested optimizers and exhibited faster convergence.

Run xlxbawi0: Started with a validation accuracy of about 37%, reaching nearly 49% by the 29th epoch. This shows a robust improvement, indicating that Adam was effective for this dataset.

Run 3zot21y3: Began at around 35% accuracy and improved to about 47% by the 29th epoch. Similar to the first run, it shows that Adam consistently facilitates good learning progress.



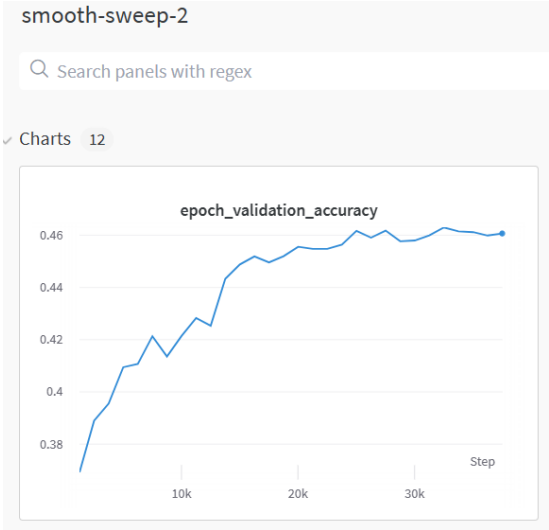


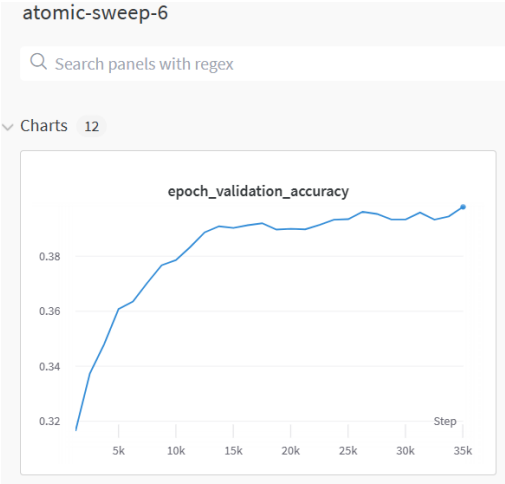
2. Stochastic Gradient Descent (SGD) (Runs: 2vbo0sz3, vwbf243o, 00lcb8hx) achieved moderate validation accuracy. It demonstrated steady, but slower convergence compared to Adam.

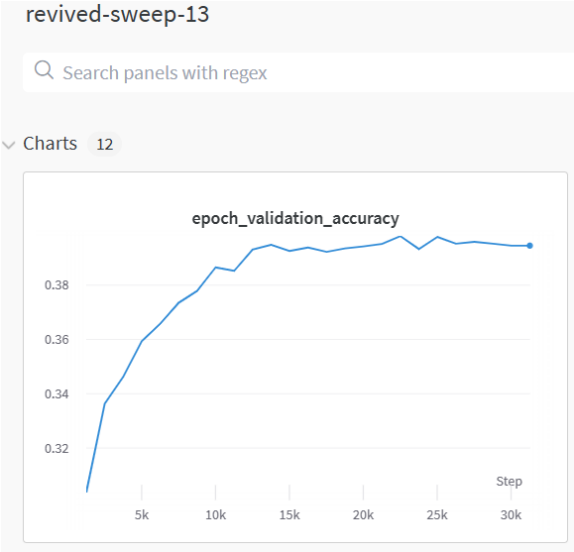
Run 2vbo0sz3: This run saw an increase from about 37% to 46% accuracy by the 29th epoch. It shows a good improvement, albeit slightly less than with Adam.

Run vwbf243o: Started around 32% and improved to near 40% by the 27th epoch.

Run 00lcb8hx: Began at about 30% accuracy and reached around 41% by the 29th epoch.







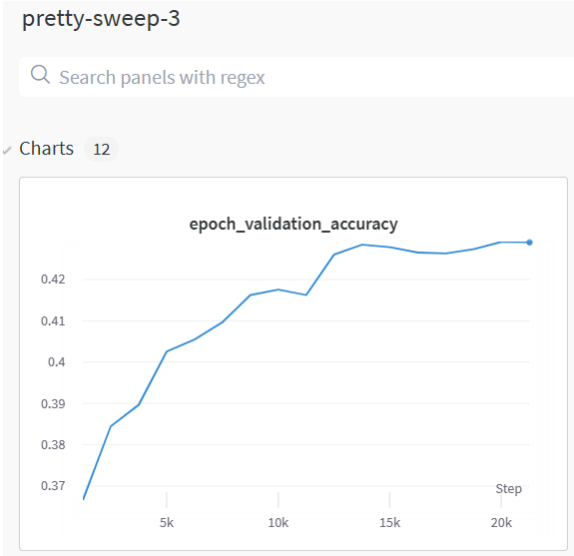
3. Adagrad (Runs: m812y960, k32xuu8x, v6jzs3ok, we29v6py) - shows the least improvement among the runs when compared to other optimizers.

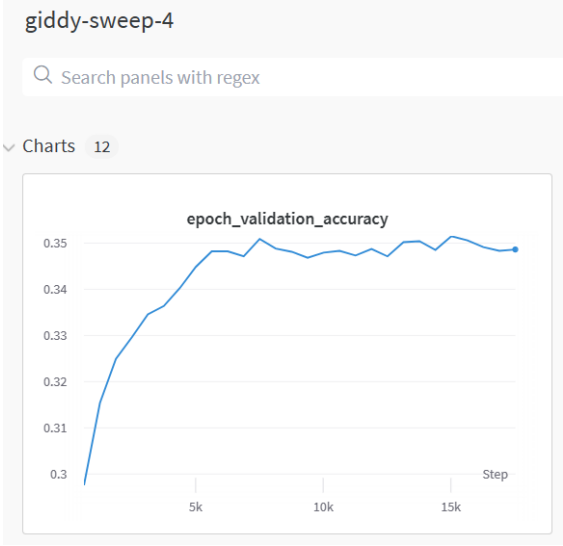
Run m812y960: Improved from around 37% to 43% by the 16th epoch, where it stopped early. This shows a moderate improvement but indicates potential issues with convergence.

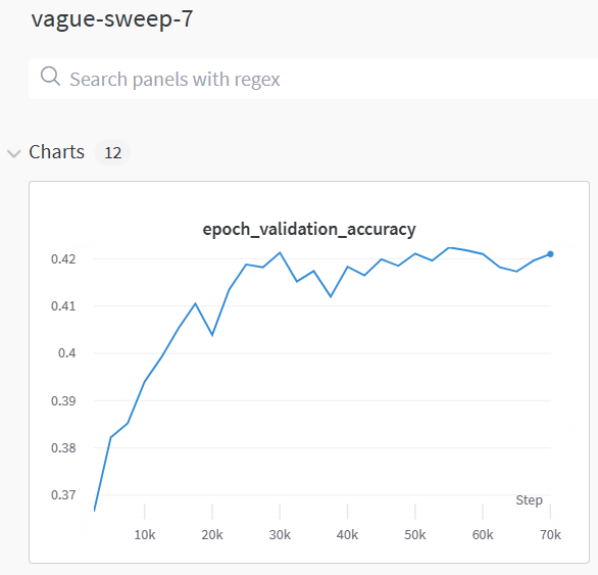
Run k32xuu8x: Started at about 30% and reached 35% by the 27th epoch

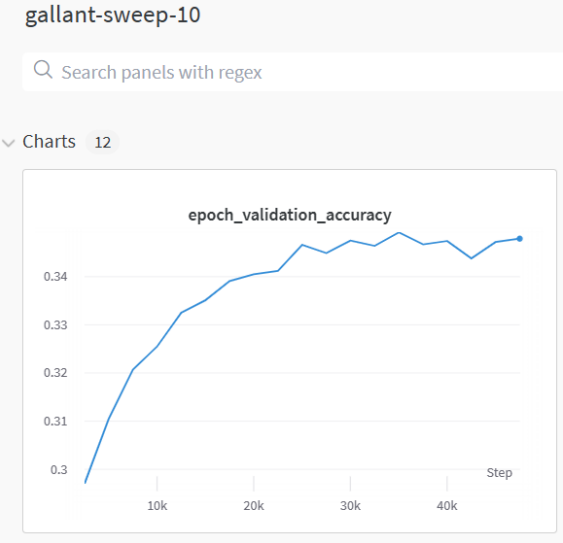
Run v6jzs3ok: Improved from about 36% to 42% by the 27th epoch

Run we29v6py: Began at around 30% and improved to 35% by the 18th epoch









4. RMSprop Optimizer (Runs: 7hjz9b7g, lq5et9sd, t6atpf5d, 2d9i05ae) - showed moderate improvements.

Run 7hjz9b7g: Started with a validation accuracy of around 35% and improved to about 49% by the 29th epoch.

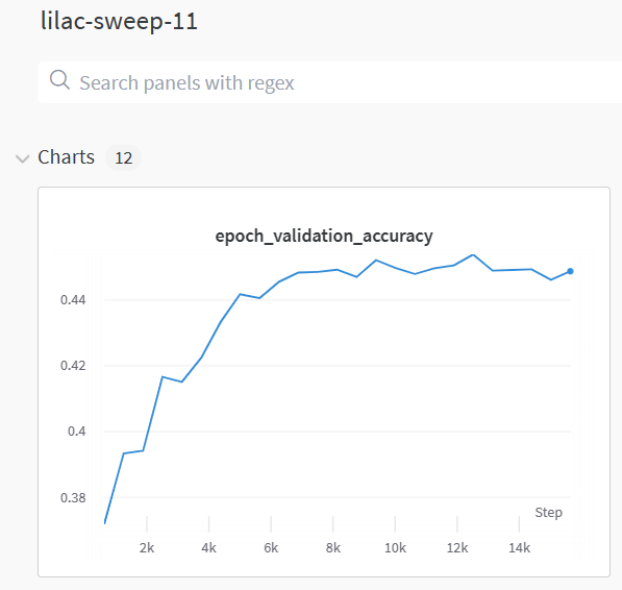
5. SGD with SAM (Runs: utgrlvsf, loclfeso)

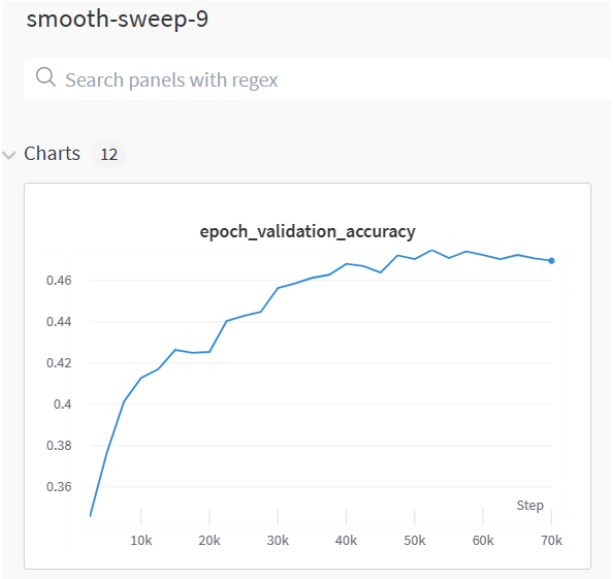
Run utgrlvsf: Started at around 37% accuracy and reached about 45% by the 24th epoch, showing good improvement.

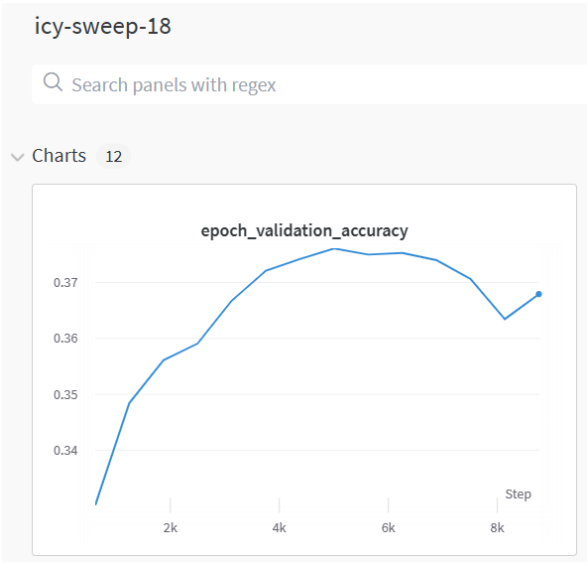
Run loclfeso: Began at around 26% accuracy and showed significant improvement, reaching about 47% by the 27th epoch.

Run jtf3as10: Began at 33% and reached 36% accuracy

Stochastic Gradient Descent with Sharpness-Aware Minimization (SGD with SAM) is designed to improve the model's generalization by considering the sharpness of the loss landscape. The runs using SGD with SAM show significant improvements, particularly 2nd run which suggests that this optimizer is well-suited for enhancing model generalization on this dataset.







These results highlight the importance of selecting an appropriate optimizer for neural network training. Each optimizer has distinct characteristics that can impact the model's learning efficiency and final performance.

6 Expected Points:

Logs: 1 point

Hyperparameter Tuning with wandb: 1point

Optimizer Evaluation:

SGD and Adam configurations (1 point).

RMSProp, AdaGrad configurations (1 point).

SGD with SAM (1 point).

Validation Accuracy: Over 40% (1 point).

W&B link:

https://wandb.ai/marius-workspace/wandb-final9/table?workspace=user-mariusmarin