Image classification using PyTorch Lightning and Wandb

Tinkering with lightning and wandb

Maniraj Sai

Introduction and Primary Objective

The primary objective of this project is to deepen our understanding of PyTorch Lightning, a high-level framework built on PyTorch, and to explore the capabilities of Weights & Biases (wandb) for experiment tracking. By applying these tools to a well known dataset, we aim to demonstrate how they can enhance the efficiency and effectiveness of developing, training, and evaluating deep learning models.

Data and Model

The central focus of this project is to train a customized ResNet-18 model for image classification on the CIFAR-10 dataset, which comprises 60,000 images across 10 distinct classes like automobiles, birds, cats, etc. This project particularly emphasizes rapid iteration and model tuning, making it an ideal testbed for experimenting with different hyperparameters and model architectures.

Code

The code to implement this project is present in the following repository.

Training a Baseline

> 💡 Note that our aim in this project was to explore popular frameworks like PyTorch Lightning and wandb. Hence we consider a rather straightforward task.

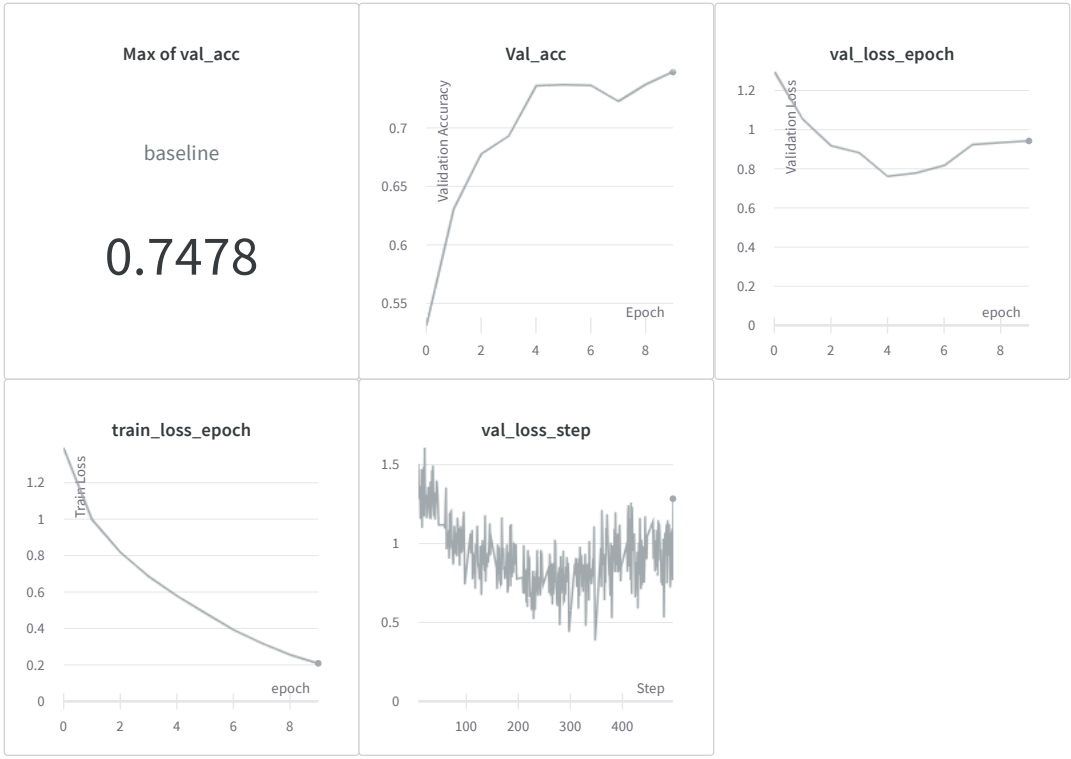
For the baseline, we use the Lightning DataModule to create the train, validation and test dataloaders. The Data Module offers significant advantages by centralizing data processing in one place, enhancing code modularity and readability.

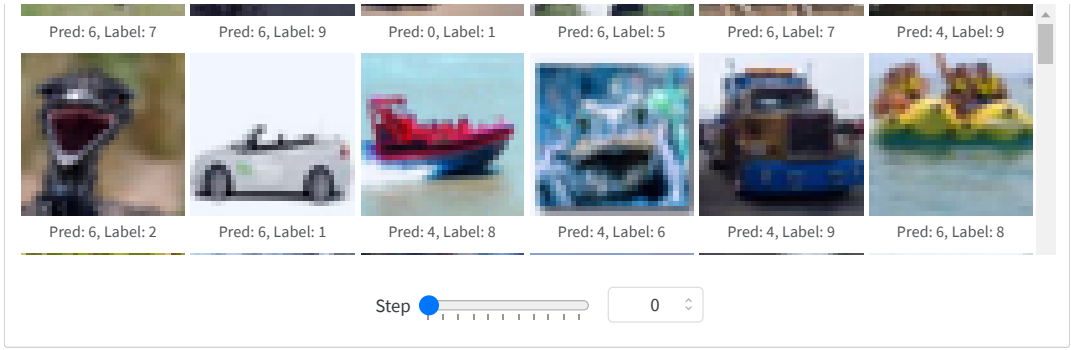
We then built the whole training script from scratch using PyTorch Lightning, a decision that greatly streamlined the development process. By leveraging Lightning's structured yet flexible

framework, we were able to focus more on the model architecture and the training logic rather than the boilerplate code.

For training the model, we made use of the ADAM optimizer with the default values of learning rate and weight decay. The batch size used was 128, and since achieving the best accuracy wasn't the primary task, we limited the training to 10 epochs throughout the experiment to work with the limited computational resources present.

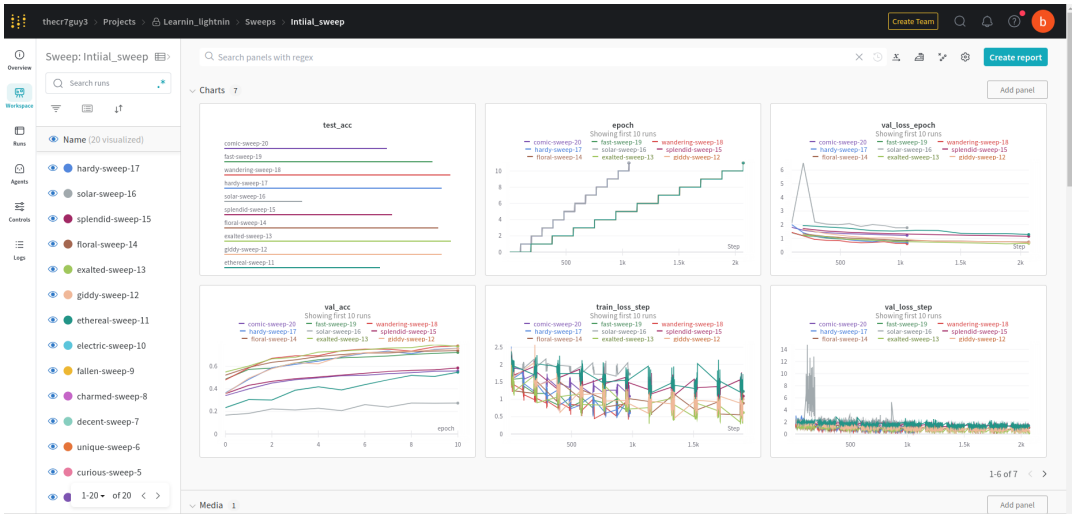
The components used for this baseline were all out of the box, plugged in without experimenting with any of the values.





Wandb Sweeps and Hyperparameter Tuning

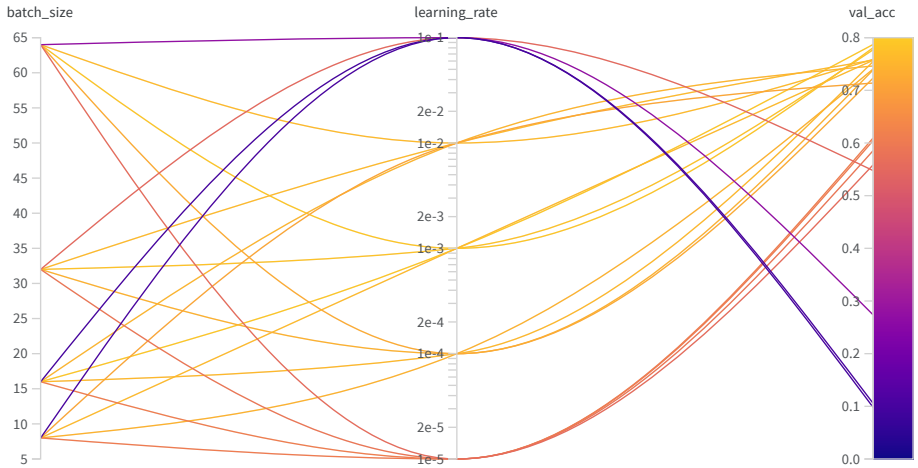
In this phase of the project, we utilized wandb sweeps to automate and streamline the hyperparameter tuning process. Our focus was on key parameters: learning rate, batch size. Since it was the first time experimenting with the sweeps feature, we decided to stick with two main hyperparameters. The sweep was configured to explore a predefined range for each, using a grid search strategy for comprehensive coverage.



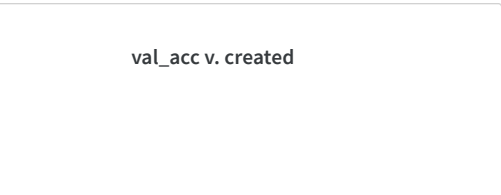
The dashboard demonstrating different runs in a sweep.

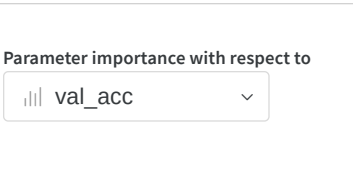
A total of 20 runs were carried throughout the sweep which was initiated through the .yaml file which triggered the automated experimentation across the specified parameter ranges. Monitoring and analysis of the sweep were conducted using wandb's dashboard. This real-time tracking enabled us to observe the performance impact of different hyperparameter combinations as the sweep progressed.

```yaml
program: train.py
name: Intiial_sweep
method: grid
metric:
  name: val_acc
  goal: minimize
parameters:
  learning_rate:
    values: [0.1,0.01,0.001,0.0001,0.00001]
  batch_size:
    values: [8,16,32,64]
  epochs:
    value: 11
```



Incorporating wandb sweeps not only streamlined the hyperparameter tuning process but also provided a level of analysis depth that would be challenging to achieve manually. The results from this exercise led to an enhanced model performance over our initial baseline and offered valuable guidelines for future model optimizations and experiments.





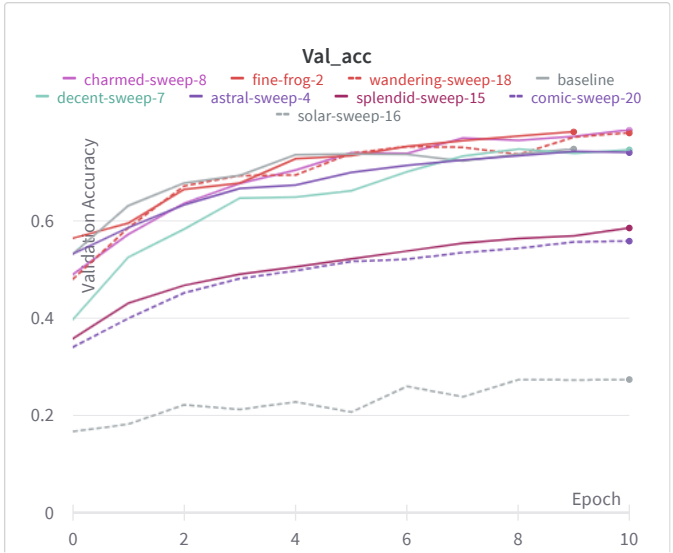
Results and Analysis

Upon completing the wandb sweeps and rigorous model training, we delved into a comprehensive analysis of the results.

Model Performance Metrics

Loss Trends: The loss reduction patterns across training and validation phases were carefully examined. The optimal hyperparameter settings led to a steady decrease in training loss without causing significant validation loss, indicating good generalization without overfitting.

Accuracy Achievements: The model reached its peak accuracy with a specific combination of hyperparameters, notably a moderate learning rate and a balanced batch size. This configuration outperformed our baseline model by a good margin, highlighting the effectiveness of our tuning approach.

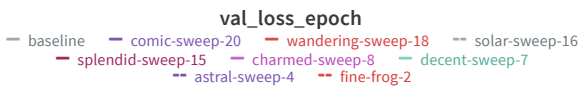


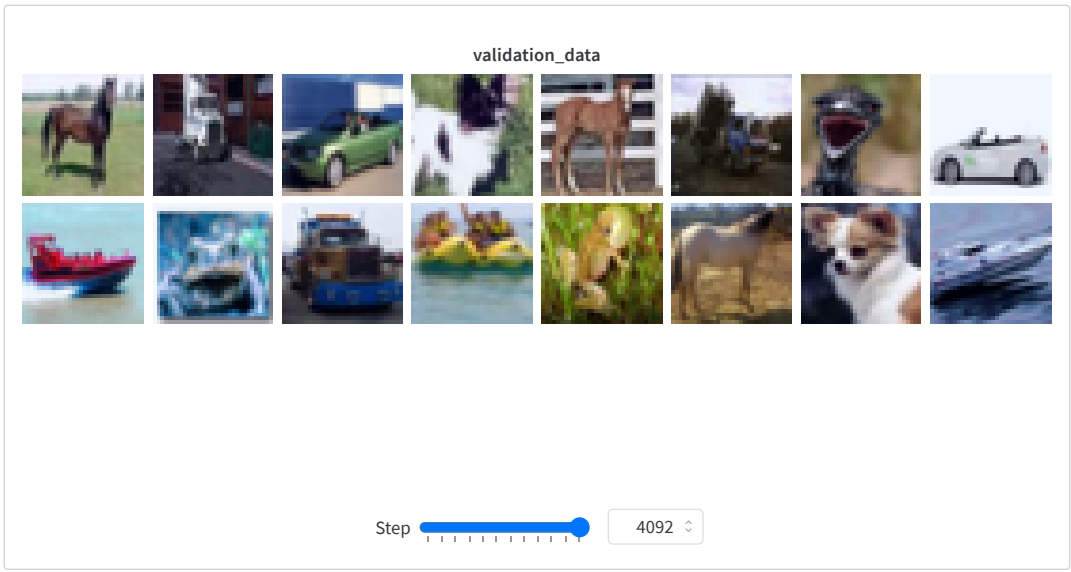
Run set 9

Max of val_acc

charmed-sweep-8

0.787





Future work and conclusion

This project, despite its certain limitations, successfully achieved its primary objectives: to gain proficiency in PyTorch Lightning and to explore the functionalities of wandb sweeps. We navigated the challenges of hyperparameter tuning, albeit with only two parameters, and worked within the constraints of limited training epochs. These conditions, while restrictive, did not hinder our main goal but rather sharpened our focus on efficiently utilizing the tools at our disposal.

The following are the things to be done to give more purposefulness to the project

- [ ] Use different Networks Backbones and Use Pretrained Networks
- [ ] Try to add different hyper-parameters to the sweep