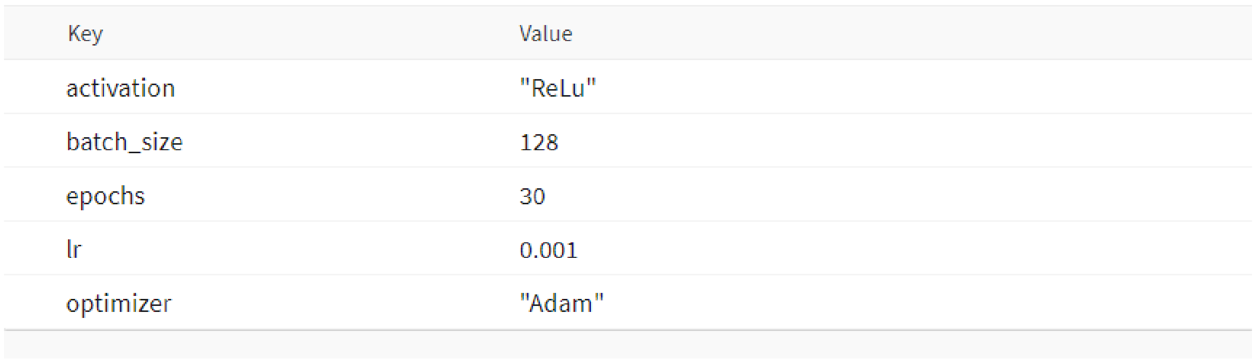
LAB 8

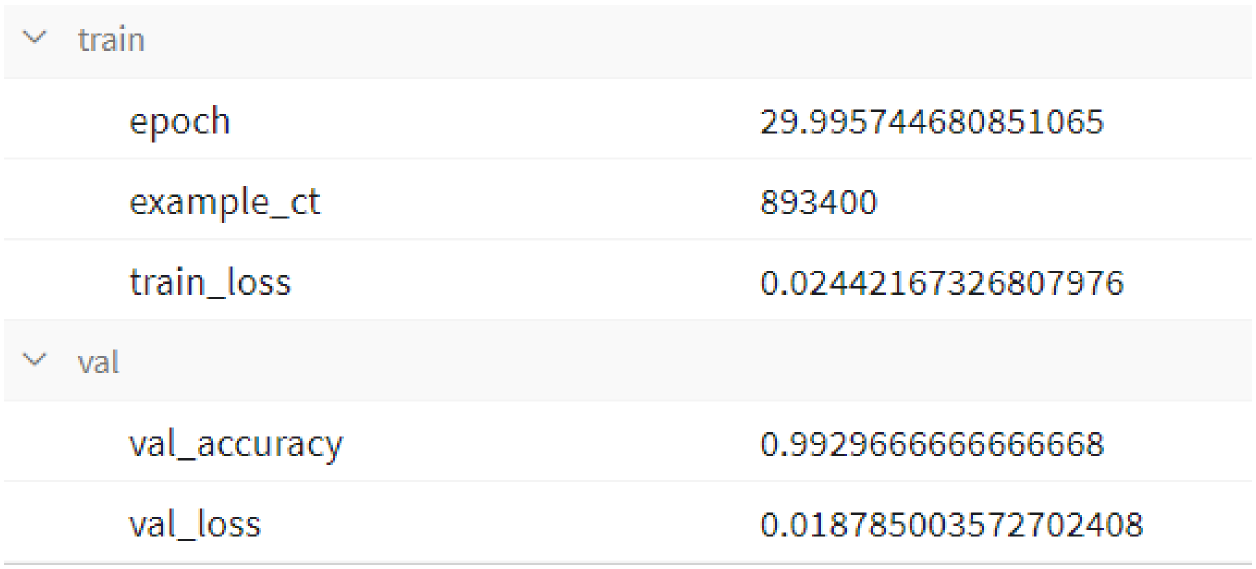
Effect of Hyperparameters

I performed the following runs with the following hyperparameters and I have reported the result of each set of hyper parameters respectively

RUN: smart-sunset1

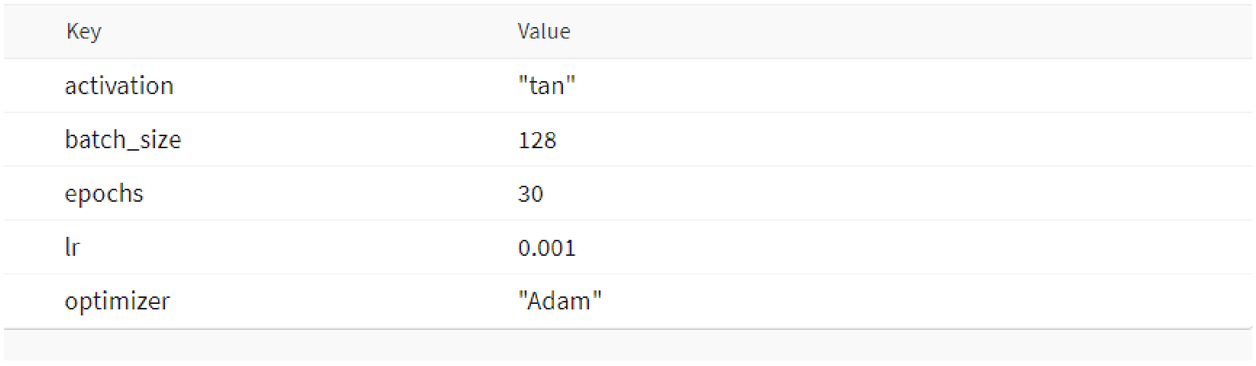


Key	Value
activation	"ReLu"
batch_size	128
epochs	30
lr	0.001
optimizer	"Adam"

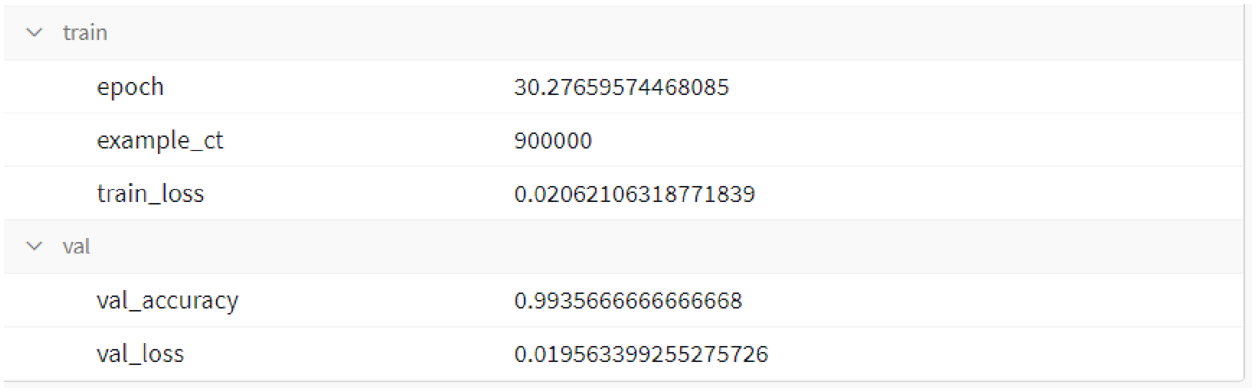


train	
epoch	29.995744680851065
example_ct	893400
train_loss	0.02442167326807976
val	
val_accuracy	0.9929666666666668
val_loss	0.018785003572702408

RUN:lyric-hill-2

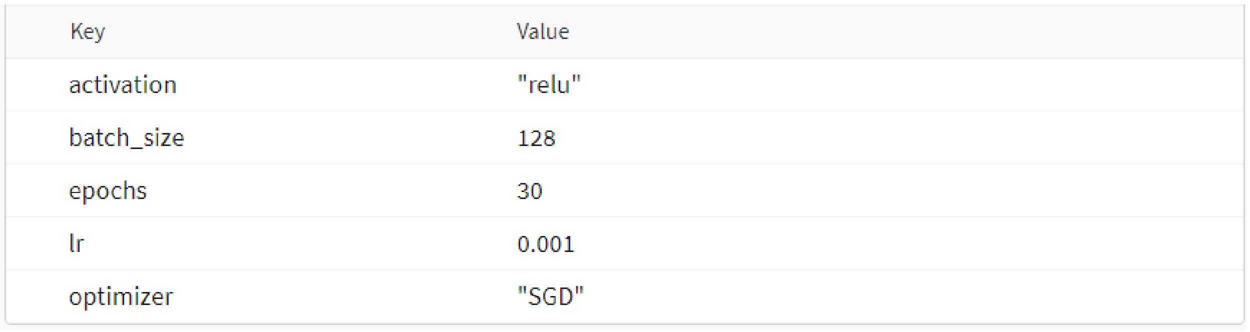


Key	Value
activation	"tan"
batch_size	128
epochs	30
lr	0.001
optimizer	"Adam"

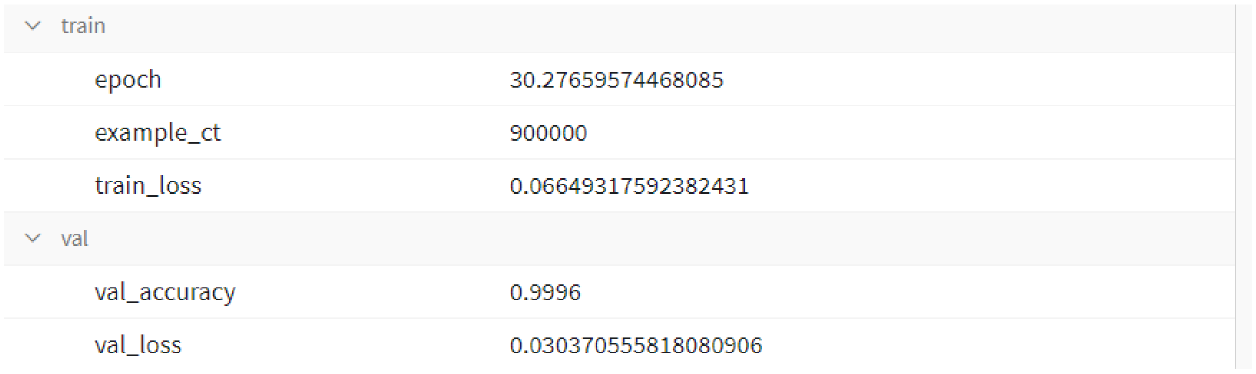


train	
epoch	30.27659574468085
example_ct	900000
train_loss	0.02062106318771839
val	
val_accuracy	0.9935666666666668
val_loss	0.019563399255275726

RUN: major-sponge3

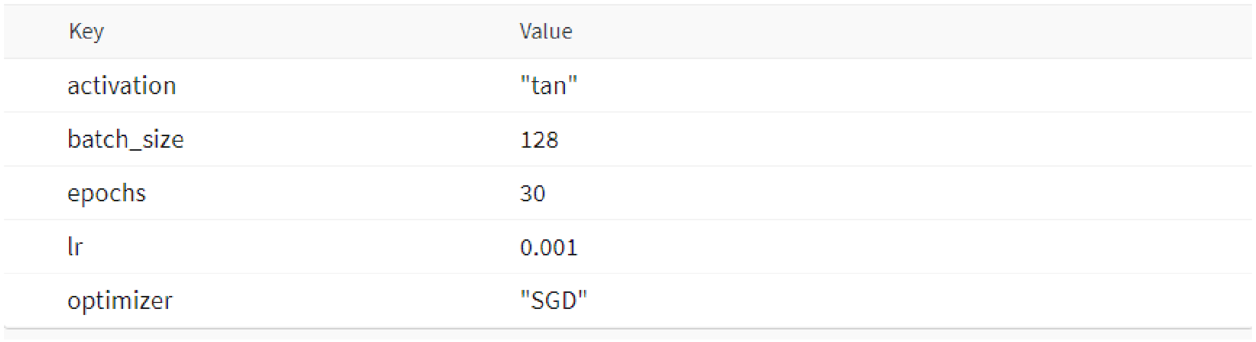


Key	Value
activation	"relu"
batch_size	128
epochs	30
lr	0.001
optimizer	"SGD"

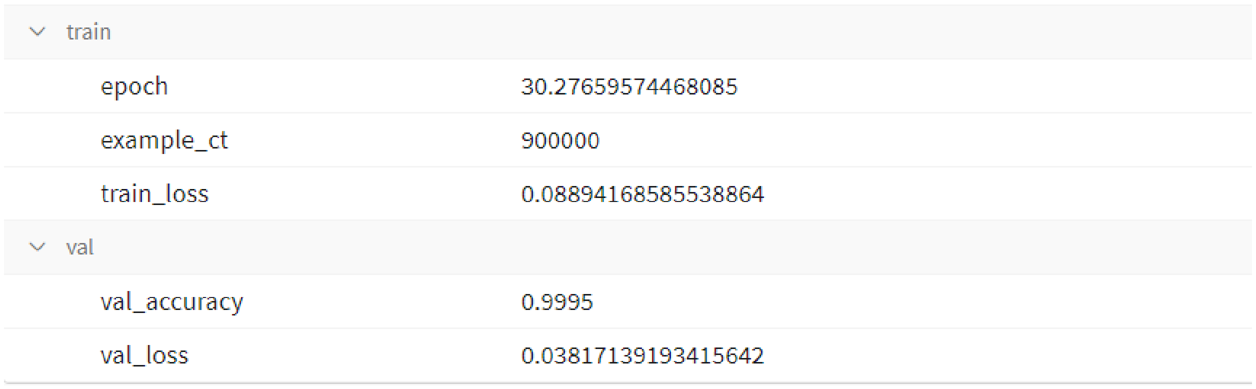


train	
epoch	30.27659574468085
example_ct	900000
train_loss	0.06649317592382431
val	
val_accuracy	0.9996
val_loss	0.030370555818080906

RUN: fiery-pond-4

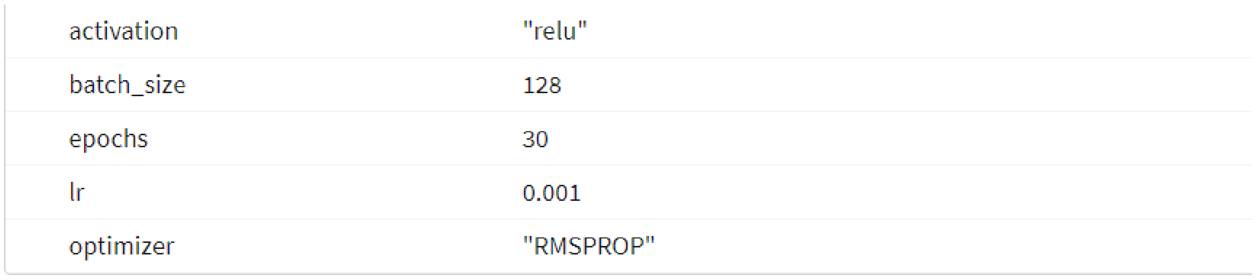


Key	Value
activation	"tan"
batch_size	128
epochs	30
lr	0.001
optimizer	"SGD"

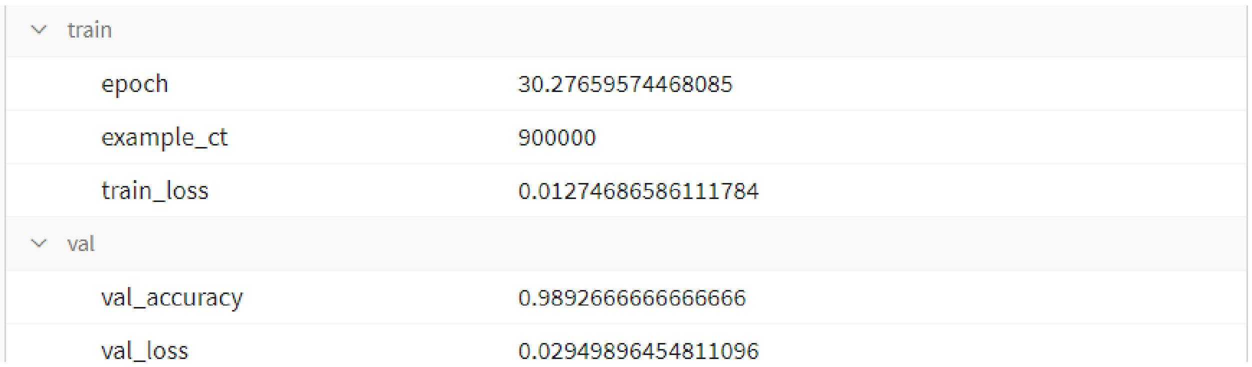


train	
epoch	30.27659574468085
example_ct	900000
train_loss	0.08894168585538864
val	
val_accuracy	0.9995
val_loss	0.03817139193415642

RUN: still-baze-5



activation	"relu"
batch_size	128
epochs	30
lr	0.001
optimizer	"RMSPROP"



train	
epoch	30.27659574468085
example_ct	900000
train_loss	0.01274686586111784
val	
val_accuracy	0.9892666666666666
val_loss	0.02949896454811096

Observations

💡 on varying the activation function from ReLu and Tanh, Tanh performed slightly worse than ReLu.The reason why Tanh might perform worse than ReLU in ResNet-18 is that ReLU has been specifically designed to address the issues that can arise in very deep networks like ResNet-18, while Tanh has some inherent limitations that can make it less suitable for these types of architectures, However the difference was very minute

💡 on varying the optimizer, SGD performed the best on this task followed by Adam and then RMSprop.One possible reason for this could be that ResNet-18 is a deep neural network, and SGD is known to work well in deep neural networks because it can help mitigate the vanishing gradient problem that can occur with other optimizers like RMSprop. On the other hand, Adam is known to work well for a wide range of problems, but may not be as effective for very deep networks.

Another possible explanation could be that the dataset you are using may have certain characteristics that make it more amenable to SGD. For example, SGD may work well for noisy or sparse datasets, while Adam may perform better for datasets with smooth or well-behaved gradients.

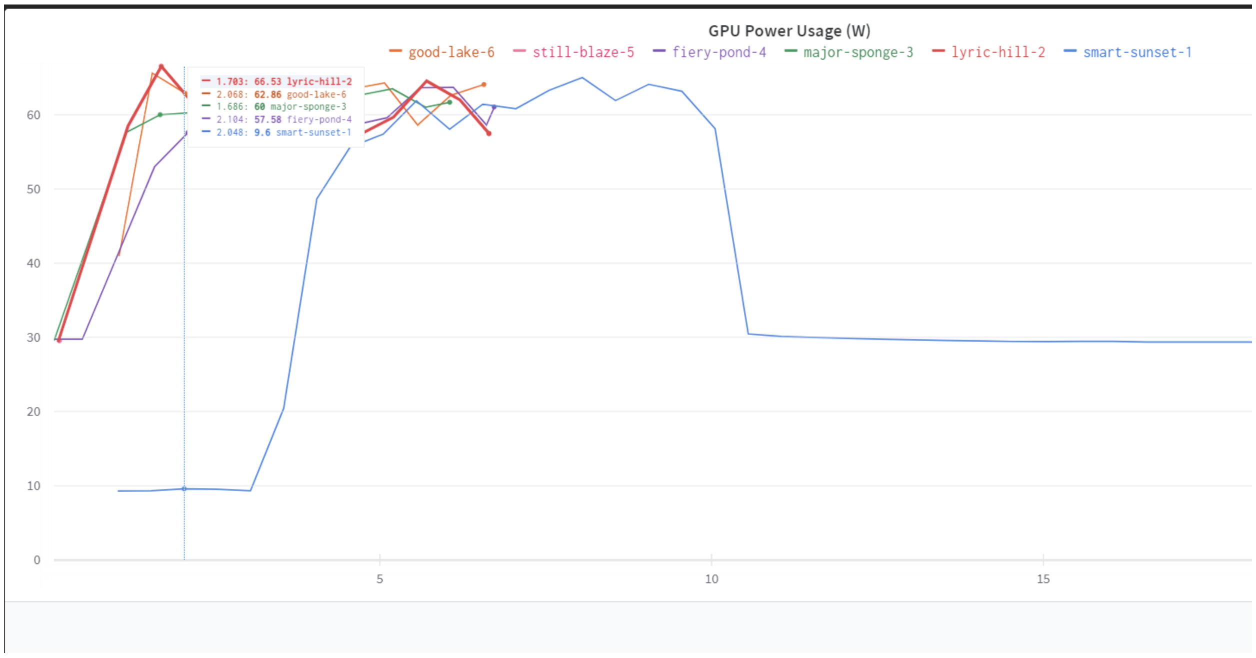
Finally, the specific hyperparameters of the optimizers can also play a role in their performance. For example, the learning rate, momentum, and weight decay parameters can all impact the effectiveness of the optimizer. It's possible that the hyperparameters you used for SGD were better suited to your specific task than the ones you used for Adam or RMSprop.

GPU Observations

GPU power usage refers to the amount of electrical power that a graphics processing unit (GPU) consumes while it is performing computations. GPUs are specialized hardware designed for parallel computing, and they are commonly used in deep learning and other computationally intensive tasks

If GPU power usage is high, it means that the graphics processing unit (GPU) is consuming a significant amount of electrical power while performing computations. High GPU power usage is generally an indication that the GPU is being utilized to its full capacity, which can be a good thing if you are trying to maximize performance.

out of all the runs, the hyperparameters for the run lyric-hill had the highest GPU usage and hence these choice of parameters were most suitable for GPU utilization



GPU temperature tells us how hot the graphics processing unit (GPU) is while performing computations. The temperature of a GPU is an important parameter to monitor because high temperatures can lead to reduced performance and even system instability if the temperature gets too high.

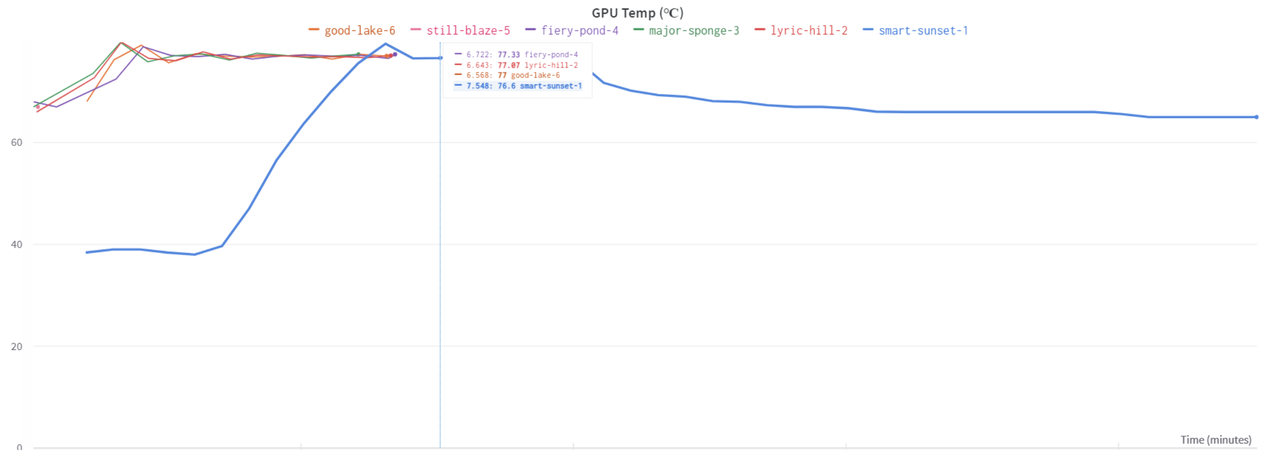
As a GPU performs computations, it generates heat due to the electrical power it consumes. The temperature of the GPU can increase rapidly if it is being utilized heavily, especially if the cooling system of the computer is not sufficient to dissipate the heat.

Monitoring the temperature of the GPU can help prevent overheating, which can lead to damage to the hardware, system instability, and reduced performance. It can also help to identify any issues with the cooling system, such as a fan malfunction or clogged air vents, which can be addressed to prevent damage to the GPU.

In addition to monitoring the temperature of the GPU, it is important to ensure that the cooling system of the computer is functioning properly and that the GPU is not being pushed beyond its safe operating limits. This can be accomplished by adjusting the power and temperature settings of the GPU, optimizing the computations to minimize unnecessary work, and ensuring that the computer has adequate cooling to dissipate the heat generated by the GPU.

Overall, monitoring the temperature of the GPU is an important aspect of maintaining the health and performance of the system, especially in situations where the GPU is being utilized heavily, such as in deep learning applications.

From all the runs, smart-sunset had the most prolonged time of being heated



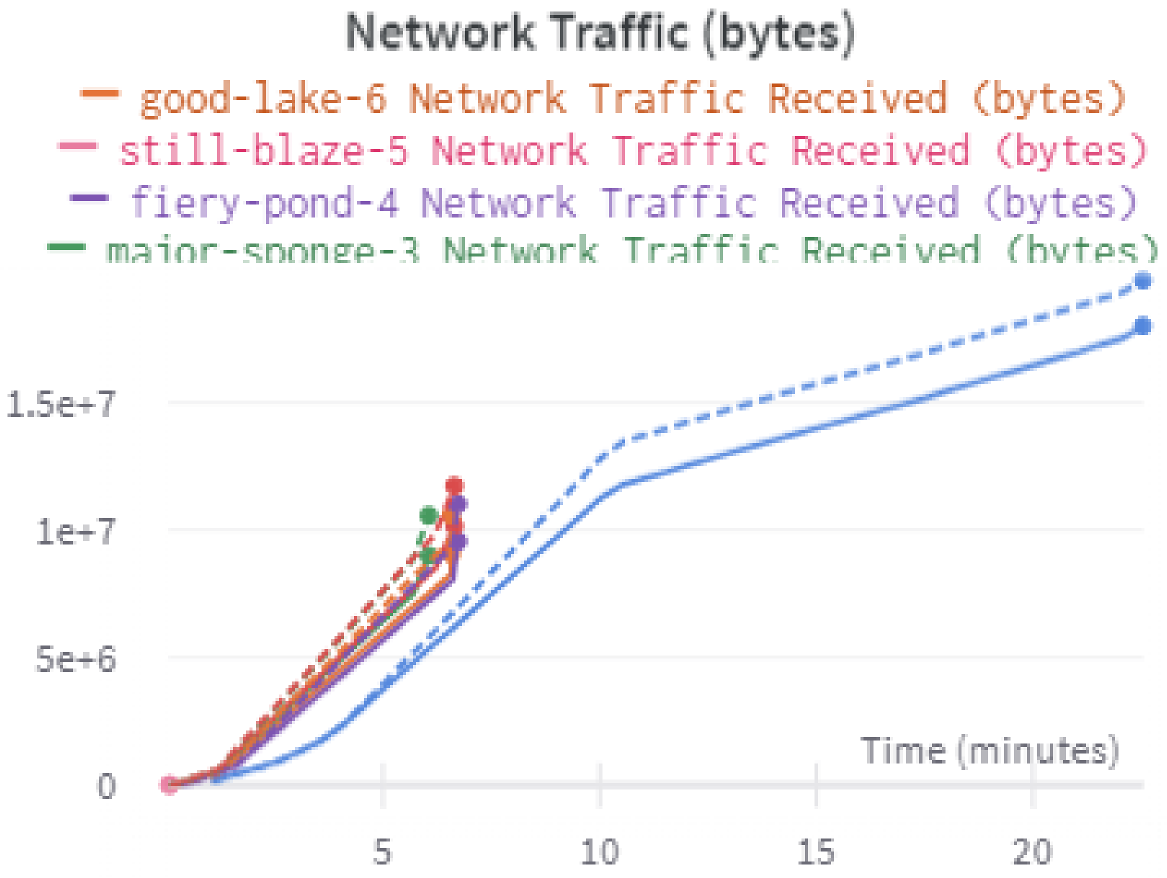
Furthermore, Network traffic during a training run can provide valuable information about the performance and efficiency of the training process. In deep learning, network traffic refers to the data transfer between the CPU and GPU during the training process, which includes transferring the input data, intermediate results, and gradients.

The amount of network traffic during a training run is affected by a number of factors, including the size of the dataset, the batch size, the architecture of the neural network, and the hardware configuration of the system.

Monitoring network traffic during a training run can provide insights into the efficiency of the data transfer between the CPU and GPU. If the network traffic is high, it could indicate that the batch size is too large or the network architecture is not optimized for the available hardware. On the other hand, if the network traffic is low, it could indicate that the batch size is too small, or the network architecture is not utilizing the available hardware to its full potential.

In addition, network traffic can also impact the overall training time, as excessive data transfer can result in bottlenecks that slow down the training process. By monitoring network traffic, we can identify potential bottlenecks and optimize the training process to minimize data transfer and reduce training time.

Overall, monitoring network traffic during a training run can provide valuable insights into the performance and efficiency of the training process, helping to optimize the training process and improve the overall performance of the deep learning model. since the networ traffic for smart-sunset-1 is too high, we can infer that the batch size maybe a bit large

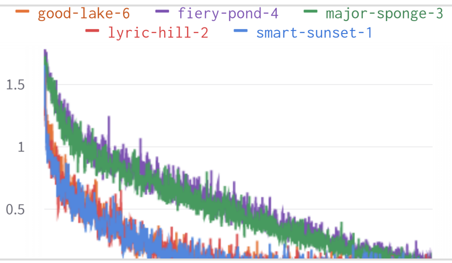


Weights & Biases Report :

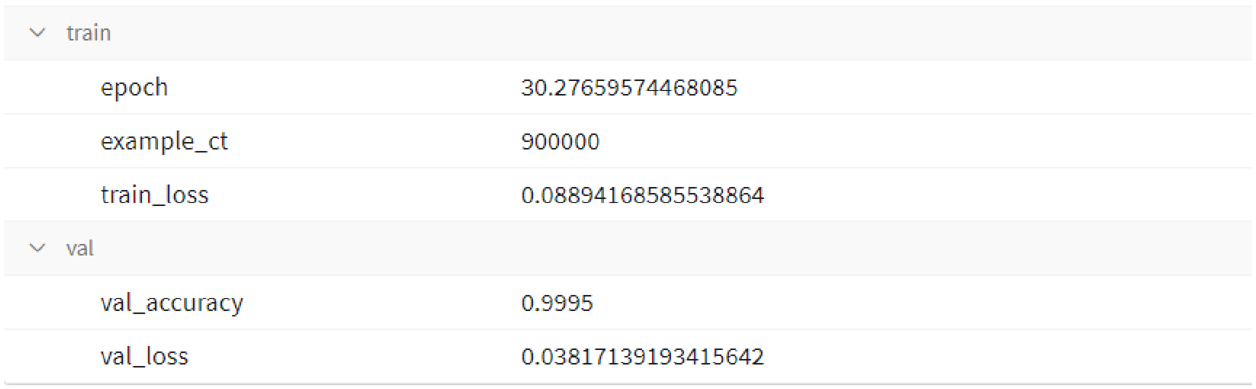
The Weights & Biases Report can be accessed on the following link :

Lab 8 report

https://api.wandb.ai/links/sharma-87/pyp17e56



https://api.wandb.ai/links/sharma-87/pyp17e56



train	
epoch	30.27659574468085
example_ct	900000
train_loss	0.08894168585538864
val	
val_accuracy	0.9995
val_loss	0.03817139193415642