Tool Comparison : WanDB and neptune.ai

Submitted by: Somya Goel[1], Madhu Samhitha[2], Sushrita Yerra[3], Mahima Choudha[4]

Abstract

In this document, we are going to be comparing two Machine Learning (ML) lifecycle platforms, namely **Neptune.AI** and **Weights & Biases (Wandb)**. Developing machine learning projects can be difficult due to their iterative and dynamic nature. This document outlines the various stages involved in the process and explores the challenges associated with each. The focus is on the two most widely used MLOps frameworks: 'neptune.ai' and 'Wandb', and a comprehensive comparison is provided to highlight the most important features for rapid development and deployment of ML solutions.

1. Introduction:

The Machine Learning (ML) development workflow poses several challenges that are unique and distinct from those faced in software development. Alongside the standard challenges such as code quality, maintainability, and scalability, ML development requires addressing issues like data quality and availability, model selection and tuning, interpretability and explainability, deployment and maintenance, reproducibility, and more. While there are numerous tools available in the market addressing each problem at each stage of the life cycle, identifying the appropriate tools for a given set of requirements can be a challenge. This has created a need for Machine Learning Platforms that offer the flexibility required by the development lifecycle while helping manage the entire lifecycle.

insights gained from the evaluation will help teams and projects make informed decisions by understanding the strengths and weaknesses of both tools.

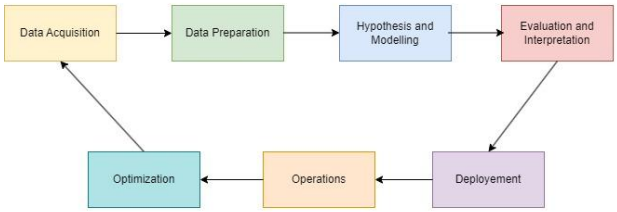


Fig 1.1 ML Lifecycle

1.1 Problem:

One unique challenge in the ML development lifecycle is the absence of a one-size-fits-all solution, and developers often experiment with multiple available tools to improve results. Therefore, it is crucial to identify each tool's strengths and weaknesses and see how they perform concerning the various requirements unique to this workflow.

1.2 Research Question:

This study aims to compare two popular data science tools, WanDB and neptune.ai, to determine which is better suited for various stages of the ML lifecycle, including data preprocessing, model training, hyperparameter optimization, result analysis, and model deployment. The assessment of each tool will consider a set of parameters such as ease of use, collaboration features, integration with popular ML frameworks, and overall performance. The

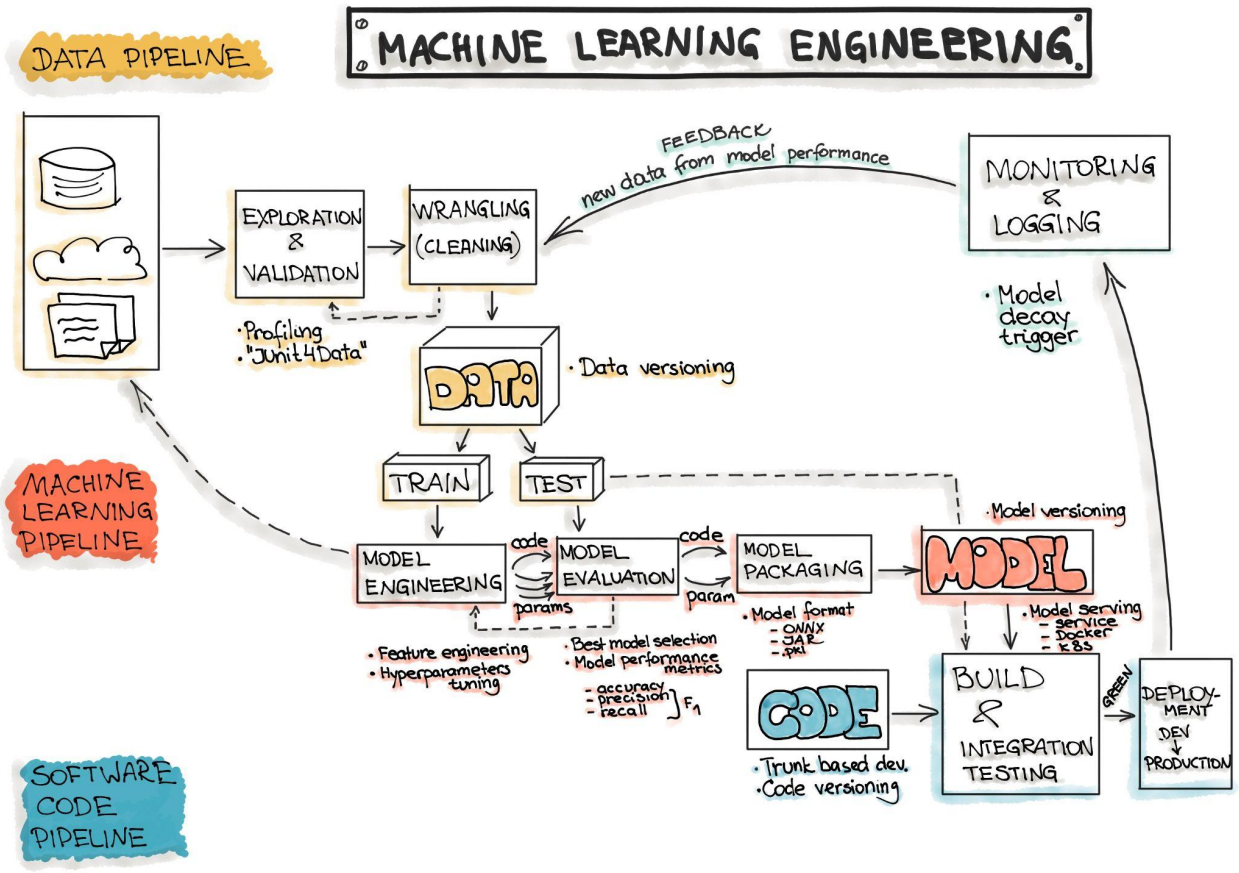


Fig 1.2 Machine Learning Pipeline
(Reference : https://ml-ops.org/img/ml-engineering.jpg)

2. Approach:

In this section, we provide a formal introduction to the goals of the workflow, as well as the machine learning task that we address in order to evaluate the effectiveness of the chosen machine learning toolkits.

2.1 Objective

The available systems and tools offer assistance for various tasks such as data cleaning, labeling, model training, and more. However, ensuring consistency and traceability of data and model artifacts throughout the entire lifecycle and iterations remains difficult. The aim is to evaluate and contrast two frameworks based on their ability to provide these features and how feasible they are for implementation in current systems.

2.2 Task

We are going to compare the 2 tools on a common task: Image Classification. We are running a Convolutional Neural Network (CNN) model on **CIFAR-10** dataset to perform Image classification.

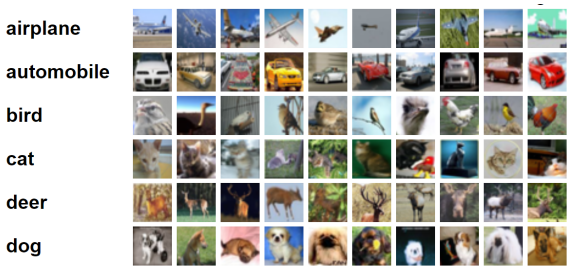


Fig 2.1 CIFAR-10 dataset

2.3 Experimental Setup

In the initial implementation, Python 3 is used along with various libraries including PyTorch, TorchVision, numpy etc. These libraries are installed using Conda,

and the experiments were conducted using macOS M1 in Google Colab environments. Google Colab is a hosted environment that allows users to run Python code on Jupyter notebooks and share them with collaborators. It also provides GPU support (specifically Nvidia K80 Series) to speed up the experimentation.

2.3.1 Uniformity

To ensure reproducibility we have set up a seed value. Training Neural networks involves random elements, such as weight initialization, dropout, or sampling of mini-batches. Setting a seed value, fixes the initial random state, every time you run the training process with the same seed, we obtain the same random numbers and, the same initial conditions for your model. This is useful to compare different training runs, debug issues, or reproduce specific results. Below is a code snippet highlighting the same.

```
SEED = 1234

random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed(SEED)
torch.backends.cudnn.deterministic = True
```

Fig 2.2 Code snippet of SEED value

- random.seed(SEED) - this ensures that random number generated within the Python environment is consistent across different parts the code
- np.random.seed(SEED)- this ensures the random number generated in the NumPy library is consistent.
- torch.manual_seed(SEED)- this ensures the random number generated in the Pytorch library is consistent.
- torch.cuda.manual_seed(SEED) -random number generator for CUDA operations in PyTorch on GPU for consistency.

3. Methodology:

This section elaborates on the dataset and model being used in the image classification task and the steps being used for training the model. It also explains the integration of 'neptune.ai' and 'Wandb'.

3.1 Model Training

Dataset: 'CIFAR-10' consists of 60,000 color images of 32*32 pixels arranged in 10 distinct classes. This dataset is widely used for image classification tasks and is recognized as a significant benchmark in the field of Machine Learning and Computer Vision research. It is frequently used to evaluate the performance of different deep learning models, including CNNs.

Model: AlexNet is a deep convolutional neural network architecture that was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012. It is one of the first deep learning models to achieve state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which is a popular benchmark for image classification.

AlexNet consists of eight layers, including five convolutional layers and three fully connected layers. The convolutional layers are designed to learn feature representations of the input image at different scales and orientations, while the fully connected layers are used for classification. The model also uses a technique called ReLU activation, which helps to speed up training and prevent overfitting by introducing non-linearity into the model. It also performs well on the CIFAR-10 dataset.

Data Preparation: CIFAR-10 is already divided into 50,000 images for training and the remaining 10,000 images for testing. We then split the training images further into training images and validation images in a 90-10 split. Then, the images were resized to the appropriate input size for AlexNet, and pixel values were normalized.

Model Training: We trained the model over 25 epochs, with learning rate of 10^{-3}, batch size of 256 and using Cross Entropy Loss. 'Adam' optimizer was also further used.

Model Evaluation: The model was then evaluated on the basis of training, validation and test accuracy.

3.2 Integrating the model to Neptune AI

1. Sign up for a Neptune account and create a new project.
2. Install the neptune-client library in your Python environment using pip.
3. Run the code snippet shown below:

```
[ ] import neptune
    from getpass import getpass

    api_token = getpass("Enter your private Neptune API token: ")
```

```
[ ] workspace = "somyagoel"
    project_name = "520-final-project"
    project = f"{workspace}/{project_name}"
```

```
run = neptune.init_run(
    project=project,
    api_token=api_token,
    capture_hardware_metrics=True,
    capture_stderr=True,
    capture_stdout=True,
)
```

Fig 3.1 Code snippet to connect to neptune.ai

4. Access the private API token from a personal Neptune account.
5. Before the training process, log in the hyperparameters such as learning rate, dataset size, optimizer, loss function, epochs etc.
6. During the training process, log in the training loss/accuracy as well as validation loss/accuracy. Neptune AI also automatically plots a chart of these metrics in real time.
7. Model architecture, model weights are also logged into the tool for easy visualization of the model.
8. View and analyze the experiment results in the Neptune web interface.

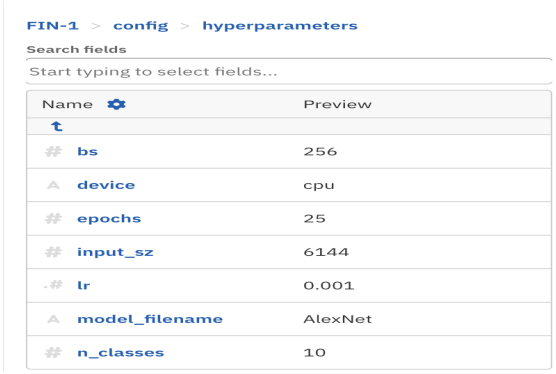


Fig 3.2 Logging of hyperparameters in neptune.ai

```
train_loss, train_acc = train(model, train_iterator, optimizer, criterion, device)
run["training/batch/loss"].append(train_loss)
run["training/batch/acc"].append(train_acc)
valid_loss, valid_acc = evaluate(model, valid_iterator, criterion, device)
run["validation/batch/loss"].append(valid_loss)
run["validation/batch/acc"].append(valid_acc)
```

Fig 3.3 Code snippet to log training and evaluation results to neptune.ai

```
#Logging the model architecture to Neptune AI

with open("model_arch.txt", "w") as f:
    f.write(str(model))

run["config/model_arch"].upload(f"./model_arch.txt")
```

Fig 3.4 Code snippet for uploading model architecture to neptune.ai

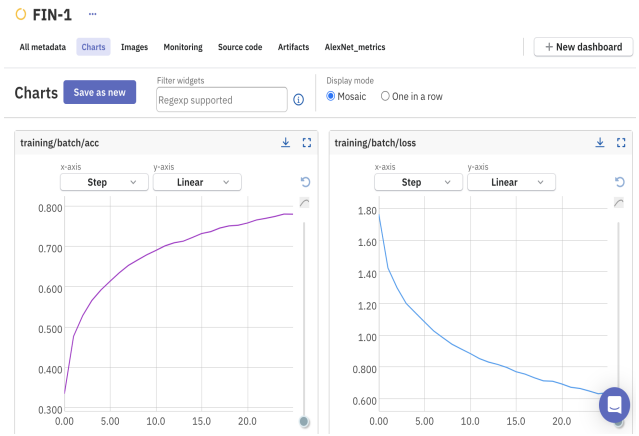


Fig 3.5 Training results displayed on neptune.ai

3.3 Steps to integrate WanDB

1) Sign up for Wandb to create an account and set up a project.
2) These are the following requisites to authorize the Wandb execution in the development environment:

   i) Install 'wandb' using pip
   ii) Authorize the connection to your account using the API key provided.
   iii) Connect to the project using wandb.init() shown in the figure below.

```
import wandb
import random
#wandb.finish()
# start a new wandb run to track this script
wandb.init(
    # set the wandb project where this run will be logged
    project="520",

    # track hyperparameters and run metadata
    config={
    "learning_rate": 1e-3,
    "architecture": "Alex-Net",
    "dataset": "CIFAR-10",
    "epochs": 25,
    "Optimizer": "Adam",
    "Criterion" : "CrossEntropyLoss",

    }
)
```

Fig 3.6 Code snippet to connect to Wandb

A successful connection should log the run as shown below -

```
wandb: Currently logged in as: mvangara (ssmm). Use `wandb login --relogin` to force relogin
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230419_014715-gi0dvch7
Syncing run ethereal-cherry-4 to Weights & Biases (docs)
View project at https://wandb.ai/ssmm/520
View run at https://wandb.ai/ssmm/520/runs/gi0dvch7
Display W&B run
```

3) Metadata and hyperparameters such as Learning Rate, dataset, number of epochs, optimizer, loss function used are stored in the tool.

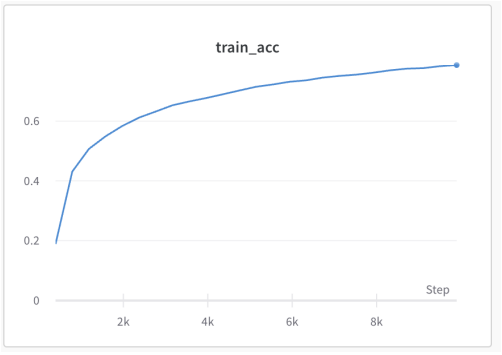
4) Real-time training and validation accuracies were logged to help understand the performance of the model better.

```
wandb.log({"train_loss":train_loss})
wandb.log({"train_acc":train_acc})

wandb.log({"valid_loss":valid_loss})
wandb.log({"valid_acc":valid_acc})
```

Fig3.7 Code snippet to log training and evaluation results to Wandb

5) Results were visualized for each epoch.



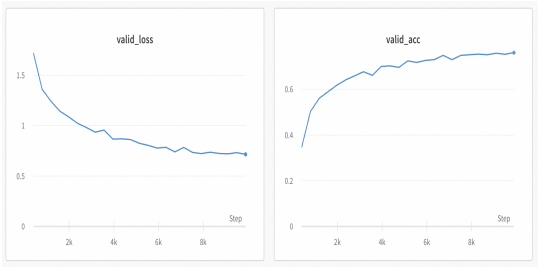
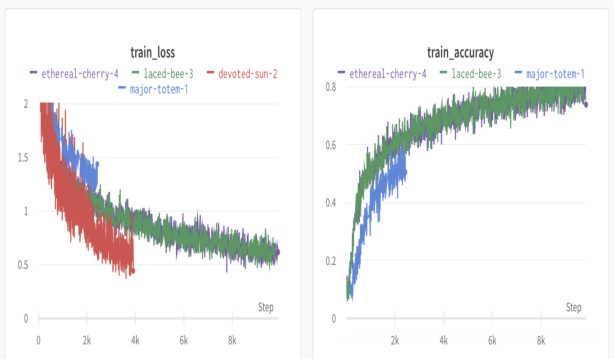


Fig 3.8 Training and validation results displayed on Wandb.

7) Wandb allowed for model performance comparison across multiple runs.

a) Performance metrics -



b) System performance metrics -

Fig 3.9 Model and system performance metrics across different runs of the model.

3.4 Evaluation Criteria

In this section, we have identified a set of evaluation criteria that will be used to perform a comparative analysis of the two mentioned tools. These criteria have been carefully chosen by examining various parameters that are crucial to the machine learning lifecycle. The objective is to assess the overall effectiveness of the

tools and provide insights into how they fare in comparison to each other.

3.4.1 Dataset versioning: It is essential to keep track of the different versions of the dataset used in a machine learning workflow. This ensures that the data used for model training and evaluation is consistent and reproducible. It helps in identifying any changes or errors in the dataset and comparing the performance of the model across different versions of the dataset.

3.4.2 Model versioning: Similar to dataset versioning, it is important to keep track of the different versions of the model. This helps in reproducing the results, comparing the performance of different models, and identifying the changes made to the model during development.

3.4.3 Collaboration: Collaboration is crucial in a machine learning workflow, where multiple developers, data scientists, and stakeholders work together on a project. Collaborative tools allow team members to share code, data, and insights, and work together towards a common goal.

3.4.4 Framework integration: Framework integration enables developers to use popular machine learning frameworks, such as TensorFlow or PyTorch, in their workflow. This provides access to a rich set of tools and libraries, simplifies the development process, and improves the ease of working with the workflow tool.

3.4.5 Hyperparameter optimization integration: Hyperparameter optimization helps in finding the optimal values for these parameters, which can significantly improve the performance of the model. Providing different integration options is essential for developers to test out the most suitable one for their use case.

3.4.8 Report generation and management: Report generation and management tools help in generating reports that summarize the results of the machine learning workflow. These reports can be used for communicating the results to stakeholders, validating the results, and improving the workflow.

3.4.9 Model visualization: Model visualization helps in understanding the behavior of the model. Model visualization ability helps us to see the model architecture in terms of layers and visually see the results of model performance in the form of images and charts.

3.4.10 Understandability: This metric is to assess the documentation available in the form of quick guides or tutorials explaining the various features of the workflow tool. Since, the workflow tool involves a series of integrations and stages, it can become very complex making understandability a vital measure to estimate the learning curve.

3.4.11 Usability: This metric is to assess how easy is the workflow tool for day to day usage in terms of how much time and effort needs to be spent in the usage. Ideally, the tool should become second nature on usage, i.e, being very seamless.

3.4 Evaluation Scale

We are using a 3 point numeric scale to rate how well a tool performs across different evaluation criterias.
This rating is given to every tool for each of the aforementioned evaluation criterias. This rating is given on the basis of some predefined rubrics.
They are as follows:

‘1’	**Tool does not support this functionality**
‘2’	**Tool supports this functionality and performs satisfactorily.**
‘3’	**Tool supports this functionality and performs really well.** (Tool performs “really well” here means that it is very straightforward to use and has additional features as well.)

We have provided these ratings based on our experience with working with these tools and after performing a thorough analysis of the tools based on each evaluation criteria.

4. Comparative Analysis and Results

4.1 Comparison on functional requirements

1. Dataset Versioning:

Both Wandb and Neptune AI offer dataset versioning capabilities. Neptune.ai has a straightforward display where you can log in the name of the dataset used as well as the size of training and validation sets (as shown in Figure 4.1)
Wandb has a similar display. The user chooses which information to log in. It is very straightforward to use. Each run, the user can log in updated dataset information, which makes it really easy to store different versions of the dataset.

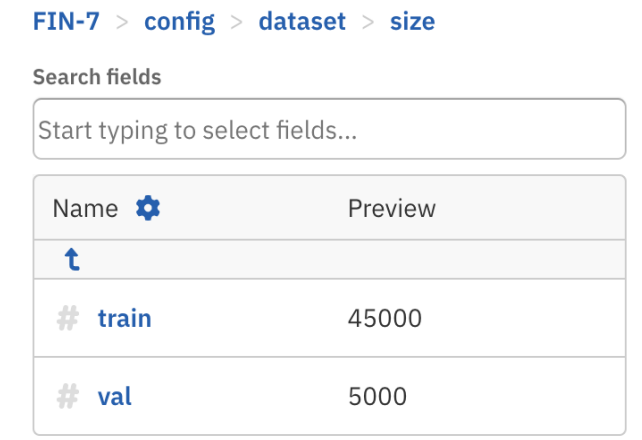


Fig 4.1 Dataset information displayed on neptune.ai

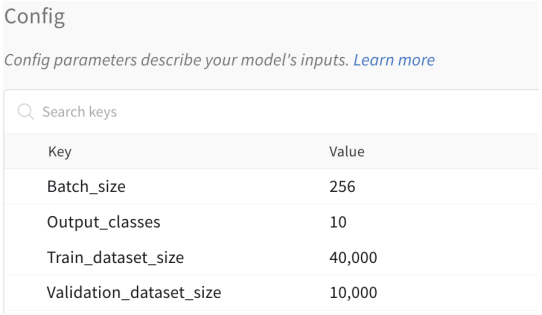


Fig 4.2 Dataset information displayed on Wandb

2. Model Versioning:

- Wandb provides a comprehensive model Registry which allows to not only log the models but also store them as artifacts.
- The Model registry at Wandb has the following features -
    - Documentation and annotation of hyperparameters
    - Tracking and lineage of different experiments
    - Model governance and performance tracking
    - Lineage and visualization of nodes.

- Neptune.ai also allows the user to store model information and store them as artifacts. It is very straightforward to use. Neptune.ai also provides us with code snippets and proper documentation to store models as artifacts.

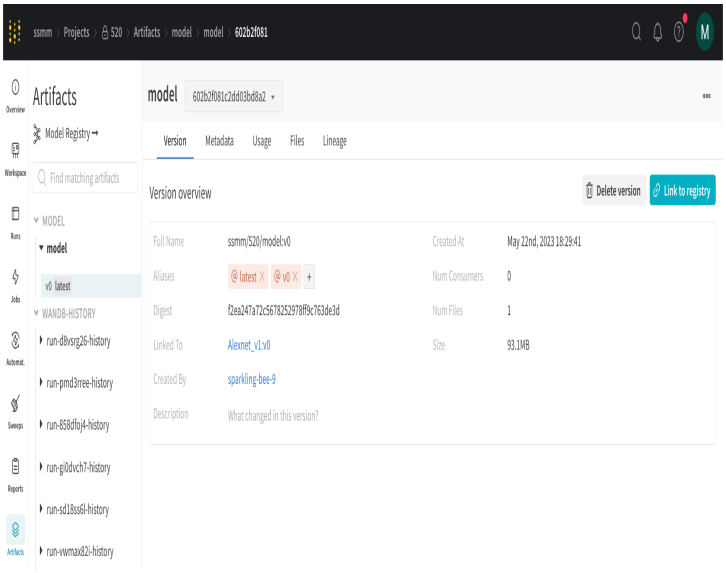


Fig 4.5 Single Run View

Fig 4.6 Single Run Performance

Fig 4.7 Run Summary

Fig 4.8 Multiple Run Views

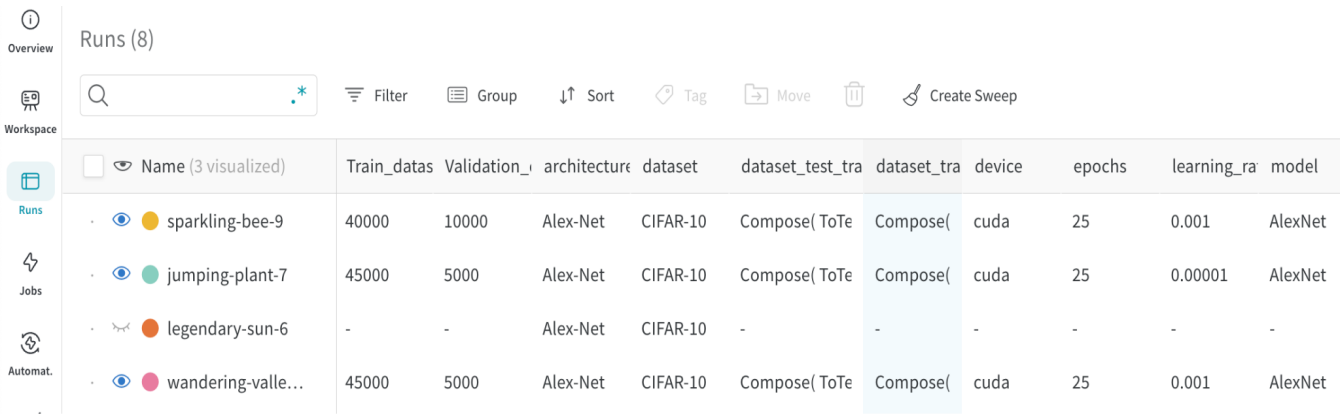
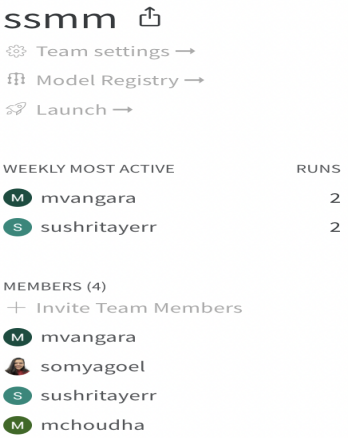


Fig 4.9 Table View of Multiple Run

3. Collaboration:

Wandb and neptune.ai offer collaborative support to facilitate teamwork. neptune.ai offers a unique capability allowing comments enabling teams for an easy, interactive way to share feedback and collaborate. Though, NeptuneAI's collaborative features are all paid (for individual workspaces) whereas Wandb supports collaboration in the free version.

- Wandb offered a single point of logging for various users.
- The different runs across the contributors were plotted in a single place for efficient comparisons.



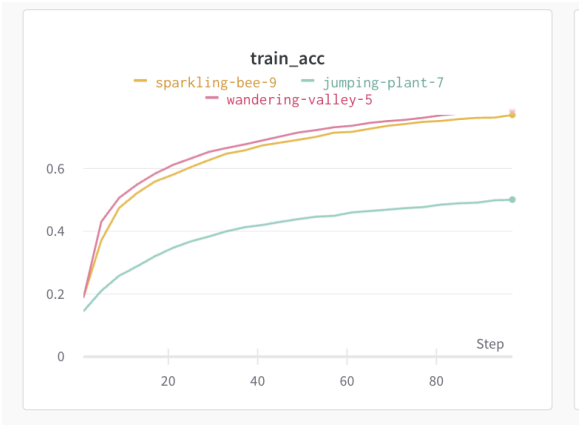


Fig 4.10 List of collaborators on Wandb's UI

1. **Framework Integration:** Both these tools provide integration for an array of popular ML frameworks (eg. pytorch, keras, tensorflow etc.)in the market. The integrations are equally straightforward and simple in both these tools making it easy for developers to begin working or integrating their existing work with these tools.

2. **Report generation and management:** Both tools provide capabilities to capture experiment data, including metrics, hyperparameters, code snapshots and organize them in a structured manner. Neptune.ai allows for the user to download their results in the form of .jpeg files or .csv files. So, all this information can be

presented to stakeholders. Wand additionally provides to assist with generating reports automatically based on the captured experiment data through customizable report templates. These reports are also interactive to modify the views being displayed. Here's the link to one of the reports we generated through Wandb :

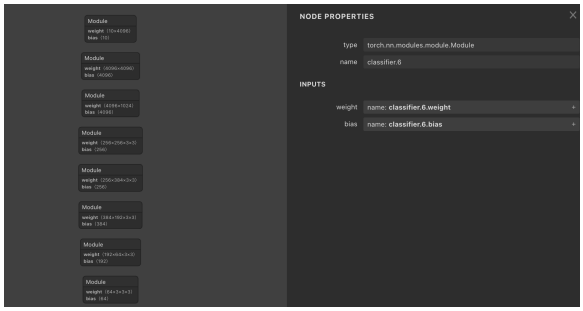
https://wandb.ai/ssmm/520/reports/Report-Generation-for-CIFAR-10--Vmlldzo0NDMwMzM4?accessToken=buko1oaq9xd9ewwl0mjsarjkvpkmyd49ew6iagk4z4g4uqf2qau267pfydrqp3j2

3. **Model Visualization:**

Neptune.ai allows users to upload their model architecture to the tool as a .txt or image file. It even allows the users to upload model results as images to the tool. Users can also upload model information as '.pth' files. This helps to free up space on their local working directories and also allows for better model visualization. Neptune.ai's ability to upload all this information on its tool allows for better model visualization. Neptune.ai also allows you to view model performance results in the form of charts.

Wandb doesn't allow you to upload model architecture in a .txt file for better model visualization. You can upload model information as a .pt file but that gives a very limited view of the model information. .It also doesn't allow you to upload results as .jpeg or .png files to the tool. But, you can view model performance results in the form of charts.

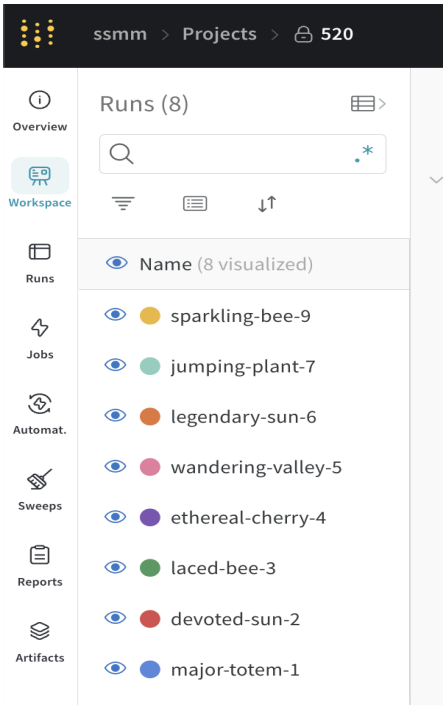
Saved model limited visualization on wandb -



4.2 Comparison on non-functional requirements

1. **Understandability:** Both the frameworks have extremely high quality documentation and support through readmes, tutorials, blogs and an active and engaged community. WanDB has a git repository with examples of the many integrations that are supported as part of the application. Neptune AI also has examples of the many integrations it supports as part of it's website and the git repository. For a first-time user, there are step-by-step quick start guides that help one get started in minutes without issues. Neptune also additionally offers video tutorials especially for the starter tutorials. While both tutorials are excellent and beneficial in their own right, Neptune AI gains a slight advantage by catering to a wider range of users through the inclusion of videos and interactive tutorials in the website.

2. **Usability:**



3. **Debuggability:** WanDB and NeptuneAI both allow users to log and monitor errors that occur during training and inference. Neptune AI allows users to console log, print stack traces which offers additional resources to identify the source of the error. Both these tools also provide interactive visualizations which makes it useful to debug by visually inspecting the model's behavior. They also provide systems metrics

logging and monitoring which provides further information into the consumption of the models helping us guage the source of any error.

4.3 Final comparison

Criteria	Wa nd B	Neptune AI	Commen ts
Dataset Versioning	**3**	**3**	Both offer the same functiona lity.
Model versioning			
Collaboration	**3**	**2**	Neptune. ai only allows for collabora tion in its paid version.
Framework Integration	**3**	**3**	
Report Generation and Management	**3**	**2**	
Model Visualization	**2**	**3**	
Understandab ility			
Usability			
Debuggability			

RELATED WORK

There are many other tools in this space, this section lists a few other popular ones below:

TensorBoard: a visualization toolkit for TensorFlow that can be used for tracking and visualizing experiment metrics, graph structures, and other aspects of model training and evaluation.

MLflow: an open-source platform for managing end-to-end machine learning workflows, including tracking experiments, packaging code, and deploying models.

Sacred: a tool for experiment management and configuration, particularly focused on reproducibility and organizing experiments as a series of "configurations" with associated results.

Comet.ml: a cloud-based platform for experiment tracking and collaboration, with features such as interactive visualizations, real-time logging, and integrations with popular machine learning libraries.

Polyaxon: an open-source platform for building, training, and deploying machine learning models, with features for experiment tracking, hyperparameter optimization, and model versioning.

Conclusion

References

1. https://neptune.ai/vs/wandb
2. https://ml-ops.org/img/ml-engineering.jpg
3. https://www.cs.toronto.edu/~kriz/cifar.html
4. https://pytorch.org/hub/pytorch_vision_alexnet/
5. https://pytorch.org/docs/stable/nn.init.html
6.
7.