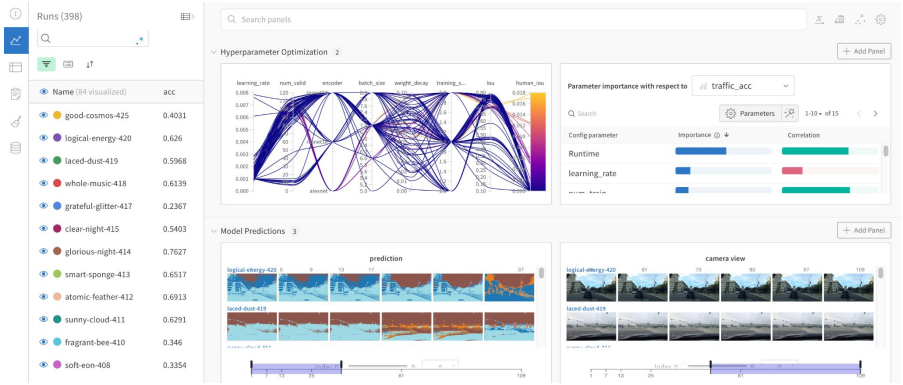
Introduction to Wandb

By Michael Kireeff and Aisha Opaluwa

What is WandB?

WandB, or Weights and Biases, is a powerful tool designed to help data scientist and machine learning engineers to track and manage their experiments easily through performance visualization and hyperparameter tuning.

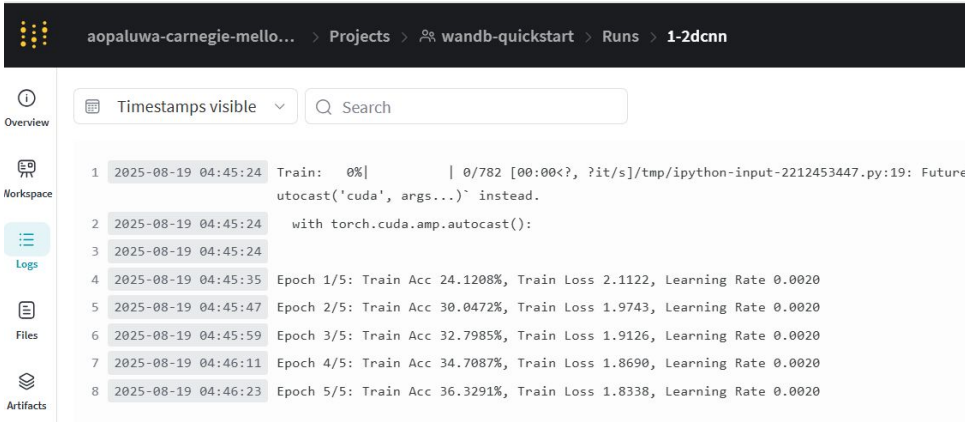
It integrates seamlessly with popular machine learning frameworks like PyTorch and Tensorflow through its client library.

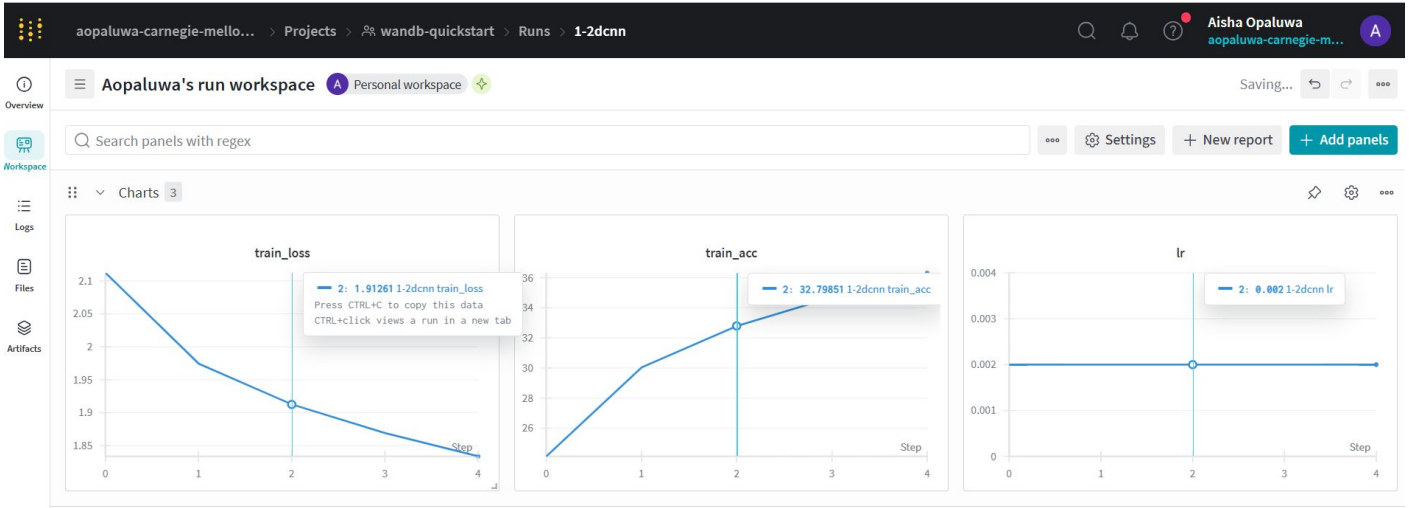


What are the Key Features of WandB

Performance Logging and Visualization

- Log various metrics and visualize them in real time
- Keep track of different model versions and training runs

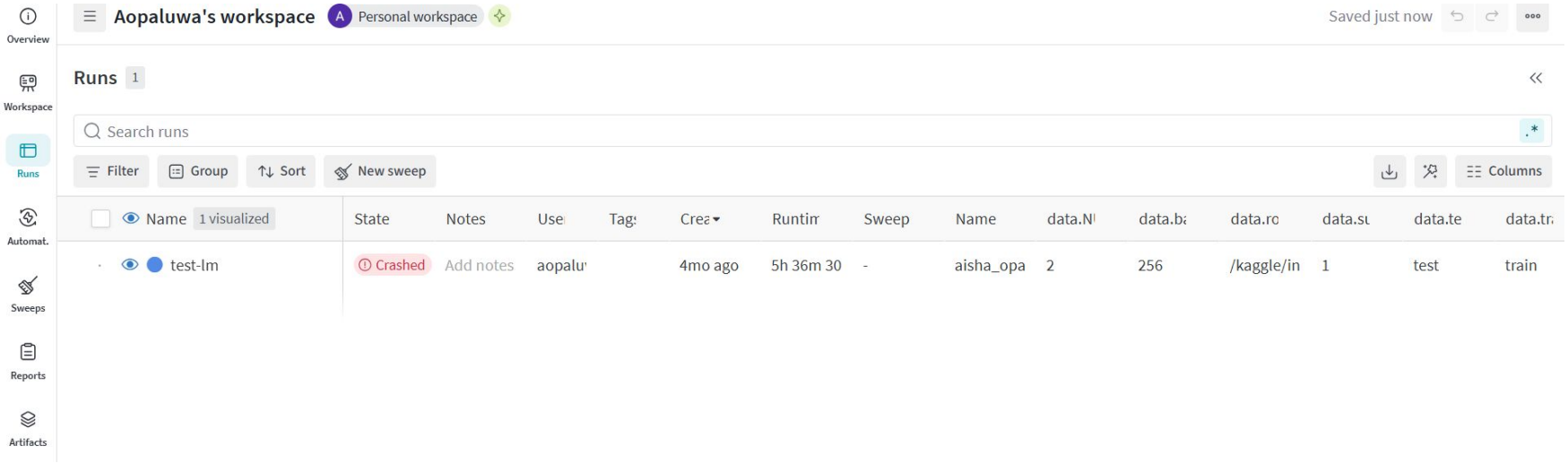




What are the Key Features of WandB

Experiment Resumption

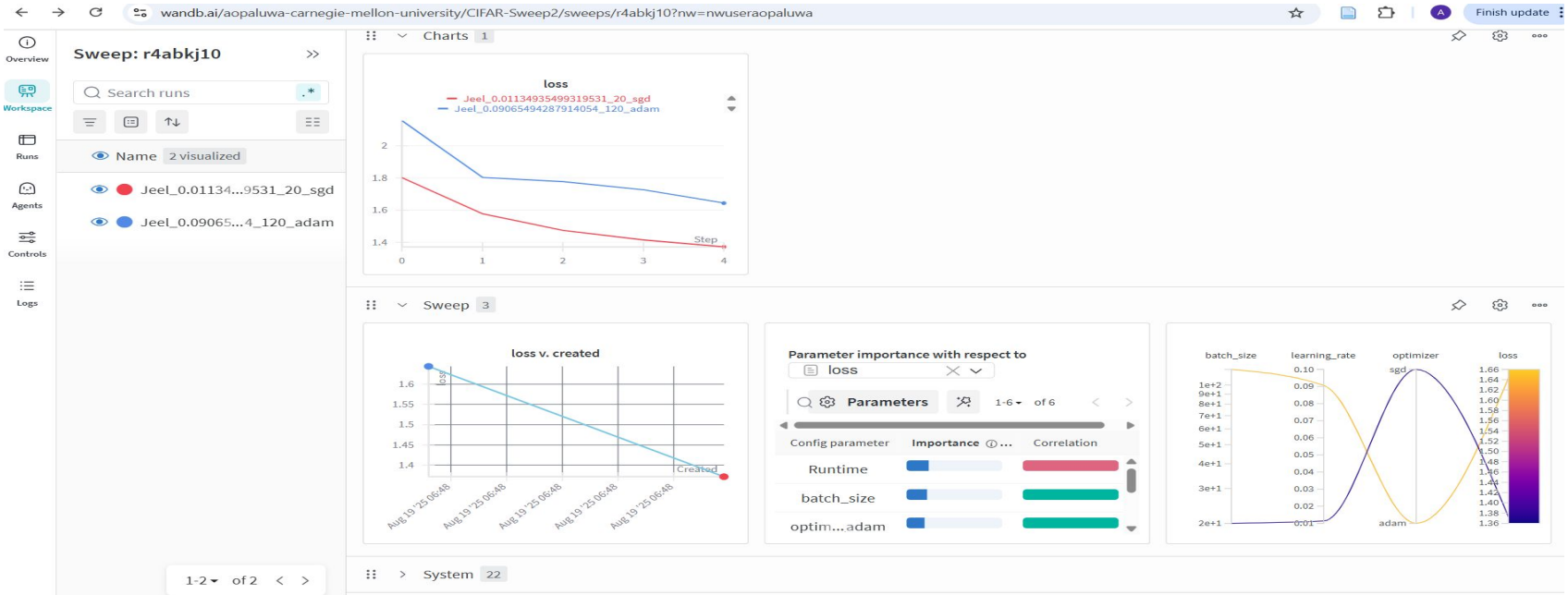
- Easily restore model's state and resume experiments in case of runtime crashes



What are the Key Features of WandB

Hyperparameter Tuning

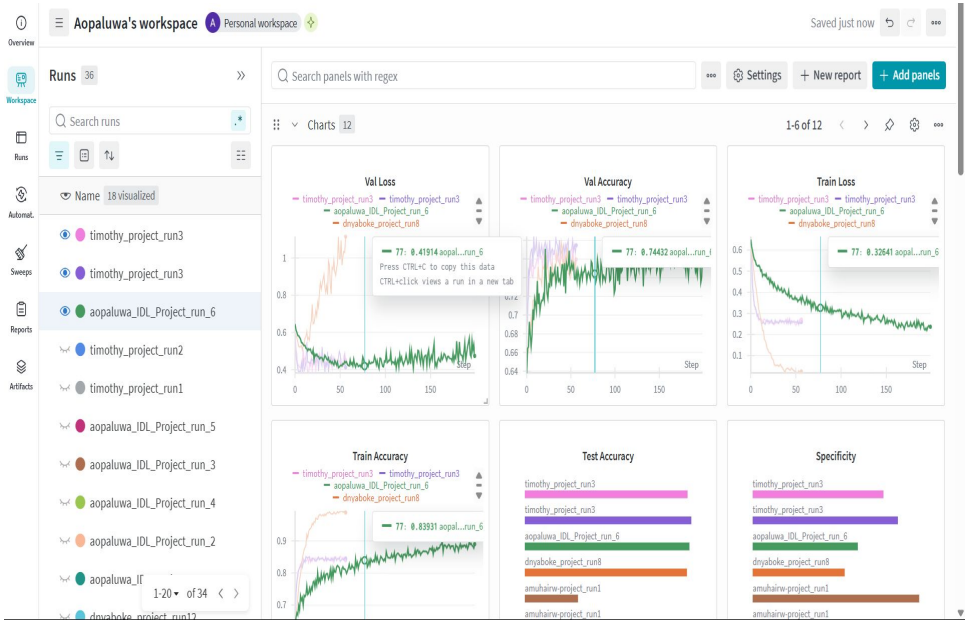
- Automate hyperparameter tuning to find the best configuration for your models through sweeps

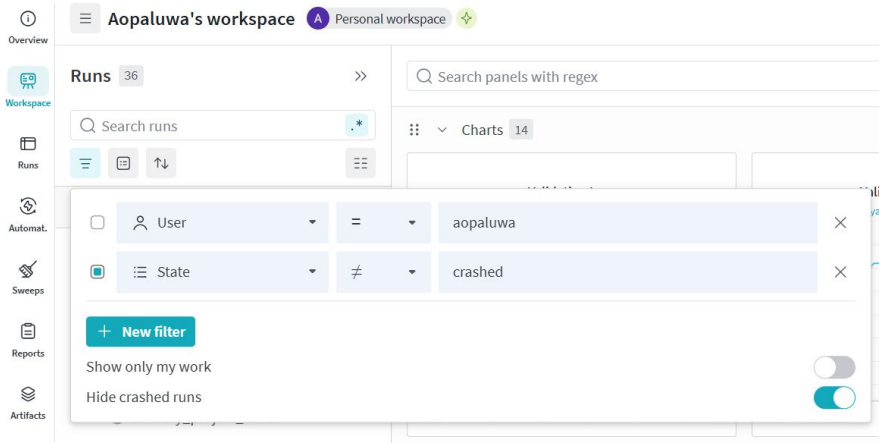


What are the Key Features of WandB

Collaboration

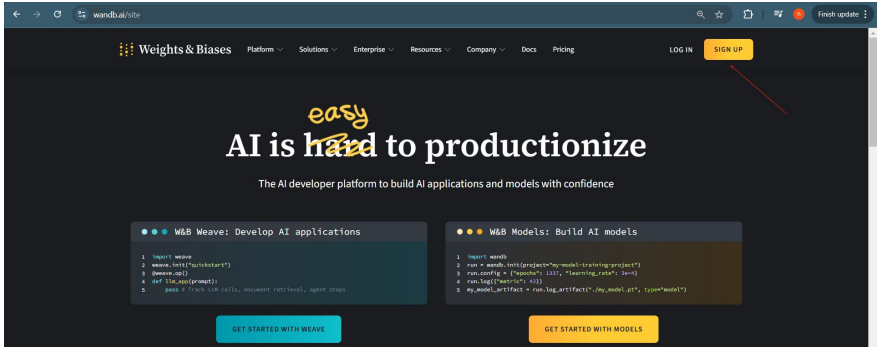
- Share and collaborate your projects with team members and experiment on different ablations while keeping track of others progress

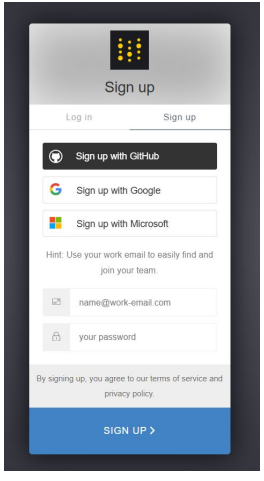




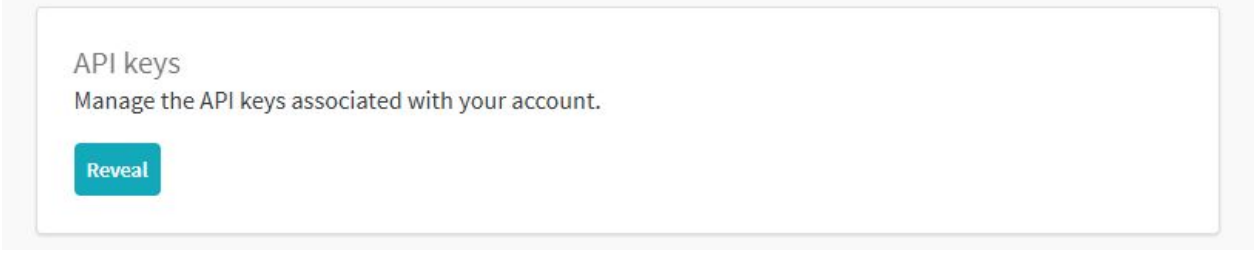
Setting Up WandB Account and an API Key

Creating an account: https://wandb.ai/site





Generating an API key: https://wandb.ai/settings



WandB Installation and Notebook Setup

Wandb Installation for Python

Installation and Libraries

```
[ ]  ## Installing WandB
     !pip install wandb -qqq
```

Notebook Setup

- Dataset: CIFAR10
- Neural Network: Convolutional Neural Network
- Code Format: all functionalities are built in functional blocks for automated access

Credits to Fall 2024 TAs for the Notebook

WandB Login and Initializing Project

Wandb Login in Notebook:

```
[ ] import wandb, os
    os.environ['WANDB_API_KEY'] = "7328d336610dec777bdecff906dc57e2a464d0b4"#your key here
    wandb.login()
```

```
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more information.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit: ··········
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
True
```

Initializing Project:

```
[ ] run_config = {
        'model': '1-2dcnn',
        'optimizer':'sgd',
        'lr': 2e-3,
        'batch_size':64,
        'epochs': 5
    }
```

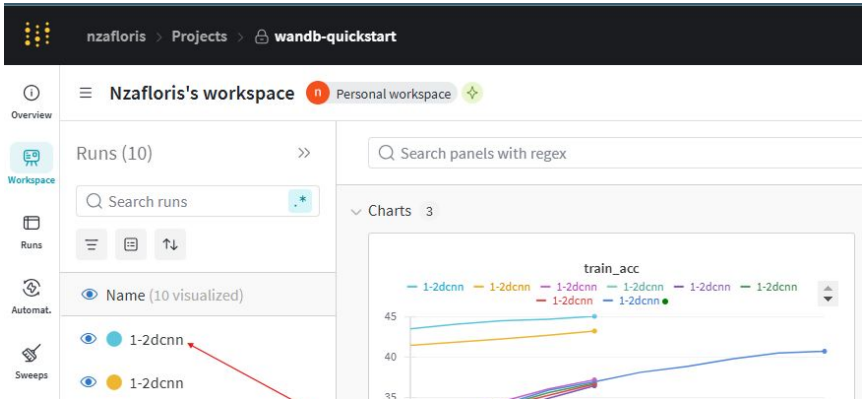
```
[ ] run = wandb.init(
        project="wandb-quickstart",
        job_type="model-training",
        name=run_config['model'],
        config=run_config
        )
```

```
wandb: Currently logged in as: nzafloris. Use `wandb login --relogin` to force relogin
Tracking run with wandb version 0.18.7
Run data is saved locally in /content/wandb/run-20241218_123851-96ynisfu
Syncing run 1-2dcnn to Weights & Biases (docs)
View project at https://wandb.ai/nzafloris/wandb-quickstart
View run at https://wandb.ai/nzafloris/wandb-quickstart/runs/96ynisfu
```

Logging Metrics and Saving Models

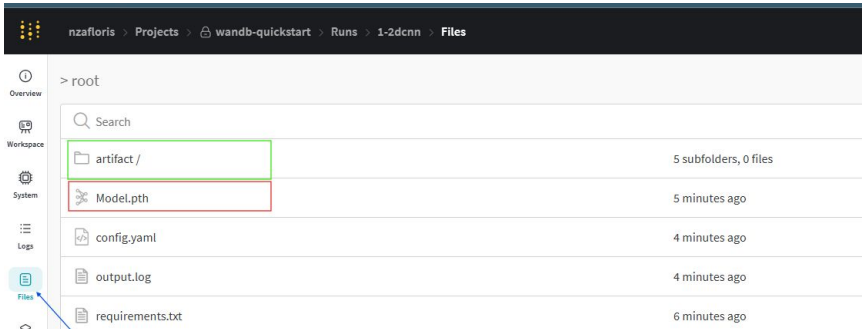
Logging Metrics

```
# What to log

metrics = {
    "train_loss":train_loss,
    "train_acc": train_acc,
    'lr': lr
}

# Log to run
wandb.log(metrics)
```



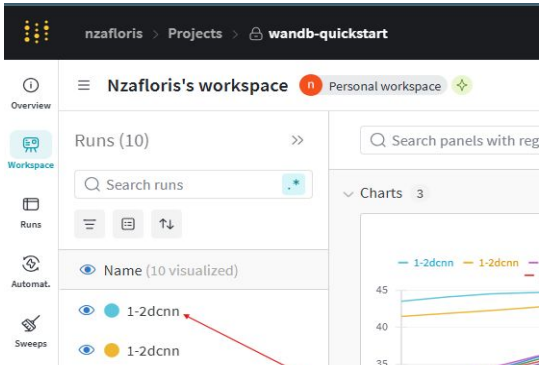
Saving Model to WandB

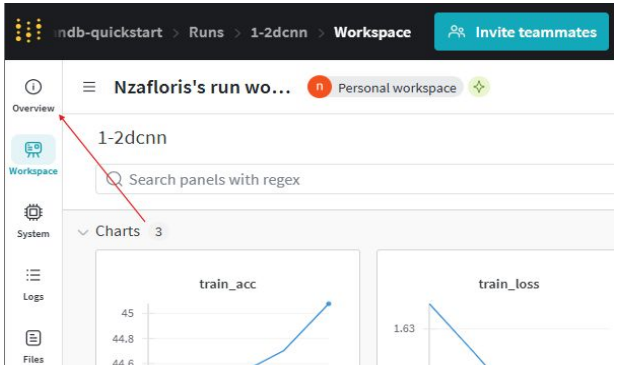
```
        # Saving the model and optimizer states

        torch.save({
              'model_state_dict': model.state_dict(),
              'optimizer_state_dict': optimizer.state_dict()
              }, "Model.pth")

        # ALTERNATIVE 1: Saving Files as Artifacts
        # Creating Artifact
        model_artifact = wandb.Artifact(run_config['model'], type='model')

        # Adding model file to Artifact
        model_artifact.add_file("Model.pth")

        # Saving Artifact to WandB
        run.log_artifact(model_artifact)

        # ALTERNATIVE 2: Saving Files as Files
        wandb.save("Model.pth")

if finish:
  wandb.finish()
```

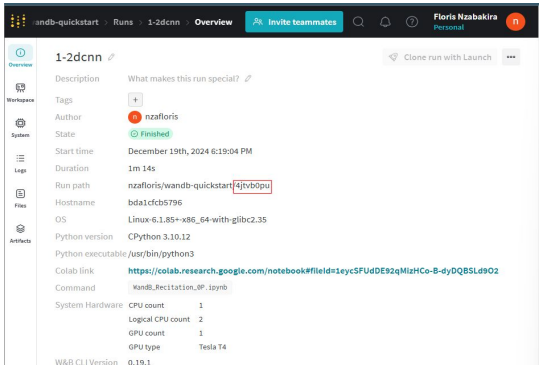


Resuming a Run

Getting Run ID







Resuming Run in Notebook

```
run_id = "4jtvb0pu" ### Replace with run id string
run = wandb.init(
    id     = run_id, ### Insert specific run id here if you want to resume a previous run
    resume = "must", ### You need this to resume previous runs, but comment out reinit = True when using this
    project = "wandb-quickstart", ### Project should be created in your wandb account
)
```

Automated Hyperparameter Tuning with Sweeps

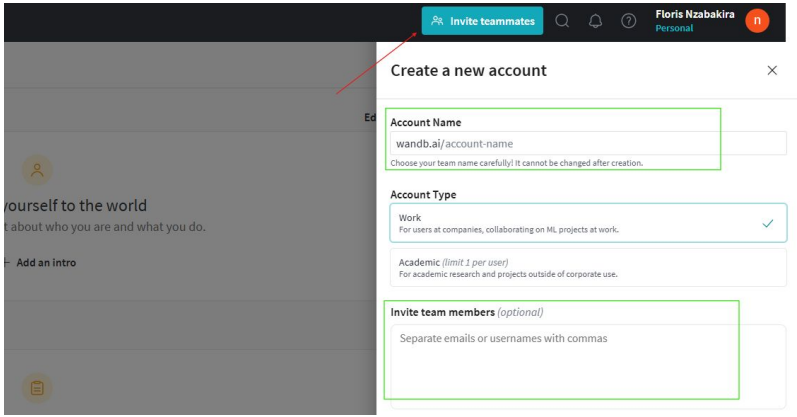
```
[ ]  # Hyperparameters to work with

     parameters_dict = {
         'optimizer':{
             'values': ['sgd', 'adam']
         },
         'learning_rate':{
             'distribution':'uniform',
             'min':2e-4,
             'max':1e-1
         },
         'batch_size': {
             'distribution': 'q_log_uniform_values',
             'q':4,
             'min': 16,
             'max': 128
         },
         'epochs':{
             'value': 5
         }
     }
     sweep_config['parameters'] = parameters_dict
```

```
[ ]  # Initalizing the sweep

     sweep_id = wandb.sweep(sweep_config, project="CIFAR-Sweep2")
```

```
def train_sweep(config = None):
    with wandb.init(config=config) as run:
        run.name=f"Jeel_{wandb.config.learning_rate}_{wandb.config.batch_size}_{wandb.config.optimizer}"
        config = wandb.config

        train_loader, test_loader = build_data(config.batch_size, data_train, data_test)

        model = Network().to(device)

        optimizer = get_optim(config.optimizer, config.learning_rate, model)

        criterion = nn.CrossEntropyLoss()

        scaler = torch.cuda.amp.GradScaler()

        for epoch in range(config.epochs):

            model, loss = train_epoch(model, train_loader, optimizer, criterion, scaler)

            wandb.log({'loss': loss})
```

```
[ ]  # Running the sweep

     wandb.agent(sweep_id, train_sweep, count=2)
```

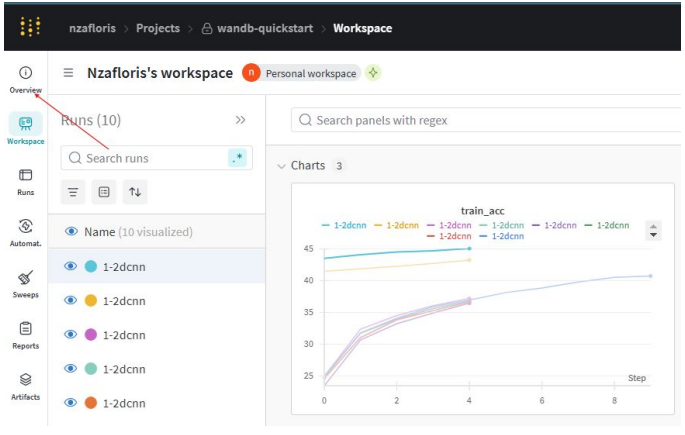
Collaboration on WandB

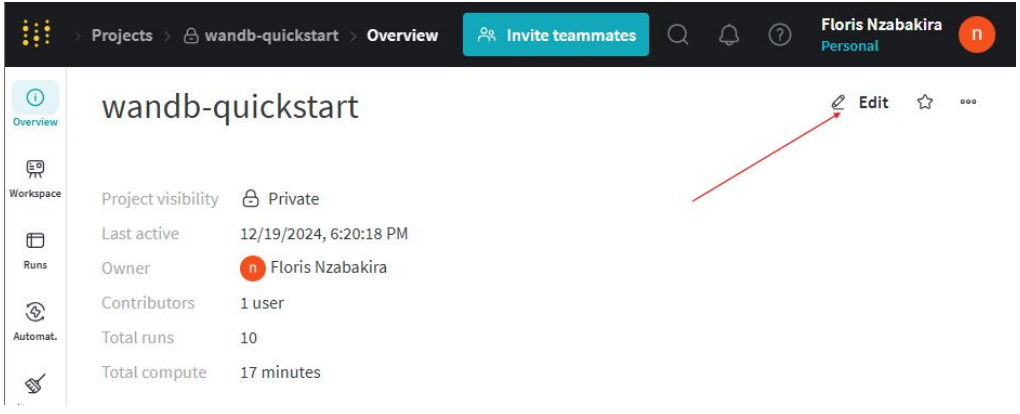


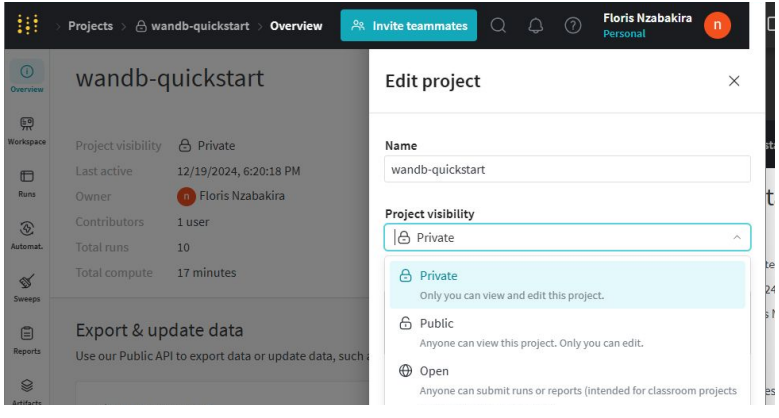

```
[16] run = wandb.init(
         entity="wandb-starter",
         project="wandb-quickstart",
         job_type="model-training",
         name=run_config['model'],
         config=run_config
         )
```

```
wandb: Currently logged in as: nzafloris (wandb-starter). Use `wandb login --relogin` to force relogin
Tracking run with wandb version 0.19.1
Run data is saved locally in /content/wandb/run-20241219_225136-k7cu7hle
Syncing run 1-2dcnn to Weights & Biases (docs)
View project at https://wandb.ai/wandb-starter/wandb-quickstart
View run at https://wandb.ai/wandb-starter/wandb-quickstart/runs/k7cu7hle
```

Making your WandB Project Public







WandB Storage for Individual and Group Accounts

NB: Once your WandB free storage is full, you won't be able to access your runs.

You can view your storage status in the following way

Individual Account:

- Provides 100GB free
- Can be accessed through https://wandb.ai/subscriptions

Group Account:

- Provides 5GB free per team
- Can be accessed through https://wandb.ai/account-settings/<Team-Name>/billing