Tutorial on Weights & Biases

陈悦 & 郑馨

Okubo Lab

2024.06.26

Overview of the tutorial

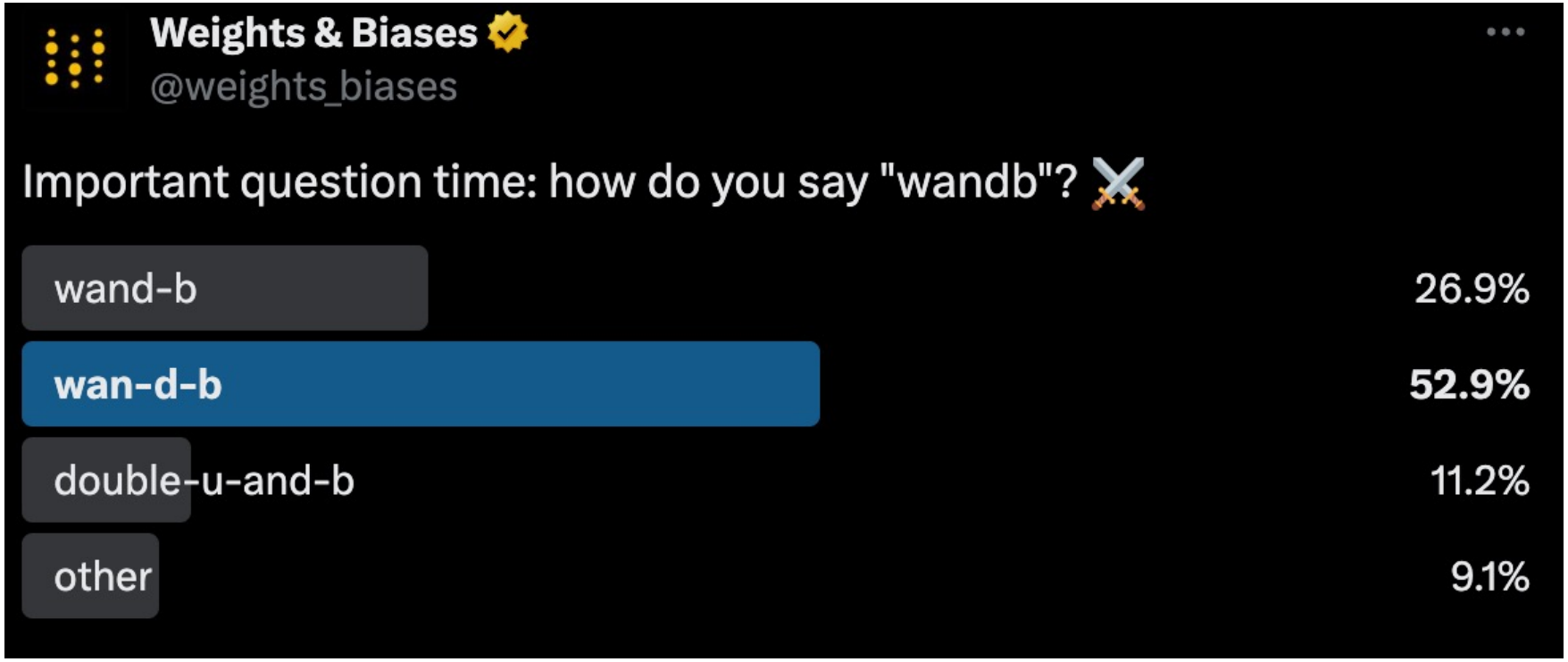
- Why we need W&B?
- Explanation on what W&B can do
- Hands-on exercises

What is W&B?

What is W&B?

- W&B is a software for "experimental tracking" for machine learning
- Just like any scientific experiments, machine learning requires you to test multiple different settings and recording the results.

How do you pronounce W&B?

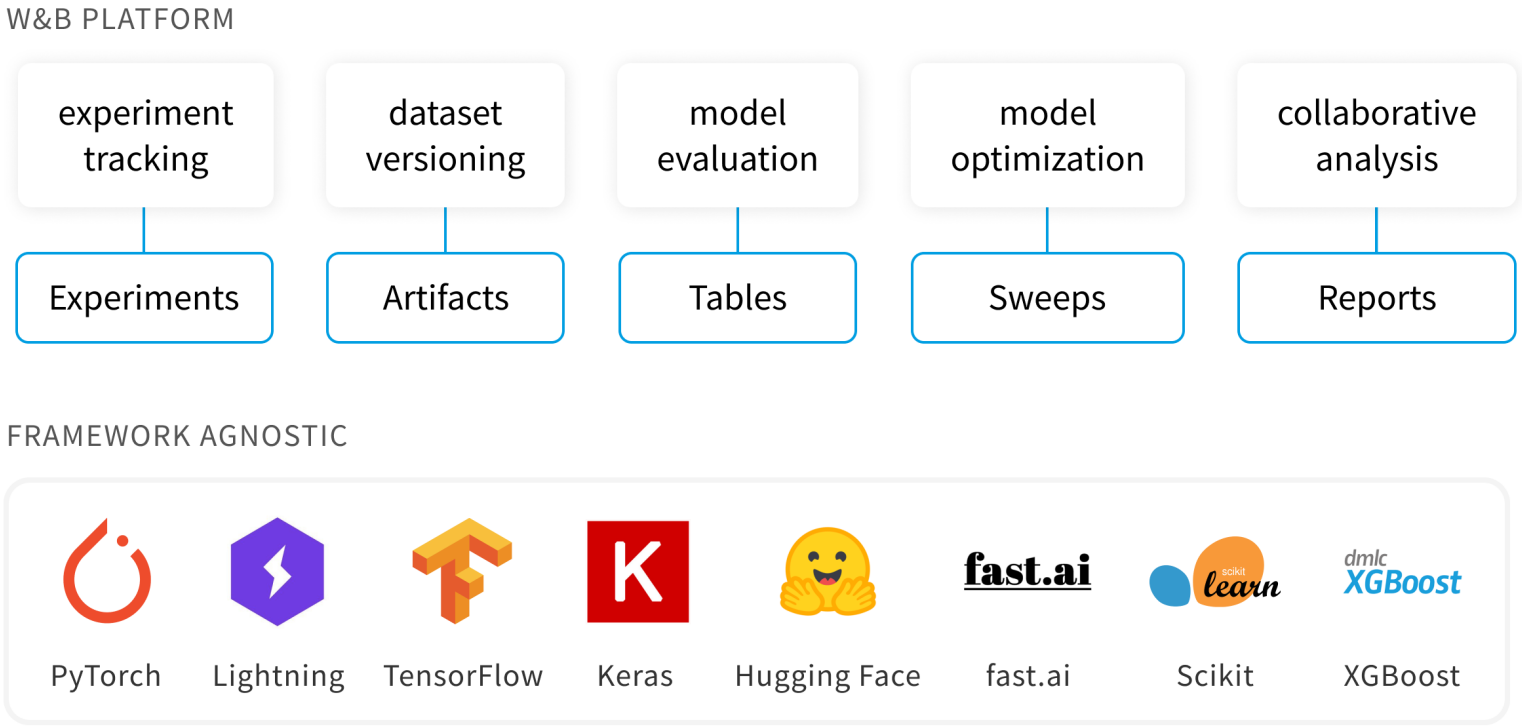


Why do you need "experiment tracking"?

- Debugging
  - How can we visualize the training process in real-time?
- Model comparison
  - Which models are performing well on your task?
- Reproducibility and traceability
  - How can we make sure other people can get the same result?
- Collaboration
  - How to share results to other people?

W&B

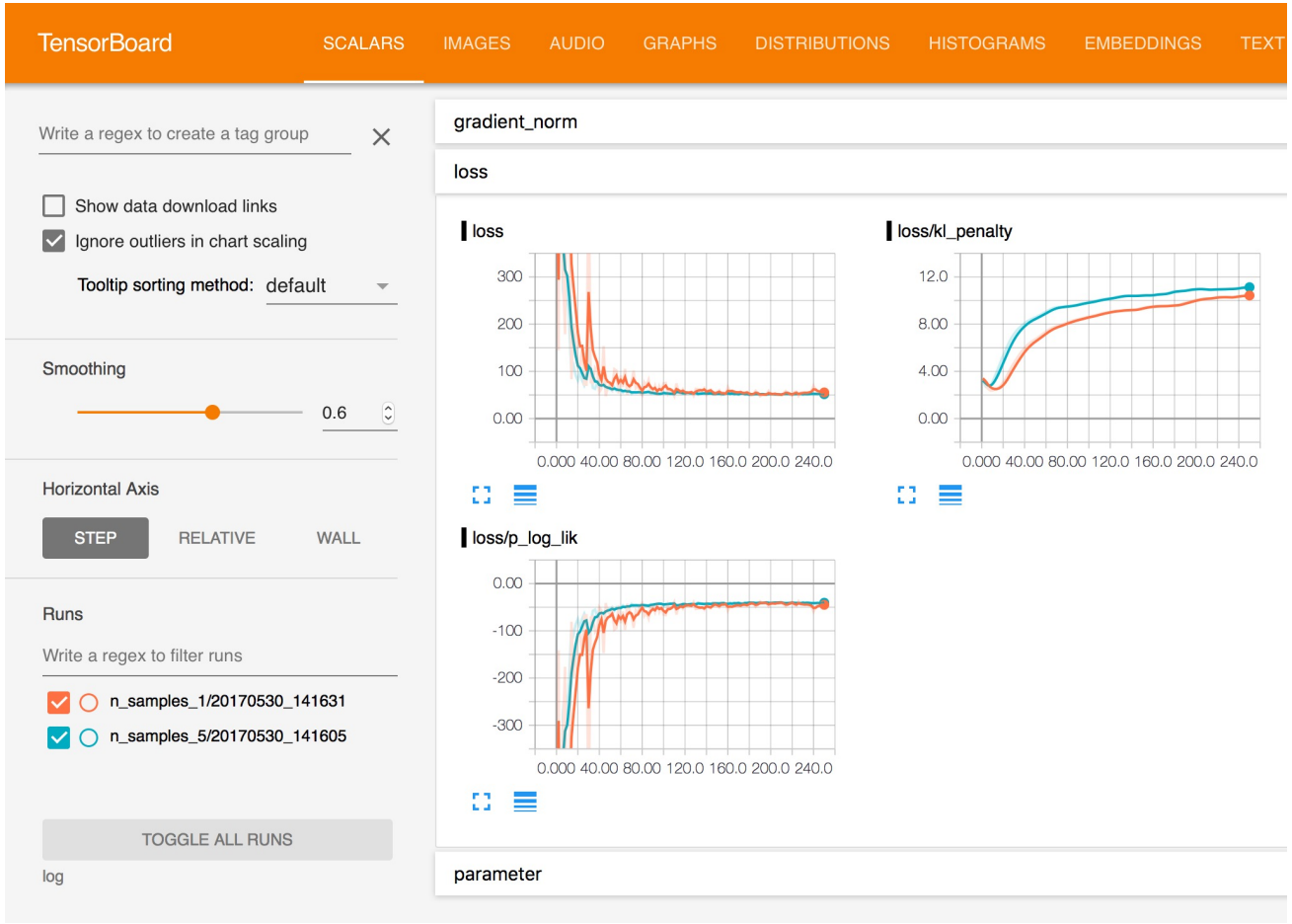
- Online
- Version management
- Case analysis
- Automated optimization
- Teamwork
- Powerful scalability



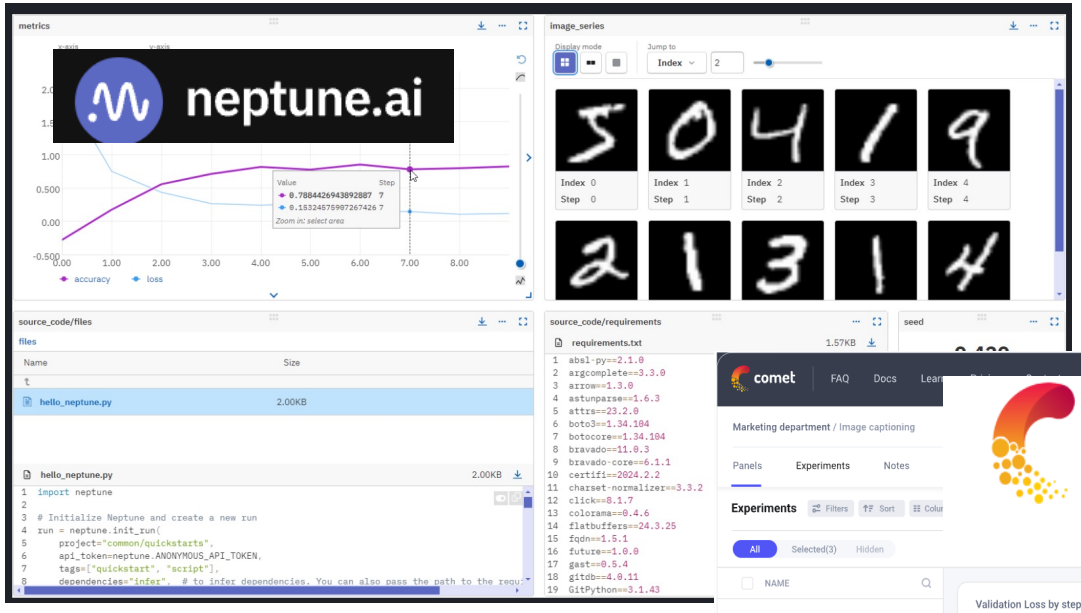
Isn't Tensorboard enough?

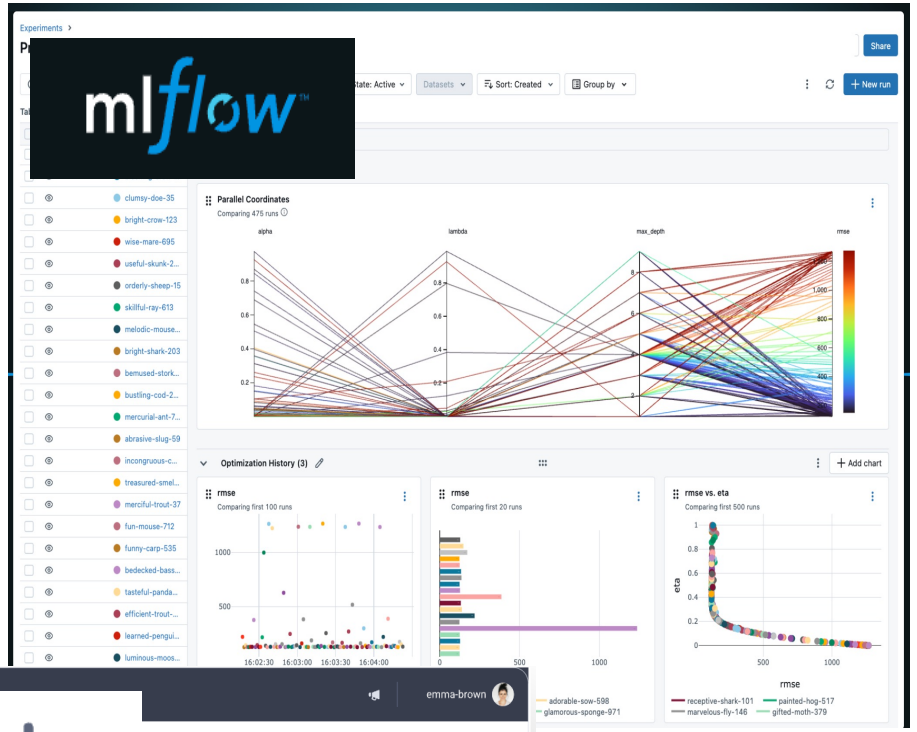
- (+) basic visualization
- (+) long history, large community

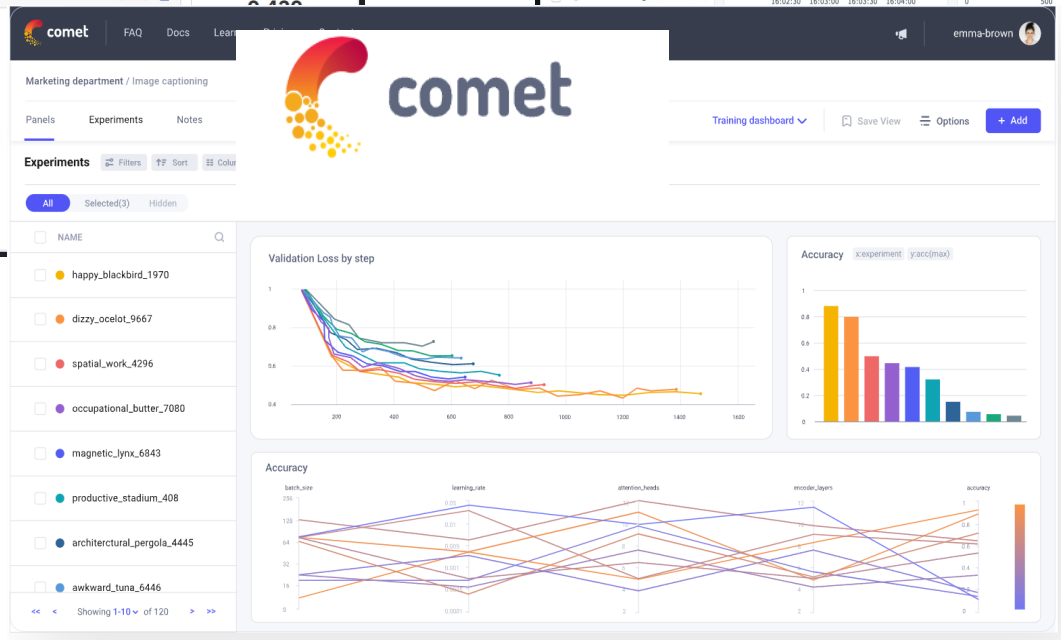
- (-) local
- (-) limited scalability
- (-) no collaboration function



Other options



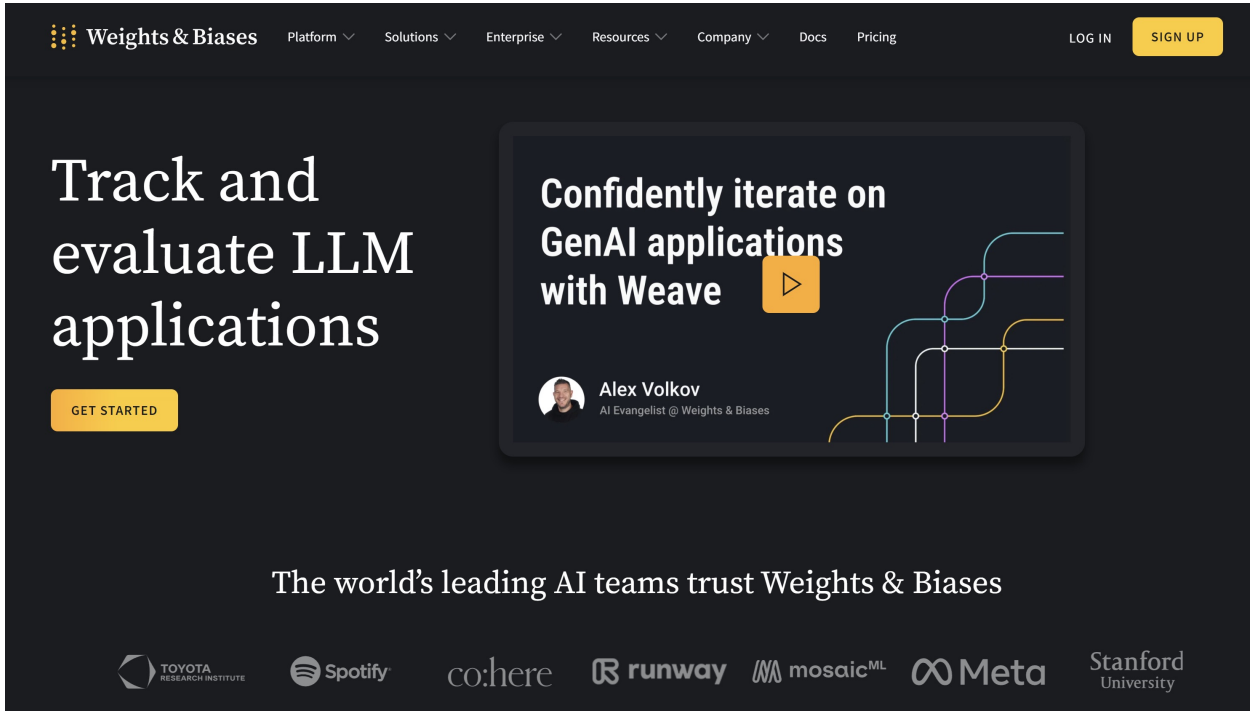


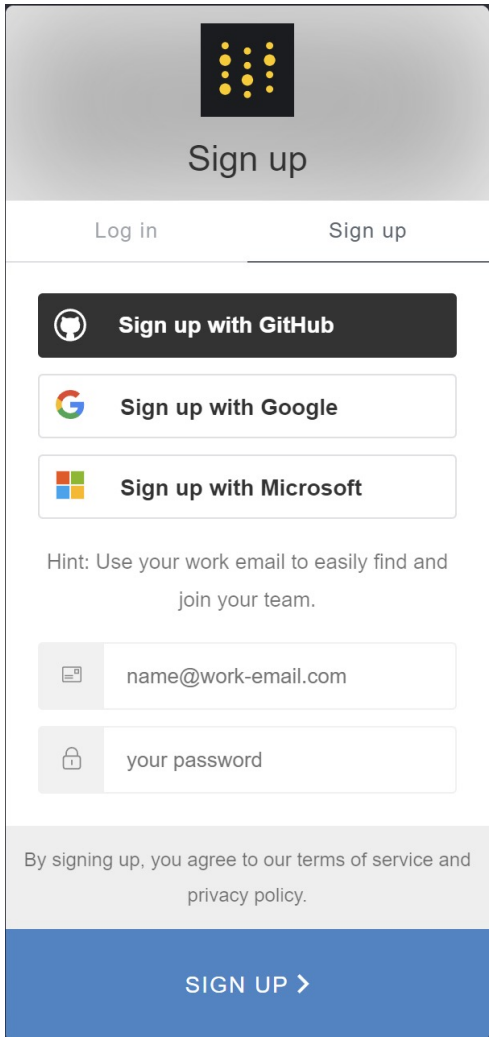


W&B basics

How can I create a W&B account?

- Visit W&B website: https://wandb.ai
- Sign up





Exercise overview

- You are training a neural network that can classify hand-written digits (MNIST dataset).
- The data loading and neural network training codes (PyTorch) are already provided to you.

- Can you add experiment tracking features using W&B?

What is a 'project'? What is a 'run'?

- Project
  - a research project or a large collection of experiments
  - in a project, you can organize different experiments and share information
  - allow team members to collaborate
- Run
  - a single experiment process, smallest unit in experiment tracking
  - in a run, you can record the parameters of the model, metrics during the training process, model logs, charts, etc.

Important commands

- W&B.login()
- W&B.init()
- W&B.log()
- W&B.finish()

How do I initiate a run?

- Use `wandb.init()` to start a new run to track and log to W&B
- During this step, we can specify project, group, name etc., we can also save the configurations

What should I put in my config?

- Advice on what to put in config (doesn't change during a single run) vs. what to log each training step.

Quiz 1

Please modify the code to specify the project name and run name as following:

- project: "wandb_demo"
- name: current time (`nowtime`)

Please also record experiment hyperparameters

```
def train(config):
    train_loader, test_loader = create_dataloaders(config)
    model = cnn(config)
    optimizer = torch.optim.__dict__[config['optimizer']](params=model.parameters(), lr=config['lr'])
    nowtime = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    '''
    Please modify here
    '''

    for epoch in range(1, config['epochs']+1):
        model, train_loss = train_epoch(model, train_loader, optimizer)
        val_acc = eval_epoch(model, test_loader)
        print(f"epoch {epoch}: train_loss={train_loss:.2f}, val_acc= {100 * val_acc:.2f}%")

    wandb.finish() # Notify wandb that your run has ended and upload all log data to wandb

    return model

# model = train(config)
```

Answer for quiz 1

```
'''
Here is the answer
'''
wandb.init(project='wandb_demo', name=nowtime, config=config)
```

W&B Log

https://docs.wandb.ai/ref/python/log

How can we track the experimental results?

- We can use `wandb.log()` to log a dictionary of data to the current run's history
- The most basic usage is

```
wandb.log({"train-loss": 0.5, "accuracy": 0.9})
```

Quiz 2

Please modify the code to record training loss, validation loss, and validation accuracy.

```python
def train(config):
    train_loader, test_loader = create_dataloaders(config)
    model = cnn(config)
    optimizer = torch.optim.__dict__[config['optimizer']](params=model.parameters(), lr=config['lr'])
    nowtime = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    wandb.init(project='wandb_demo', name=nowtime, config=config)

    for epoch in range(1, config['epochs']+1):
        model, train_loss = train_epoch(model, train_loader, optimizer)
        val_acc, val_loss = eval_epoch(model, test_loader)
        print(f"epoch {epoch}: train_loss={train_loss:.2f}, val_loss={val_loss:.2f}, val_acc= {100 * val_acc:.2f}%")

        '''
        Please modify here
        '''

    wandb.finish() # Notify wandb that your run has ended and upload all log data to wandb

    return model

# model = train(config)
```

Answer for quiz 2

```
'''
Here is the answer
'''
wandb.log({"train/loss": train_loss, "val/loss": val_loss, "val/acc": val_acc})
```

W&B Artifact

https://docs.wandb.ai/ref/python/artifact

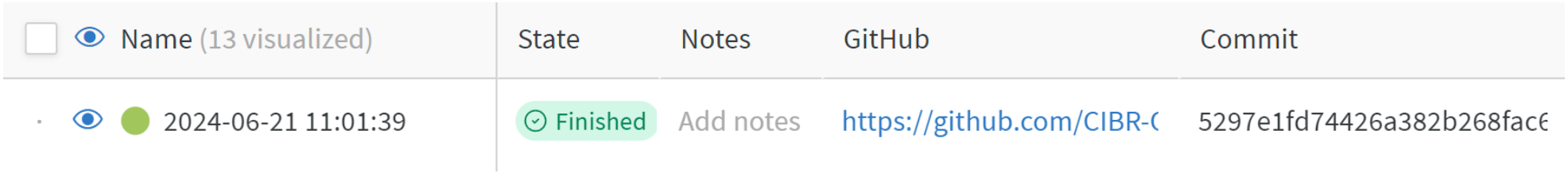
Motivation

- The dataset is not fixed. Everytime I collect new data, I need to include it in the training set.
  - concept called "data lineage"

- The code gets updated very frequently. I want to keep track of which version of the code that was used to run the experiment.

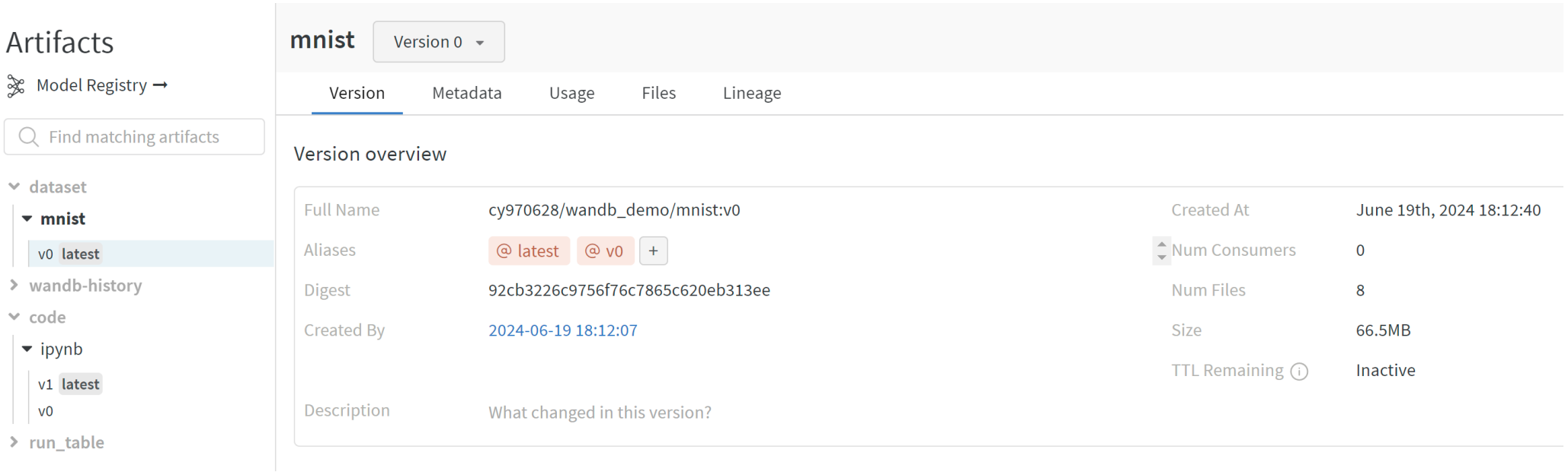
How to manage the version?

- Code version
  - integrated with Github to automatically capture the submission history of the code and associate it with the experimental run
  - link the code folder to existing Github repository



How to manage the version?

- wandb.Artifact()
- Save files and re-use
  - data, codes, model etc.

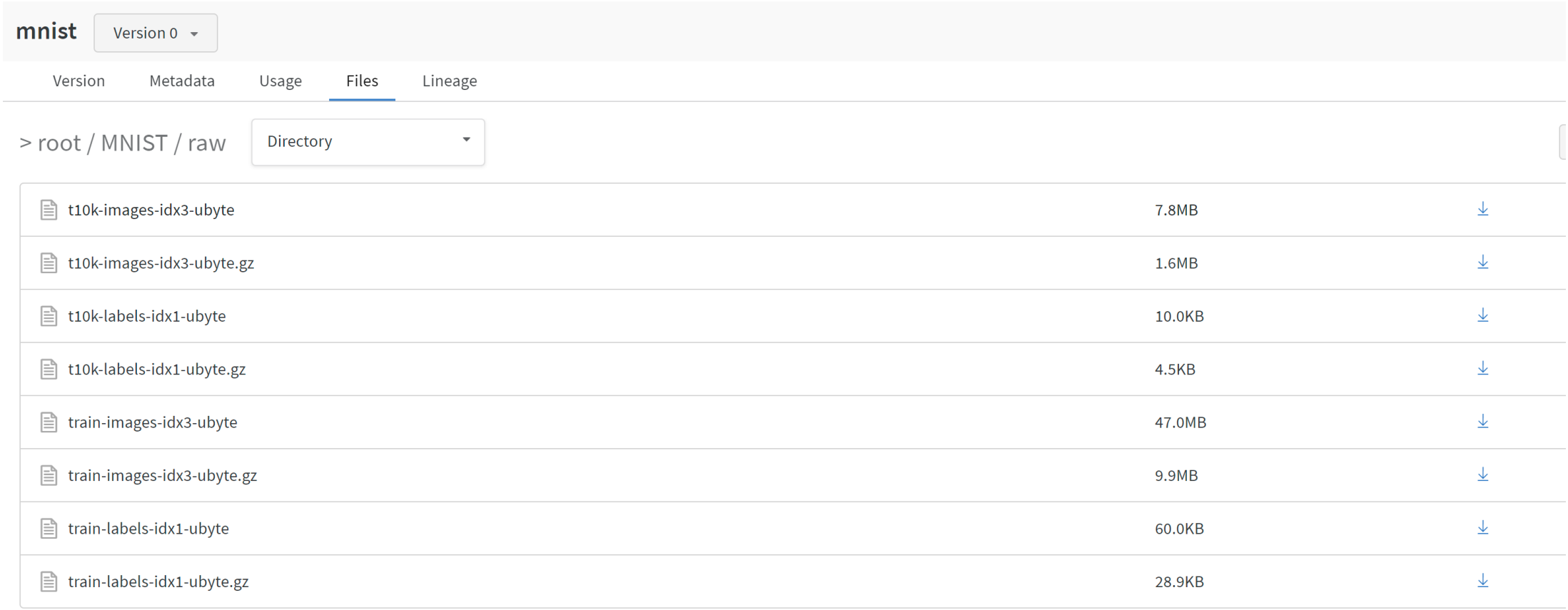


Example: save dataset

- Create an Artifact object, specify the name and type
- Add files to the object
- Log the files

```
arti_dataset = wandb.Artifact('mnist', type='dataset')
arti_dataset.add_dir('data/')
wandb.log_artifact(arti_dataset)
```

Example: save dataset



Quiz 3

Please modify the code to save the jupyter notebook file

Hint:

- Create a new `Artifact` object, specify name and type
- Use `add_file` instead of `add_dir`

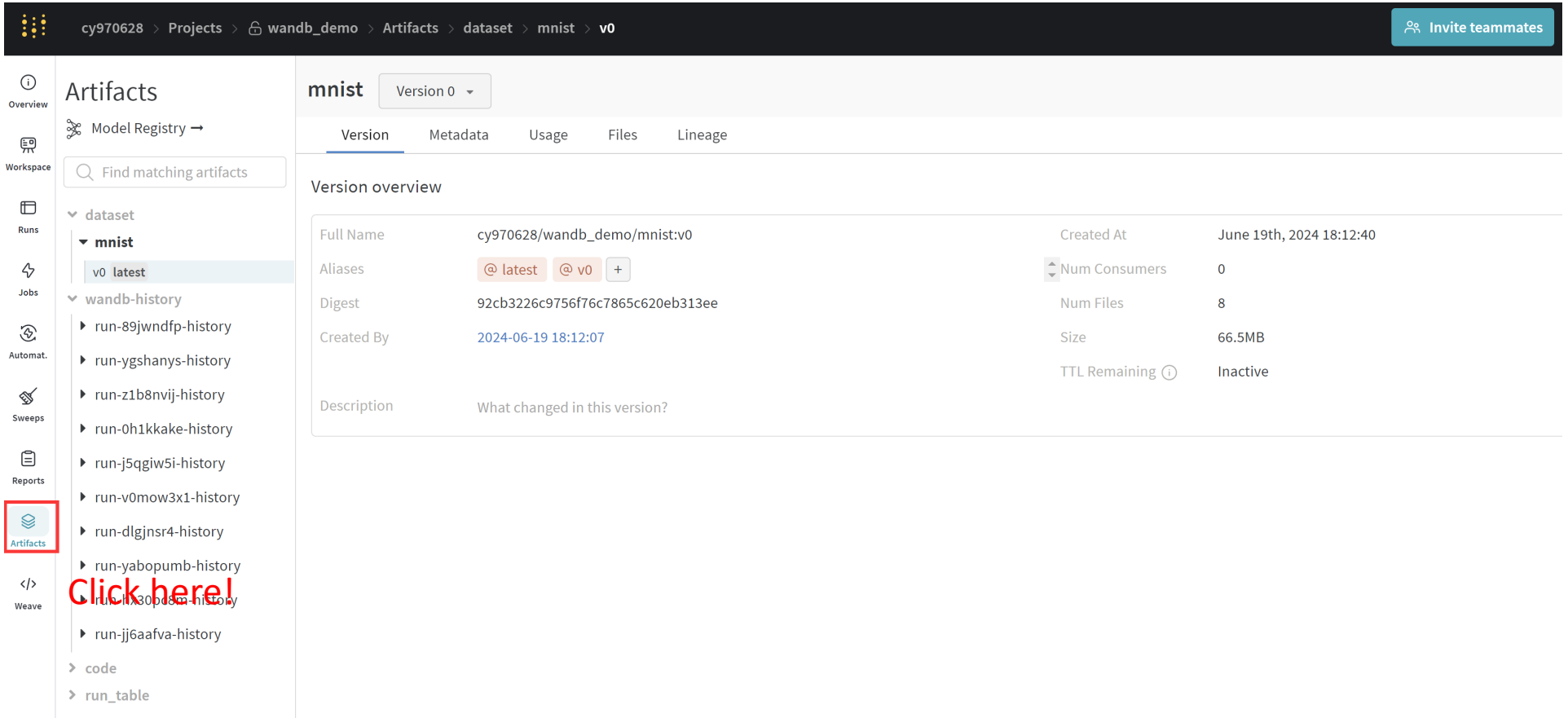
```
def train(config):
    train_loader, test_loader = create_dataloaders(config)
    model = cnn(config)
    optimizer = torch.optim.__dict__[config['optimizer']](params=model.parameters(), lr=config['lr'])
    nowtime = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    wandb.init(project='wandb_demo', name=nowtime, config=config)

    for epoch in range(1, config['epochs']+1):
        model, train_loss = train_epoch(model, train_loader, optimizer)
        val_acc = eval_epoch(model, test_loader)
        print(f"epoch {epoch}: train_loss={train_loss:.2f}, val_acc= {100 * val_acc:.2f}%")
        wandb.log({"train_loss": train_loss, "val_acc": val_acc})

    '''
    Please modify here
    '''

    wandb.finish() # Notify wandb that your run has ended and upload all log data to wandb

    return model

# model = train(config)
```

Answer for quiz 3

```
'''
Here is the answer
'''
arti_code = wandb.Artifact('ipynb', type='code')
arti_code.add_file('wandb_demo.ipynb')
wandb.log_artifact(arti_code)
```

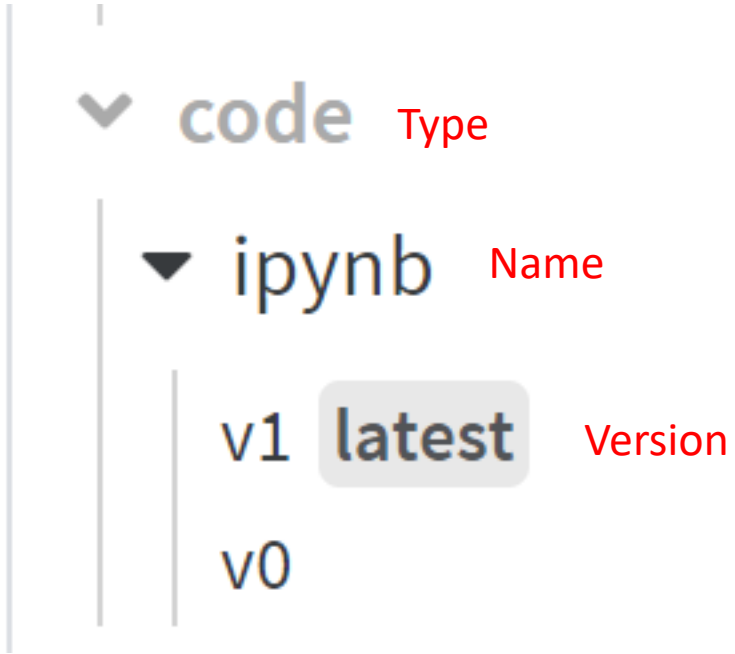
Where can I find my files?

- Visit "artifact"



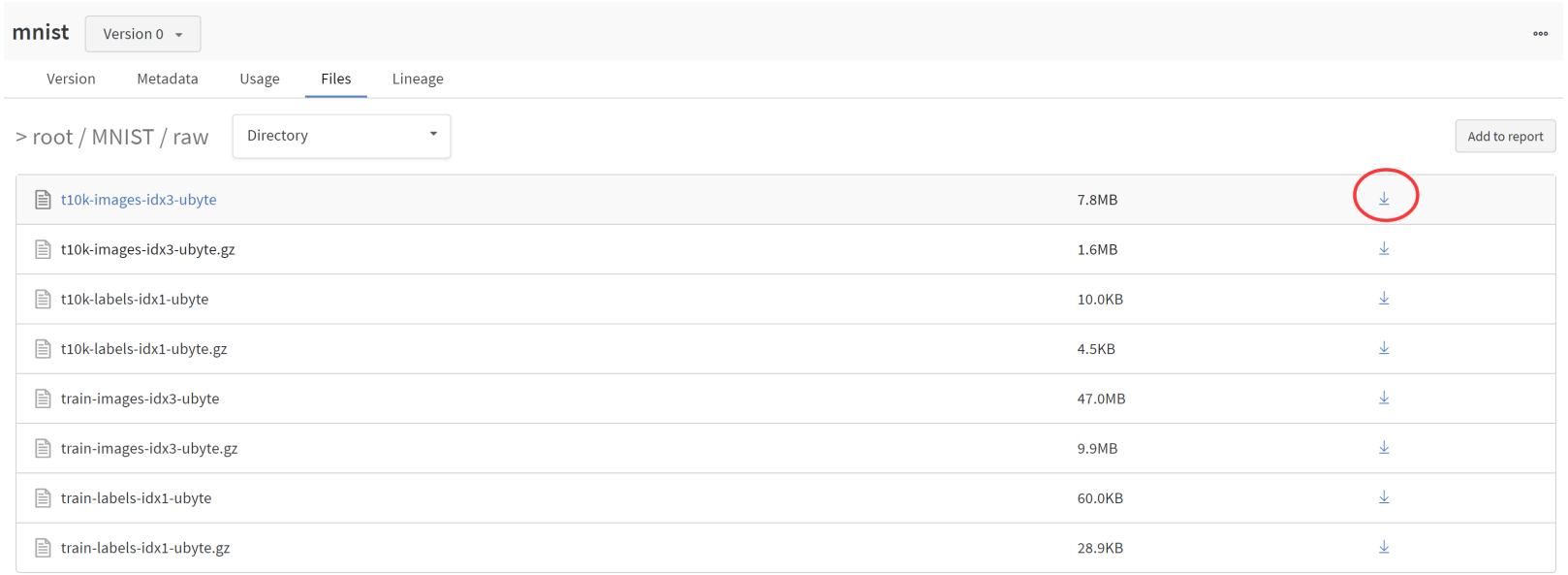
Where can I find my files?

- Choose type and name



How to reuse the files?

- Directly download from website



- Use API

```
import wandb
run = wandb.init()
artifact = run.use_artifact('cy970628/wandb_demo/mnist:v0', type='dataset')
artifact_dir = artifact.download()
```

W&B Tables

https://docs.wandb.ai/guides/tables

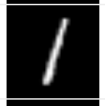
How to easily visualize the experiment data/results?

- E.g. Visualize images and compare the targets & predictions

Tables 1

Add panel

runs.summary["cases"]

	Image	Target	Prediction
1		7	7
2		2	2
3		1	1
4		0	0
5		4	4

How to easily visualize the experiment data/results?

- wandb.Table()
- Like a DataFrame
- Example

```
my_table = wandb.Table(columns=["a", "b"], data=[["a1", "b1"], ["a2", "b2"]])
wandb.log({"Table Name": my_table})
```

Quiz 4

Please modify the code to create and log a table to show 10 validation results.

The table should include:

- Image: original input image
- Target: target class of the image
- Prediction: predicted class of the image

Notice the `results` have already been formatted using `show_cases` `results` format:

- list of lists
- [n_cases*[Image, Target, Prediction]]

Quiz 4

```python
def show_cases(model, show_num):
    cases = []
    test_data = datasets.MNIST(root = 'data', train = False, transform = ToTensor(),download = True)
    for i in range(show_num):
        features, label = test_data[i]
        tensor = features.to(device)
        pred = torch.argmax(model(tensor[None,...]))
        image = wandb.Image(features.permute(1,2,0).numpy())
        cases.append([image, label, pred])
    return cases
```

```python
from utils import show_cases

def train(config):
    train_loader, test_loader = create
    model = cnn(config)
    optimizer = torch.optim.__dict__[
    nowtime = datetime.datetime.now().

    wandb.init(project='wandb_demo', name=nowtime, config=config)

    for epoch in range(1, config['epochs']+1):
        model, train_loss = train_epoch(model, train_loader, optimizer)
        val_acc = eval_epoch(model, test_loader)
        print(f"epoch {epoch}: train_loss={train_loss:.2f}, val_acc= {100 * val_acc:.2f}%")
        wandb.log({"train_loss": train_loss, "val_acc": val_acc})


    # results to show
    results = show_cases(model, 10)
    '''
    Please modify here
    '''

    wandb.finish() # Notify wandb that your run has ended and upload all log data to wandb

    return model

# model = train(config)
```

Answer for quiz 4

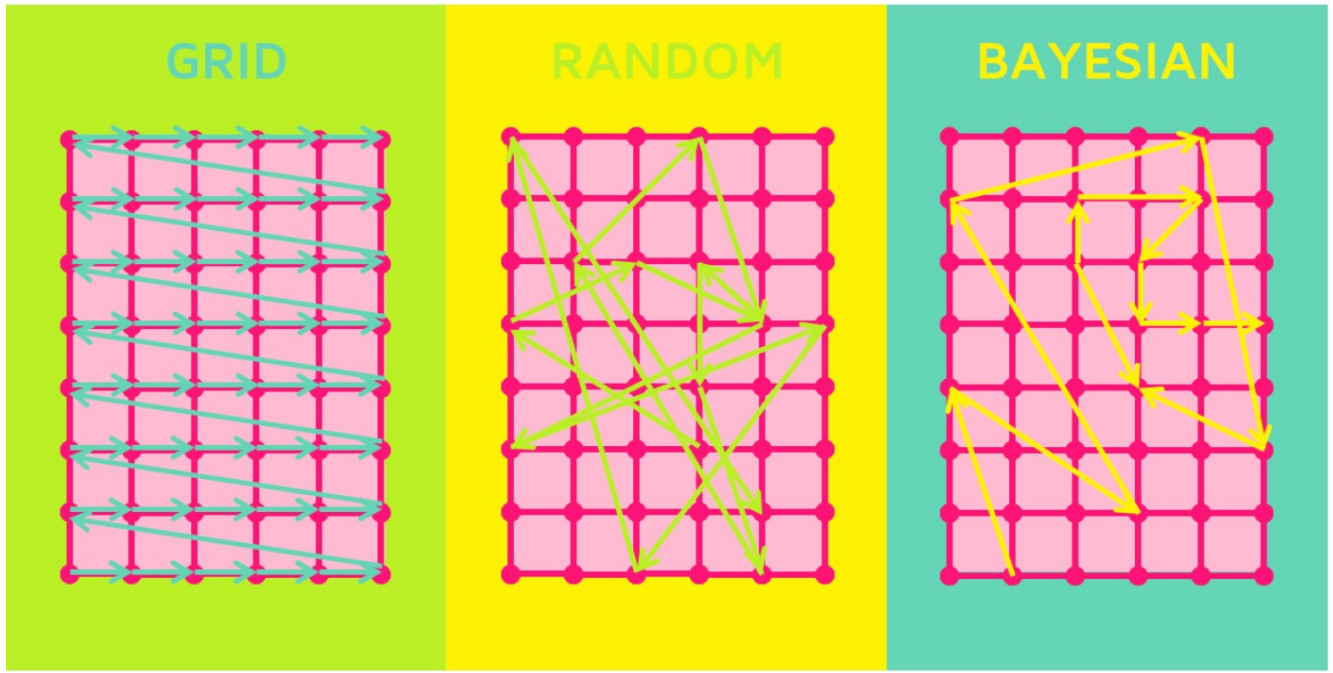
```
# results to show
results = show_cases(model, 10)
'''

Here is the answer
'''

column_name = ['Image', 'Target', 'Prediction']
cases = wandb.Table(columns=column_name, data=results)
wandb.log({'cases':cases})
```

W&B Sweeps

https://docs.W&B.ai/guides/sweeps

Why sweeps?

- Hyperparameter tuning: find the optimal value for hyperparameters
- Strategies
  - Manual
  - Grid
  - Random
  - Bayesian
  - Others



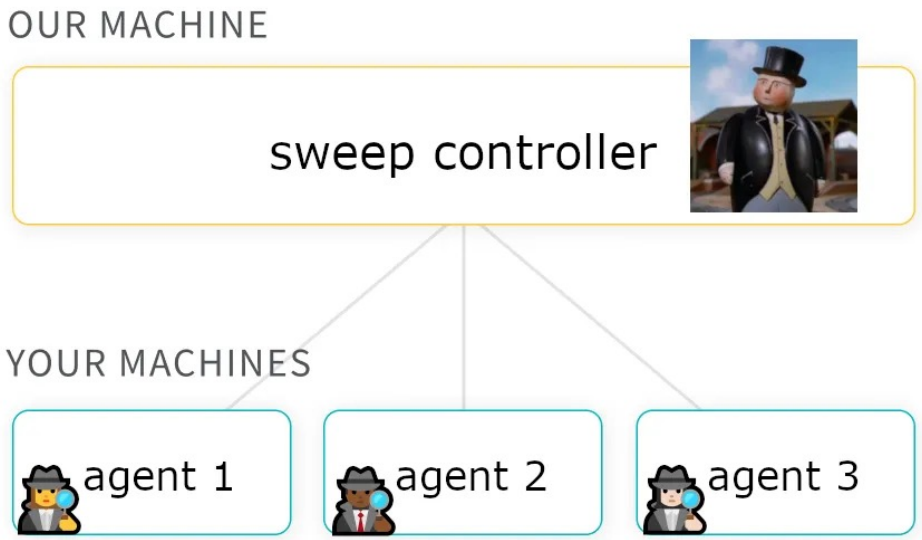
What is a sweep

- A way to automate hyperparameter search and visualize rich, interactive experiment tracking

- "sweep over" hyperparameter space

Overview of how to create sweeps

- Define the search space with sweep configuration
- Initialize the sweep
- Run the sweep with agent
  - Can be easily distributed



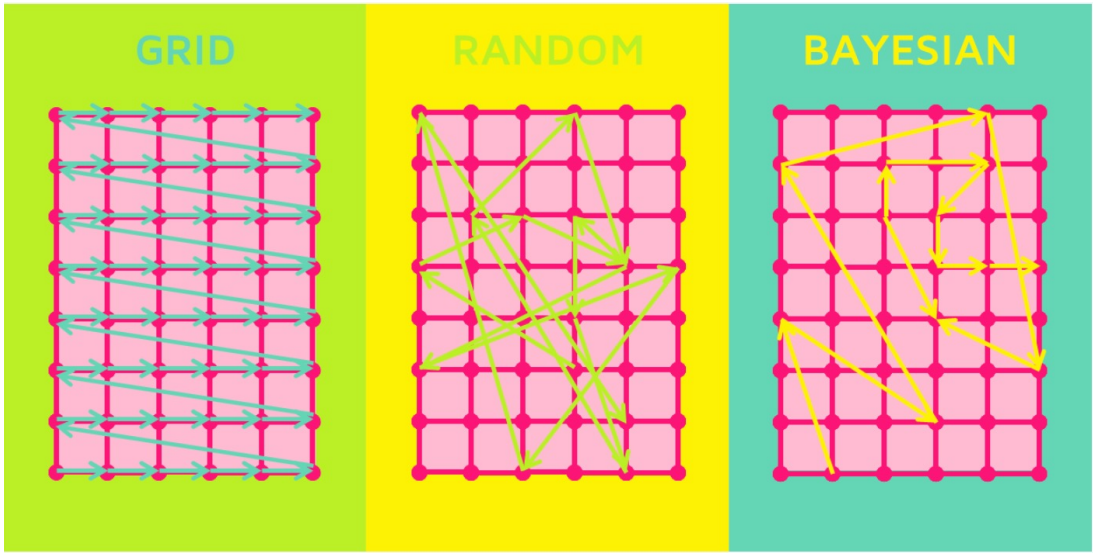
Sweep configuration

- Nested dictionary or YAML file
- Keys
  - `name`
  - `method` (required)
  - `metric`
  - `parameters` (required)
- Other
  - `run_cap`
  - `early_terminate`

example configuration

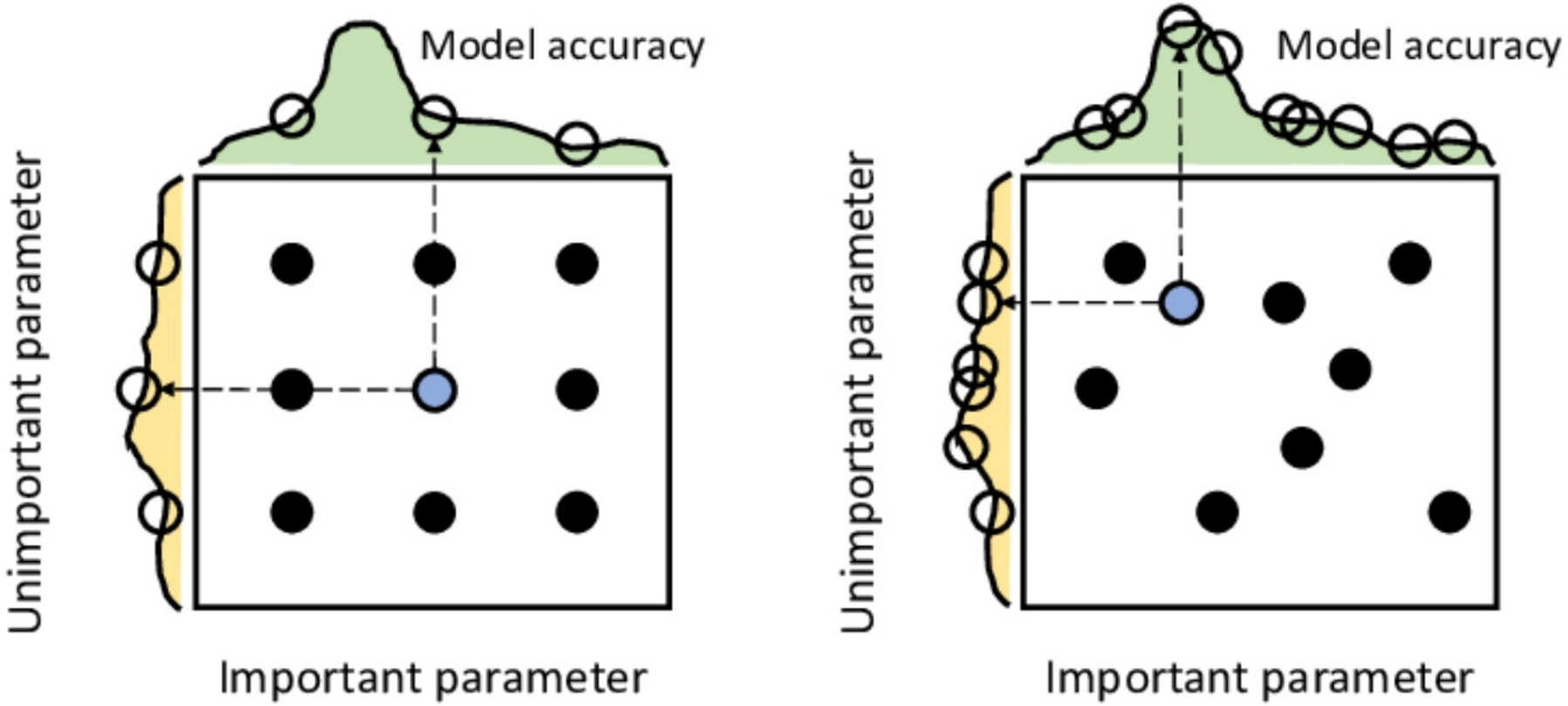
```
{'method': 'random',
 'metric': {'goal': 'maximize', 'name': 'val_acc'},
 'name': 'random_exp',
 'parameters': {'batch_size': {'values': [128, 256, 512, 1024]},
                'dropout_rate': {'values': [0.0, 0.2, 0.4, 0.6, 0.8]},
                'epochs': {'value': 5},
                'hidden_layer_width': {'values': [32, 64, 128, 256]},
                'lr': {'distribution': 'log_uniform_values',
                       'max': 0.1,
                       'min': 1e-06}}}
```

Method



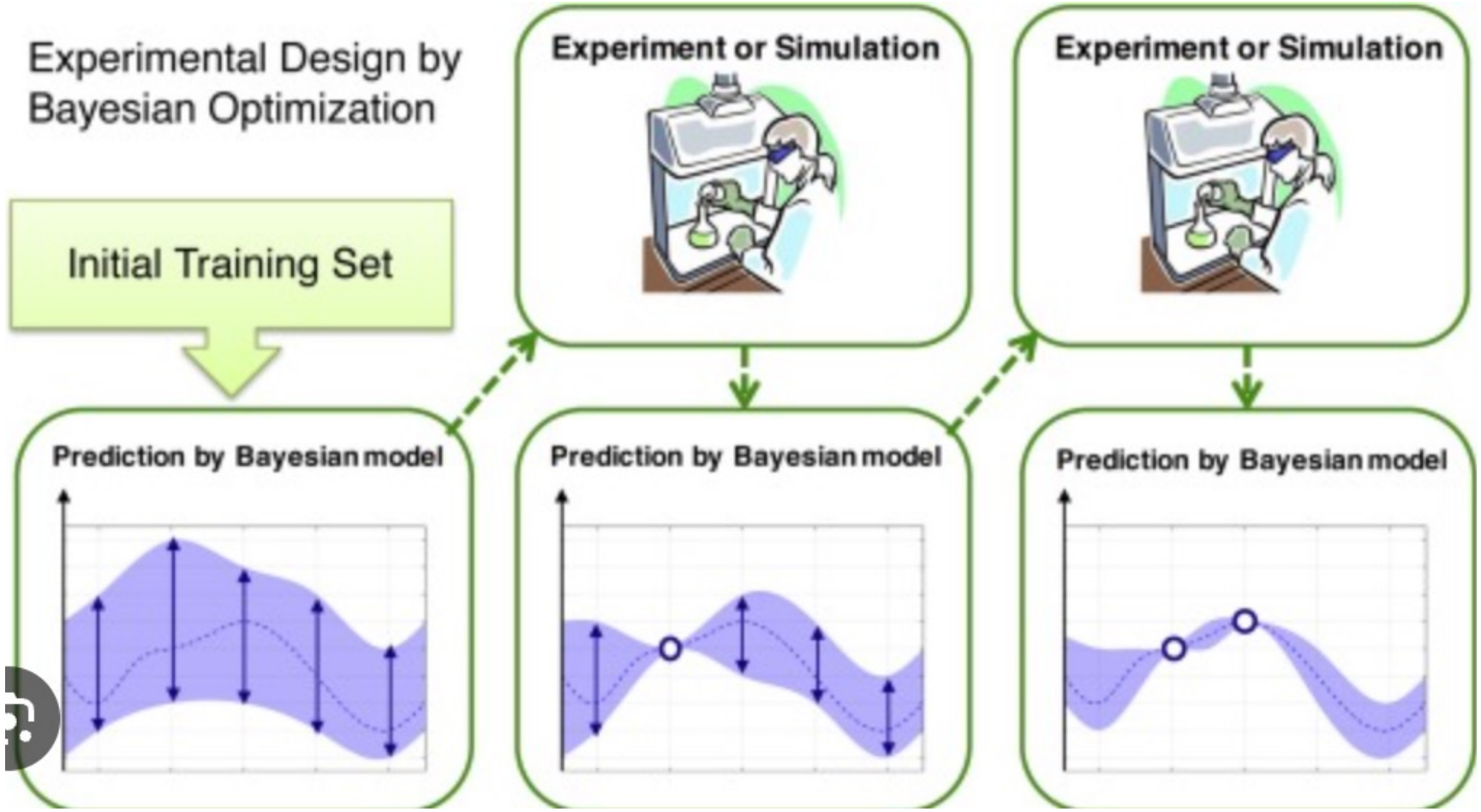
- grid
  - test all combinations
- random
  - number of runs needs to be specified
- bayes
  - number of runs needs to be specified

Grid vs. Random search



based on Bergstra & Bengio (2012)

Bayesian Optimization (BO)



Metric

- Specify the metric to optimize
- Keys
  - `name`: name of the metric; must be one of the **logged** variables
  - `goal`: can be `'maximize'` or `'minimize'`
  - `target`: target value of the metric; stop runs after reaching this value
- Optional for grid search and random search; required for bayes

Parameters

- Define hyperparameter search space
- Discrete values: specify a list of discrete values to choose from
  - key: `value/values`

```
'dropout_rate': {
    'values': [0.0, 0.2, 0.4, 0.6, 0.8],
},
'epochs': {
    'value': 5
},
```

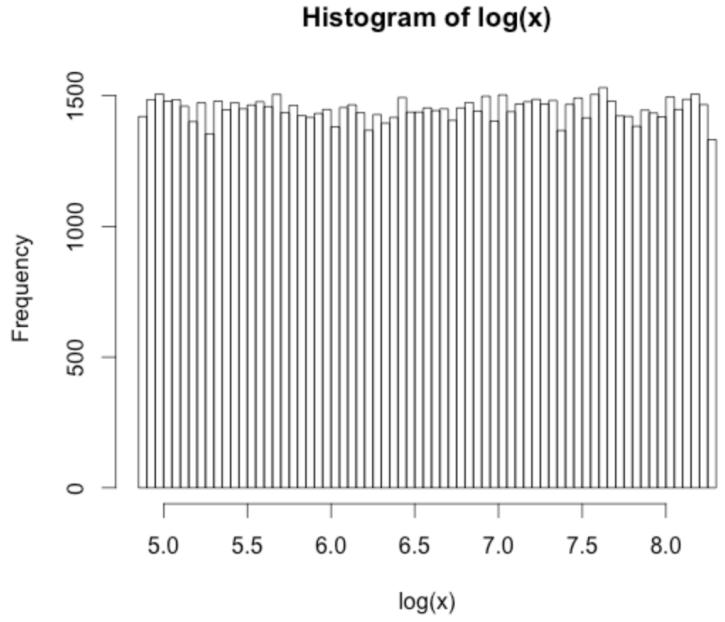
- Distribution: specify a distribution to sample from
  - key: `distribution, min, max`

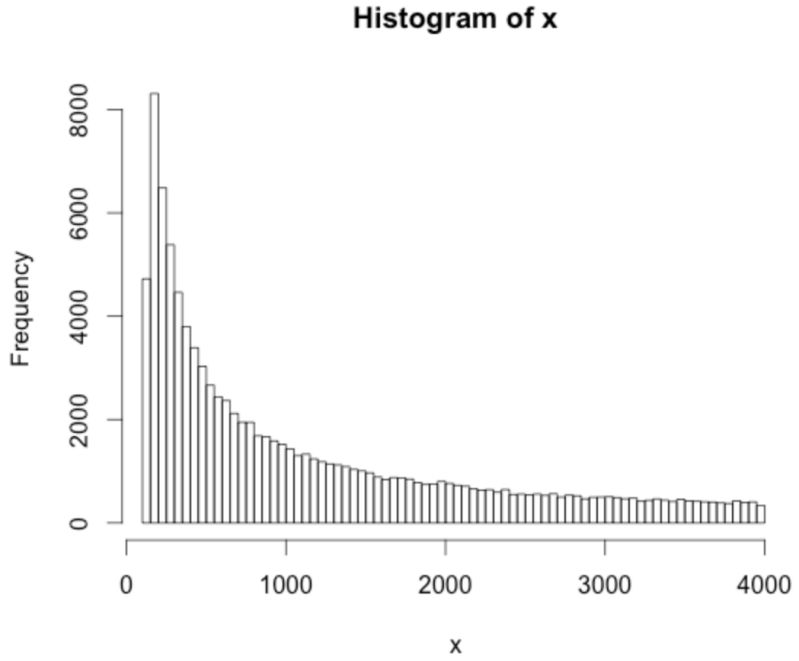
```
'lr': {
    'distribution': 'log_uniform_values',
    'min': 1e-6,
    'max': 0.1,
},
```

Value for `distribution` key	Description
`constant`	Constant distribution. Must specify the constant value (`value`) to use.
`categorical`	Categorical distribution. Must specify all valid values (`values`) for this hyperparameter.
`int_uniform`	Discrete uniform distribution on integers. Must specify `max` and `min` as integers.
`uniform`	Continuous uniform distribution. Must specify `max` and `min` as floats.
`q_uniform`	Quantized uniform distribution. Returns `round(X / q) * q` where X is uniform. `q` defaults to `1`.
`log_uniform`	Log-uniform distribution. Returns a value `X` between `exp(min)` and `exp(max)` such that the natural logarithm is uniformly distributed between `min` and `max`.
`log_uniform_values`	Log-uniform distribution. Returns a value `X` between `min` and `max` such that `log(X)` is uniformly distributed between `log(min)` and `log(max)`.
`q_log_uniform`	Quantized log uniform. Returns `round(X / q) * q` where `X` is `log_uniform`. `q` defaults to `1`.
`q_log_uniform_values`	Quantized log uniform. Returns `round(X / q) * q` where `X` is `log_uniform_values`. `q` defaults to `1`.
`inv_log_uniform`	Inverse log uniform distribution. Returns `X`, where `log(1/X)` is uniformly distributed between `min` and `max`.

log-uniform distribution

- Log of the varible is uniformly distributed.
- More likely to draw smaller values





Quiz 5: define a sweep config

1). Define a sweep configuration with `name`, `method` and `metric`

- `name`: assign a meaningful name, such as "mnist_random".
- `method`: use `random` search for this exercise.
- `metric`: specify the `metric` with `"name"` being `"val/acc"` and `goal` being `"maximize"`.

Answer for quiz 5.1

```
sweep_config = {
    'name': 'mnist_random',
    'method': 'random',
    'metric': {
        'name': 'val/acc',
        'goal': 'maximize'
    }
}
```

Quiz 5: define a sweep config

```
# parameter configuration
parameters_dict = {
    'hidden_layer_width': {
        'values': [32, 64, 128, 256],
    },
    'dropout_rate': {
        'values': [0.0, 0.2, 0.4, 0.6, 0.8],
    },
    'epochs': {
        'value': 5
    },
    'batch_size': {
        'values': [128, 256, 512, 1024]
    }
}
```

2). Update the sweep config with `parameters`

- Update the given hyperparameter configuration to include the learning rate. The key for learning rate is `"lr"`.
  - `distribution: "log_uniform_values"`
  - `min: 1e-6`
  - `max: 0.1`
- Update the sweep configuration with this hyperparameter search space. The key in sweep_config is `"parameters"`.

Answer for quiz 5

```
# update with learning rate
parameters_dict['lr'] = {
    "distribution": "log_uniform_values",
    "min": 1e-6,
    "max": 0.1
}
```

Sweep initialization

- Initialize the sweep controller by `wandb.sweep()`.
  - `sweep`
  - `project`
  - `entity`
- Sweep id will be returned by `wandb.sweep()`. You can use the sweep id to start runs in this sweep.

```
sweep_id = wandb.sweep(sweep=sweep_config, project="wandb-demo", entity='okubo-lab')
```

✓ 0.5s

```
Create sweep with ID: 03j0ty10
Sweep URL: https://wandb.ai/okubo-lab/wandb-demo/sweeps/03j0ty10
```

Quiz 6: initialize a sweep

- Initialize a sweep using `wandb.sweep()`
  - Specify `sweep` with `sweep_config` you defined
  - Specify `project`
- Name the returned sweep id as `sweep_id`

Answer for quiz 6

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

Modify training code

- Training code

```
def train(config):
    train_loader, test_loader = create_dataloaders(config)
    model = cnn(config)
    optimizer = torch.optim.__dict__[config['optimizer']](params=model.parameters(), lr=con
    nowtime = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    wandb.init(project='wandb_demo', name=nowtime, config=config)

    for epoch in range(1, config['epochs']+1):
        model, train_loss = train_epoch(model, train_loader, optimizer)
        val_acc = eval_epoch(model, test_loader)
        print(f"epoch {epoch}: train_loss={train_loss:.2f}, val_acc= {100 * val_acc:.2f}%")

        '''
        Here is the answer
        '''
        wandb.log({"train_loss": train_loss, "val_acc": val_acc})

    wandb.finish() # Notify wandb that your run has ended and upload all log data to wandb

    return model
```

- Config, project name, and entity will be handled by controller
- Need to specify run name
- Access configuration using `wandb.config`

Quiz 7: modify training code to initialize runs

- Initialize wandb runs using `wandb.init()`. You only need to specify run name with `nowtime` variable.

- Retreive the configuration from wandb using `wandb.config` and assign it to a variable named `config`.

Answer for quiz 7

```python
def train(config=None):

    nowtime = datetime.datetime.now().strftime('%Y-%m-%d-%H:%M:%S')

    # config for specific runs will be handled by Wandb Controller

    '''
    Here is the answer
    '''
    wandb.init(name=nowtime)
    config = wandb.config

    train_loader, test_loader = create_dataloaders(config)
    model = cnn(config)
    optimizer = torch.optim.Adam(model.parameters(),
                                 lr=config['lr'])

    for epoch in range(1, config['epochs']+1):
        model, train_loss = train_epoch(model, train_loader, optimizer)
        val_acc = eval_epoch(model, test_loader)
        print(f"epoch {epoch}: train_loss={train_loss:.2f}, val_acc= {100 * val_acc:.2f}%")
        wandb.log({"train_loss": train_loss, "val_acc": val_acc})

    wandb.finish() # Notify wandb that your run has ended and upload all log data to wandb

    return model
```

Run with agent

- Sweep controller will wait for agents to start runs.

- Run agent with `wandb.agent()` by specifying:
  - `sweep_id`
  - `function`: training function
  - `count` (optional): The number of sweep config trials to try.

```
wandb.agent(sweep_id, train, count=100)
```

Quiz 8: Run sweep with sweep agent

- Use `wandb.agent()` to start the sweep
- Define
  - `sweep_id`
  - `function`
  - `count`
- If you have access to GPU from server, set `count` to be 20.

  If you are using Google Colab, set `count` to be 10.

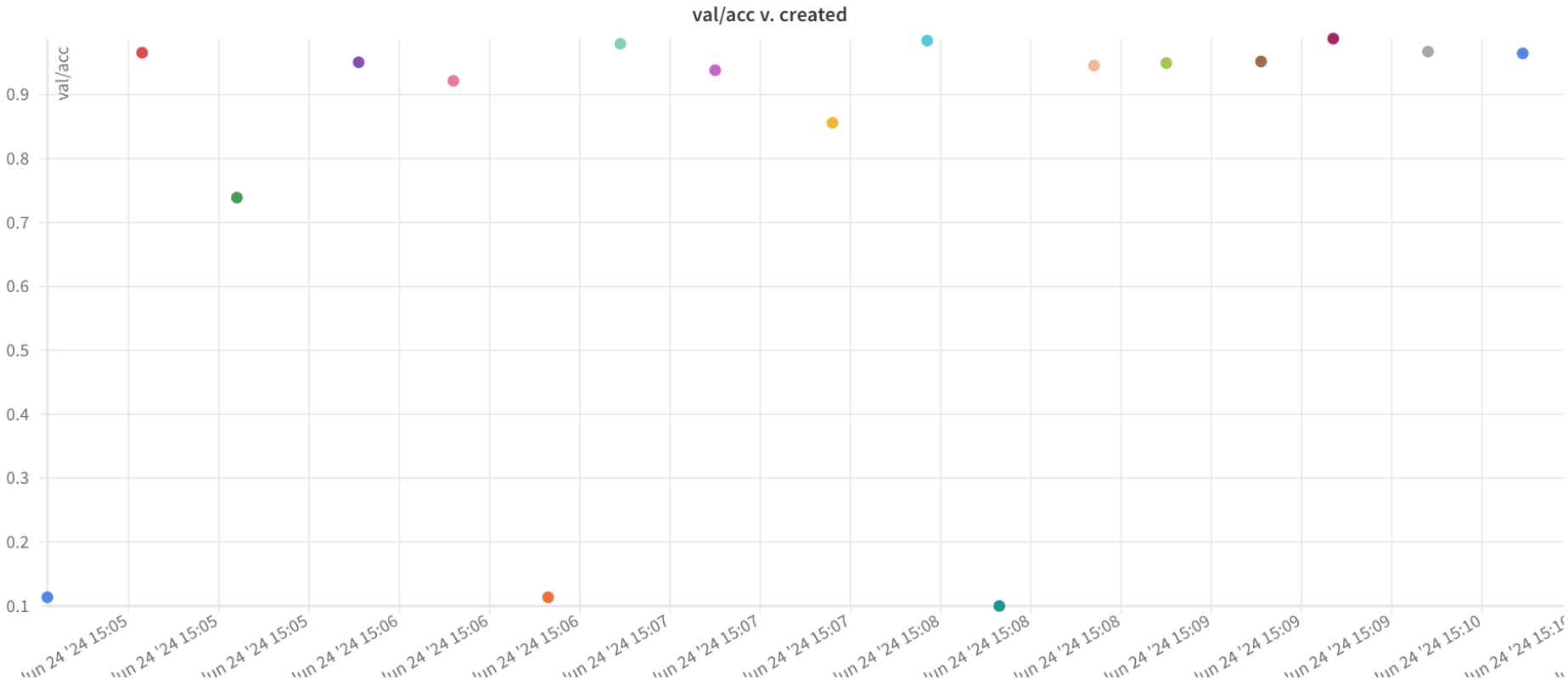
  If you are using CPU, set `count` to be 5.

Answer for quiz 8

```
# start the sweep using agent
wandb.agent(sweep_id=sweep_id, function=train, count=20)
```

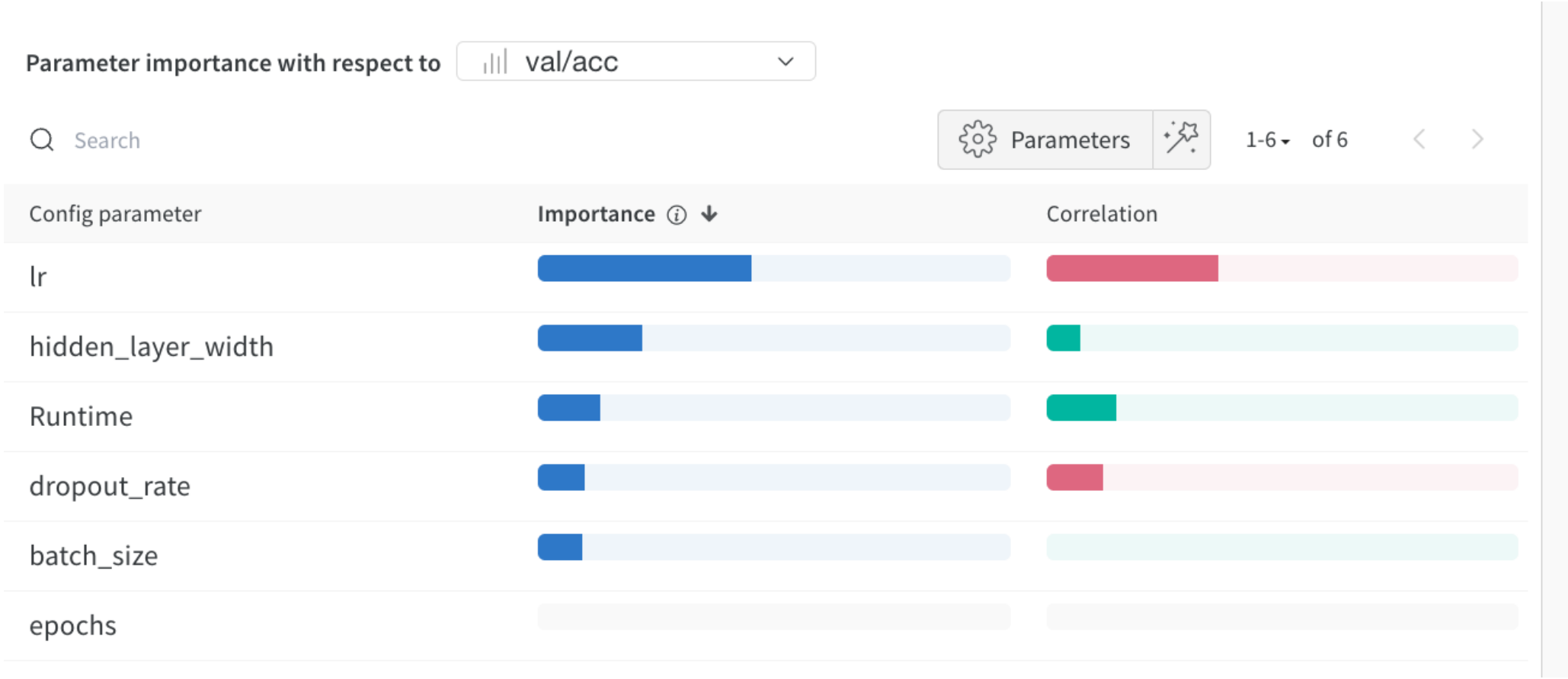
Result visualization

- "created time" v.s. "val/acc" scatter plot



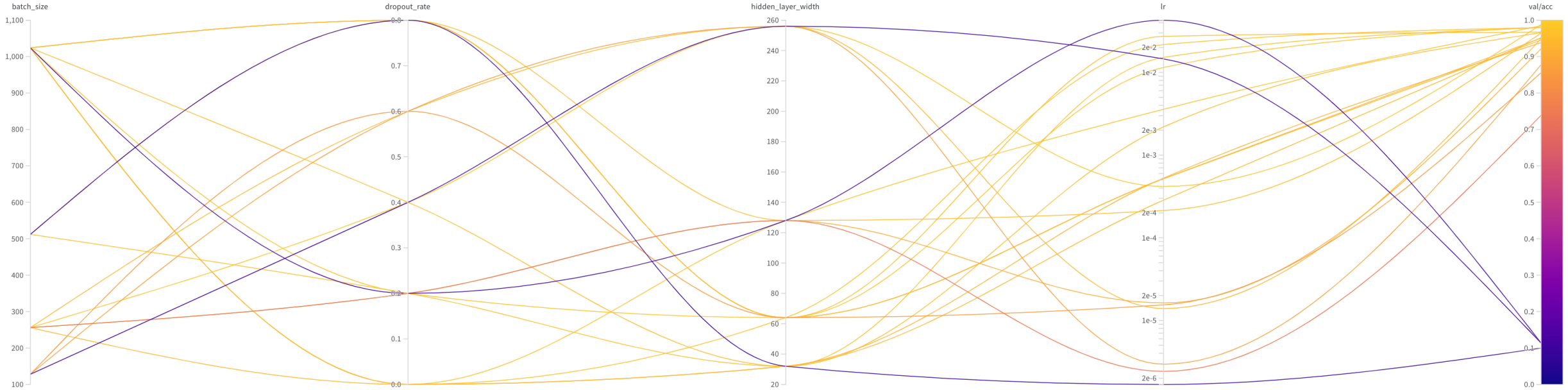
Result visualization

- parameter importance plot



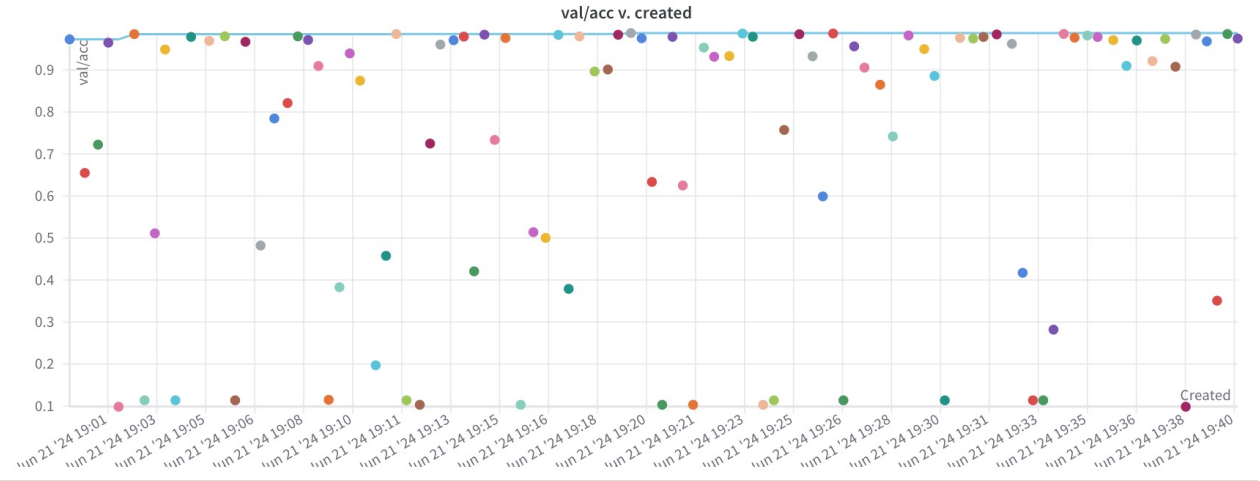
Result visualization

- parallel coordinates chart

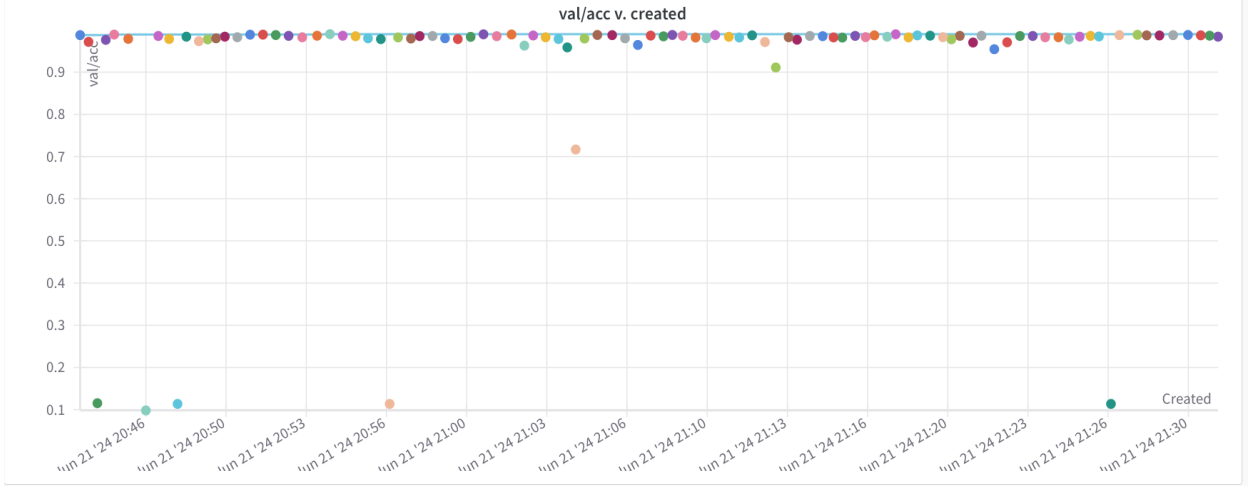


Random vs. Bayes

- Random



- Bayes

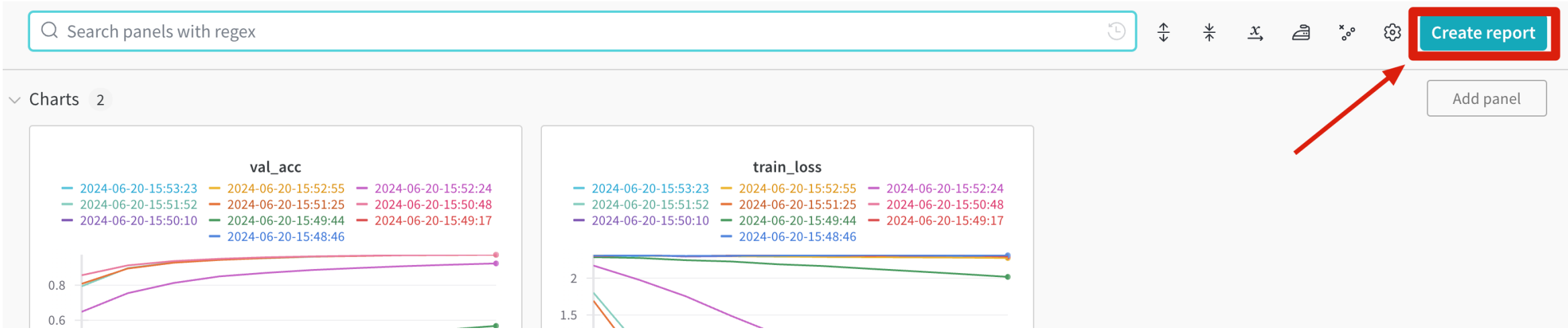


W&B Reports

https://docs.W&B.ai/guides/reports

W&B Reports

- (+) Better format than copy-and-paste
- (+) Can include code and equations.
- (+) The plots remain interactive.
- (-) Might take some time to get used to it.



Speech BCI experiments

Our results on the speech BCI benchmark 2024.

Tatsuo Okubo, Yurika, Xin Zheng

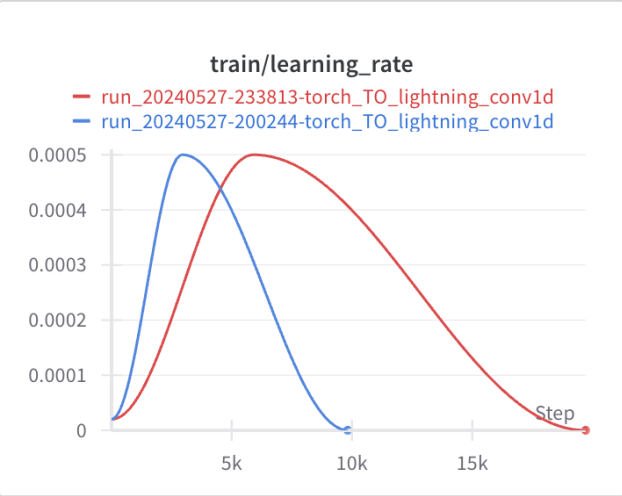
Model architecture

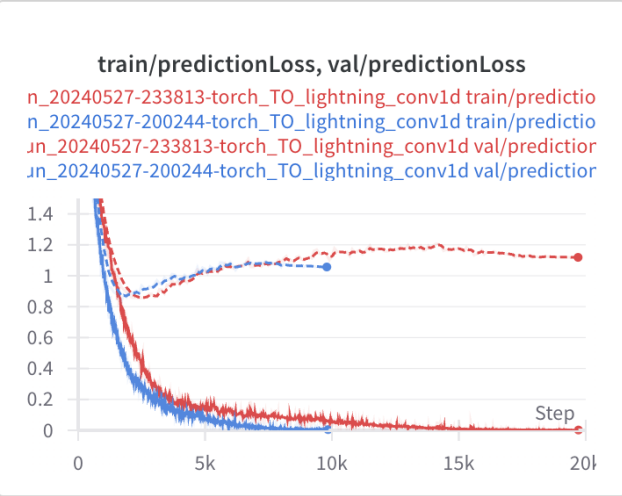
`conv1d` replaced the stacking (kernel size=65, stride=4, output dim=512).

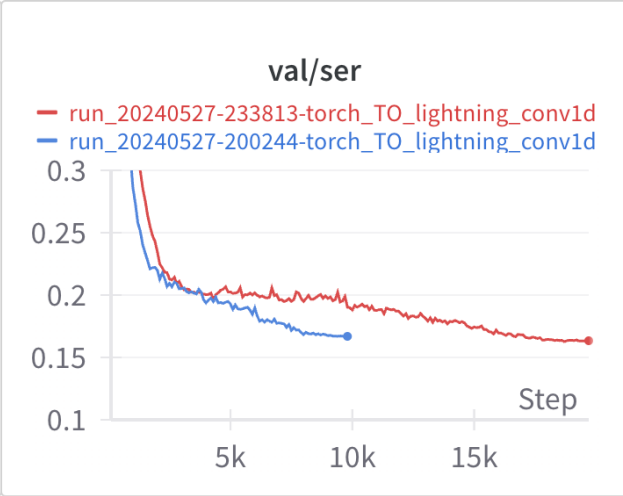
Batch size was 128 and Adam optimizer (eta=5e-4, beta_1=0.9, beta_2=0.999, eps=1e-8) was used together with OneCycleLR.

Longer training

Using the same maximum learning rate and OneCycleLR (cosine annealing), we tested the effect of training longer (more batches). The final val/ser was lower for the longer training (0.1634) compared to shorter (0.1669).







Team collaboration

W&B team

- Project sharing
  - allow multiple people to work on the same project, view and analyze experimental data.

speech_BCI_torch

Project Visibility	Team
Last active	2024/6/11 15:36:37
Owner	Xin Zheng
Contributors	2 users
Total runs	383
Total compute	10 days

run_20240606-11295…	Finished	Add notes	shin-zh
run_20240606-11051…	Finished	Add notes	shin-zh
run_20240603-16402…	Finished	Add notes	t-ok
run_20240603-16360…	Finished	Add notes	t-ok

How to collaborate with others?

- Create/join a team
- Specify `entity` when initialize wandb: `wandb.init(entity=...)`
- `Default`