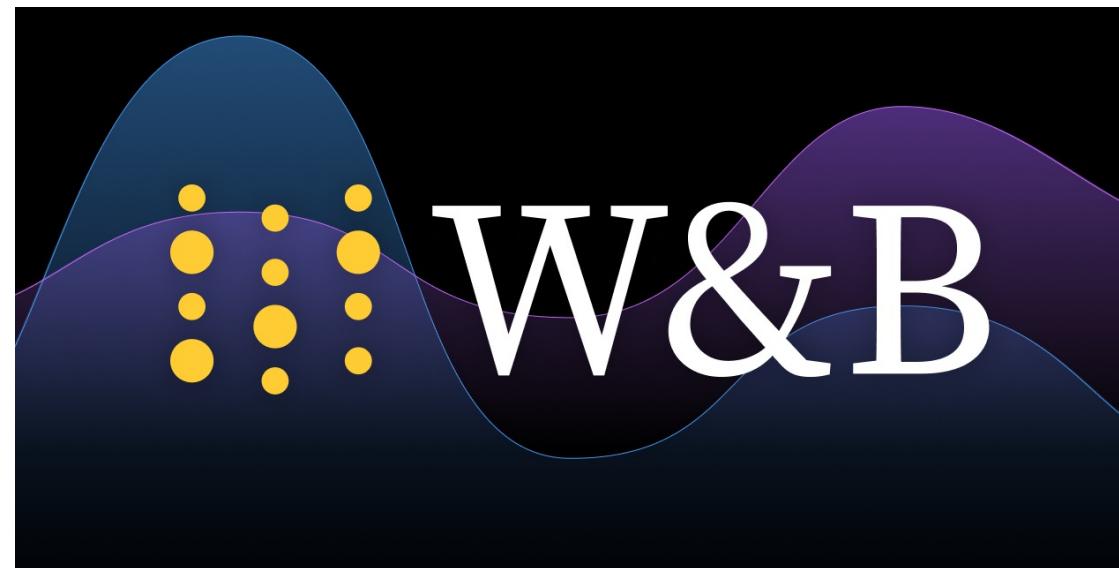


Tutorial on Weights & Biases

陈悦 & 郑馨

Okubo Lab

2024.06.26



Overview of the tutorial

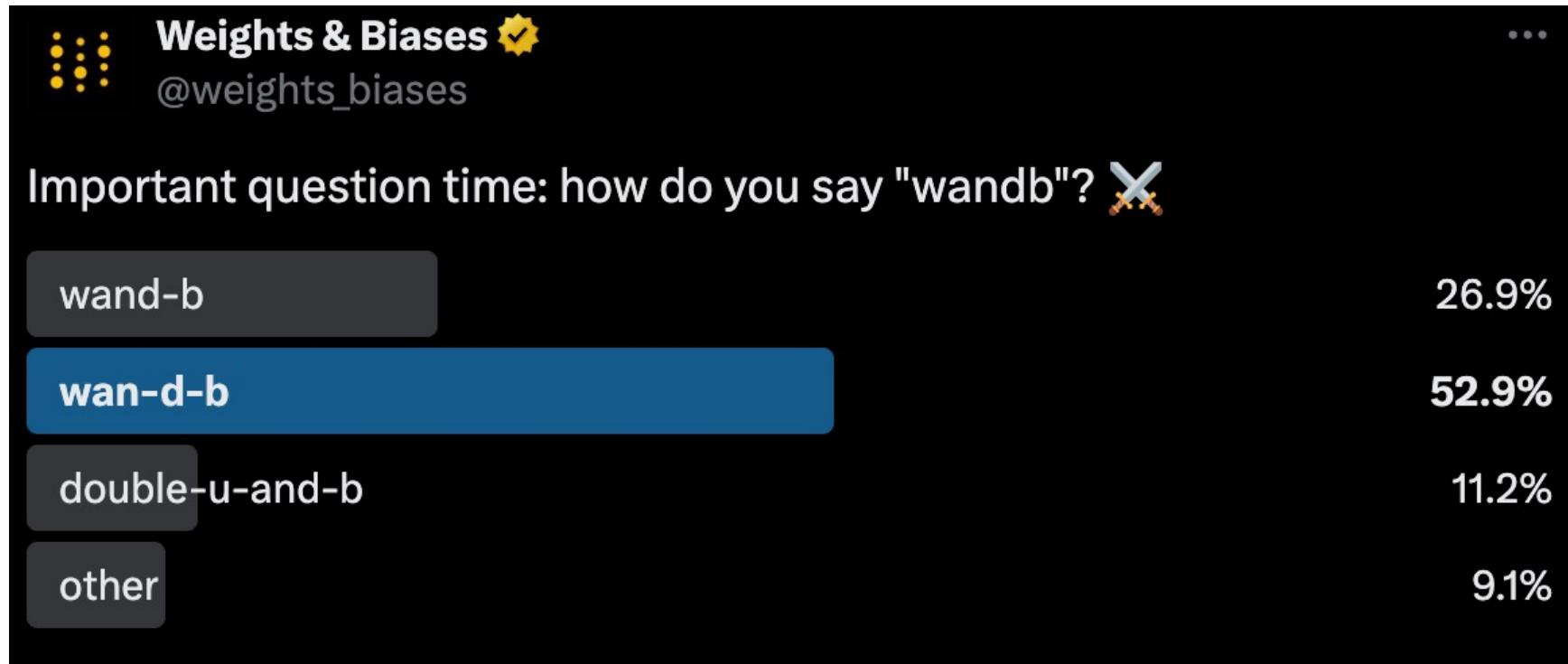
- Why we need W&B?
- Explanation on what W&B can do
- Hands-on exercises

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?

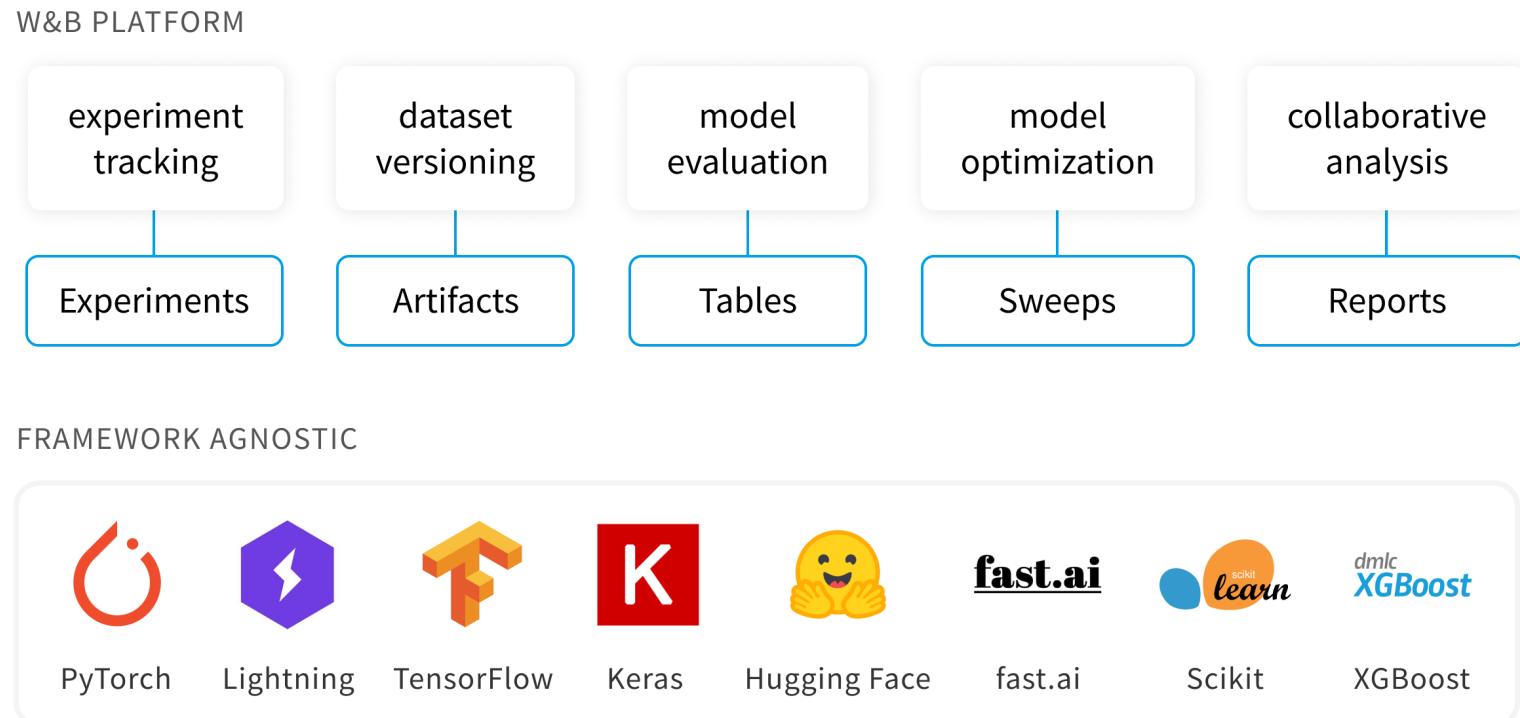


Why do you need "experiment tracking"?

- 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?

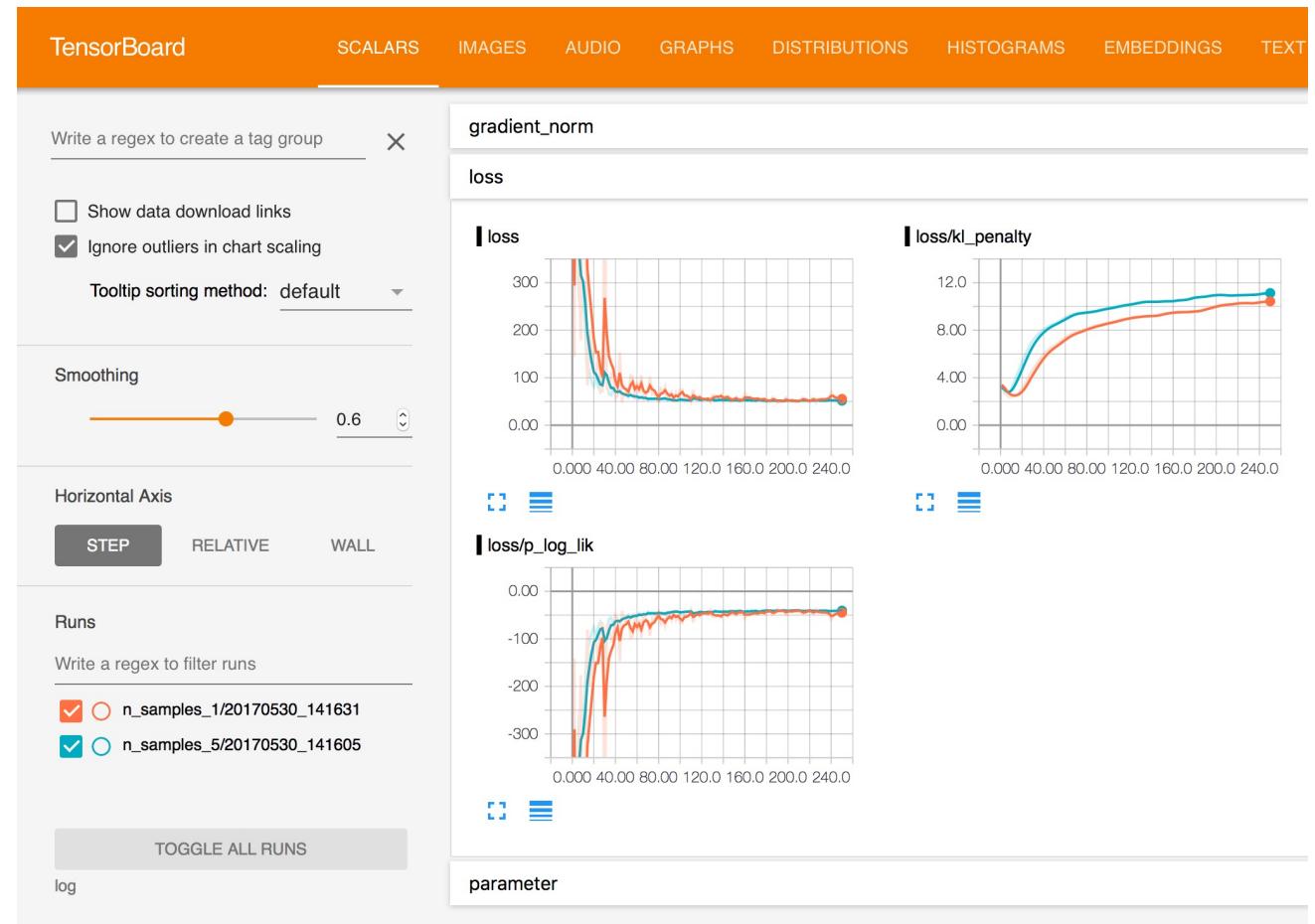
W&B

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



Isn't Tensorboard enough?

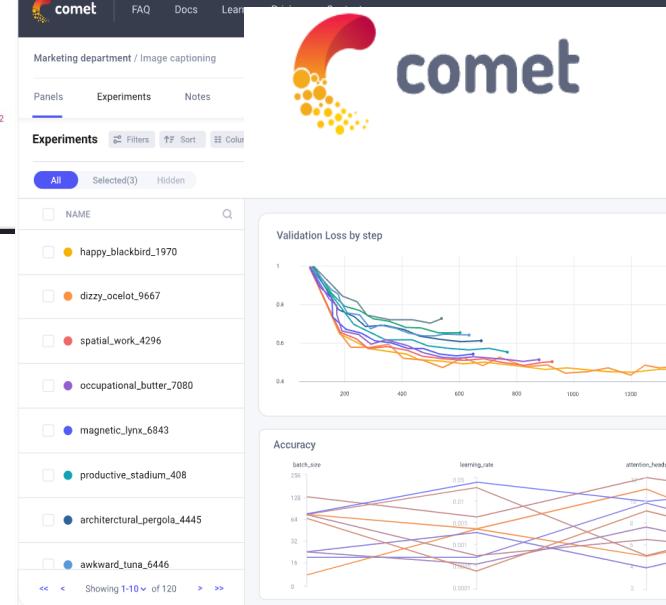
- (+) basic visualization
- (+) long history, large community
- (-) local
- (-) limited scalability
- (-) no collaboration function



Other options

The Neptune.ai interface displays various data visualizations and code snippets. At the top, there's a chart showing 'accuracy' and 'loss' over time. Below it, a 'source_code/files' section shows a file named 'hello_neptune.py'. The code snippet is as follows:

```
1 import neptune
2
3 # Initialize Neptune and create a new run
4 run = neptune.init_run(
5     project="common/quicksarts",
6     api_token=neptune.ANONYMOUS_API_TOKEN,
7     tags=["quickstart", "script"],
8     dependencies="infer", # to infer dependencies. You can also pass the path to the requirements.txt file
9 )
```

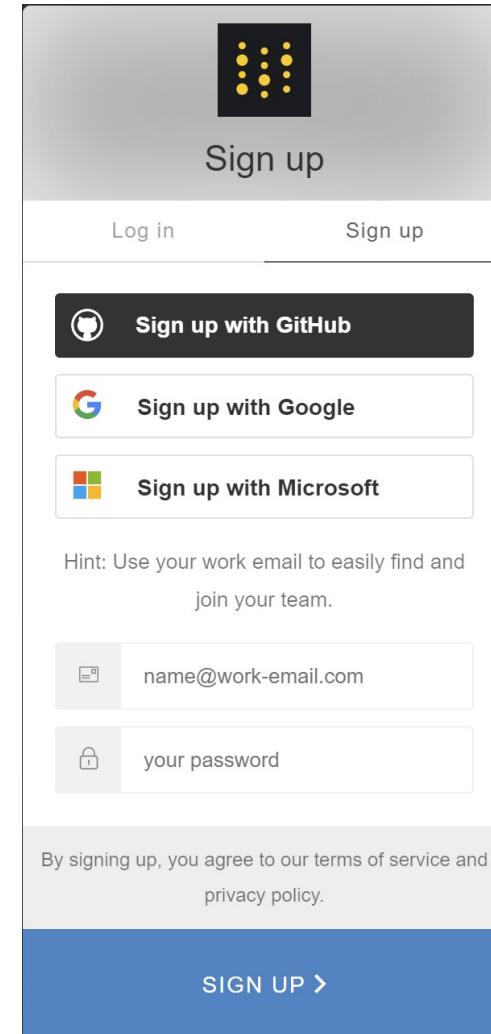


W&B basics

How can I create a W&B account?

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

The screenshot shows the Weights & Biases homepage. At the top, there's a navigation bar with links for Platform, Solutions, Enterprise, Resources, Company, Docs, and Pricing. On the right side of the nav bar are 'LOG IN' and 'SIGN UP' buttons. Below the nav bar, there's a large call-to-action button labeled 'GET STARTED'. To the right of this button is a dark rectangular box containing the text 'Confidently iterate on GenAI applications with Weave' and a yellow '▶' button. Below this box is a photo of a man with the caption 'Alex Volkov AI Evangelist @ Weights & Biases'. The background of the page features a dark theme with some abstract light-colored lines. At the bottom, it says 'The world's leading AI teams trust Weights & Biases' and lists several logos: Toyota Research Institute, Spotify, co:here, runway, mosaic^{ML}, Meta, and Stanford University.





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.

```
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

<input type="checkbox"/>  Name (13 visualized)	State	Notes	GitHub	Commit
 2024-06-21 11:01:39	 Finished	Add notes	https://github.com/CIBR-C/...	5297e1fd74426a382b268fac6

How to manage the version?

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

The screenshot shows the Wandb Artifacts interface. On the left, there's a sidebar with a tree view of artifacts:

- dataset/mnist/v0 [latest]
- wandb-history
- code/ipynb/v1 [latest]
- v0
- run_table

On the right, the main panel displays the details for the selected artifact:

mnist Version 0

Version overview

Full Name	cy970628/wandb_demo/mnist:v0	Created At	June 19th, 2024 18:12:40
Aliases	@ latest @ v0 +	Num Consumers	0
Digest	92cb3226c9756f76c7865c620eb313ee	Num Files	8
Created By	2024-06-19 18:12:07	Size	66.5MB
Description	What changed in this version?	TTL Remaining ⓘ	Inactive

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

mnist		
Version 0		▼
Version	Metadata	Usage
Files	Lineage	
> root / MNIST / raw	Directory	▼
 t10k-images-idx3-ubyte	7.8MB	Download
 t10k-images-idx3-ubyte.gz	1.6MB	Download
 t10k-labels-idx1-ubyte	10.0KB	Download
 t10k-labels-idx1-ubyte.gz	4.5KB	Download
 train-images-idx3-ubyte	47.0MB	Download
 train-images-idx3-ubyte.gz	9.9MB	Download
 train-labels-idx1-ubyte	60.0KB	Download
 train-labels-idx1-ubyte.gz	28.9KB	Download

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"

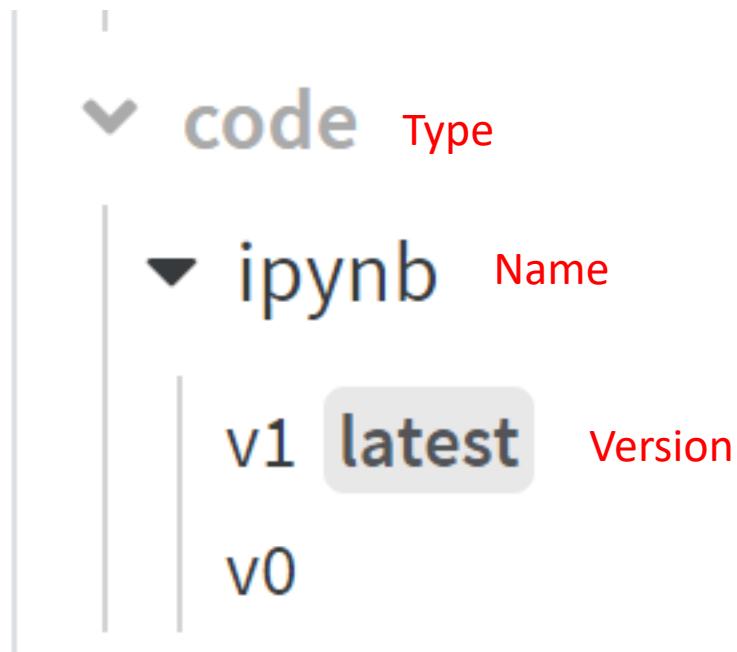
The screenshot shows the WandB web interface. The top navigation bar includes the project name 'cy970628 > Projects > wandb_demo > Artifacts > dataset > mnist > v0'. On the right, there's a 'Invite teammates' button. The left sidebar has sections for Overview, Workspace, Runs, Jobs, Automat., Sweeps, Reports, and Artifacts, with the 'Artifacts' icon highlighted by a red box. Below the sidebar is a search bar labeled 'Find matching artifacts'. The main content area is titled 'mnist Version 0'. It features tabs for Version, Metadata, Usage, Files, and Lineage, with 'Version' selected. The 'Version overview' section displays the following details:

Full Name	cy970628/wandb_demo/mnist:v0	Created At	June 19th, 2024 18:12:40
Aliases	@latest @v0 +	Num Consumers	0
Digest	92cb3226c9756f76c7865c620eb313ee	Num Files	8
Created By	2024-06-19 18:12:07	Size	66.5MB
Description	What changed in this version?	TTL Remaining ⓘ	Inactive

A large red box covers the bottom portion of the sidebar and the bottom of the main content area, with the text 'Click here!' overlaid.

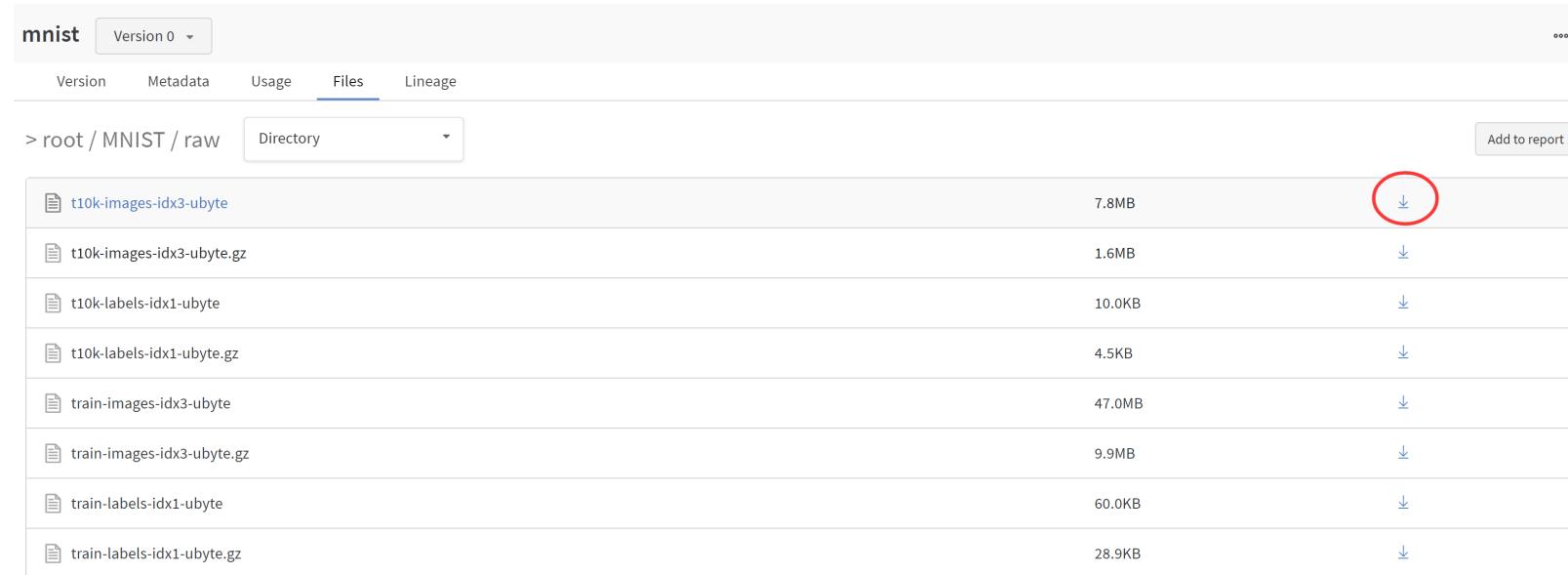
Where can I find my files?

- Choose type and name



How to reuse the files?

- Directly download from website



A screenshot of a dataset page titled "mnist Version 0". The page has tabs for "Version", "Metadata", "Usage", "Files", and "Lineage". The "Files" tab is selected. The path is shown as "> root / MNIST / raw". A dropdown menu shows "Directory". A button "Add to report" is visible. The table lists the following files:

File	Size	Action
t10k-images-idx3-ubyte	7.8MB	
t10k-images-idx3-ubyte.gz	1.6MB	
t10k-labels-idx1-ubyte	10.0KB	
t10k-labels-idx1-ubyte.gz	4.5KB	
train-images-idx3-ubyte	47.0MB	
train-images-idx3-ubyte.gz	9.9MB	
train-labels-idx1-ubyte	60.0KB	
train-labels-idx1-ubyte.gz	28.9KB	

- 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>

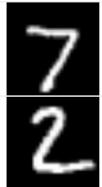
How to easily visualize the experiment data/results?

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

Tables 1

Add panel

runs.summary["cases"]

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

How to easily visualize the experiment data/results?

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

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

```
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

from utils import show_cases

def train(config):
    train_loader, test_loader = create_dataloaders(config)
    model = cnn(config)
    optimizer = torch.optim.__dict__[config['optimizer']](
        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})

# 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

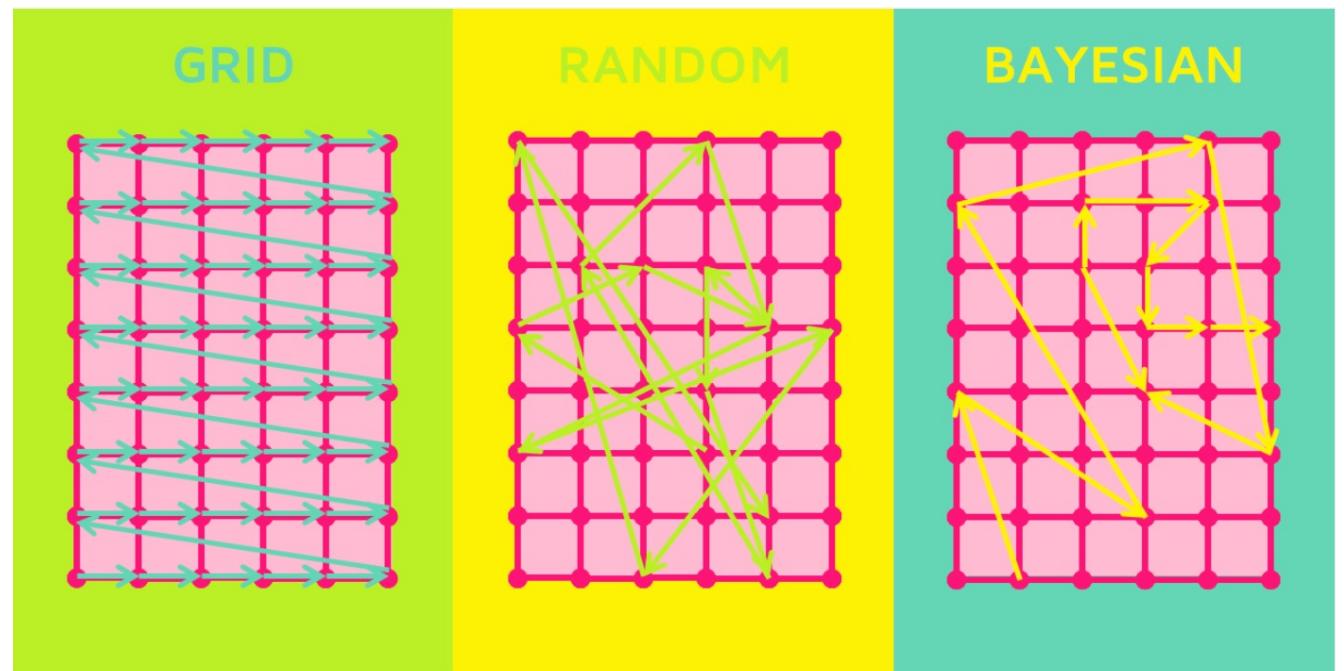
```
'''  
  
column_name = ['Image', 'Target', 'Prediction']  
cases = wandb.Table(columns=column_name, data=results)  
wandb.log({'cases':cases})
```

W&B Sweeps

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

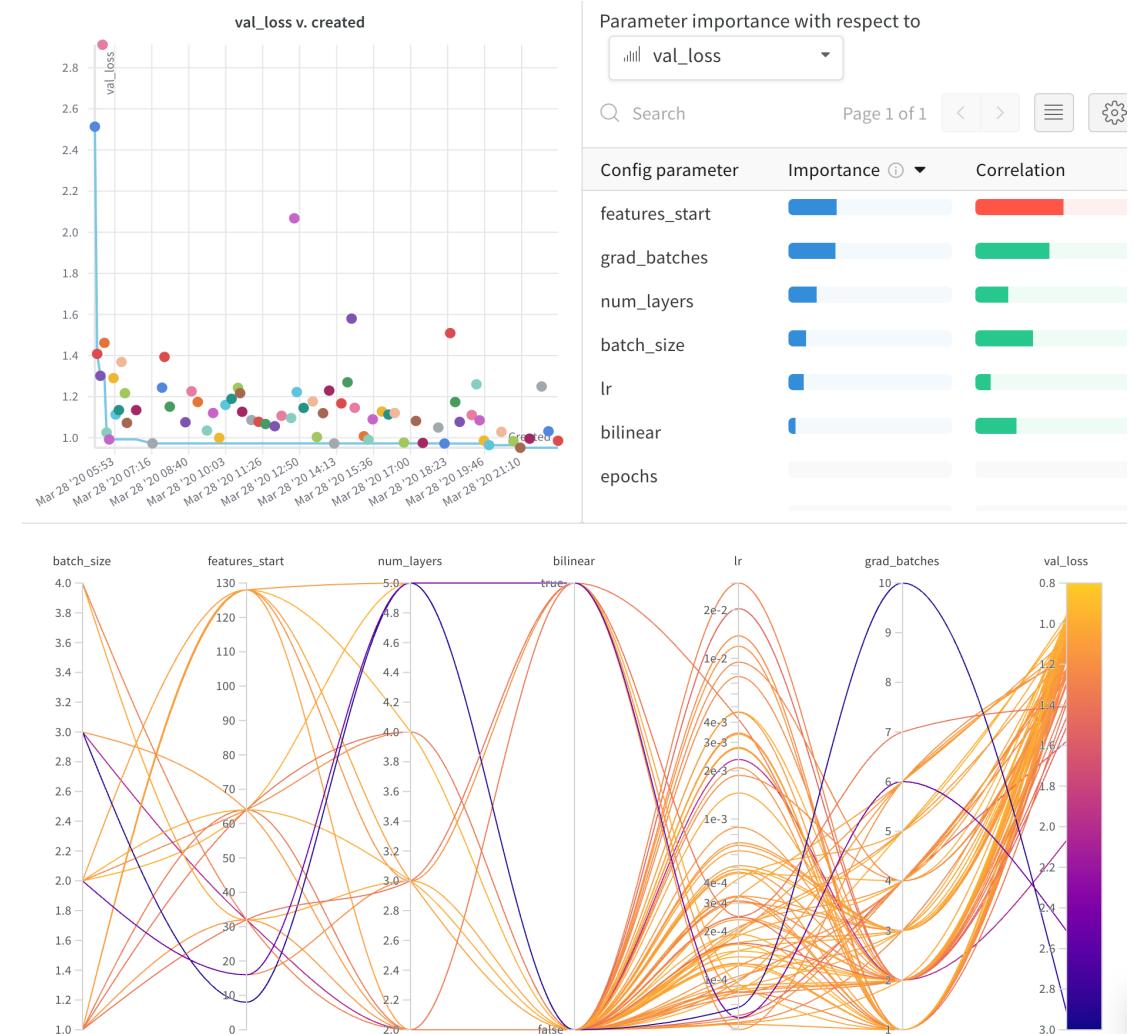
Why sweeps?

- 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

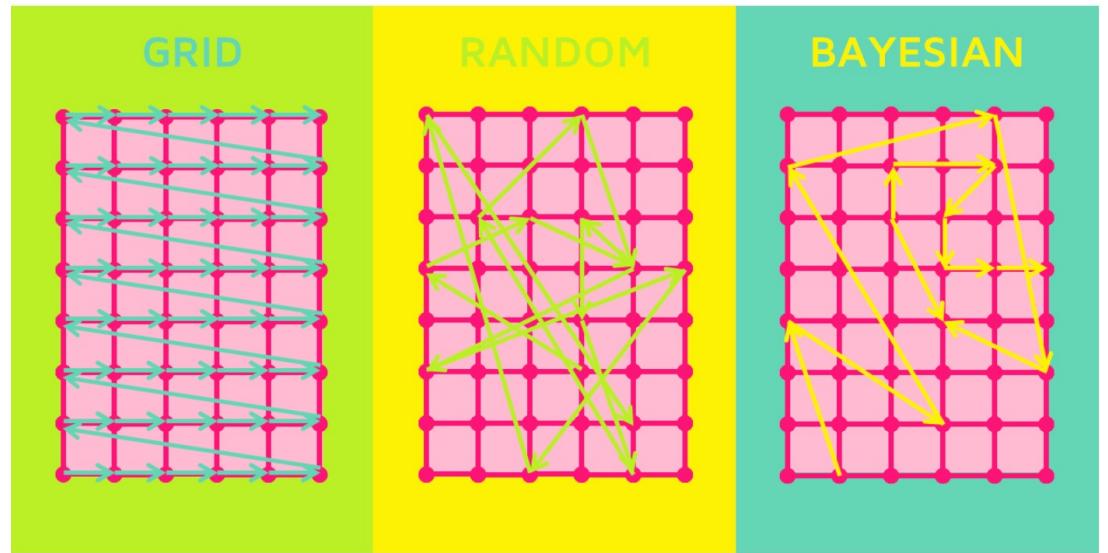
- run_cap
- early_terminate

example configuration

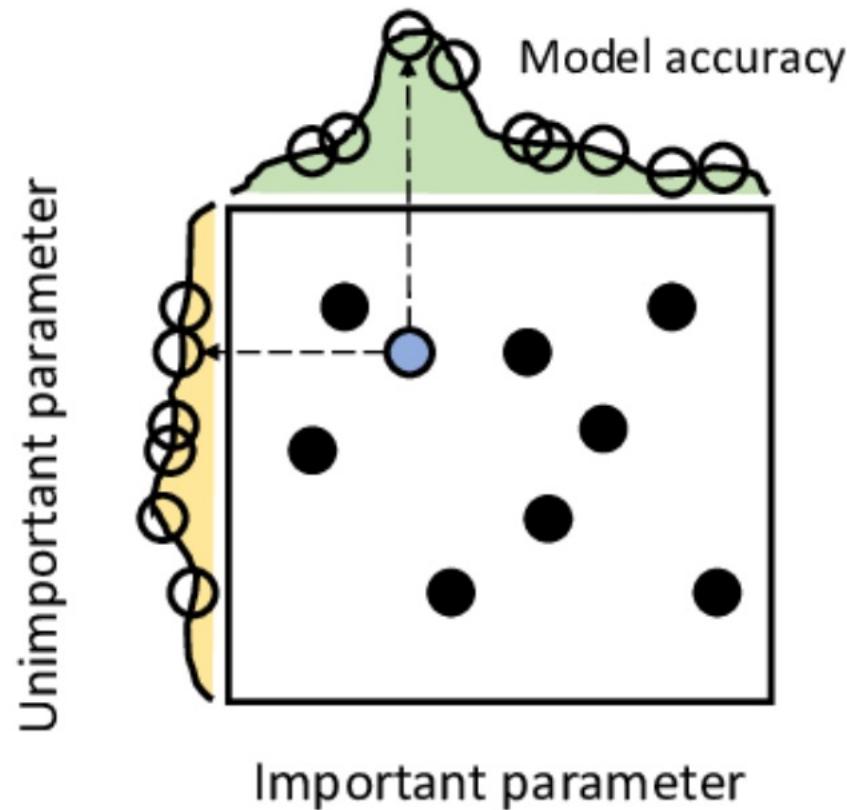
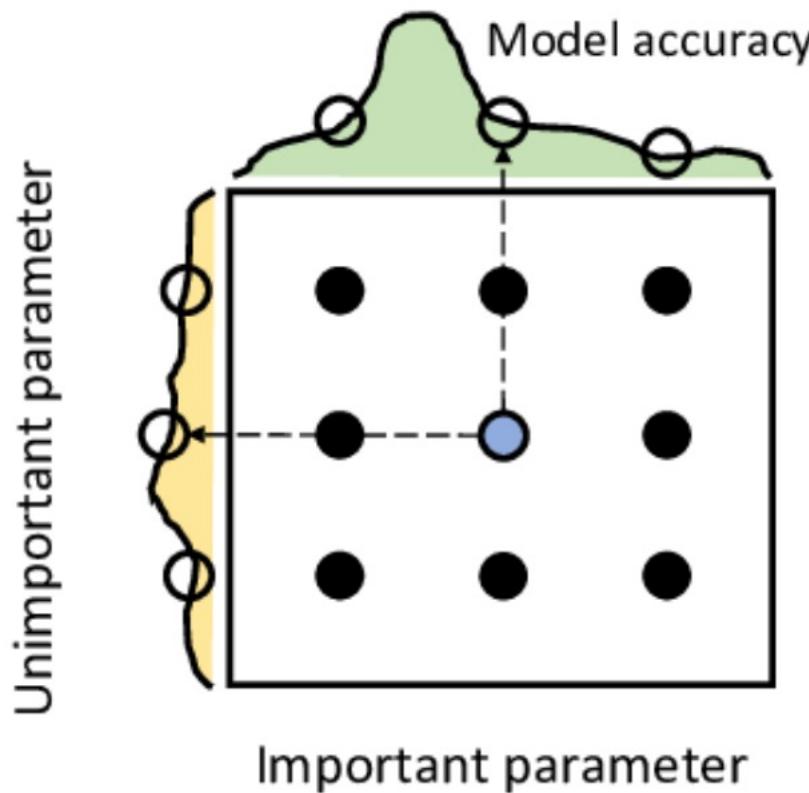
```
{'method': 'random',
'metric': {'goal': 'maximize', 'name': 'val_acc'},
'name': 'random_exp',
'parameters': {'batch_size': {'values': [128, 256, 512, 1024]},
'dropout_rate': {'values': [0.0, 0.2, 0.4, 0.6, 0.8]},
'epochs': {'value': 5},
'hidden_layer_width': {'values': [32, 64, 128, 256]},
'lr':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

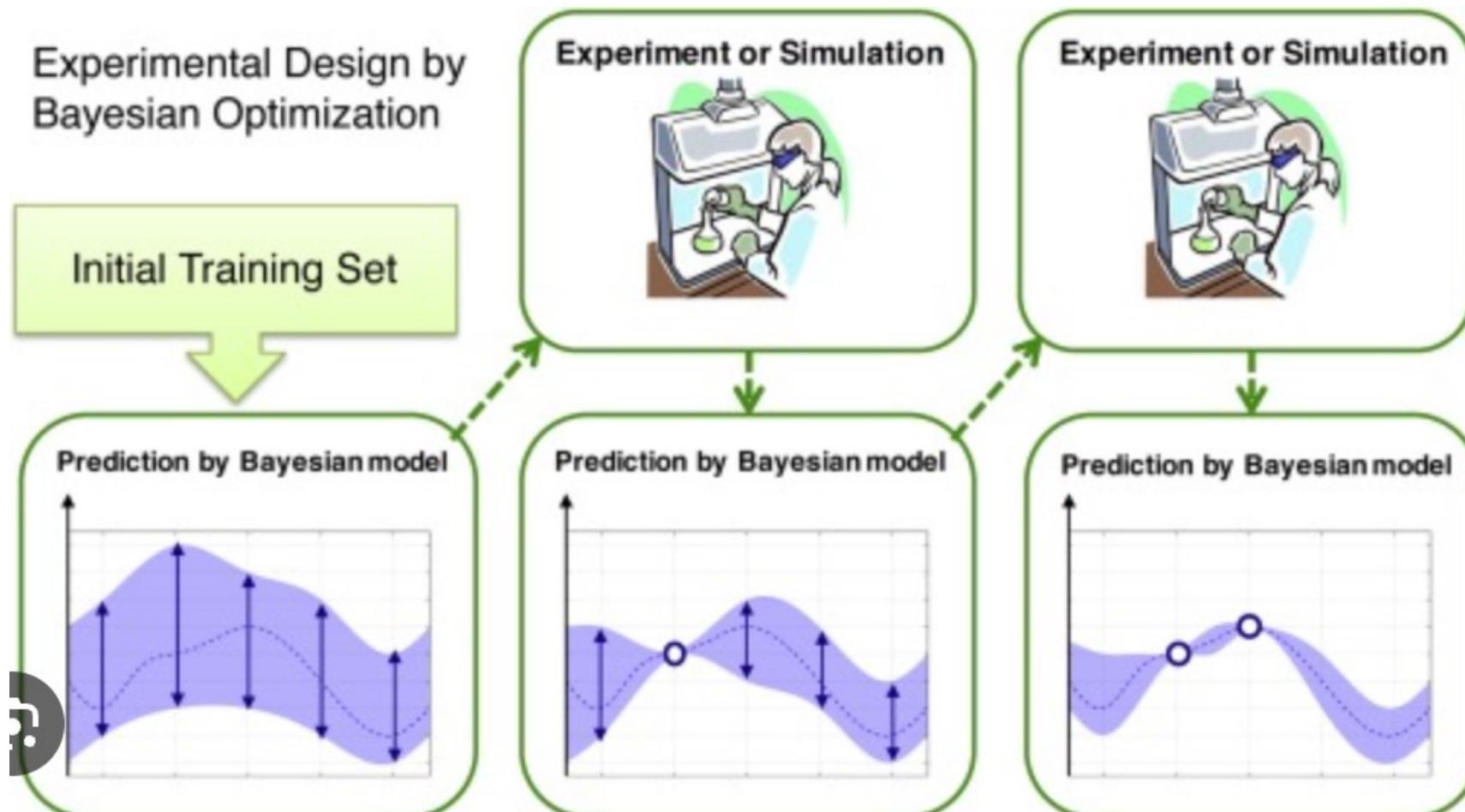


Grid vs. Random search



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  
},
```

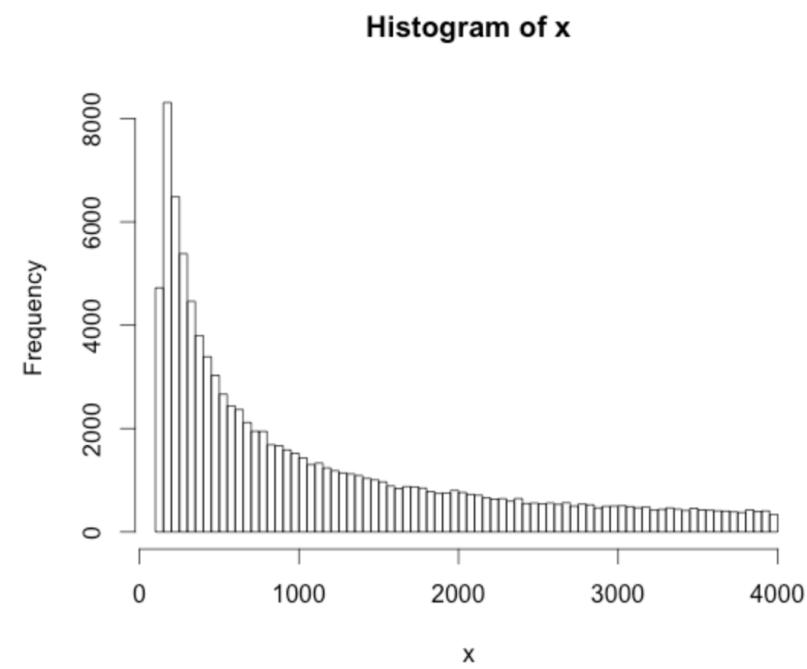
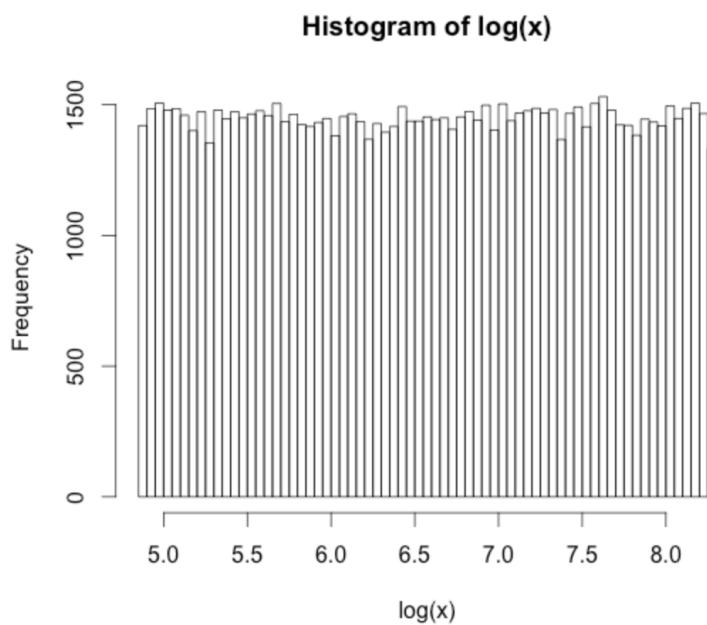
- Distribution: specify a distribution to sample from
 - key: distribution, min, max

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

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

log-uniform distribution

- Log of the variable 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'  
    }  
}
```

```
# 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]
    }
}
```

Quiz 5: define a sweep config

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

```
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.

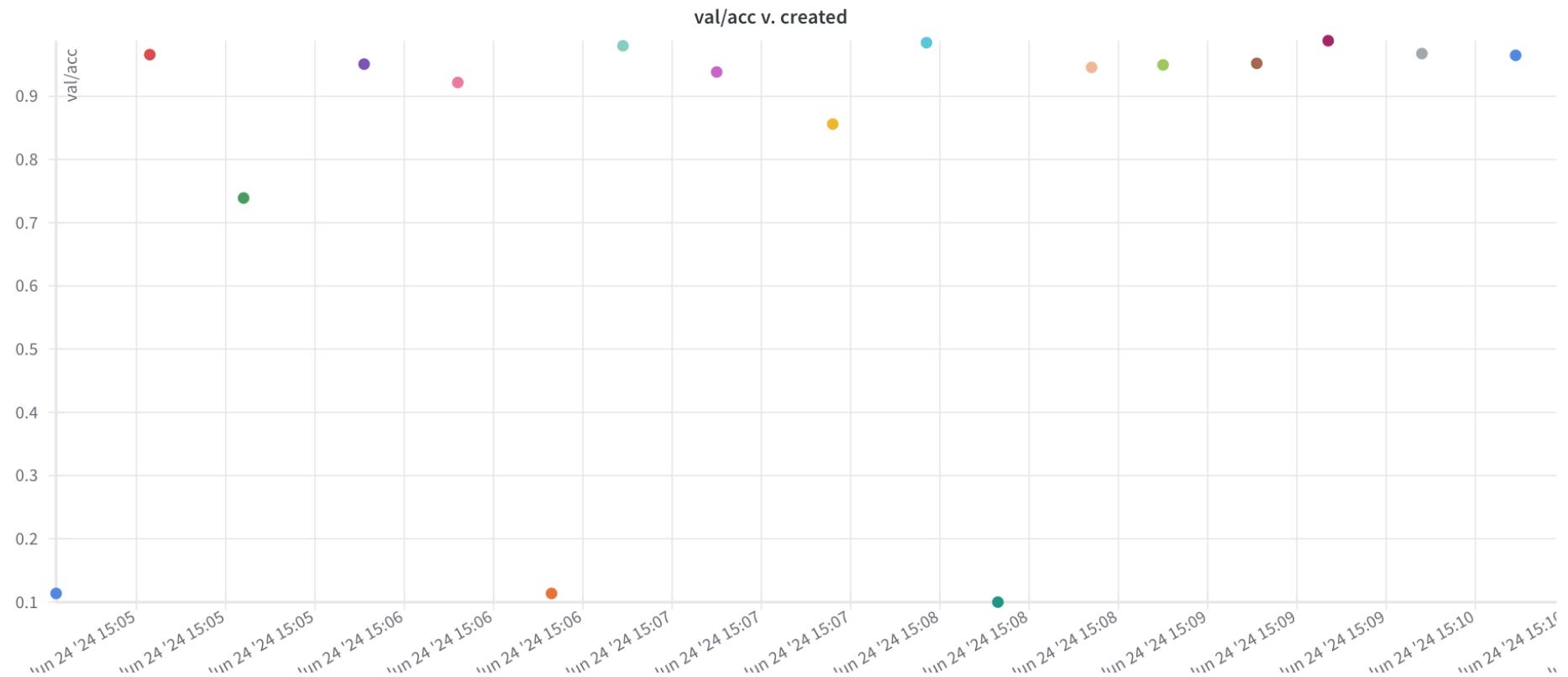
If you are using CPU, set `count` to be 5.

Answer for quiz 8

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

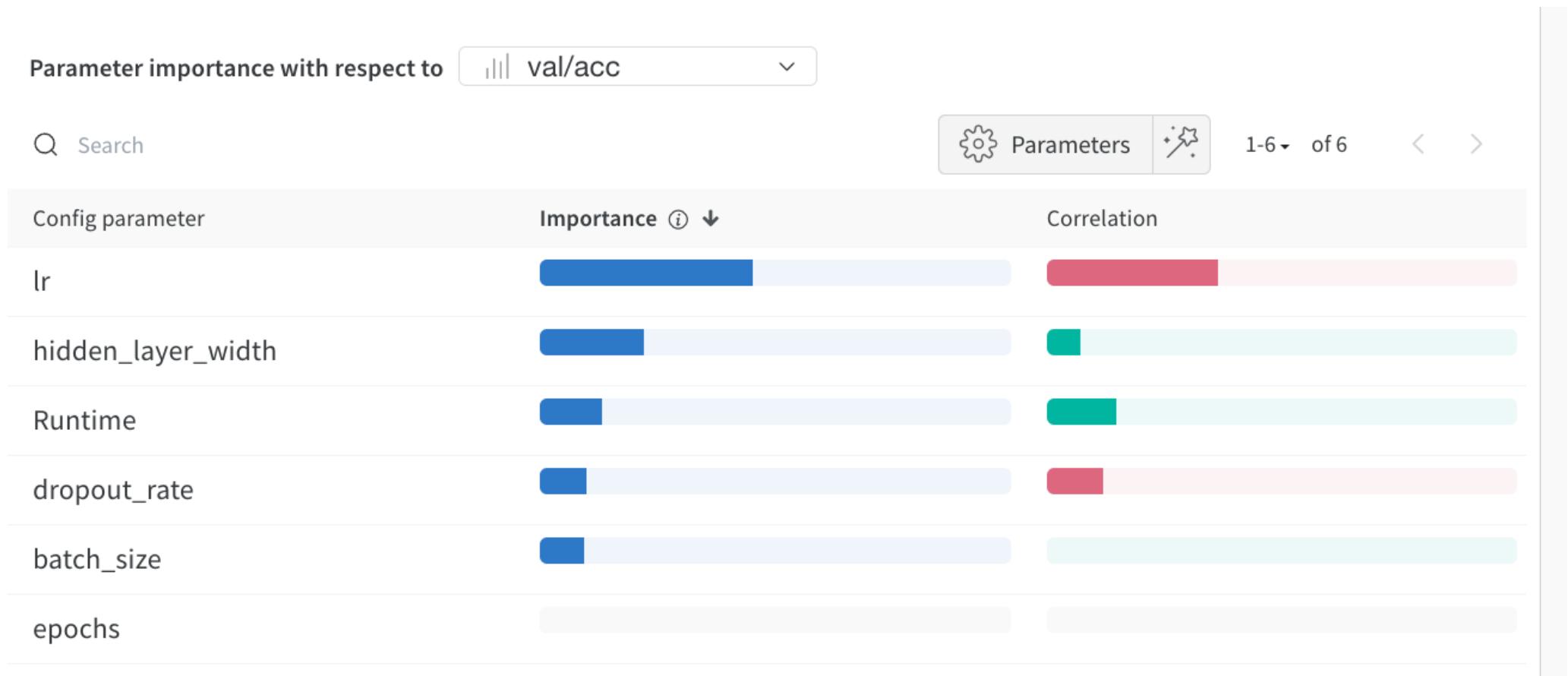
Result visualization

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



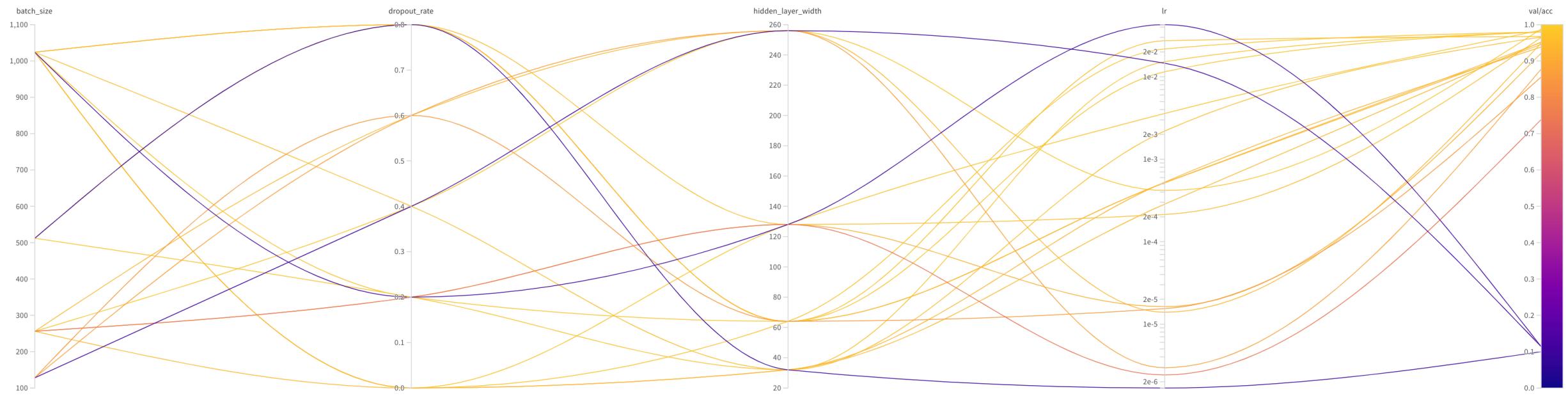
Result visualization

- parameter importance plot



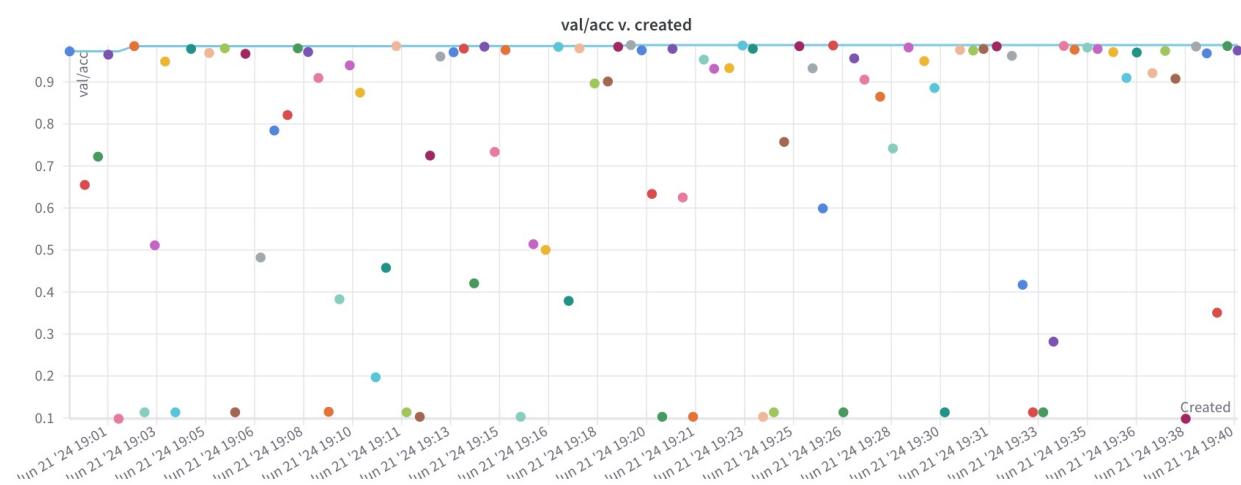
Result visualization

- parallel coordinates chart

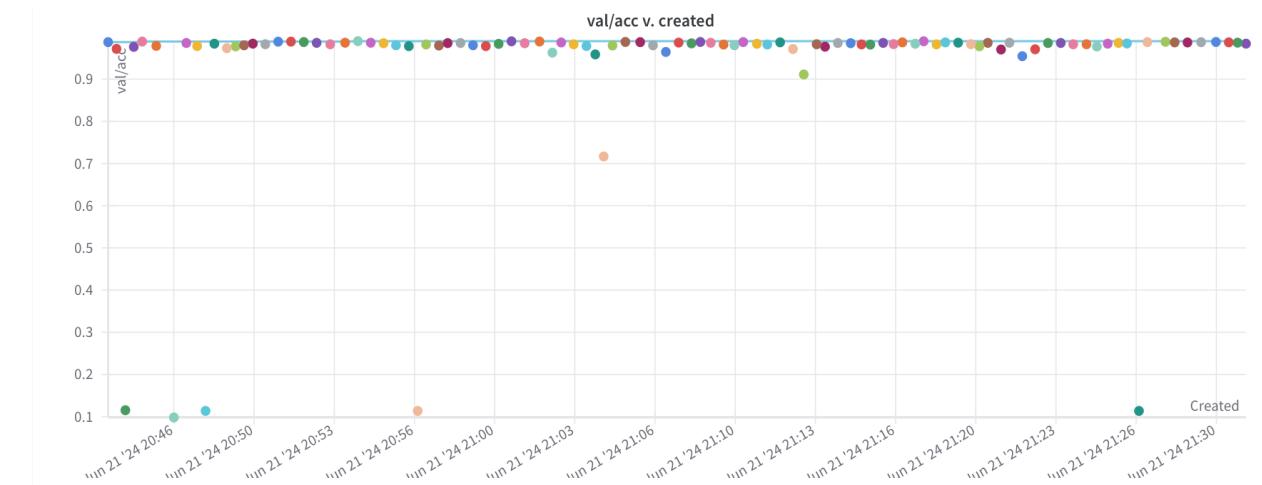


Random vs. Bayes

- Random



- Bayes

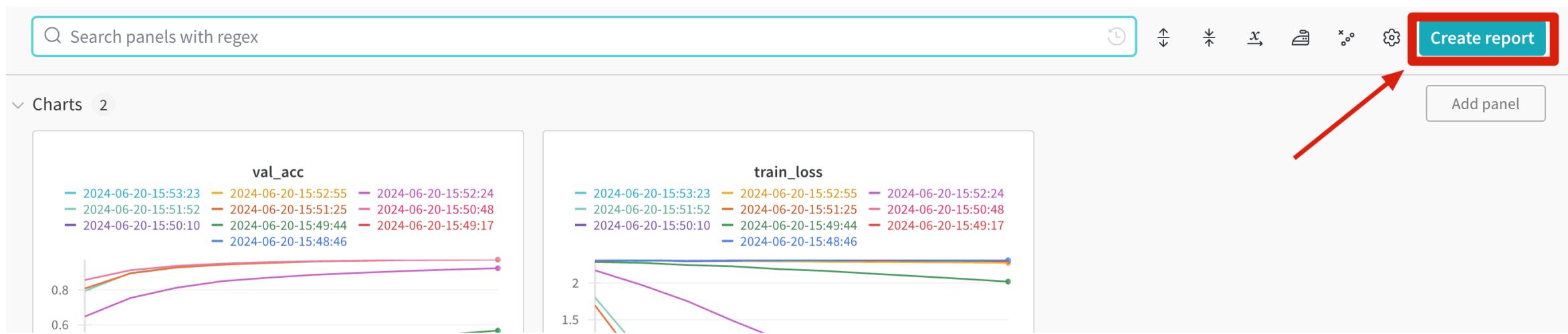


W&B Reports

<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

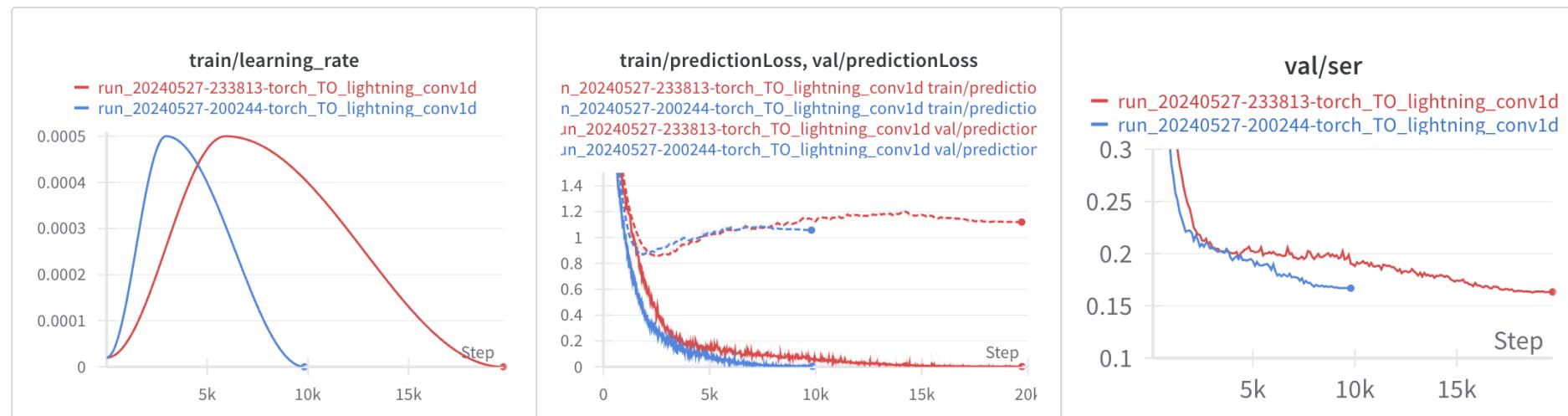
Model architecture

`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.

- run_20240606-11295...
- run_20240606-11051...
- run_20240603-16402...
- run_20240603-16360...

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	

- Finished Add notes shin-zh
- Finished Add notes shin-zh
- Finished Add notes t-ok
- Finished Add notes t-ok

How to collaborate with others?

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

The screenshot shows the Wandb web interface. On the left, there's a sidebar with navigation links: Profile, User settings (which is highlighted with a red oval), Auto-refresh (with a toggle switch), Account (showing 'okubo-lab-org Academic · 3 users'), and Usage and billing. The main area displays 'Default team' settings, which include 'Default location to create new projects' (set to 'cy970628') and 'Default project privacy in your personal account' (set to 'okubo-lab'). A dropdown menu also shows 'cy970628' and 'okubo-lab'. At the bottom, there's an option to 'Enable code saving in your personal account'.