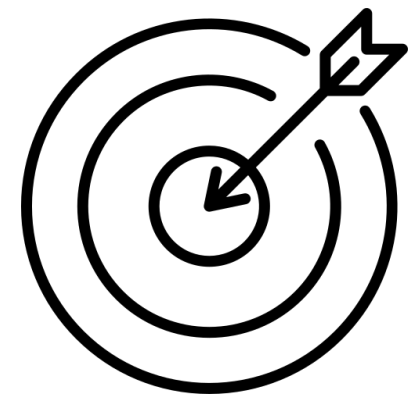


# Monitoring ML experiments with **Weights & Biases**

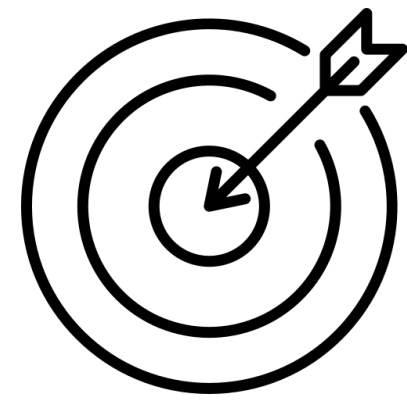
Tutorial 3  
Sungho Bae

## Intro: Motivation

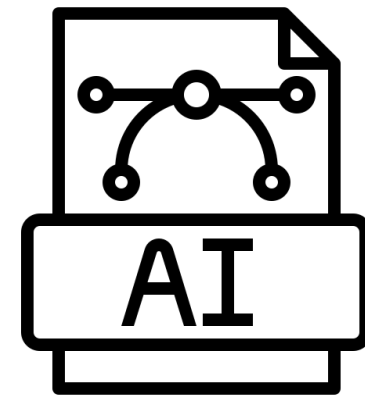


Goal

# Intro: Motivation



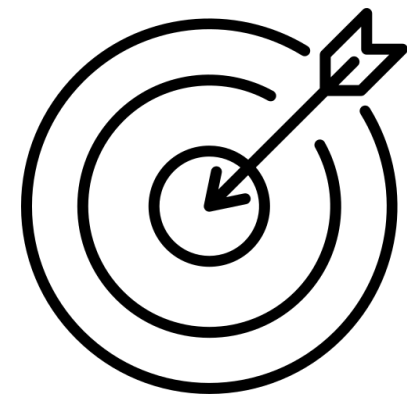
Goal



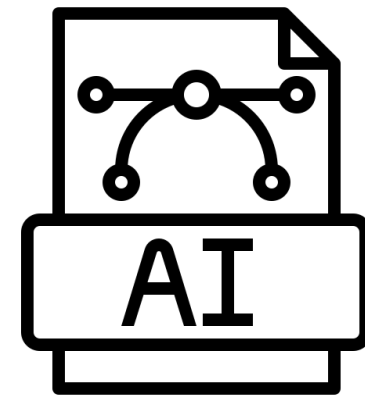
Model

# Intro: Motivation

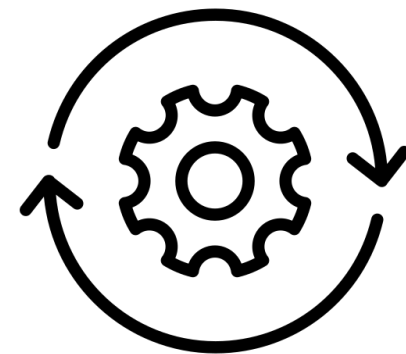
Q. Have you ever done **train/valid metric monitoring** in Jupyter Notebook or colab as follows?



Goal



Model



Experiments

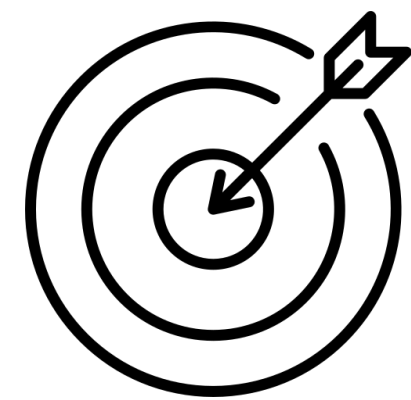
a lots of,,,,,,

```
Train on 2395 samples, validate on 295 samples
Epoch 1/500
2395/2395 [=====] - 0s 132us/step - loss: 0.0559 - val_loss: 0.0151
Epoch 2/500
2395/2395 [=====] - 0s 67us/step - loss: 0.0101 - val_loss: 0.0095
Epoch 3/500
2395/2395 [=====] - 0s 66us/step - loss: 0.0071 - val_loss: 0.0067
Epoch 4/500
2395/2395 [=====] - 0s 68us/step - loss: 0.0060 - val_loss: 0.0064
Epoch 5/500
2395/2395 [=====] - 0s 64us/step - loss: 0.0057 - val_loss: 0.0063
Epoch 6/500
2395/2395 [=====] - 0s 68us/step - loss: 0.0056 - val_loss: 0.0063
```

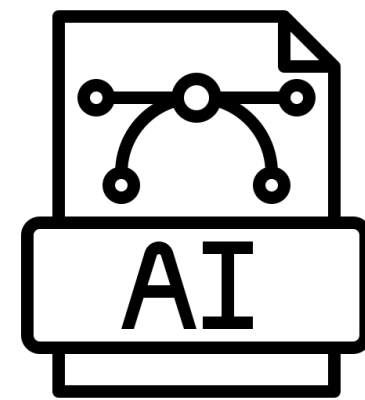


# Intro: Motivation

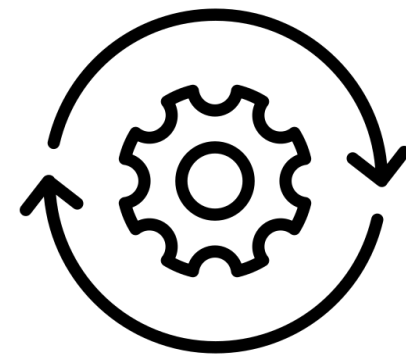
Q. Have you ever struggled to **compare and navigate the results** and experiences for each experiment?



Goal

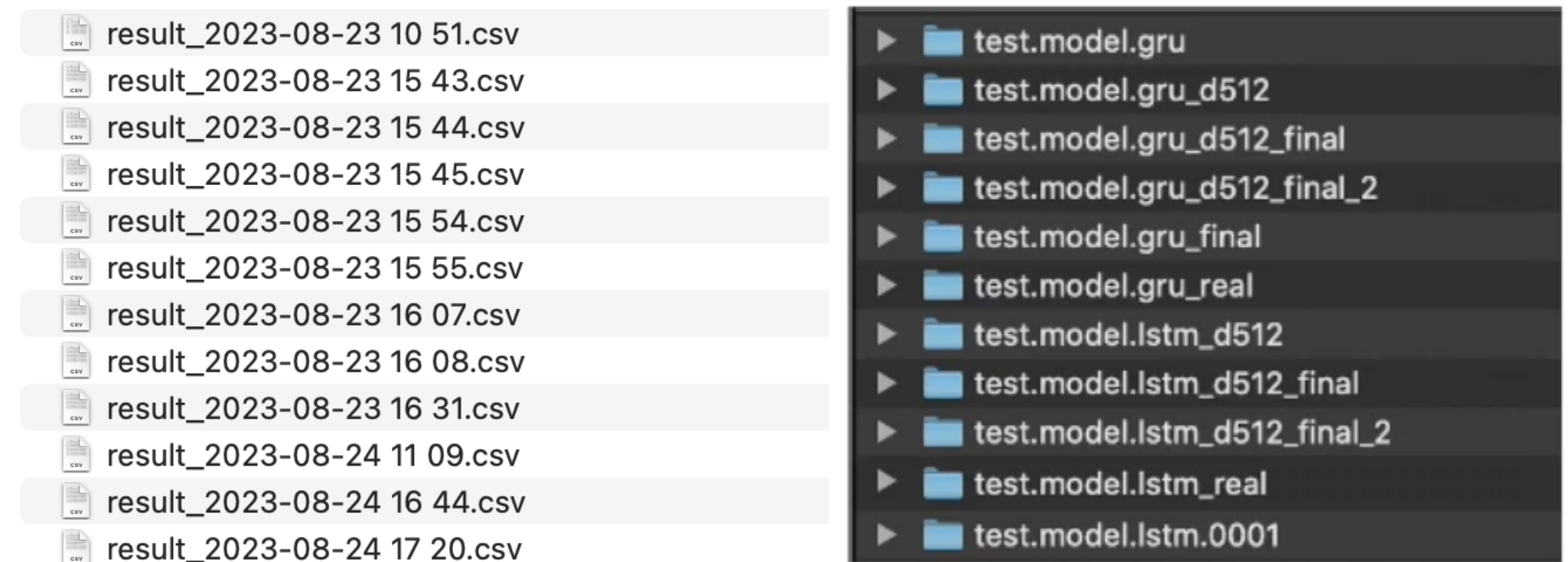


Model



Experiments

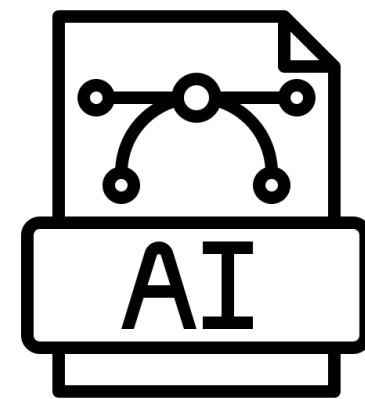
a lots of,,,,,,



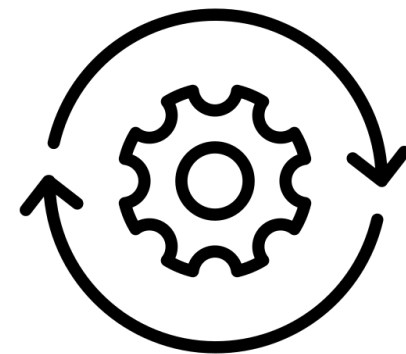
# Intro: Motivation



Me



Model

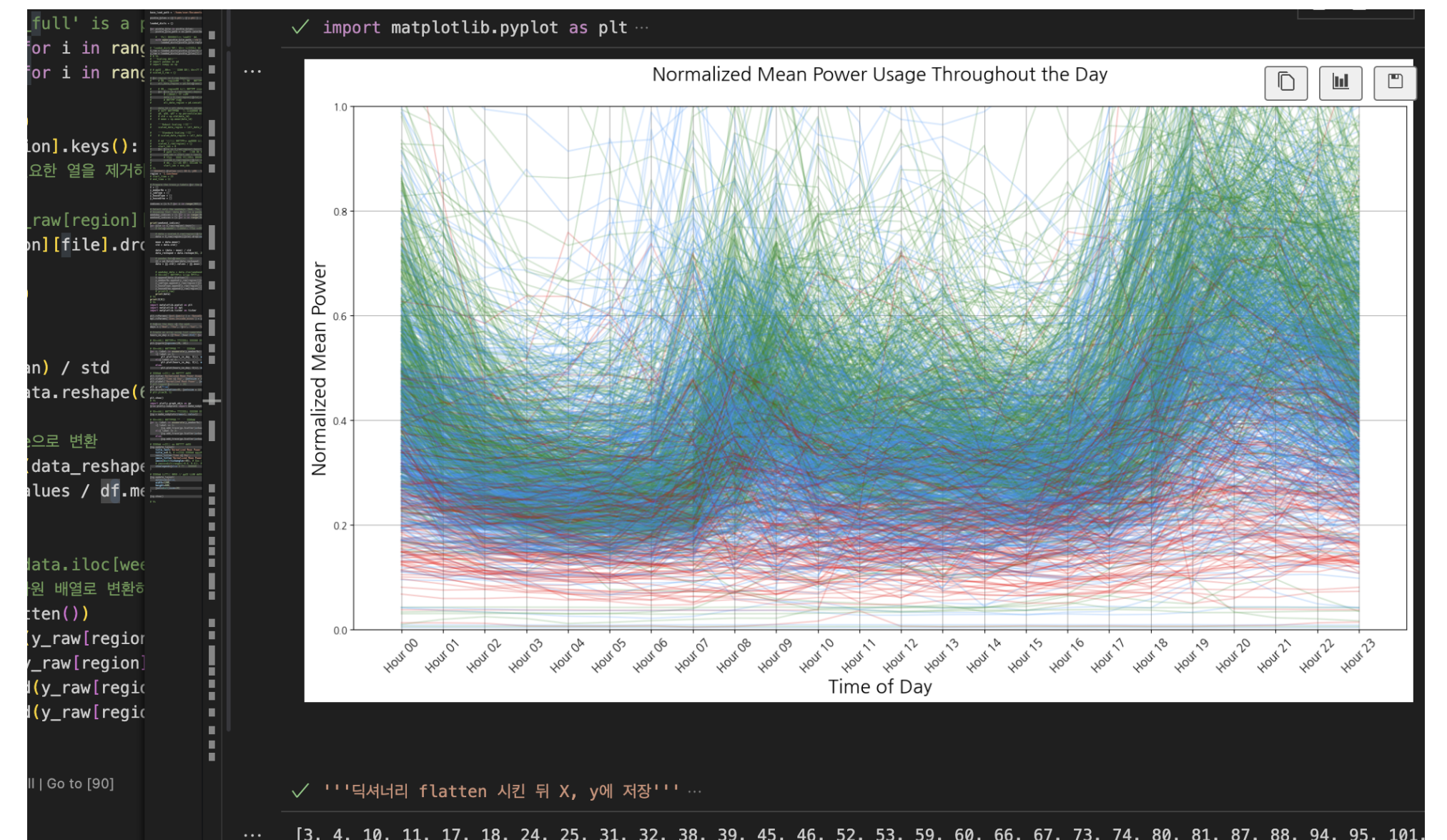


Experiments  
a lots of,,,,,,

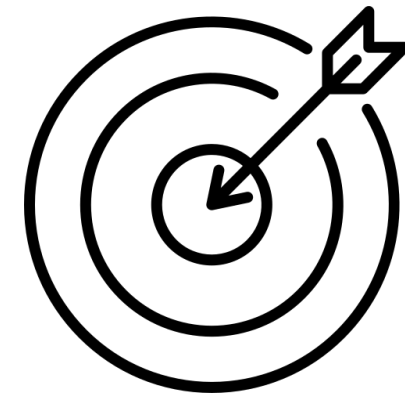


Colleague

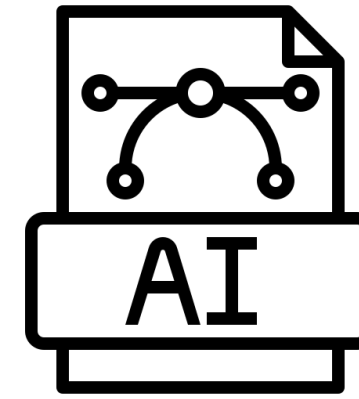
When **sharing or discussing an experiment** with a colleague, have you ever **captured an image of the results** and sent it to them?



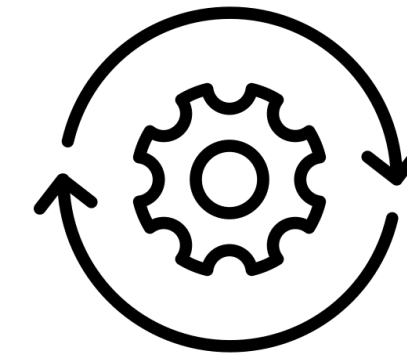
# Intro: Motivation



Goal



Model



Experiments

*a lots of,,,,,,*

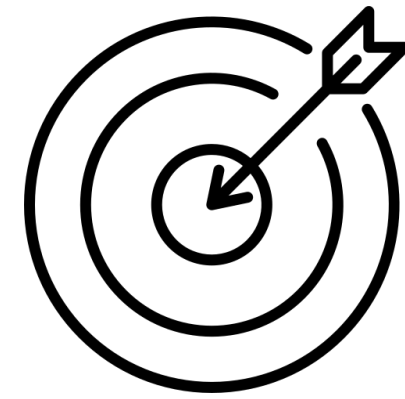
---

Adjust **hyperparameters**, verify models and data, monitor Gradient flow and GPU systems, and more...

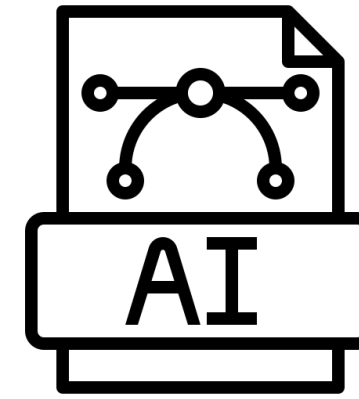
*there's a lot to consider.*



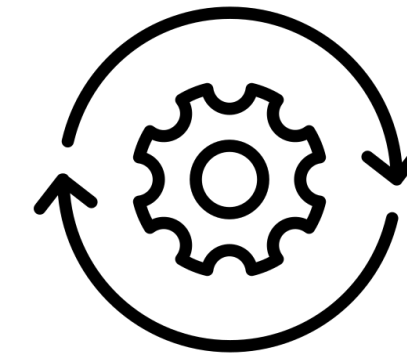
## Intro: Motivation



Goal



Model



Experiments

*a lots of,,,,,*



---

Adjust **hyperparameters**, verify models and data,  
monitor Gradient flow and GPU systems, and more...

**JUST WITH  
5 LINES?**

## Intro: Motivation

```
import wandb

# 1. Start a New run
wandb.init(project="gpt-3")

# 2. Save model inputs and hyperparameters
config = wandb.config
config.learning_rate = 0.01

# 3. Log gradients and model parameters
wandb.watch(model)
for batch_idx, (data, target) in enumerate(train_loader):
    ...
    if batch_idx % args.log_interval == 0:
        # 4. Log metrics to visualize performance
        wandb.log({"loss": loss})
```

## Intro: Motivation

Those who don't track training are doomed to repeat it.

From. Weights & Biases.

Ch1. What is Wandb?

 Weights & Biases

# What is **Wandb**?

# Weight & Bias?

A tool to **easily track and visualise the progress**  
of your deep learning experiments

**W (weights) & b (bias),**  
commonly used in deep learning, called wandb for short

Can be **integrated with various frameworks**  
such as pytorch, tensorflow, keras, huggingface, etc.



# Ch1. What is Wandb?

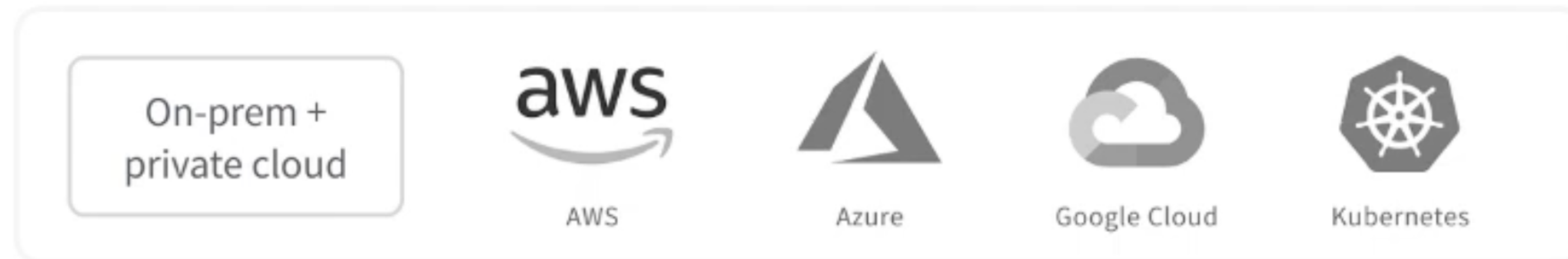
## MODULAR TOOLS



## FRAMEWORK AGNOSTIC



## ENVIRONMENT AGNOSTIC



# What Wandb can do?

- Save the **hyper-parameters** used during training
- **Explore, compare, and visualise each experiment**
- Analyse systems in the learning environment
- **Collaborate** with others
- **Replicate** the results of past experiments
- Allows **hyper-parameter tuning**
- **Permanently store a record** of all your experiments

# What Wandb can do?

- Save the **hyper-parameters** used during training
- **Explore, compare, and visualise each experiment**
- Analyse systems in the learning environment
- **Collaborate** with others
- **Replicate** the results of past experiments
- Allows **hyper-parameter tuning**
- **Permanently store a record** of all your experiments

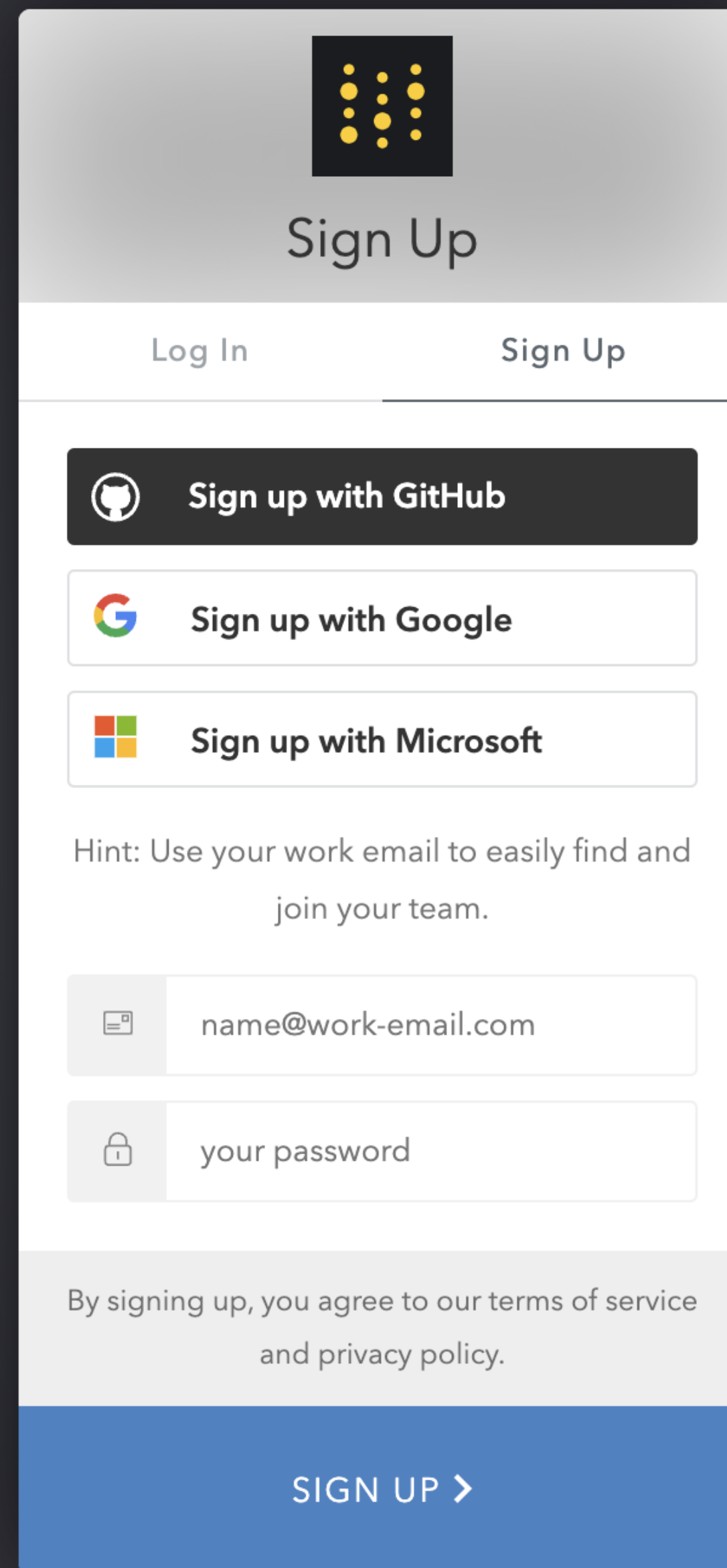
Ch2. tutorial for wandb

 Weights & Biases

# Wandb Tutorial

Ch2. tutorial for wandb

# Sign Up & Setting



The image shows a mobile-style sign-up interface for Wandb. At the top, there is a logo consisting of a 3x3 grid of yellow dots on a black square background, with the text "Sign Up" below it. Below the logo is a navigation bar with "Log In" and "Sign Up" options, where "Sign Up" is the active tab. The main content area features three social sign-up buttons: "Sign up with GitHub" (black button with white icon), "Sign up with Google" (white button with Google logo), and "Sign up with Microsoft" (white button with Microsoft logo). Below these is a hint: "Hint: Use your work email to easily find and join your team." There are two input fields: one for an email address containing "name@work-email.com" and one for a password containing "your password". At the bottom, there is a blue bar with the text "SIGN UP >".

<https://kr.wandb.ai/>

## Ch2. tutorial for wandb

Colab URL: <https://colab.research.google.com/drive/1m52CXom83J6IXn1GqrDq6auFNx1ILBY-?usp=sharing>

 **Click!**

### Quickstart: Tracking your first run in Weights & Biases

Weights & Biases' tools make it easy for you to quickly track experiments, visualize results, spot regressions, and more. Simply put, Weights & Biases enables you to build better models faster and easily share findings with colleagues.

#### Visualize your model training with

 Python /  Pytorch  or  Open in Colab



#### 1. Set up the wandb library

Install the CLI and Python library for interacting with the Weights and Biases API.

```
pip install wandb
```

Next, log in and paste your API key when prompted.

```
wandb login
```

 **Your API key** for logging in to the wandb library.  

## Ch2. tutorial for wandb

### ✓ Run an experiment

1. Start a new run and pass in hyperparameters to track
2. Log metrics from training or evaluation
3. Visualize results in the dashboard

✓  
26行

```
import random

# Launch 5 simulated experiments
total_runs = 5
for run in range(total_runs):
    # 🐛 1 Start a new run to track this script
    wandb.init(
        # Set the project where this run will be logged
        project="basic-intro",
        # We pass a run name (otherwise it'll be randomly assigned, like sunshine-lollypop-10)
        name=f"experiment_{run}",
        # Track hyperparameters and run metadata
        config={
            "learning_rate": 0.02,
            "architecture": "CNN",
            "dataset": "CIFAR-100",
            "epochs": 10,
        })

    # This simple block simulates a training loop logging metrics
    epochs = 10
    offset = random.random() / 5
    for epoch in range(2, epochs):
        acc = 1 - 2 ** -epoch - random.random() / epoch - offset
        loss = 2 ** -epoch + random.random() / epoch + offset

        # 🐛 2 Log metrics from your script to W&B
        wandb.log({"acc": acc, "loss": loss})

# Mark the run as finished
wandb.finish()
```

doesn't actually train the model.

Just a **sample code** that simply shows  
what **ACC and LOSS graph** look like at Wandb Page



## Ch2. tutorial for wandb

```
# 🐛 2 Log metrics from your script to W&B  
wandb.log({"acc": acc, "loss": loss})
```

```
# Mark the run as finished  
wandb.finish()
```

wandb: Currently logged in as: **oy6uns** (**oy6uns team**). Use `wandb login --relogin` to force relogin  
Tracking run with wandb version 0.16.5  
Run data is saved locally in /content/wandb/run-20240401\_080837-av2dpar6  
Syncing run **experiment\_0** to [Weights & Biases \(docs\)](#)  
View project at [https://wandb.ai/oy6uns\\_team/basic-intro](https://wandb.ai/oy6uns_team/basic-intro)  
View run at [https://wandb.ai/oy6uns\\_team/basic-intro/runs/av2dpar6/workspace](https://wandb.ai/oy6uns_team/basic-intro/runs/av2dpar6/workspace)

### Run history:

acc   
loss 

### Run summary:

acc 0.90902  
loss 0.11327

View run **experiment\_0** at: [https://wandb.ai/oy6uns\\_team/basic-intro/runs/av2dpar6/workspace](https://wandb.ai/oy6uns_team/basic-intro/runs/av2dpar6/workspace)  
Synced 4 W&B file(s), 0 media file(s), 0 artifact file(s) and 0 other file(s)  
Find logs at: `./wandb/run-20240401_080837-av2dpar6/logs`  
Tracking run with wandb version 0.16.5  
Run data is saved locally in /content/wandb/run-20240401\_080842-8lyllfgy  
Syncing run **experiment\_1** to [Weights & Biases \(docs\)](#)  
View project at [https://wandb.ai/oy6uns\\_team/basic-intro](https://wandb.ai/oy6uns_team/basic-intro)  
View run at [https://wandb.ai/oy6uns\\_team/basic-intro/runs/8lyllfgy/workspace](https://wandb.ai/oy6uns_team/basic-intro/runs/8lyllfgy/workspace)



# Ch2. tutorial for wandb

oy6uns\_team > Projects > basic-intro

Sungho Bae  
oy6uns\_team-org

Autosaved just now

Oy6uns's workspace Personal workspace

Overview

Workspace

Runs

Jobs

Automat.

Sweeps

Reports

Artifacts

Runs (5)

Search runs

Name (5 visualized)

- experiment\_4
- experiment\_3
- experiment\_2
- experiment\_1
- experiment\_0

1-5 of 5

Search panels with regex

Charts 2

### loss

Step	experiment_4	experiment_3	experiment_2	experiment_1	experiment_0
0	0.85	0.80	0.40	0.70	0.75
1	0.55	0.20	0.45	0.50	0.20
2	0.40	0.10	0.20	0.40	0.15
3	0.40	0.05	0.15	0.35	0.10
4	0.35	0.05	0.30	0.30	0.05
5	0.25	0.05	0.15	0.25	0.05
6	0.25	0.05	0.15	0.25	0.05
7	0.25	0.05	0.15	0.25	0.05

### acc

Step	experiment_4	experiment_3	experiment_2	experiment_1	experiment_0
0	0.45	0.30	0.40	0.35	0.30
1	0.70	0.45	0.55	0.60	0.40
2	0.80	0.70	0.60	0.75	0.50
3	0.85	0.75	0.80	0.80	0.70
4	0.85	0.80	0.70	0.65	0.85
5	0.90	0.75	0.75	0.85	0.80
6	0.85	0.75	0.75	0.80	0.80
7	0.90	0.85	0.75	0.80	0.80

System 11

Hidden Panels 0

Add section

Create report

Add panel

Ch3. tutorial for wandb + pytorch

 Weights & Biases

# Wandb + PyTorch Tutorial

# Ch3. tutorial for wandb + pytorch

In pseudocode, what we'll do is:

```
# import the library
import wandb

# start a new experiment
wandb.init(project="new-sota-model")

# capture a dictionary of hyperparameters with config
wandb.config = {"learning_rate": 0.001, "epochs": 100, "batch_size": 128}

# set up model and data
model, dataloader = get_model(), get_data()

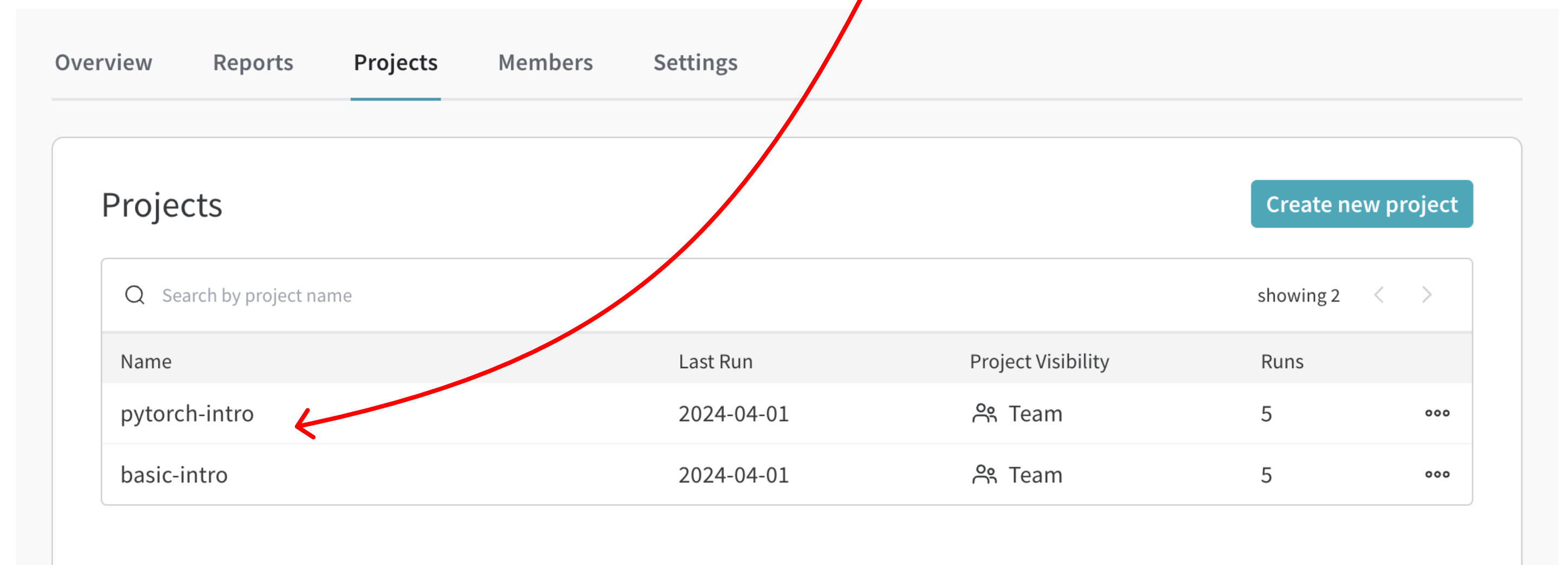
# optional: track gradients
wandb.watch(model)

for batch in dataloader:
    metrics = model.training_step()
    # log metrics inside your training loop to visualize model performance
    wandb.log(metrics)

# optional: save model at the end
model.to_onnx()
wandb.save("model.onnx")
```

```
# import the library
import wandb

# start a new experiment
wandb.init(project="new-sota-model")
```



The screenshot shows the 'Projects' tab in the wandb interface. At the top, there are navigation tabs: Overview, Reports, Projects (selected), Members, and Settings. Below the tabs is a 'Create new project' button. A search bar is present with the text 'Search by project name'. Below the search bar is a table with the following data:

Name	Last Run	Project Visibility	Runs
pytorch-intro	2024-04-01	👤 Team	5
basic-intro	2024-04-01	👤 Team	5

The project name you set will be **displayed on the wandb homepage**

## Ch3. tutorial for wandb + pytorch

In pseudocode, what we'll do is:

```
# import the library
import wandb

# start a new experiment
wandb.init(project="new-sota-model")

# capture a dictionary of hyperparameters with config
wandb.config = {"learning_rate": 0.001, "epochs": 100, "batch_size": 128}

# set up model and data
model, dataloader = get_model(), get_data()

# optional: track gradients
wandb.watch(model)

for batch in dataloader:
    metrics = model.training_step()
    # log metrics inside your training loop to visualize model performance
    wandb.log(metrics)

# optional: save model at the end
model.to_onnx()
wandb.save("model.onnx")
```

```
# capture a dictionary of hyperparameters with config
wandb.config = {"learning_rate": 0.001, "epochs": 100, "batch_size": 128}
```

Declare the **model's hyperparameters**  
in the python **dictionary type**

## Ch3. tutorial for wandb + pytorch

In pseudocode, what we'll do is:

```
# import the library
import wandb

# start a new experiment
wandb.init(project="new-sota-model")

# capture a dictionary of hyperparameters with config
wandb.config = {"learning_rate": 0.001, "epochs": 100, "batch_size": 128}

# set up model and data
model, dataloader = get_model(), get_data()

# optional: track gradients
wandb.watch(model)

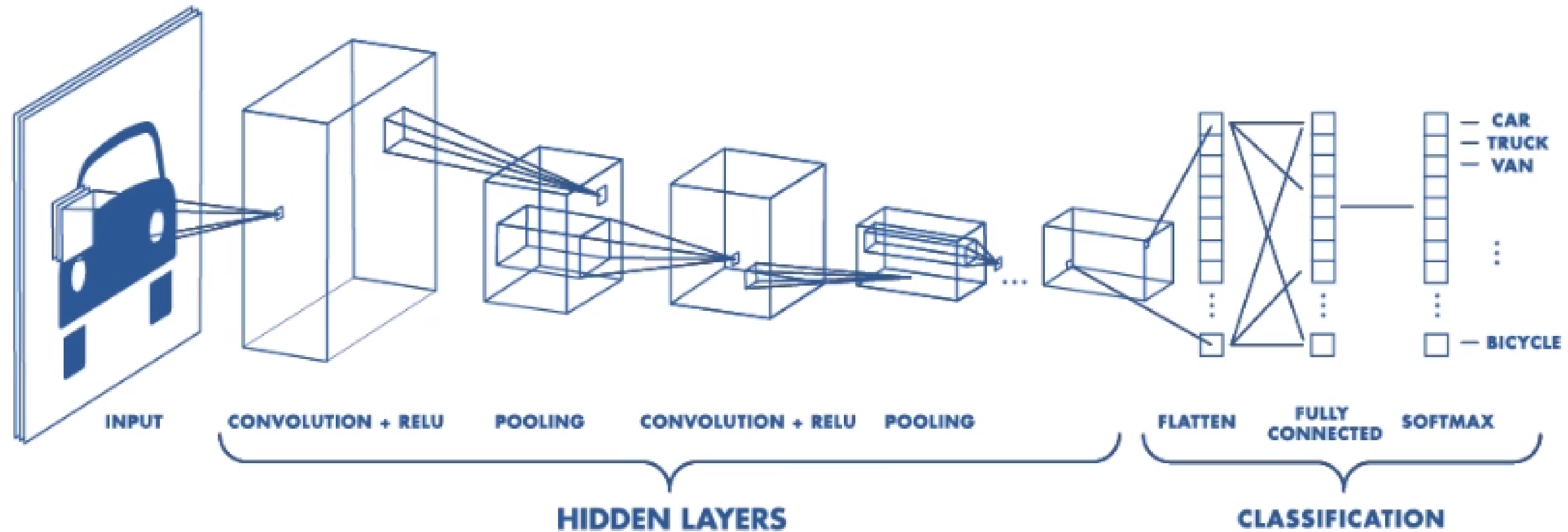
for batch in dataloader:
    metrics = model.training_step()
    # log metrics inside your training loop to visualize model performance
    wandb.log(metrics)

# optional: save model at the end
model.to_onnx()
wandb.save("model.onnx")
```

```
# set up model and data
model, dataloader = get_model(), get_data()
```

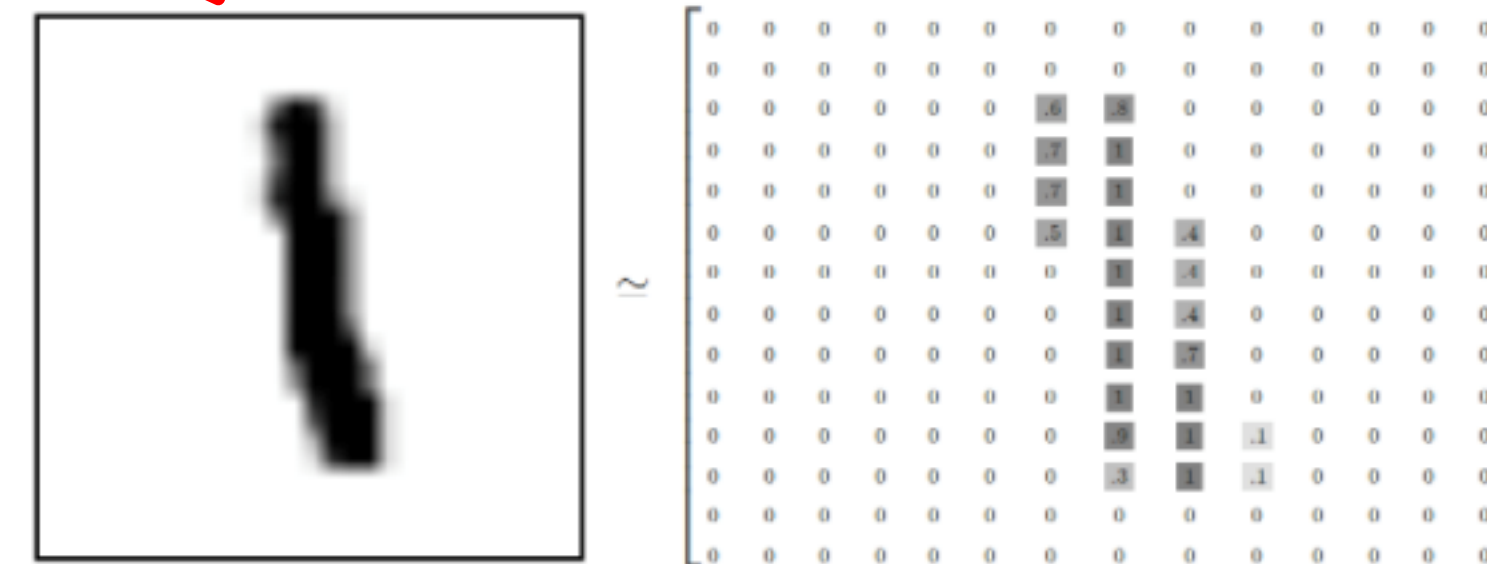
in this tutorial, we will use    model: **CNN**  
data: **MNIST**

# CNN(Convolutional Neural Network)



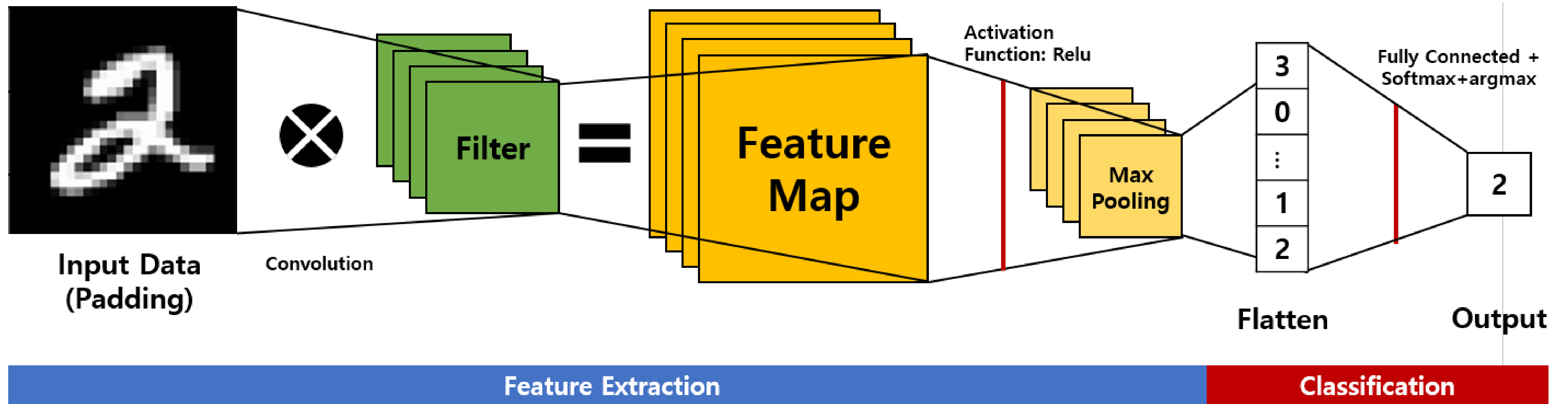
- Models that mimic human visual structure
- Useful for finding patterns to recognize images

# MNIST Dataset



consists of **black and white images** with values between 0 to 9, **28x28** in size, and **70,000 images** (60,000 in the training set and 10,000 in the test set)

# CNN with MNIST





# Ch3. tutorial for wandb + pytorch

In pseudocode, what we'll do is:

```
# import the library
import wandb

# start a new experiment
wandb.init(project="new-sota-model")

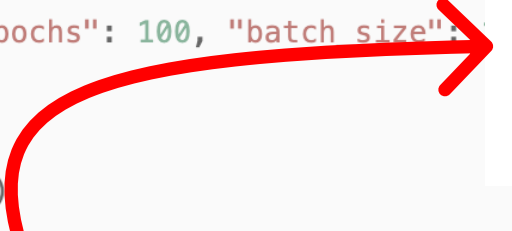
# capture a dictionary of hyperparameters with config
wandb.config = {"learning_rate": 0.001, "epochs": 100, "batch_size": 16}

# set up model and data
model, dataloader = get_model(), get_data()

# optional: track gradients
wandb.watch(model)

for batch in dataloader:
    metrics = model.training_step()
    # log metrics inside your training loop to visualize model performance
    wandb.log(metrics)

# optional: save model at the end
model.to_onnx()
wandb.save("model.onnx")
```



```
# optional: track gradients
wandb.watch(model)
```

Allows wandb to track the process of training the model.

## Ch3. tutorial for wandb + pytorch

In pseudocode, what we'll do is:

```
# import the library
import wandb

# start a new experiment
wandb.init(project="new-sota-model")


# capture a dictionary of hyperparameters with config
wandb.config = {"learning_rate": 0.001, "epochs": 100, "batch_size": 128}

# set up model and data
model, dataloader = get_model(), get_data_loader()

# optional: track gradients
wandb.watch(model)

for batch in dataloader:
    metrics = model.training_step()
    # log metrics inside your training loop to visualize model performance
    wandb.log(metrics)

# optional: save model at the end
model.to_onnx()
wandb.save("model.onnx")
```



```
for batch in dataloader:
    metrics = model.training_step()
    # log metrics inside your training loop to visualize model performance
    wandb.log(metrics)
```

**Train the model and log to wandb.**

# Ch3. tutorial for wandb + pytorch

In pseudocode, what we'll do is:

```
# import the library
import wandb

# start a new experiment
wandb.init(project="new-sota-model")

# capture a dictionary of hyperparameters with config
wandb.config = {"learning_rate": 0.001, "epochs": 100, "batch_size": 128}

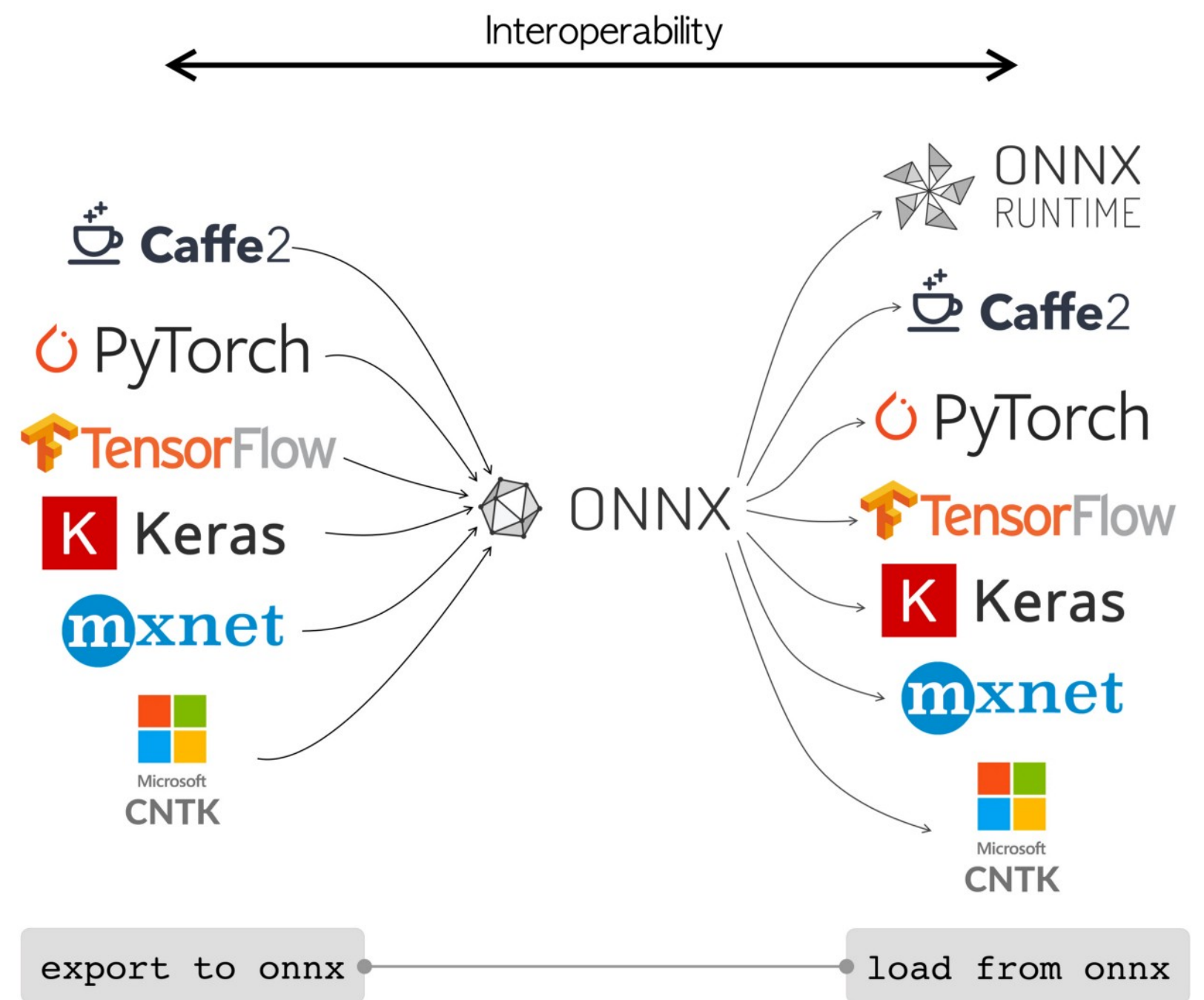
# set up model and data
model, dataloader = get_model(), get_data()

# optional: track gradients
wandb.watch(model)

for batch in dataloader:
    metrics = model.training_step()
    # log metrics inside your training loop to visualize model performance
    wandb.log(metrics)

# optional: save model at the end
model.to_onnx()
wandb.save("model.onnx")
```

```
# optional: save model at the end
model.to_onnx()
wandb.save("model.onnx")
```



**ONNX is short for Open Neural Network Exchange**

- It is a **sharing platform** designed to make models created in **different DNN framework environments** (ex Tensorflow, PyTorch, etc.) compatible with each other.

**Save the model**

## Ch3. tutorial for wandb + pytorch

Colab URL: <https://colab.research.google.com/drive/1pxMw2Z6cMNSgwCCaPrRTwleiE1l6zjt7?usp=sharing>

`model_pipeline(config)` — `wandb.init(config)`

👉 **Click!**

`config = wandb.config`

`make(config)` : define model, train\_loader, test\_loader, criterion, optimizer

`get_data()` : make CustomDataset

`make_loader()` : make DataLoader

`class ConvNet()` : define CNN model structure

`train()` : train the model

`wandb.watch`

`train_batch()` : calculate the loss for each epoch

`train_log()` : log to wandb by using `wandb.log`

`test()` : test model's final performance

`wandb.log`

`wandb.save`

# Ch3. tutorial for wandb + pytorch

oy6uns\_team > Projects > DE24sp-pytorch-demo > Workspace

Sungho Bae  
oy6uns\_team-org

Oy6uns's workspace Personal workspace

Autosaved 10 minutes ago

Runs (2)

Search runs

Name (2 visualized)

16, 32

8, 8

**Click!**

loss

test\_accuracy

epoch

System 11

Network Traffic (bytes)

Disk I/O Utilization (MB)

Disk Utilization (GB)

1-2 of 2

Check out the information  
of the best model



# Ch3. tutorial for wandb + pytorch

The screenshot shows the wandb interface for a project named 'DE24sp-pytorch-demo'. The breadcrumb navigation is 'oy6uns\_team > Projects > DE24sp-pytorch-demo > Runs > 16, 32 > Logs'. The user 'Sungho Bae' from 'oy6uns\_team-org' is logged in. On the left sidebar, the 'Logs' icon is highlighted with a red box and a yellow arrow pointing to it, with the text 'Click!' next to it. The main content area displays the following code and logs:

```
1 ConvNet(  
2   (layer1): Sequential(  
3     (0): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
4     (1): ReLU()  
5     (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
6   )  
7   (layer2): Sequential(  
8     (0): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
9     (1): ReLU()  
10    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
11  )  
12  (fc): Linear(in_features=1568, out_features=10, bias=True)  
13 )  
14 Loss after 01536 examples: 0.516  
15 Loss after 03136 examples: 0.347  
16 Loss after 04736 examples: 0.139  
17 Loss after 06336 examples: 0.149  
18 Loss after 07936 examples: 0.186  
19 Loss after 09536 examples: 0.132  
20 Loss after 11136 examples: 0.065  
21 Loss after 12704 examples: 0.040  
22 Loss after 14304 examples: 0.272  
23 Loss after 15904 examples: 0.155  
24 Loss after 17504 examples: 0.077  
25 Loss after 19104 examples: 0.049  
26 Loss after 20704 examples: 0.043  
27 Loss after 22304 examples: 0.016  
28 Loss after 23904 examples: 0.015  
29 Loss after 25472 examples: 0.068  
30 Loss after 27072 examples: 0.029  
31 Loss after 28672 examples: 0.009  
32 Loss after 30272 examples: 0.013  
33 Loss after 31872 examples: 0.010  
34 Loss after 33472 examples: 0.019
```

**we can understand the structure of the model.**

# Ch3. tutorial for wandb + pytorch

oy6uns\_team > Projects > DE24sp-pytorch-demo > Runs > 16, 32 > Files

Sungho Bae  
oy6uns\_team-org

> root

Search

artifact /	1 subfolder, 0 files		
config.yaml	17 minutes ago	882.0B	↓
<b>model.onnx</b>	17 minutes ago	117.3KB	↓
output.log	17 minutes ago	1.8KB	↓
requirements.txt	17 minutes ago	9.0KB	↓
wandb-metadata.json	17 minutes ago	1.2KB	↓
wandb-summary.json	17 minutes ago	22.6KB	↓

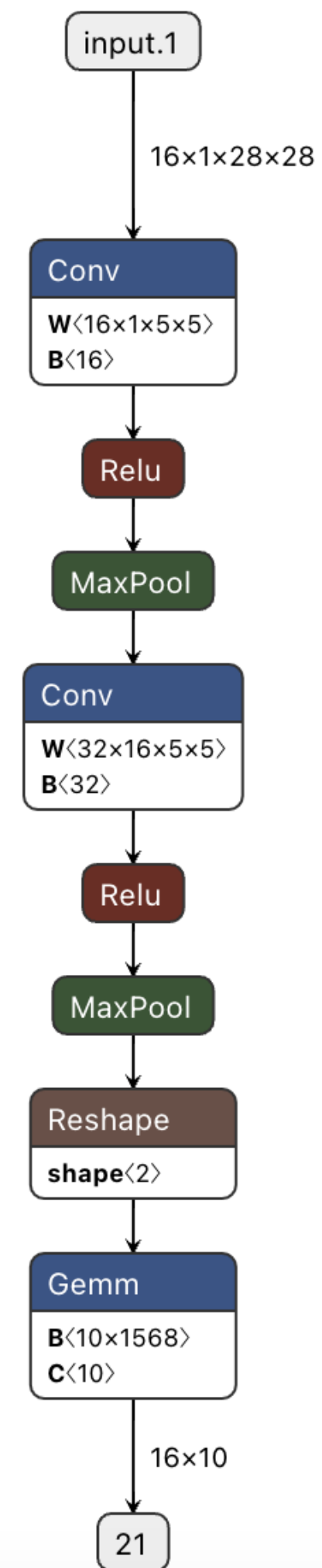
Click! 🖱️

1

2

# Ch3. tutorial for wandb + pytorch

- Workspace
- System
- Logs
- Files
- Artifacts



We can see the structure of the model as a picture.



Ch4. hyperparameters sweeps

 Weights & Biases

# Wandb Sweeps

way to find best parameters

## Ch4. hyperparameters sweeps

💡 simple task : minimize the **value  $3x+y$**

```
# Import the W&B Python Library and log into W&B
import wandb

wandb.login()

# 1: Define objective/training function
def objective(config):
    score = config.x**3 + config.y
    return score

def main():
    wandb.init(project="my-first-sweep")
    score = objective(wandb.config)
    wandb.log({"score": score})

# 2: Define the search space
sweep_configuration = {
    "method": "random",
    "metric": {"goal": "minimize", "name": "score"},
    "parameters": {
        "x": {"max": 0.1, "min": 0.01},
        "y": {"values": [1, 3, 7]},
    },
}

# 3: Start the sweep
sweep_id = wandb.sweep(sweep=sweep_configuration, project="my-first-sweep")

wandb.agent(sweep_id, function=main, count=10)
```

*# 1: Define objective/training function*

```
def objective(config):
    score = config.x**3 + config.y
    return score
```

```
def main():
    wandb.init(project="my-first-sweep")
    score = objective(wandb.config)
    wandb.log({"score": score})
```

## Ch4. hyperparameters sweeps

💡 simple task : minimize the **value  $3x+y$**

```
# Import the W&B Python Library and log into W&B
import wandb

wandb.login()

# 1: Define objective/training function
def objective(config):
    score = config.x**3 + config.y
    return score

def main():
    wandb.init(project="my-first-sweep")
    score = objective(wandb.config)
    wandb.log({"score": score})
```

```
# 2: Define the search space
sweep_configuration = {
    "method": "random",
    "metric": {"goal": "minimize", "name": "score"},
    "parameters": {
        "x": {"max": 0.1, "min": 0.01},
        "y": {"values": [1, 3, 7]},
    },
}
```

```
# 3: Start the sweep
sweep_id = wandb.sweep(sweep=sweep_configuration, project="my-first-sweep")

wandb.agent(sweep_id, function=main, count=10)
```

```
# 2: Define the search space
sweep_configuration = {
    "method": "random",
    "metric": {"goal": "minimize", "name": "score"},
    "parameters": {
        "x": {"max": 0.1, "min": 0.01},
        "y": {"values": [1, 3, 7]},
    },
}
```

randomly search

**x in  $0.01 < x < 0.1$**

**y in {1, 3, 7}**

ex) x = 0.312, y = 3

## Ch4. hyperparameters sweeps

```
# Import the W&B Python Library and log into W&B
import wandb

wandb.login()

# 1: Define objective/training function
def objective(config):
    score = config.x**3 + config.y
    return score

def main():
    wandb.init(project="my-first-sweep")
    score = objective(wandb.config)
    wandb.log({"score": score})

# 2: Define the search space
sweep_configuration = {
    "method": "random",
    "metric": {"goal": "minimize", "name": "score"},
    "parameters": {
        "x": {"max": 0.1, "min": 0.01},
        "y": {"values": [1, 3, 7]},
    },
}
```

```
# 3: Start the sweep
sweep_id = wandb.sweep(sweep=sweep_configuration, project="my-first-sweep")

wandb.agent(sweep_id, function=main, count=10)
```

*# 3: Start the sweep*

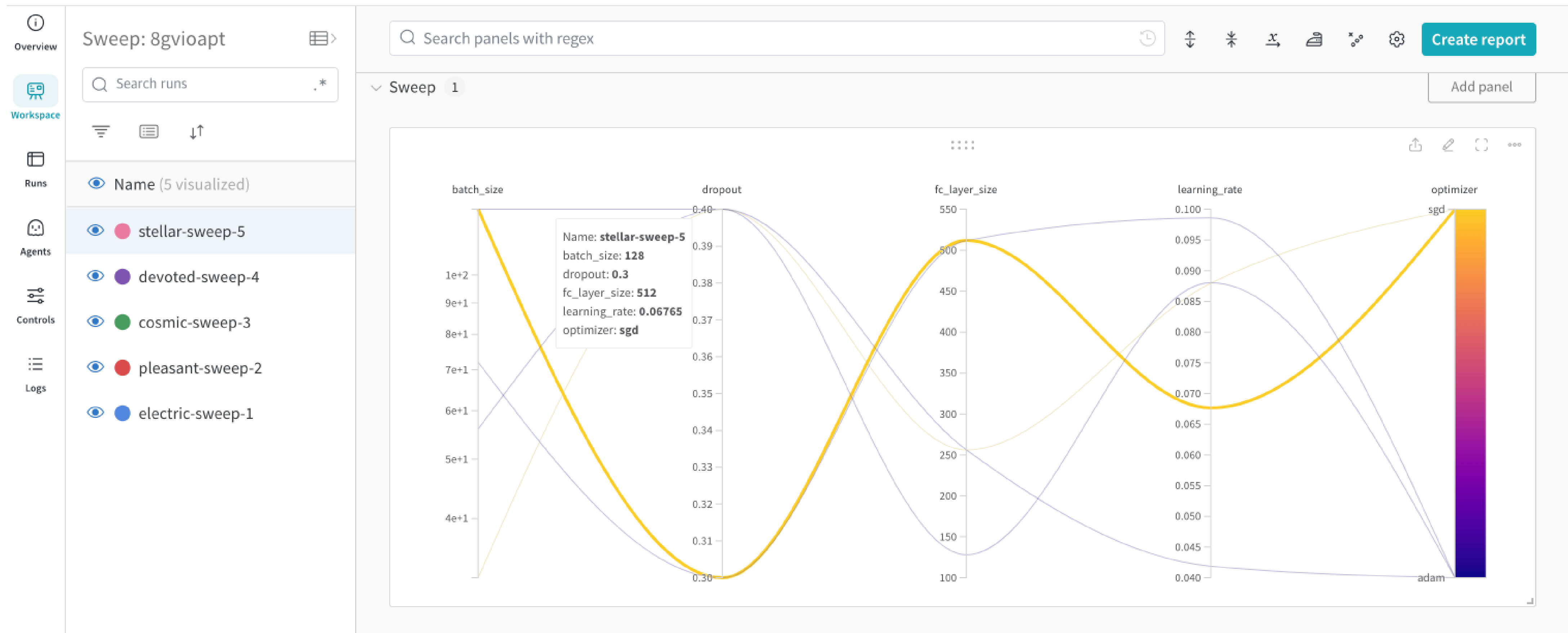
```
sweep_id = wandb.sweep(sweep=sweep_configuration, project="my-first-sweep")
```

```
wandb.agent(sweep_id, function=main, count=10)
```

Randomize the parameters and search **10 times**

# Ch4. hyperparameters sweeps

Colab URL: [https://colab.research.google.com/drive/1BaG2rD\\_0tAWm7hSzKgqhfH3tz9BvIDKr?usp=sharing](https://colab.research.google.com/drive/1BaG2rD_0tAWm7hSzKgqhfH3tz9BvIDKr?usp=sharing) 🖱️ Click!



 Weights & Biases

**Thank you**