```
In [1]: !wget https://raw.githubusercontent.com/Berkeley-CS182/cs182fa23_public/main/q_wand
        !pip install wandb
```

```
--2023-10-17 21:26:48--  https://raw.githubusercontent.com/Berkeley-CS182/cs182fa
23_public/main/q_wandbai/architectures.py (https://raw.githubusercontent.com/Berk
eley-CS182/cs182fa23_public/main/q_wandbai/architectures.py)
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.13
3, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.1
33|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1618 (1.6K) [text/plain]
Saving to: 'architectures.py'

     OK .                                                      100%  744K=0.002s

2023-10-17 21:26:48 (744 KB/s) - 'architectures.py' saved [1618/1618]
```

```
Collecting wandb
  Downloading wandb-0.15.12-py3-none-any.whl (2.1 MB)
     ---------------------------------------- 2.1/2.1 MB 14.8 MB/s eta 0:00:00
Collecting docker-pycreds>=0.4.0
  Downloading docker_pycreds-0.4.0-py2.py3-none-any.whl (9.0 kB)
Requirement already satisfied: appdirs>=1.4.3 in d:\anaconda\anaconda_setup\envs
\malning\lib\site-packages (from wandb) (1.4.4)
Requirement already satisfied: setuptools in d:\anaconda\anaconda_setup\envs\maln
ing\lib\site-packages (from wandb) (63.4.1)
Collecting pathtools
  Downloading pathtools-0.1.2.tar.gz (11 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting setproctitle
  Downloading setproctitle-1.3.3-cp37-cp37m-win_amd64.whl (11 kB)
Collecting GitPython!=3.1.29,>=1.0.0
  Downloading GitPython-3.1.38-py3-none-any.whl (190 kB)
     ---------------------------------------- 190.6/190.6 kB ? eta 0:00:00
Requirement already satisfied: PyYAML in d:\anaconda\anaconda_setup\envs\malning
\lib\site-packages (from wandb) (6.0)
Requirement already satisfied: typing-extensions in d:\anaconda\anaconda_setup\en
vs\malning\lib\site-packages (from wandb) (4.4.0)
Requirement already satisfied: protobuf!=4.21.0,<5,>=3.19.0 in d:\anaconda\anacon
da_setup\envs\malning\lib\site-packages (from wandb) (3.19.6)
Requirement already satisfied: requests<3,>=2.0.0 in d:\anaconda\anaconda_setup\e
nvs\malning\lib\site-packages (from wandb) (2.28.1)
Requirement already satisfied: Click!=8.0.0,>=7.1 in d:\anaconda\anaconda_setup\e
nvs\malning\lib\site-packages (from wandb) (8.1.3)
Requirement already satisfied: psutil>=5.0.0 in d:\anaconda\anaconda_setup\envs\m
alning\lib\site-packages (from wandb) (5.9.3)
Collecting sentry-sdk>=1.0.0
  Downloading sentry_sdk-1.32.0-py2.py3-none-any.whl (240 kB)
     ------------------------------------ 241.0/241.0 kB 15.4 MB/s eta 0:00:00
Requirement already satisfied: importlib-metadata in d:\anaconda\anaconda_setup\e
nvs\malning\lib\site-packages (from Click!=8.0.0,>=7.1->wandb) (5.0.0)
Requirement already satisfied: colorama in d:\anaconda\anaconda_setup\envs\malnin
g\lib\site-packages (from Click!=8.0.0,>=7.1->wandb) (0.4.6)
Requirement already satisfied: six>=1.4.0 in d:\anaconda\anaconda_setup\envs\maln
ing\lib\site-packages (from docker-pycreds>=0.4.0->wandb) (1.16.0)
Collecting gitdb<5,>=4.0.1
  Downloading gitdb-4.0.10-py3-none-any.whl (62 kB)
     ---------------------------------------- 62.7/62.7 kB ? eta 0:00:00
Requirement already satisfied: charset-normalizer<3,>=2 in d:\anaconda\anaconda_s
etup\envs\malning\lib\site-packages (from requests<3,>=2.0.0->wandb) (2.1.1)
Requirement already satisfied: idna<4,>=2.5 in d:\anaconda\anaconda_setup\envs\ma
lning\lib\site-packages (from requests<3,>=2.0.0->wandb) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in d:\anaconda\anaconda_setu
p\envs\malning\lib\site-packages (from requests<3,>=2.0.0->wandb) (1.26.12)
Requirement already satisfied: certifi>=2017.4.17 in d:\anaconda\anaconda_setup\e
nvs\malning\lib\site-packages (from requests<3,>=2.0.0->wandb) (2023.7.22)
Collecting smmap<6,>=3.0.1
  Downloading smmap-5.0.1-py3-none-any.whl (24 kB)
Requirement already satisfied: zipp>=0.5 in d:\anaconda\anaconda_setup\envs\malni
ng\lib\site-packages (from importlib-metadata->Click!=8.0.0,>=7.1->wandb) (3.10.
0)
Building wheels for collected packages: pathtools
  Building wheel for pathtools (setup.py): started
  Building wheel for pathtools (setup.py): finished with status 'done'
  Created wheel for pathtools: filename=pathtools-0.1.2-py3-none-any.whl size=879
2 sha256=3a43b3d11799db09ec92120b06a6c8f6deccae8ba0055f23c6f991ae93b4b1cf
  Stored in directory: c:\users\cyt\appdata\local\pip\cache\wheels\3e\31\09\fa59c
```

```
ef12cdcfecc627b3d24273699f390e71828921b2cbba2
Successfully built pathtools
Installing collected packages: pathtools, smmap, setproctitle, sentry-sdk, docker
-pycreds, gitdb, GitPython, wandb
Successfully installed GitPython-3.1.38 docker-pycreds-0.4.0 gitdb-4.0.10 pathtoo
ls-0.1.2 sentry-sdk-1.32.0 setproctitle-1.3.3 smmap-5.0.1 wandb-0.15.12
```

In [1]:
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import wandb
from architectures import BasicConvNet, ResNet18, MLP
from torch.utils.tensorboard import SummaryWriter
from tqdm import tqdm
from torch.utils.data import DataLoader
```

executed in 1.57s, finished 13:53:56 2023-10-18

# Exploring Tensorboard

Tensorboard is a local tool for visualizing images, metrics, histograms, and more. It is designed for tensorflow, but can be integrated with torch. Let's explore tensorboard usage with an example:

```
from torch.utils.tensorboard import SummaryWriter

# To start a run, call the following
writer = SummaryWriter(comment=f'Name_of_Run')

# When you want to log a value, use the writer. When adding a scalar, the f
ormat is as follows:
# add_scalar(tag, scalar_value, global_step=None, walltime=None, new_style=
False, double_precision=False)
writer.add_scalar('Training Loss', loss.item(), step)

# Finally, when you are done logging values, close the writer
writer.close()
```

There are many other functionalities and methods that you are free to explore, but will not be mentioned in this notebook.

## Your Task

We will be once again building classifiers for the CIFAR-10. There are various architectures set up for you to use in the architectures.py file. Using tensorboard, please search through 5 different hyperparameter configurations. Examples of choices include: learning rate, batch size, architecture, optimization algorithm, etc. Please submit the generated plots on your pdf and answer question A.

In [2]:
```
epochs = 2
cuda = torch.cuda.is_available()
device = torch.device("cuda" if cuda else "cpu")
```
executed in 1.35s, finished 13:53:59 2023-10-18

In [3]:
```
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

trainset = torchvision.datasets.CIFAR10(root='./../cifar-10/', train=True,
                                        download=True, transform=transform)
testset = torchvision.datasets.CIFAR10(root='./../cifar-10/', train=False,
                                       download=True, transform=transform)
```
executed in 2.60s, finished 13:54:02 2023-10-18

```
Files already downloaded and verified
Files already downloaded and verified
```

In [4]:
```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
device
```
executed in 16ms, finished 13:54:02 2023-10-18

Out[4]:
```
device(type='cuda')
```

In [5]:
```
def get_optimizer(params, optim_type, lr):
    if optim_type == "sgd":
        optimizer = optim.SGD(params, lr=lr)
    elif optim_type == "adam":
        optimizer = optim.Adam(params, lr=lr)
    else:
        raise ValueError(optim_type)

    return optimizer

def get_model(model_type):
    if model_type == "basicconvnet":
        model = BasicConvNet()
    elif model_type == "resnet18":
        model = ResNet18()
    elif model_type == "mlp":
        model = MLP()
    else:
        raise ValueError(model_type)

    return model

def get_criterion(loss_type):
    if(loss_type == "mse"):
        criterion = nn.MSELoss()
    elif(loss_type == "cross"):
        criterion = nn.CrossEntropyLoss()
    else:
        raise ValueError(loss_type)

    return criterion
```
executed in 13ms, finished 13:54:03 2023-10-18

In [6]:
```python
def train(writer, dataloader, model, loss_fn, optimizer, epoch):
    size = len(dataloader.dataset)
    num_batch = len(dataloader)
    model.train()

    total_loss = 0
    correct = 0

    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)

        pred = model(X)
        loss = loss_fn(pred, y)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        total_loss += loss.item()
        correct += (pred.argmax(1) == y).type(torch.float).sum().item()

        if(batch % 100 == 0):
            loss, current = loss.item(), batch * len(X)
            print(f"loss: {loss:>7f} [{current:>5d} / {size:>5d}]")

    avg_loss = total_loss / num_batch
    correct /= size

    # write into tensorboard
    writer.add_scalar("Train Loss", avg_loss, epoch)
    writer.add_scalar("Train Acc", correct, epoch)

    print(f"Train Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss: {avg_loss:>8


def test(writer, dataloader, model, loss_fn, epoch):

    size = len(dataloader.dataset)
    num_batches = len(dataloader)
    model.eval()

    test_loss = 0
    correct = 0.1
    with torch.no_grad():
        for batch, (X, y) in enumerate(dataloader):
            X, y = X.cuda(), y.cuda()
            pred = model(X)
            test_loss += loss_fn(pred, y).item()
            correct += (pred.argmax(1) == y).type(torch.float).sum().item()

    test_loss /= num_batches
    correct /= size

    # write into tensorboard
    writer.add_scalar('Test Loss', test_loss, epoch)
    writer.add_scalar('Test Acc', correct, epoch)

    print(f"Evaluation Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss: {test_l
```

executed in 20ms, finished 13:54:04 2023-10-18

In [10]:
```
def run_training(trainset, testset, hyperparameters, log_dir = "logs"):

    print("---------------------------config------------------------------")
    print(hyperparameters)
    print("---------------------------------------------------------------")

    name = ""
    for i, key in enumerate(hyperparameters.keys()):
        value = hyperparameters[key]
        if i != (len(hyperparameters.keys()) - 1):
            item = key + "_" + str(value) + "_"
        else:
            item = key + "_" + str(value)
        name = name + item

    model_type = hyperparameters['model']
    model = get_model(model_type)
    loss_type = hyperparameters['loss_fn']
    criterion = get_criterion(loss_type)
    learning_rate = hyperparameters['lr']
    optim_type = hyperparameters['optimizer']
    optimizer = get_optimizer(model.parameters(), optim_type, lr=learning_rate)
    batch_size = hyperparameters['batch_size']
    num_epochs = hyperparameters['epochs']

    # build train data loader
    trainloader = DataLoader(trainset, batch_size=batch_size, shuffle=True)
    # build test data loader
    testloader = DataLoader(testset, batch_size=batch_size, shuffle=False)

    # create a tensorboard writer
    path = log_dir + "/" + name
    writer = SummaryWriter(path)
    print(f"log will be written to {path}")

    model.cuda()

    for t in range(num_epochs):
        print(f"Epoch {t+1}\n-------------------------------")
        train(writer, trainloader, model, criterion, optimizer, t+1)
        test(writer, testloader, model, criterion, t+1)

    writer.close()
```

executed in 15ms, finished 14:05:20 2023-10-18

In [11]:
```
hyperparameters1 = {
    "model" : "basicconvnet",
    "lr" : 0.0001,
    "loss_fn" : "cross",
    "optimizer" : "adam",
    "epochs" : 20,
    "batch_size" : 16
}

hyperparameters2 = {
    "model" : "resnet18",
    "lr" : 0.0001,
    "loss_fn" : "cross",
    "optimizer" : "adam",
    "epochs" : 20,
    "batch_size" : 16
}

hyperparameters3 = {
    "model" : "mlp",
    "lr" : 0.0001,
    "loss_fn" : "cross",
    "optimizer" : "adam",
    "epochs" : 20,
    "batch_size" : 16
}
```
executed in 8ms, finished 14:05:23 2023-10-18

In [12]:
```
def run():
    # Perhaps you want to make a function to train on a certain set of hyperparamete
    # Don't forget to use tensorboard
    run_training(trainset, testset, hyperparameters1)
    run_training(trainset, testset, hyperparameters2)
    run_training(trainset, testset, hyperparameters3)
```
executed in 6ms, finished 14:05:45 2023-10-18

In [ ]: