```
from pathlib import Path
import sys

if 'google.colab' in str(get_ipython()):
    from google.colab import drive  # Import Google Drive mounting
utility
    drive.mount('/content/drive')  # Mount Google Drive

    # REPLACE WITH YOUR FOLDER

    base_folder =
Path('/content/drive/MyDrive/Colab_Notebooks/BUAN_6382_Applied_DeepLea
rning')

    data_folder =  Path('/content')

    !pip install pytorch-lightning==2.0.9 -qq
    !pip install torchmetrics -U -qq
    !pip install fastdownload -U -qq
    !pip install fastai -U -qq
    !pip install wandb -U -qq
    !pip install torchinfo

else:
    # Set base folder path for storing files on local machine
    # REPLACE WITH YOUR FOLDER
    # FILL THIS ONLY IF YOU ARE RUNNING ON A LOCAL MACHINE
    print('Path is
/Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/Data')

    base_folder =
Path('/Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning')
    data_folder =
Path('/Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/Custom_files')
    !pip install pytorch-lightning==2.0.9 -qq
    !pip install torchmetrics -U -qq
    !pip install fastdownload -U -qq
    !pip install fastai -U -qq
    !pip install wandb -U -qq
```

```
Mounted at /content/drive
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 727.7/727.7 kB 8.2 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 805.2/805.2 kB 14.2 MB/s eta
```

```
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 14.1 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 190.6/190.6 kB 15.4 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 243.2/243.2 kB 17.9 MB/s eta
0:00:00
etadata (setup.py) ... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
62.7/62.7 kB 10.5 MB/s eta 0:00:00

from pathlib import Path
import sys

# Determine the storage location based on the execution environment
# If running on Google Colab, use Google Drive as storage
if 'google.colab' in str(get_ipython()):
    custom_function_folder =
Path('/content/drive/MyDrive/Colab_Notebooks/BUAN_6382_Applied_DeepLea
rning/Custom_files') # Your Google Drive

    sys.path.append(str(custom_function_folder))
    model_folder =
Path('/content/drive/MyDrive/Colab_Notebooks/BUAN_6382_Applied_DeepLea
rning/Data') # Google drive folder where you want to save model and
logs
    model_folder.mkdir(parents=True, exist_ok=True)
    project_folder = model_folder
    # project_folder =
Path('/content/drive/MyDrive/Colab_Notebooks/BUAN_6382_Applied_DeepLea
rning/Class/Class - 6/Imagenette_project')
    kaggle_api_folder = base_folder/'data/.kaggle'

# If running locally, specify a different path
else:
    # Set base folder path for storing files on local machine
    # REPLACE WITH YOUR FOLDER
    # FILL THIS ONLY IF YOU ARE RUNNING ON A LOCAL MACHINE
    print('Path is
/Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/Custom_files')
    custom_function_folder =
Path('/Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/Custom_files') #
Your Google Drive

    sys.path.append(str(custom_function_folder))
    model_folder =
Path('/Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
```

```
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/Data') # Google
drive folder where you want to save model and logs
    model_folder.mkdir(parents=True, exist_ok=True)
    # project_folder =
Path('/Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/Class/Class -
6/Imagenette_project')
    kaggle_api_folder = base_folder/'data/.kaggle'
    # project_folder =
Path('/Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/Data')

# import Libraries
import yaml

import torch
import torch.nn as nn
import torchmetrics
from torchvision import transforms
import pytorch_lightning as pl
from pytorch_lightning import seed_everything
from pytorch_lightning.tuner import Tuner
from pytorch_lightning.callbacks import ModelCheckpoint,
EarlyStopping, LearningRateMonitor
from pytorch_lightning.loggers import CSVLogger, WandbLogger
import wandb
import gc

from data_module_fmnist import FashionMNISTDataModule
from multiclass_lightning_module_v0 import MultiClassLightningModule
from resnet import SimpleResNet
from shared_utils import  plot_losses_acc

SimpleResNet??

def count_parameters(model):
    total_params = sum(p.numel() for p in model.parameters())
    trainable_params = sum(p.numel() for p in model.parameters() if
p.requires_grad)
    return total_params, trainable_params

model = SimpleResNet(num_classes=10)
total_params, trainable_params = count_parameters(model)
print(f"Total parameters: {total_params}")
print(f"Trainable parameters: {trainable_params}")
```

```
Total parameters: 831914
Trainable parameters: 831914
```

```
trans1 = transforms.ToTensor()

# Transform 2: Normalize the tensor images.
# The specified mean and standard deviation values are dataset-
specific.
trans2 = transforms.Normalize((0.2857,), (0.3528))

# Combine the above transformations into a single composite transform.
trans = transforms.Compose([trans1, trans2])

def load_datamodule(config,data_folder):
    # Fetch the correct transform function based on config and pass
the appropriate arguments
    dm = FashionMNISTDataModule(
        data_dir=data_folder,
        train_transform=trans,
        test_transform=trans,
        **config['data_module']
    )
    return dm

# Function to load the model
def load_model(model_config):
    model = SimpleResNet(num_classes=10)
    return model

def load_lightning_module(config, model):
    optimizer_cls = eval(config['optimizer_cls'])
    loss_fn = eval(config['loss_fn'])()  # directly instantiate the
loss function
    metric_cls = eval(config['metric_cls'])

    # If scheduler is defined, convert its string to class as well
    if config.get('scheduler_cls'):
        scheduler_cls = eval(config['scheduler_cls'])
        scheduler_options = config['scheduler_options']
        scheduler_params =   config['scheduler_params']
    else:
        scheduler_cls = None

    lightning_module = MultiClassLightningModule(model=model,

optimizer_cls=optimizer_cls,
                                                 loss_fn=loss_fn,

metric_cls=metric_cls,

scheduler_cls=scheduler_cls,
```

```
scheduler_options=scheduler_options,

scheduler_params=scheduler_params,
                                                **config['others']
)
    return lightning_module

def load_trainer(model, trainer_config, cl_config, batch_size,
model_folder,  logging=False, checkpointing=True,
early_stopping=False):

    lr_monitor = LearningRateMonitor(**cl_config['lr_monitor'])
    callbacks = [lr_monitor]
    if checkpointing:
        model_checkpoint_callback =
ModelCheckpoint(dirpath=model_folder/cl_config['log_dir'],

**cl_config['model_checkpoint'])
        callbacks.append(model_checkpoint_callback)

    if early_stopping:
        early_stop_callback =
EarlyStopping(**cl_config['early_stopping'] )
        callbacks.append(early_stop_callback)

    if logging:
        # For WandB logger:
        wandb_logger = WandbLogger(project=cl_config['wandb']
['project'], name=cl_config['wandb']['name'],
save_dir=model_folder/cl_config['log_dir'])
        wandb_logger.experiment.config.update({'batch_size':
batch_size, 'epochs': trainer_config['max_epochs']})
        wandb_logger.watch(model)

        # For CSV logger:
        csv_logger =
CSVLogger(save_dir=model_folder/cl_config['log_dir'],
name=cl_config['csv']['name'])
        csv_logger.log_hyperparams(params={'batch_size': batch_size,
'epochs': trainer_config['max_epochs']})

        trainer = pl.Trainer(callbacks=callbacks,
                             logger=[csv_logger, wandb_logger],
                             **trainer_config)
    else:
        trainer = pl.Trainer(callbacks=callbacks,
                             **trainer_config
                )
    return trainer
```

```
def load_components(model_config, data_module_config,
lightning_module_config, data_folder, trainer_config,
cl_config, batch_size,logging=False, checkpointing=True,
early_stopping=False):

    # Load the model
    model = load_model(model_config)

    # Load the data module
    dm = load_datamodule(data_module_config, data_folder)

    # Load the lightning module
    lightning_module = load_lightning_module(lightning_module_config,
model)

    # Load the trainer
    trainer = load_trainer(model, trainer_config, cl_config,
batch_size, model_folder,  logging=logging,
                           checkpointing=checkpointing,
early_stopping=early_stopping)

    return model, dm, lightning_module, trainer

def load_yaml(filepath):
    with open(filepath, 'r') as file:
        return yaml.safe_load(file)

project_folder = custom_function_folder

# Load configurations from YAML files
def load_all_configs():
    model_config =
load_yaml(project_folder/'model_config_fminst.yaml')
    data_module_config =
load_yaml(project_folder/'data_module_config_fminst.yaml')
    lightning_module_config =
load_yaml(project_folder/'lightning_module_config.yaml')
    cl_config =
load_yaml(project_folder/'callbacks_loggers_config_fminst.yaml')
    trainer_config = load_yaml(project_folder/'trainer_config.yaml')
    return model_config, data_module_config, lightning_module_config,
cl_config, trainer_config

def free_memory():
    """
    Attempts to free up memory by deleting variables and running
Python's garbage collector.
    """
    gc.collect()
    for device_id in range(torch.cuda.device_count()):
        torch.cuda.set_device(device_id)
```

```
        torch.cuda.empty_cache()
    gc.collect()

model_config, data_module_config, lightning_module_config, cl_config,
trainer_config = load_all_configs()
trainer_config

{'max_epochs': 2,
 'accelerator': 'auto',
 'devices': 'auto',
 'deterministic': False,
 'log_every_n_steps': 1,
 'gradient_clip_algorithm': 'norm',
 'gradient_clip_val': 0,
 'fast_dev_run': False,
 'overfit_batches': 0.0,
 'accumulate_grad_batches': 1,
 'limit_train_batches': 1.0,
 'limit_val_batches': 1.0,
 'limit_test_batches': 1.0}

cl_config

{'log_dir': 'logs',
 'lr_monitor': {'logging_interval': 'step'},
 'model_checkpoint': {'monitor': 'val_metric',
  'mode': 'max',
  'save_top_k': 1,
  'save_last': True},
 'early_stopping': {'monitor': 'val_metric',
  'patience': 5,
  'mode': 'max',
  'verbose': True},
 'wandb': {'project': 'FMINST', 'name': 'resnet'},
 'csv': {'name': 'csvlogger'}}

model_config

{'num_classes': 10}

lightning_module_config

{'optimizer_cls': 'torch.optim.AdamW',
 'loss_fn': 'torch.nn.CrossEntropyLoss',
 'metric_cls': 'torchmetrics.Accuracy',
 'scheduler_cls': 'None',
 'scheduler_options': 'None',
 'scheduler_params': 'None',
 'others': {'optimizer_params': {'weight_decay': 0},
  'num_classes': 10,
  'learning_rate': 0.0001,
```

```
  'log_every_n_steps': 1,
  'log_test_metrics': True,
  'display_metrics': True}}

data_module_config

{'data_module': {'batch_size': 64, 'seed': 42}}

data_module_config['data_module']['batch_size']

64

# Load components
free_memory()
seed_everything(42)
model_config, data_module_config, lightning_module_config, cl_config,
trainer_config = load_all_configs()
# override default values
trainer_config['fast_dev_run']=True
model, dm, lightning_module, trainer = load_components(model_config,
data_module_config,

lightning_module_config, data_folder, trainer_config,
                                                       cl_config,
batch_size=data_module_config['data_module']['batch_size'],
                                                       logging=False,
checkpointing=False, early_stopping=False)
dm.prepare_data()
trainer.fit(lightning_module, dm)

INFO:lightning_fabric.utilities.seed:Global seed set to 42
INFO:pytorch_lightning.utilities.rank_zero:GPU available: True (cuda),
used: True
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False,
using: 0 TPU cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False,
using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False,
using: 0 HPUs
INFO:pytorch_lightning.utilities.rank_zero:Running in `fast_dev_run`
mode: will run the requested loop using 1 batch(es). Logging and
checkpointing is suppressed.

Downloading http://fashion-mnist.s3-website.eu-central-
1.amazonaws.com/train-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-
1.amazonaws.com/train-images-idx3-ubyte.gz to
/content/FashionMNIST/raw/train-images-idx3-ubyte.gz

100%|██████████| 26421880/26421880 [00:02<00:00, 12654198.63it/s]
```

```
Extracting /content/FashionMNIST/raw/train-images-idx3-ubyte.gz to
/content/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-
1.amazonaws.com/train-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-
1.amazonaws.com/train-labels-idx1-ubyte.gz to
/content/FashionMNIST/raw/train-labels-idx1-ubyte.gz

100%|██████████| 29515/29515 [00:00<00:00, 197429.30it/s]

Extracting /content/FashionMNIST/raw/train-labels-idx1-ubyte.gz to
/content/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-
1.amazonaws.com/t10k-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-
1.amazonaws.com/t10k-images-idx3-ubyte.gz to
/content/FashionMNIST/raw/t10k-images-idx3-ubyte.gz

100%|██████████| 4422102/4422102 [00:01<00:00, 3776077.61it/s]

Extracting /content/FashionMNIST/raw/t10k-images-idx3-ubyte.gz to
/content/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-
1.amazonaws.com/t10k-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-
1.amazonaws.com/t10k-labels-idx1-ubyte.gz to
/content/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz

100%|██████████| 5148/5148 [00:00<00:00, 19955893.71it/s]

Extracting /content/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz to
/content/FashionMNIST/raw


INFO:pytorch_lightning.accelerators.cuda:LOCAL_RANK: 0 -
CUDA_VISIBLE_DEVICES: [0]
INFO:pytorch_lightning.callbacks.model_summary:
  | Name         | Type               | Params
-------------------------------------------------
0 | model        | SimpleResNet       | 831 K
1 | loss_fn      | CrossEntropyLoss   | 0
2 | train_metric | MulticlassAccuracy | 0
3 | val_metric   | MulticlassAccuracy | 0
4 | test_metric  | MulticlassAccuracy | 0
-------------------------------------------------
831 K     Trainable params
0         Non-trainable params
```

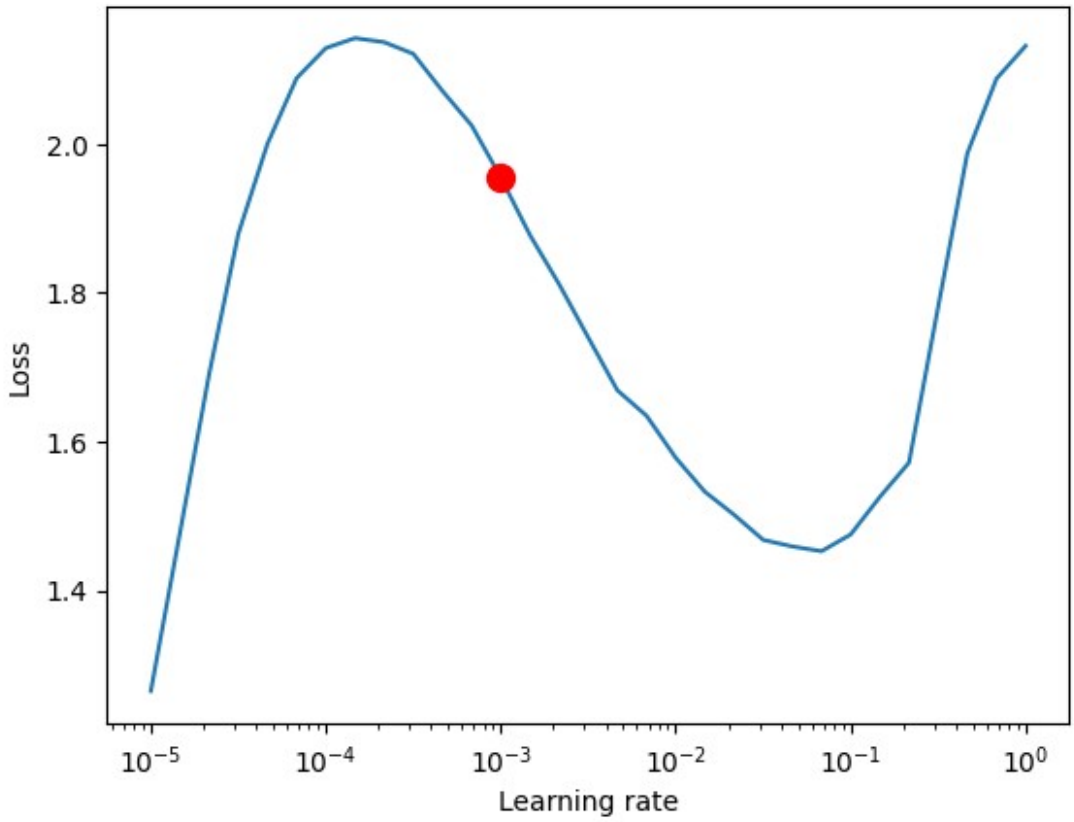
```
831 K     Total params
3.328     Total estimated model params size (MB)
```


```
{"model_id":"73592027a6434bb1aaaf5fd1870e717d","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"639f02b8bdc44b68a9dbab15cd311b24","version_major":2,"vers
ion_minor":0}
```


```
Epoch 1: Val_Loss: 2.29, Val_Metric: 0.14 |

INFO:pytorch_lightning.utilities.rank_zero:`Trainer.fit` stopped:
`max_steps=1` reached.

Train_Loss: 2.60, Train_Metric: 0.08

# Load components
free_memory()
seed_everything(42)
model_config, data_module_config, lightning_module_config, cl_config,
trainer_config = load_all_configs()
# override default values
trainer_config['max_epochs']=5
data_module_config['data_module']['batch_size']=64

model, dm, lightning_module, trainer = load_components(model_config,
data_module_config,

lightning_module_config, data_folder, trainer_config,

cl_config,batch_size=data_module_config['data_module']['batch_size'],
                                                    logging=False,
checkpointing=False, early_stopping=False)
dm.setup('fit')
tuner = Tuner(trainer)
lr_finder = tuner.lr_find(lightning_module, datamodule=dm, min_lr=1e-
5, max_lr=1, num_training=30, mode='exponential')
fig = lr_finder.plot(suggest=True)
new_lr = lr_finder.suggestion()
print(new_lr)

INFO:lightning_fabric.utilities.seed:Global seed set to 42
INFO:pytorch_lightning.utilities.rank_zero:GPU available: True (cuda),
used: True
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False,
using: 0 TPU cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False,
using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False,
using: 0 HPUs
INFO:pytorch_lightning.utilities.rank_zero:`Trainer(limit_train_batche
```

```
s=1.0)` was configured so 100% of the batches per epoch will be used..
INFO:pytorch_lightning.utilities.rank_zero:`Trainer(limit_val_batches=
1.0)` was configured so 100% of the batches will be used..
INFO:pytorch_lightning.utilities.rank_zero:`Trainer(limit_test_batches
=1.0)` was configured so 100% of the batches will be used..
WARNING:pytorch_lightning.loggers.tensorboard:Missing logger
folder: /content/lightning_logs
INFO:pytorch_lightning.accelerators.cuda:LOCAL_RANK: 0 -
CUDA_VISIBLE_DEVICES: [0]

Epoch 1: Val_Loss: 2.30, Val_Metric: 0.09 |

{"model_id":"d336a09e4fa64d5fa6fbb3354a947f9e","version_major":2,"vers
ion_minor":0}

INFO:pytorch_lightning.utilities.rank_zero:`Trainer.fit` stopped:
`max_steps=30` reached.
INFO:pytorch_lightning.tuner.lr_finder:Learning rate set to
0.0010000000000000002
INFO:pytorch_lightning.utilities.rank_zero:Restoring states from the
checkpoint path at /content/.lr_find_b79fc161-0e40-47ef-9ff3-
4416062ec29e.ckpt
INFO:pytorch_lightning.utilities.rank_zero:Restored all states from
the checkpoint at /content/.lr_find_b79fc161-0e40-47ef-9ff3-
4416062ec29e.ckpt

Train_Loss: 2.10, Train_Metric: 0.42
0.0010000000000000002
```




```
free_memory()
seed_everything(42)
model_config, data_module_config, lightning_module_config, cl_config,
trainer_config = load_all_configs()

# override default values
data_module_config['data_module']['batch_size']=128
lightning_module_config['others']['learning_rate']=0.003
trainer_config['max_epochs']=10
trainer_config['gradient_clip_val']=2
trainer_config['log_every_n_steps']=20

lightning_module_config['others']['optimizer_params']
['weight_decay']=1
lightning_module_config['others']['learning_rate']=0.003
lightning_module_config['scheduler_cls']='torch.optim.lr_scheduler.Red
uceLROnPlateau'
lightning_module_config['scheduler_params']= {'mode': 'max',
'patience': 0, 'factor': 0.5, 'verbose': True}
lightning_module_config['scheduler_options']= {'monitor': 'val_loss',
'interval': 'epoch', 'frequency': 1}
cl_config['lr_monitor']['logging_interval']='epoch'
cl_config['wandb']['project']='fminst'
cl_config['wandb']['name']='resnet'
```

```
# data_module_config['data_module']['small_subset']=True
# data_module_config['data_module']['num_samples_small']=0.5

model, dm, lightning_module, trainer = load_components(model_config,
data_module_config,

lightning_module_config, data_folder, trainer_config,
                                                       cl_config,
batch_size=data_module_config['data_module']['batch_size'],
                                                       logging=True,
checkpointing=True, early_stopping=True)
dm.setup('fit')
trainer.fit(lightning_module, dm)
```

```
INFO:lightning_fabric.utilities.seed:Global seed set to 42
```

```
<IPython.core.display.Javascript object>
```

```
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
wandb: logging graph, to disable use `wandb.watch(log_graph=False)`
INFO:pytorch_lightning.utilities.rank_zero:GPU available: True (cuda),
used: True
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False,
using: 0 TPU cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False,
using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False,
using: 0 HPUs
INFO:pytorch_lightning.utilities.rank_zero:`Trainer(limit_train_batche
s=1.0)` was configured so 100% of the batches per epoch will be used..
INFO:pytorch_lightning.utilities.rank_zero:`Trainer(limit_val_batches=
1.0)` was configured so 100% of the batches will be used..
INFO:pytorch_lightning.utilities.rank_zero:`Trainer(limit_test_batches
=1.0)` was configured so 100% of the batches will be used..
/usr/local/lib/python3.10/dist-packages/pytorch_lightning/callbacks/
model_checkpoint.py:617: UserWarning: Checkpoint directory
/content/drive/MyDrive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/
Data/logs exists and is not empty.
  rank_zero_warn(f"Checkpoint directory {dirpath} exists and is not
empty.")
```

```
INFO:pytorch_lightning.accelerators.cuda:LOCAL_RANK: 0 -
CUDA_VISIBLE_DEVICES: [0]
INFO:pytorch_lightning.callbacks.model_summary:
  | Name         | Type               | Params
-------------------------------------------------
0 | model        | SimpleResNet       | 831 K
1 | loss_fn      | CrossEntropyLoss   | 0
2 | train_metric | MulticlassAccuracy | 0
3 | val_metric   | MulticlassAccuracy | 0
4 | test_metric  | MulticlassAccuracy | 0
-------------------------------------------------
831 K     Trainable params
0         Non-trainable params
831 K     Total params
3.328     Total estimated model params size (MB)

{"model_id":"77f85926eb4a475ba8bd40f706f02b5c","version_major":2,"vers
ion_minor":0}

Epoch 1: Val_Loss: 2.30, Val_Metric: 0.09 |

{"model_id":"24168b5f2a5f49dfaf55ba6e3464bbbc","version_major":2,"vers
ion_minor":0}

{"model_id":"5fbdfa1e1ac44dc3b0330fa0655e63f2","version_major":2,"vers
ion_minor":0}

Epoch 1: Val_Loss: 0.40, Val_Metric: 0.86 |

INFO:pytorch_lightning.callbacks.early_stopping:Metric val_metric
improved. New best score: 0.859

Train_Loss: 0.45, Train_Metric: 0.84

{"model_id":"846b3d284666427aa2ca297b948ee537","version_major":2,"vers
ion_minor":0}

Epoch 2: Val_Loss: 0.41, Val_Metric: 0.86 | Train_Loss: 0.37,
Train_Metric: 0.86

{"model_id":"2ade38f85eb64c2b892ddb19e0c20952","version_major":2,"vers
ion_minor":0}

Epoch 3: Val_Loss: 0.42, Val_Metric: 0.84 | Train_Loss: 0.37,
Train_Metric: 0.87

{"model_id":"44000b5714bd431ba4a46ff5db0035a3","version_major":2,"vers
ion_minor":0}

Epoch 4: Val_Loss: 0.54, Val_Metric: 0.81 | Train_Loss: 0.37,
Train_Metric: 0.87
```

{"model_id":"e84e5b72704d43fd83f6a28d9b6e67fe","version_major":2,"version_minor":0}

```
Epoch 5: Val_Loss: 0.48, Val_Metric: 0.81 | Train_Loss: 0.37,
Train_Metric: 0.87
Epoch 00005: reducing learning rate of group 0 to 1.5000e-03.
```

{"model_id":"bdfecea3779b49bc9d9d2876667f1354","version_major":2,"version_minor":0}

```
Epoch 6: Val_Loss: 0.41, Val_Metric: 0.86 |

INFO:pytorch_lightning.callbacks.early_stopping:Metric val_metric
improved by 0.005 >= min_delta = 0.0. New best score: 0.865

Train_Loss: 0.31, Train_Metric: 0.89
Epoch 00006: reducing learning rate of group 0 to 7.5000e-04.
```

{"model_id":"9f468649e7be4f6aac143b81b308f606","version_major":2,"version_minor":0}

```
Epoch 7: Val_Loss: 0.28, Val_Metric: 0.90 |

INFO:pytorch_lightning.callbacks.early_stopping:Metric val_metric
improved by 0.035 >= min_delta = 0.0. New best score: 0.900

Train_Loss: 0.27, Train_Metric: 0.90
Epoch 00007: reducing learning rate of group 0 to 3.7500e-04.
```

{"model_id":"63d6d18d4bfd499ba73e93d6f26582f6","version_major":2,"version_minor":0}

```
Epoch 8: Val_Loss: 0.28, Val_Metric: 0.90 |

INFO:pytorch_lightning.callbacks.early_stopping:Metric val_metric
improved by 0.001 >= min_delta = 0.0. New best score: 0.900

Train_Loss: 0.23, Train_Metric: 0.92
Epoch 00008: reducing learning rate of group 0 to 1.8750e-04.
```

{"model_id":"6e4060eb16cc4b45b37d6ec1585fff26","version_major":2,"version_minor":0}

```
Epoch 9: Val_Loss: 0.25, Val_Metric: 0.91 |

INFO:pytorch_lightning.callbacks.early_stopping:Metric val_metric
improved by 0.009 >= min_delta = 0.0. New best score: 0.910

Train_Loss: 0.20, Train_Metric: 0.93
Epoch 00009: reducing learning rate of group 0 to 9.3750e-05.
```

{"model_id":"0bcb4d0e2ed347b89aeed761745e2b79","version_major":2,"version_minor":0}

```
Epoch 10: Val_Loss: 0.23, Val_Metric: 0.92 |

INFO:pytorch_lightning.callbacks.early_stopping:Metric val_metric
improved by 0.008 >= min_delta = 0.0. New best score: 0.918

Train_Loss: 0.18, Train_Metric: 0.94
Epoch 00010: reducing learning rate of group 0 to 4.6875e-05.

INFO:pytorch_lightning.utilities.rank_zero:`Trainer.fit` stopped:
`max_epochs=10` reached.
```

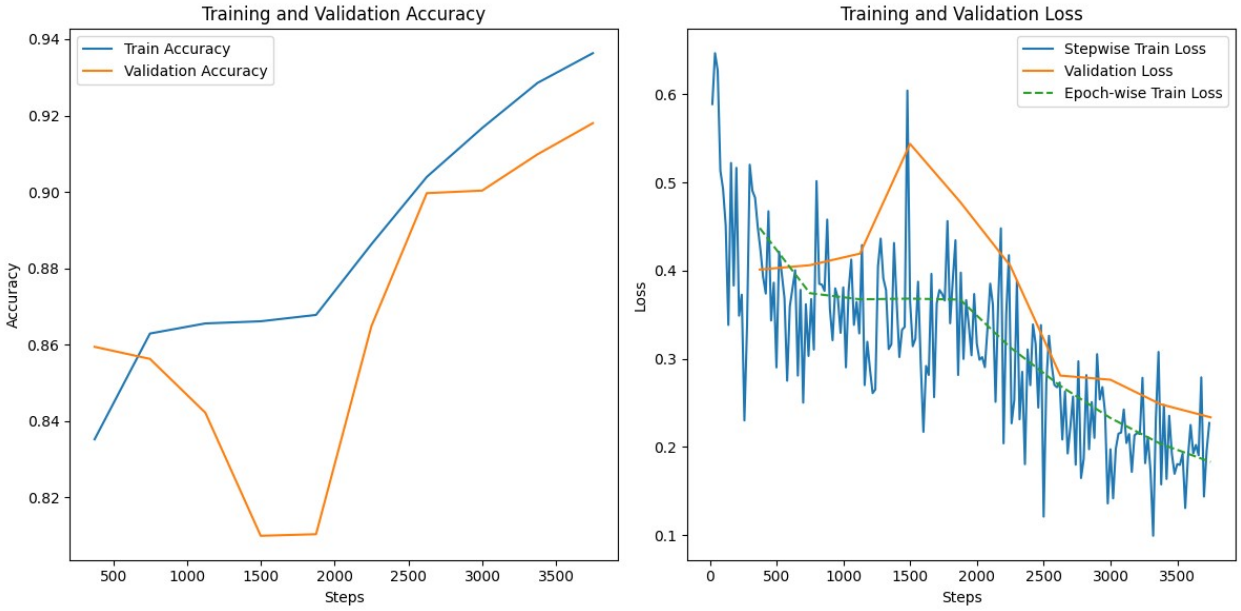
```
file = f"{trainer.logger.log_dir}/metrics.csv"
print(file)
```

```
/content/drive/MyDrive/Colab_Notebooks/
BUAN_6382_Applied_DeepLearning/Data/logs/csvlogger/version_42/
metrics.csv
```

```
import pandas as pd
df = pd.read_csv(file)
pd.DataFrame(df['val_metric'].dropna())
```

```
     val_metric
19     0.859417
41     0.856250
63     0.842167
85     0.809917
106    0.810333
128    0.864833
150    0.899667
172    0.900333
193    0.909833
215    0.918000
```

```
plot_losses_acc(file)
```




```
free_memory()
seed_everything(42)
model_config, data_module_config, lightning_module_config, cl_config,
trainer_config = load_all_configs()

# override default values
data_module_config['data_module']['batch_size']=128
lightning_module_config['others']['learning_rate']=0.003
trainer_config['max_epochs']=10
trainer_config['gradient_clip_val']=2
trainer_config['log_every_n_steps']=20

lightning_module_config['others']['optimizer_params']
['weight_decay']=10
lightning_module_config['others']['learning_rate']=0.003
# Setting the scheduler class
lightning_module_config['scheduler_cls'] =
'torch.optim.lr_scheduler.StepLR' # CODE HERE

# Parameters for the OneCycleLR
# Note: 'max_lr' is a required parameter for OneCycleLR; you'll need
to specify it based on your needs
lightning_module_config['scheduler_params'] = {'step_size':10,
'gamma': 0.5}

# Options related to the monitoring of the scheduler (if needed)
lightning_module_config['scheduler_options'] = {'monitor': 'val_loss',
'interval': 'epoch', 'frequency': 1}

cl_config['lr_monitor']['logging_interval']='epoch'
cl_config['wandb']['project']='fminst'
```

```
cl_config['wandb']['name']='resnet'

# data_module_config['data_module']['small_subset']=True
# data_module_config['data_module']['num_samples_small']=0.5

model, dm, lightning_module, trainer = load_components(model_config,
data_module_config,

lightning_module_config, data_folder, trainer_config,
                                                    cl_config,
batch_size=data_module_config['data_module']['batch_size'],
                                                    logging=True,
checkpointing=True, early_stopping=True)
dm.setup('fit')
trainer.fit(lightning_module, dm)
```

```
INFO:lightning_fabric.utilities.seed:Global seed set to 42
/usr/local/lib/python3.10/dist-packages/pytorch_lightning/loggers/
wandb.py:398: UserWarning: There is a wandb run already in progress
and newly created instances of `WandbLogger` will reuse this run. If
this is not desired, call `wandb.finish()` before instantiating
`WandbLogger`.
  rank_zero_warn(
wandb: logging graph, to disable use `wandb.watch(log_graph=False)`
INFO:pytorch_lightning.utilities.rank_zero:GPU available: True (cuda),
used: True
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False,
using: 0 TPU cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False,
using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False,
using: 0 HPUs
INFO:pytorch_lightning.utilities.rank_zero:`Trainer(limit_train_batche
s=1.0)` was configured so 100% of the batches per epoch will be used..
INFO:pytorch_lightning.utilities.rank_zero:`Trainer(limit_val_batches=
1.0)` was configured so 100% of the batches will be used..
INFO:pytorch_lightning.utilities.rank_zero:`Trainer(limit_test_batches
=1.0)` was configured so 100% of the batches will be used..
/usr/local/lib/python3.10/dist-packages/pytorch_lightning/callbacks/
model_checkpoint.py:617: UserWarning: Checkpoint directory
/content/drive/MyDrive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/
Data/logs exists and is not empty.
  rank_zero_warn(f"Checkpoint directory {dirpath} exists and is not
empty.")
INFO:pytorch_lightning.accelerators.cuda:LOCAL_RANK: 0 -
CUDA_VISIBLE_DEVICES: [0]
INFO:pytorch_lightning.callbacks.model_summary:
  | Name         | Type               | Params
-------------------------------------------------
0 | model        | SimpleResNet       | 831 K
```

```
1 | loss_fn      | CrossEntropyLoss   | 0
2 | train_metric | MulticlassAccuracy | 0
3 | val_metric   | MulticlassAccuracy | 0
4 | test_metric  | MulticlassAccuracy | 0
----------------------------------------------------
831 K     Trainable params
0         Non-trainable params
831 K     Total params
3.328     Total estimated model params size (MB)
```

{"model_id":"a866d5cdb75744189bc82db0660ad1f5","version_major":2,"version_minor":0}

```
Epoch 1: Val_Loss: 2.30, Val_Metric: 0.09 |
```

{"model_id":"ce81eb593b6b4757b80777d783cff81d","version_major":2,"version_minor":0}

{"model_id":"a0ff5338fefb4b19a334eab0767ca53c","version_major":2,"version_minor":0}

```
Epoch 1: Val_Loss: 1.50, Val_Metric: 0.55 |

INFO:pytorch_lightning.callbacks.early_stopping:Metric val_metric
improved. New best score: 0.553

Train_Loss: 0.76, Train_Metric: 0.80
```

{"model_id":"d892200fd9534a50ad49526825ae2637","version_major":2,"version_minor":0}

```
Epoch 2: Val_Loss: 1.63, Val_Metric: 0.56 |

INFO:pytorch_lightning.callbacks.early_stopping:Metric val_metric
improved by 0.006 >= min_delta = 0.0. New best score: 0.559

Train_Loss: 0.79, Train_Metric: 0.81
```

{"model_id":"be0a304d79d346aebdf497cf1b9718c8","version_major":2,"version_minor":0}

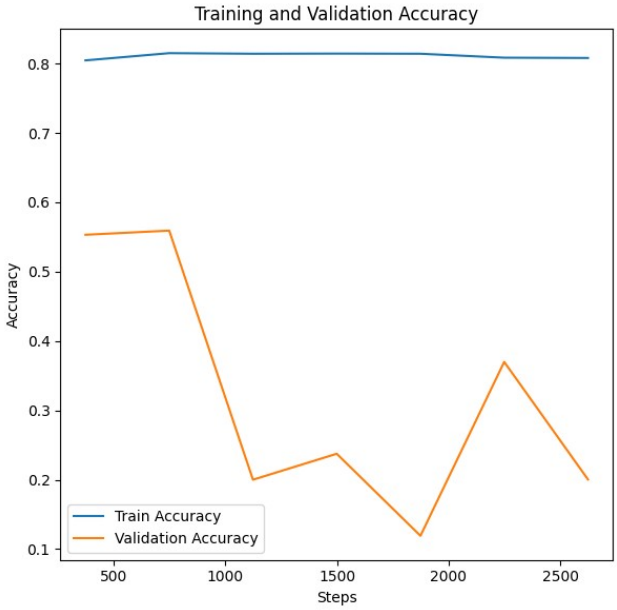
```
Epoch 3: Val_Loss: 2.16, Val_Metric: 0.20 | Train_Loss: 0.80,
Train_Metric: 0.81
```

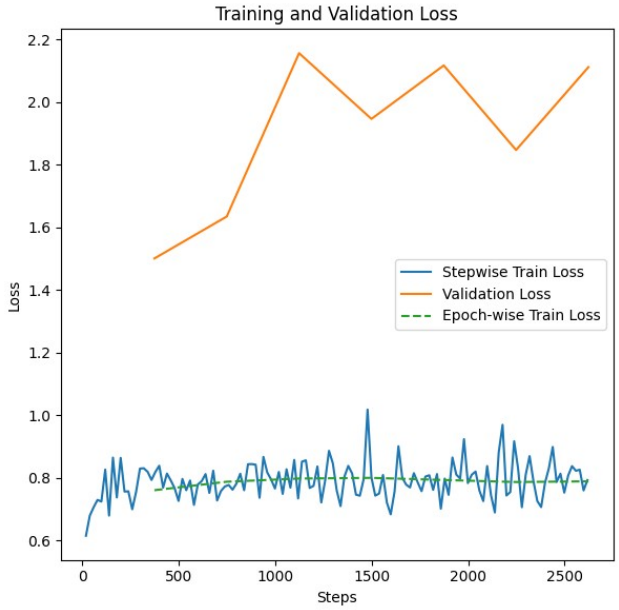
{"model_id":"5cb12174e35542b599125b7b065ba5db","version_major":2,"version_minor":0}

```
Epoch 4: Val_Loss: 1.95, Val_Metric: 0.24 | Train_Loss: 0.80,
Train_Metric: 0.81
```

{"model_id":"4fa33298644144888929c5cbc793e603","version_major":2,"version_minor":0}

```
Epoch 5: Val_Loss: 2.12, Val_Metric: 0.12 | Train_Loss: 0.79,
Train_Metric: 0.81

{"model_id":"7429f2d1c50c4be3aa2954d7fe130124","version_major":2,"vers
ion_minor":0}

Epoch 6: Val_Loss: 1.85, Val_Metric: 0.37 | Train_Loss: 0.79,
Train_Metric: 0.81

{"model_id":"429d1a9207224819bdb8eb68df99bd65","version_major":2,"vers
ion_minor":0}

Epoch 7: Val_Loss: 2.11, Val_Metric: 0.20 |

INFO:pytorch_lightning.callbacks.early_stopping:Monitored metric
val_metric did not improve in the last 5 records. Best score: 0.559.
Signaling Trainer to stop.

Train_Loss: 0.79, Train_Metric: 0.81

file = f"{trainer.logger.log_dir}/metrics.csv"
print(file)

/content/drive/MyDrive/Colab_Notebooks/
BUAN_6382_Applied_DeepLearning/Data/logs/csvlogger/version_43/
metrics.csv

import pandas as pd
df = pd.read_csv(file)
pd.DataFrame(df['val_metric'].dropna())

     val_metric
19     0.553167
41     0.559167
63     0.200083
85     0.237500
106    0.119250
128    0.369917
150    0.200500

plot_losses_acc(file)
```


Training and Validation Accuracy
Accuracy
Steps
Train Accuracy
Validation Accuracy


Training and Validation Loss
Loss
Steps
Stepwise Train Loss
Validation Loss
Epoch-wise Train Loss