Harikrishna Dev HXD220000

Setup environment

```
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
```

```
Path is /Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/Data
```

```
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')

Path is /Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/Custom_files
```

Importing libraries

```
# 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??

Init signature: SimpleResNet(num_classes=120)
Docstring:
Base class for all neural network modules.

Your models should also subclass this class.

Modules can also contain other Modules, allowing to nest them in
```

```
a tree structure. You can assign the submodules as regular
attributes::

    import torch.nn as nn
    import torch.nn.functional as F

    class Model(nn.Module):
        def __init__(self):
            super().__init__()
            self.conv1 = nn.Conv2d(1, 20, 5)
            self.conv2 = nn.Conv2d(20, 20, 5)

        def forward(self, x):
            x = F.relu(self.conv1(x))
            return F.relu(self.conv2(x))

Submodules assigned in this way will be registered, and will have
their
parameters converted too when you call :meth:`to`, etc.

.. note::
    As per the example above, an ``__init__()`` call to the parent
class
    must be made before assignment on the child.

:ivar training: Boolean represents whether this module is in training
or
                evaluation mode.
:vartype training: bool
Source:
class SimpleResNet(nn.Module):
    def __init__(self, num_classes=120):
        super(SimpleResNet, self).__init__()

        self.model = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=7, stride=2, padding=3),  #
Output: 16x250x188
            nn.BatchNorm2d(16),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2, padding=1),      #
Output: 16x125x94
            ResidualBlock(16, 32, stride=2),                       #
Output: 32x63x47
            ResidualBlock(32, 64, stride=2),                       #
Output: 64x32x24
            ResidualBlock(64, 256, stride=2),                      #
Output: 128x16x12
            nn.AdaptiveAvgPool2d((1, 1))                           #
Output: 128x1x1
        )
        self.fc = nn.Linear(256, num_classes)

    def forward(self, x):
        x = self.model(x)
        x = x.view(x.size(0), -1)
        x = self.fc(x)
        return x
File:           ~/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/Custom_files/
resnet.py
Type:           type
Subclasses:
```

Model initialisation

```
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
```

Transformation

```
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])
```

Modules for training

```
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
```

Running one training and validations batch to check bugs

```
# 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)
```

```
Global seed set to 42
GPU available: True (mps), used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
/Users/harikrishnadev/.pyenv/versions/3.11.4/lib/python3.11/site-
packages/pytorch_lightning/trainer/connectors/logger_connector/
logger_connector.py:67: UserWarning: Starting from v1.9.0,
`tensorboardX` has been removed as a dependency of the
`pytorch_lightning` package, due to potential conflicts with other
packages in the ML ecosystem. For this reason, `logger=True` will use
`CSVLogger` as the default logger, unless the `tensorboard` or
`tensorboardX` packages are found. Please `pip install
lightning[extra]` or one of them to enable TensorBoard support by
default
  warning_cache.warn(
Running in `fast_dev_run` mode: will run the requested loop using 1
batch(es). Logging and checkpointing is suppressed.

  | 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)
/Users/harikrishnadev/.pyenv/versions/3.11.4/lib/python3.11/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:442:
PossibleUserWarning: The dataloader, train_dataloader, does not have
many workers which may be a bottleneck. Consider increasing the value
of the `num_workers` argument` (try 8 which is the number of cpus on
this machine) in the `DataLoader` init to improve performance.
  rank_zero_warn(
/Users/harikrishnadev/.pyenv/versions/3.11.4/lib/python3.11/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:442:
PossibleUserWarning: The dataloader, val_dataloader, does not have
many workers which may be a bottleneck. Consider increasing the value
of the `num_workers` argument` (try 8 which is the number of cpus on
this machine) in the `DataLoader` init to improve performance.
  rank_zero_warn(
```

{"model_id":"5939a7edcc1f48b4b78bd162d65175e8","version_major":2,"version_minor":0}

```
/Users/harikrishnadev/.pyenv/versions/3.11.4/lib/python3.11/site-
packages/torchmetrics/functional/classification/accuracy.py:77:
UserWarning: MPS: no support for int64 reduction ops, casting it to
int32 (Triggered internally at
/Users/runner/work/pytorch/pytorch/pytorch/aten/src/ATen/native/mps/
operations/ReduceOps.mm:144.)
  tp = tp.sum(dim=0 if multidim_average == "global" else 1)
```

{"model_id":"d1c4cabb00b148439510149c1541abd5","version_major":2,"version_minor":0}

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

`Trainer.fit` stopped: `max_steps=1` reached.

Train_Loss: 2.60, Train_Metric: 0.08
```

Learning rate

```
# 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)
```

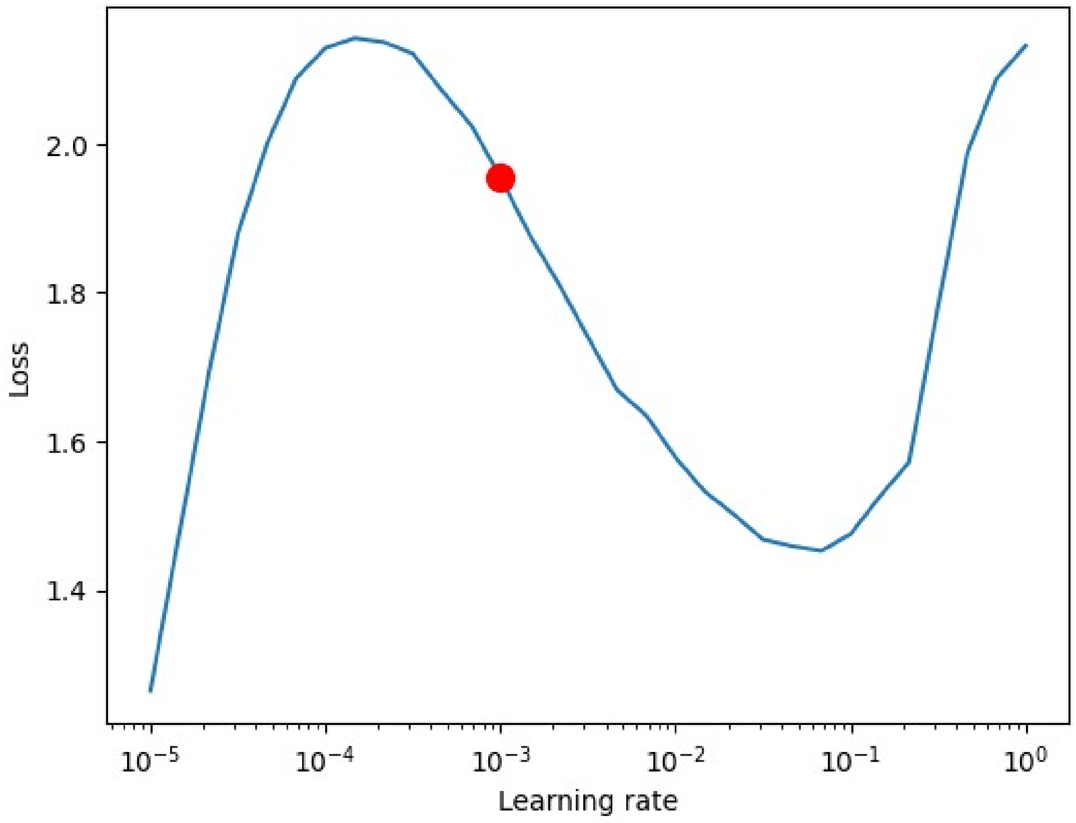
```
Global seed set to 42
GPU available: True (mps), used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
`Trainer(limit_train_batches=1.0)` was configured so 100% of the
batches per epoch will be used..
`Trainer(limit_val_batches=1.0)` was configured so 100% of the batches
will be used..
`Trainer(limit_test_batches=1.0)` was configured so 100% of the
batches will be used..

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

{"model_id":"d5ee8eac0bea4dc7a128d86de1621c62","version_major":2,"vers
ion_minor":0}

```
`Trainer.fit` stopped: `max_steps=30` reached.
Learning rate set to 0.0010000000000000002
Restoring states from the checkpoint path at
/Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/HW/.lr_find_a0572
99b-bede-4a19-8706-51abdd2a9014.ckpt
Restored all states from the checkpoint at
/Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/HW/.lr_find_a0572
99b-bede-4a19-8706-51abdd2a9014.ckpt
```

```
Train_Loss: 2.10, Train_Metric: 0.42
0.0010000000000000002
```



Model training

```
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']=0.001
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)
```

```
Global seed set to 42
Failed to detect the name of this notebook, you can set it manually
with the WANDB_NOTEBOOK_NAME environment variable to enable code
saving.
wandb: Currently logged in as: harikrish0607 (harikrishnad). Use
`wandb login --relogin` to force relogin

<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)`
GPU available: True (mps), used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
`Trainer(limit_train_batches=1.0)` was configured so 100% of the
batches per epoch will be used..
`Trainer(limit_val_batches=1.0)` was configured so 100% of the batches
will be used..
`Trainer(limit_test_batches=1.0)` was configured so 100% of the
batches will be used..
/Users/harikrishnadev/.pyenv/versions/3.11.4/lib/python3.11/site-
```

```
packages/pytorch_lightning/callbacks/model_checkpoint.py:617:
UserWarning: Checkpoint directory
/Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/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.")

  | 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":"2b212acc1bbe43afa31ad225eb1ad18a","version_major":2,"version_minor":0}

```
/Users/harikrishnadev/.pyenv/versions/3.11.4/lib/python3.11/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:442:
PossibleUserWarning: The dataloader, val_dataloader, does not have
many workers which may be a bottleneck. Consider increasing the value
of the `num_workers` argument` (try 8 which is the number of cpus on
this machine) in the `DataLoader` init to improve performance.
  rank_zero_warn(

Epoch 1: Val_Loss: 2.30, Val_Metric: 0.09 |

/Users/harikrishnadev/.pyenv/versions/3.11.4/lib/python3.11/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:442:
PossibleUserWarning: The dataloader, train_dataloader, does not have
many workers which may be a bottleneck. Consider increasing the value
of the `num_workers` argument` (try 8 which is the number of cpus on
this machine) in the `DataLoader` init to improve performance.
  rank_zero_warn(
```

{"model_id":"0cecc8f723f04c9aaae8a07ea5035255","version_major":2,"version_minor":0}

{"model_id":"107c8cd6a40f4eaca3d5b2fee4e1bcde","version_major":2,"version_minor":0}

```
Epoch 1: Val_Loss: 0.44, Val_Metric: 0.83 |
Metric val_metric improved. New best score: 0.834

Train_Loss: 0.44, Train_Metric: 0.84

{"model_id":"268017b489634a8abca22dbaeb47b201","version_major":2,"vers
ion_minor":0}

Epoch 2: Val_Loss: 0.47, Val_Metric: 0.84 |

Metric val_metric improved by 0.008 >= min_delta = 0.0. New best
score: 0.842

Train_Loss: 0.38, Train_Metric: 0.86

{"model_id":"431f44631bc540ea809003428f5ace06","version_major":2,"vers
ion_minor":0}

Epoch 3: Val_Loss: 0.94, Val_Metric: 0.69 | Train_Loss: 0.37,
Train_Metric: 0.86

{"model_id":"39175a47fbee4d7883dfa70be61cb073","version_major":2,"vers
ion_minor":0}

Epoch 4: Val_Loss: 0.70, Val_Metric: 0.74 | Train_Loss: 0.37,
Train_Metric: 0.87
Epoch 00004: reducing learning rate of group 0 to 1.5000e-03.

{"model_id":"d7dc3bcc200c4bfebc0f00b81948ae8f","version_major":2,"vers
ion_minor":0}

Epoch 5: Val_Loss: 0.32, Val_Metric: 0.89 |

Metric val_metric improved by 0.046 >= min_delta = 0.0. New best
score: 0.888

Train_Loss: 0.31, Train_Metric: 0.89
Epoch 00005: reducing learning rate of group 0 to 7.5000e-04.

{"model_id":"455680a87b984c2ea813800ad03b5a9c","version_major":2,"vers
ion_minor":0}

Epoch 6: Val_Loss: 0.35, Val_Metric: 0.87 | Train_Loss: 0.27,
Train_Metric: 0.90
Epoch 00006: reducing learning rate of group 0 to 3.7500e-04.

{"model_id":"bc491ba5bf4d48b19ad81fa4d535b09b","version_major":2,"vers
ion_minor":0}

Epoch 7: Val_Loss: 0.28, Val_Metric: 0.90 |

Metric val_metric improved by 0.014 >= min_delta = 0.0. New best
score: 0.902
```

```
Train_Loss: 0.23, Train_Metric: 0.92
Epoch 00007: reducing learning rate of group 0 to 1.8750e-04.
```

{"model_id":"3cfc66e8c9624b06a06d98e94e3167bf","version_major":2,"version_minor":0}

```
Epoch 8: Val_Loss: 0.25, Val_Metric: 0.91 |

Metric val_metric improved by 0.010 >= min_delta = 0.0. New best
score: 0.912

Train_Loss: 0.20, Train_Metric: 0.93
Epoch 00008: reducing learning rate of group 0 to 9.3750e-05.
```

{"model_id":"1ae3989fc73740f2ba0979678e441a35","version_major":2,"version_minor":0}

```
Epoch 9: Val_Loss: 0.24, Val_Metric: 0.92 |

Metric val_metric improved by 0.005 >= min_delta = 0.0. New best
score: 0.917

Train_Loss: 0.18, Train_Metric: 0.94
Epoch 00009: reducing learning rate of group 0 to 4.6875e-05.
```

{"model_id":"b7eebf1e720c4e4290008c6fad5765af","version_major":2,"version_minor":0}

```
Epoch 10: Val_Loss: 0.24, Val_Metric: 0.92 |

Metric val_metric improved by 0.001 >= min_delta = 0.0. New best
score: 0.918
`Trainer.fit` stopped: `max_epochs=10` reached.

Train_Loss: 0.17, Train_Metric: 0.94
Epoch 00010: reducing learning rate of group 0 to 2.3438e-05.
```

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

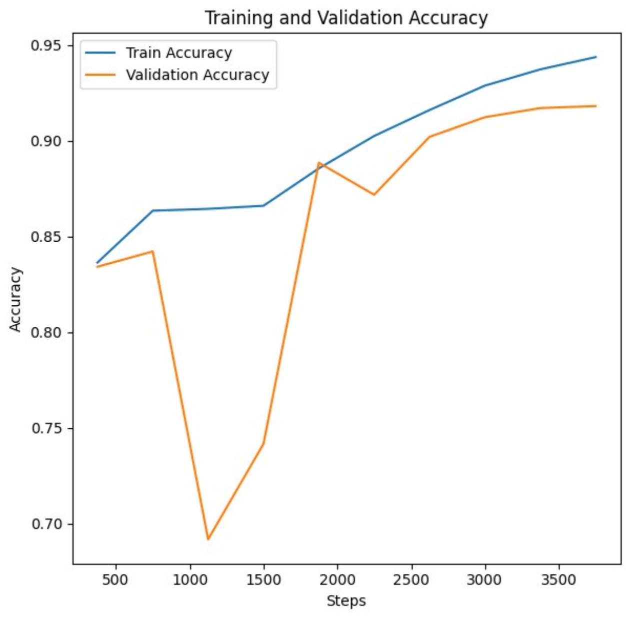
```
/Users/harikrishnadev/Library/CloudStorage/GoogleDrive-
harikrish0607@gmail.com/My
Drive/Colab_Notebooks/BUAN_6382_Applied_DeepLearning/Data/logs/csvlogg
er/version_45/metrics.csv
```

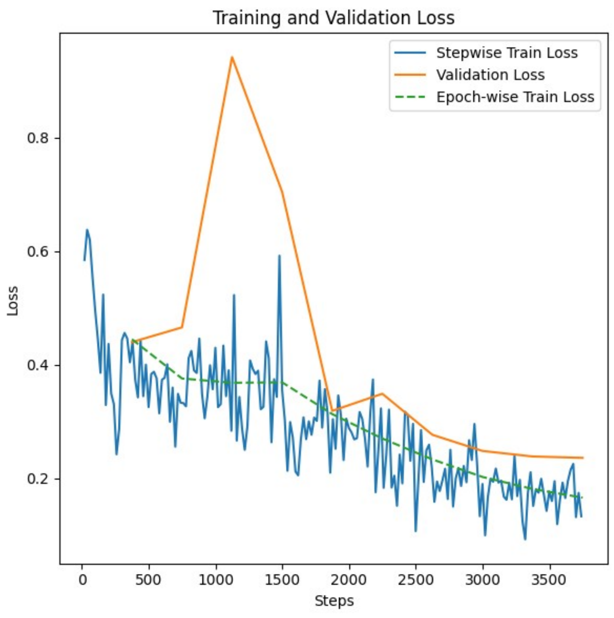
```python
import pandas as pd
df = pd.read_csv(file)
pd.DataFrame(df['val_metric'].dropna())
```

```
     val_metric
19     0.834167
41     0.842167
63     0.691833
```

```
85      0.741667
106     0.888500
128     0.871750
150     0.902000
172     0.912250
193     0.917000
215     0.918083
```

```
plot_losses_acc(file)
```


Training and Validation Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Steps


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