Navya_Bhat_HW6

November 1, 2023

[1]:
```python
from pathlib import Path
import sys

if 'google.colab' in str(get_ipython()):
    from google.colab import drive
    drive.mount('/content/drive')

    base_folder = Path('/content/drive/MyDrive/')
    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

else:
    base_folder = Path('/home/harpreet/Insync/google_drive_shaannoor')
    data_folder = Path('/home/harpreet/data')
```

```
Mounted at /content/drive
                           727.7/727.7

kB 9.9 MB/s eta 0:00:00
                           805.2/805.2

kB 41.6 MB/s eta 0:00:00
                           2.1/2.1 MB
21.6 MB/s eta 0:00:00
                           190.6/190.6

kB 22.3 MB/s eta 0:00:00
                           243.2/243.2

kB 27.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
                           62.7/62.7 kB
8.3 MB/s eta 0:00:00
  Building wheel for pathtools (setup.py) … done
```

[2]:
```
custom_function_folder = base_folder/'data/custom-functions/fall_2023'
sys.path.append('/content/drive/MyDrive/')
model_folder = base_folder/'data/models/dl_fall_2023/fmnist/oct-31'
model_folder.mkdir(parents=True, exist_ok=True)
project_folder = base_folder/'data/fmnist'
project_folder.mkdir(parents=True, exist_ok=True)
```

[3]:
```
# import Libraries
import yaml

import torch
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
import torch.nn as nn

from data_module_fmnist import FashionMNISTDataModule
from multiclass_lightning_module_v0 import MultiClassLightningModule
from shared_utils import  plot_losses_acc
```

[4]:
```
!pip show pytorch-lightning
```

```
Name: pytorch-lightning
Version: 2.0.9
Summary: PyTorch Lightning is the lightweight PyTorch wrapper for ML
researchers. Scale your models. Write less boilerplate.
Home-page: https://github.com/Lightning-AI/lightning
Author: Lightning AI et al.
Author-email: pytorch@lightning.ai
License: Apache-2.0
Location: /usr/local/lib/python3.10/dist-packages
Requires: fsspec, lightning-utilities, numpy, packaging, PyYAML, torch,
torchmetrics, tqdm, typing-extensions
Required-by:
```

[5]:
```
class ResidualBlock(nn.Module):
    def __init__(self, in_channels, out_channels, stride=1):
        super(ResidualBlock, self).__init__()

        self.main_path = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=stride,␣
↪padding=1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1),
            nn.BatchNorm2d(out_channels)
        )

        self.downsample = nn.Sequential()
        if stride != 1 or in_channels != out_channels:
            self.downsample = nn.Sequential(
                nn.Conv2d(in_channels, out_channels, kernel_size=1,␣
↪stride=stride),
                nn.BatchNorm2d(out_channels)
            )

    def forward(self, x):
        residual = x
        out = self.main_path(x)
        residual = self.downsample(residual)
        out += residual
        out = nn.ReLU(inplace=True)(out)
        return out
class SimpleResNet(nn.Module):
    def __init__(self, num_classes=10):
        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:␣
↪256x1x1
        )

        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
```

[6]:
```
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
```

[7]:
```
#Function to load the model
def load_model(model_config):
    model = SimpleResNet(num_classes=10)
    return model
```

[8]:
```
#function for transformations
def get_train_transforms(resize_height, resize_width, normalize_mean,␣
 ↪normalize_std):

    return transforms.Compose(
        [
            transforms.Resize((resize_height, resize_width)),
            transforms.ToTensor(),
            transforms.Normalize(normalize_mean, normalize_std),
        ]
    )

def get_test_transforms(resize_height, resize_width, normalize_mean,␣
 ↪normalize_std):

    return transforms.Compose(
        [
            transforms.Resize((resize_height, resize_width)),
            transforms.ToTensor(),
            transforms.Normalize(normalize_mean, normalize_std),
        ]
    )
```

```
[9]: 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])
```

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

```
[11]: # Load Lightning Module
      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
```

[12]:
```
# Load the trainer
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
```

[13]:
```
# Function to load components
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
```

[14]:
```
def load_yaml(filepath):
    with open(filepath, 'r') as file:
        return yaml.safe_load(file)
```

[15]:
```
# Load configurations from YAML files
def load_all_configs():
    model_config = load_yaml(project_folder/'model_config.yaml')
    data_module_config = load_yaml(project_folder/'data_module_config.yaml')
    lightning_module_config = load_yaml(project_folder/'lightning_module_config.
 ↪yaml')
    cl_config = load_yaml(project_folder/'callbacks_loggers_config.yaml')
    trainer_config = load_yaml(project_folder/'trainer_config.yaml')

    return model_config, data_module_config, lightning_module_config,␣
 ↪cl_config, trainer_config
```

[16]:
```
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()
```

[17]:
```
print(project_folder)
```

```
/content/drive/MyDrive/data/fmnist
```

[18]:
```
# Running a training and validation batch
# 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 [00:03<00:00, 7251610.57it/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, 137749.30it/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, 2572010.57it/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, 7961754.05it/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)
/usr/local/lib/python3.10/dist-
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(
/usr/local/lib/python3.10/dist-
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(
```

```
Training: 0it [00:00, ?it/s]
```

```
Validation: 0it [00:00, ?it/s]
```

```
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: 2.45, Train_Metric: 0.14
```

[19]:
```
print(model_config, data_module_config, lightning_module_config, cl_config,␣
  ↪trainer_config)
```

```
{'num_features': 562500, 'hidden_dim1': 500, 'hidden_dim2': 500, 'num_classes':
10} {'train_transform': {'resize_height': 500, 'resize_width': 375,
'normalize_mean': [0.5], 'normalize_std': [0.5]}, 'test_transform':
{'resize_height': 500, 'resize_width': 375, 'normalize_mean': [0.5],
'normalize_std': [0.5]}, 'data_module': {'batch_size': 64, 'seed': 42}}
{'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}} {'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': 'FMNIST', 'name':
'resnet'}, 'csv': {'name': 'csvlogger'}} {'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': True,
'overfit_batches': 0.0, 'accumulate_grad_batches': 1, 'limit_train_batches':
1.0, 'limit_val_batches': 1.0, 'limit_test_batches': 1.0}
```

[20]:
```
# 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']=10
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(stage='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)
```

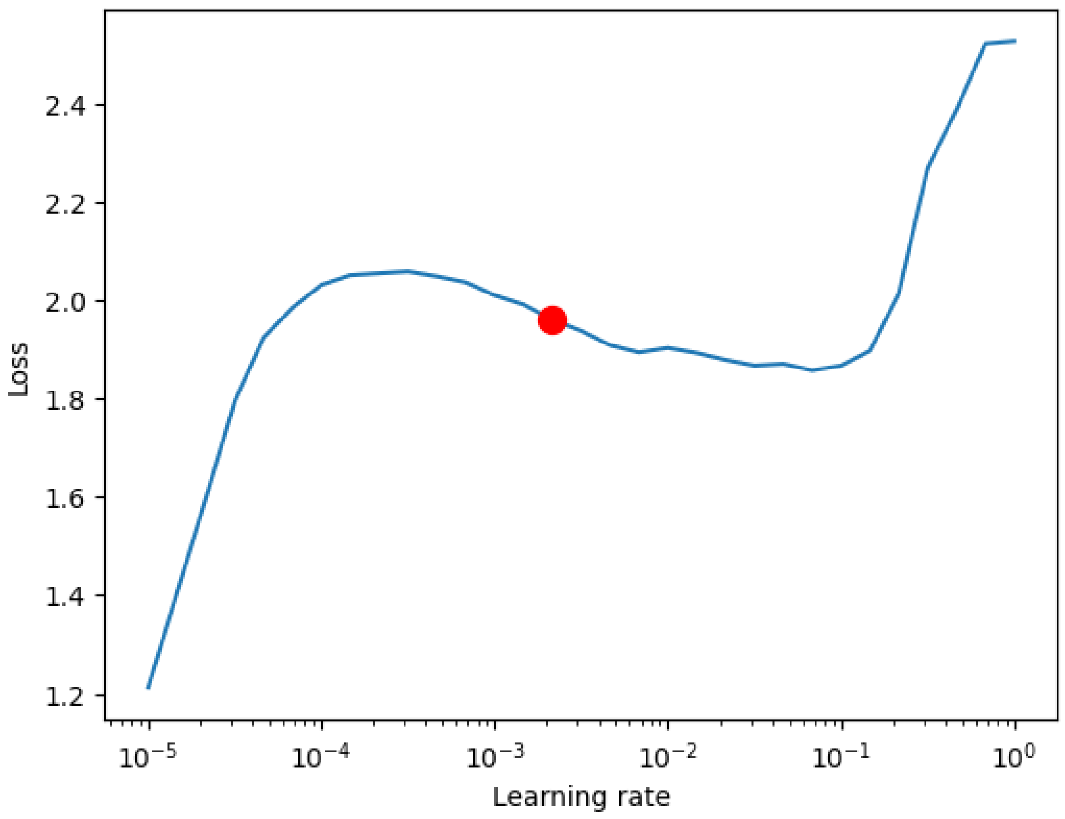
```
INFO:lightning_fabric.utilities.seed:Global seed set to 42
INFO:pytorch_lightning.utilities.rank_zero:GPU available: True (cuda), used:
True
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TPU
cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO:pytorch_lightning.utilities.rank_zero:`Trainer(limit_train_batches=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: 0.03 |
```

```
Finding best initial lr:   0%|          | 0/30 [00:00<?, ?it/s]
```

```
INFO:pytorch_lightning.utilities.rank_zero:`Trainer.fit` stopped: `max_steps=30`
reached.
INFO:pytorch_lightning.tuner.lr_finder:Learning rate set to 0.002154434690031884
INFO:pytorch_lightning.utilities.rank_zero:Restoring states from the checkpoint
path at /content/.lr_find_4c9b37df-b6d0-402f-8f32-bc0e0e36c765.ckpt
INFO:pytorch_lightning.utilities.rank_zero:Restored all states from the
checkpoint at /content/.lr_find_4c9b37df-b6d0-402f-8f32-bc0e0e36c765.ckpt
```

```
Train_Loss: 2.48, Train_Metric: 0.28
0.002154434690031884
```



[21]:
```
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']=0.002
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']=0.002
lightning_module_config['scheduler_cls']='torch.optim.lr_scheduler.
 ↪ReduceLROnPlateau'
lightning_module_config['scheduler_params']= {'mode': 'max', 'patience': 0,␣
 ↪'factor': 0.5, 'verbose': True}
lightning_module_config['scheduler_options']= {'monitor': 'val_metric',␣
 ↪'interval': 'epoch', 'frequency': 1}
```

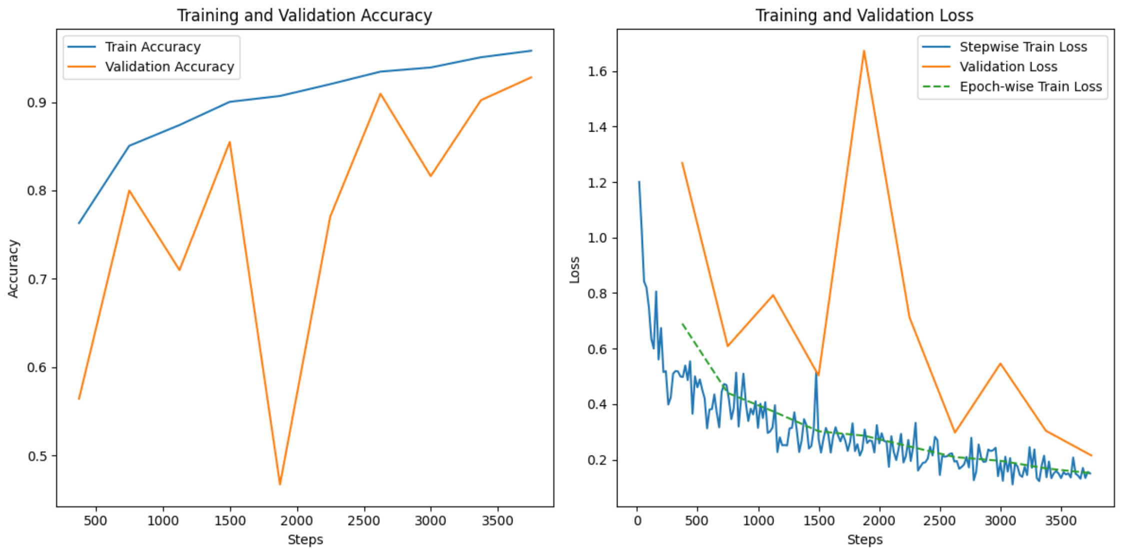
```
cl_config['lr_monitor']['logging_interval']='epoch'
cl_config['wandb']['project']='fminst'
cl_config['wandb']['name']='resnet'


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(stage='fit')
trainer.fit(lightning_module, dm)
```

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

<IPython.core.display.Javascript object>

wandb: Logging into wandb.ai. (Learn how to deploy a W&B server
locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here:
https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to
quit:

 ··········

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_batches=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/data/models/dl_fall_2023/fmnist/oct-31/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)

Sanity Checking: 0it [00:00, ?it/s]

/usr/local/lib/python3.10/dist-
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: 0.04 |

/usr/local/lib/python3.10/dist-
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(

Training: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Epoch 1: Val_Loss: 1.27, Val_Metric: 0.56 |
INFO:pytorch_lightning.callbacks.early_stopping:Metric val_metric improved. New
best score: 0.564

Train_Loss: 0.69, Train_Metric: 0.76

Validation: 0it [00:00, ?it/s]

Epoch 2: Val_Loss: 0.61, Val_Metric: 0.80 |

INFO:pytorch_lightning.callbacks.early_stopping:Metric val_metric improved by
0.236 >= min_delta = 0.0. New best score: 0.800

Train_Loss: 0.44, Train_Metric: 0.85

Validation: 0it [00:00, ?it/s]

Epoch 3: Val_Loss: 0.79, Val_Metric: 0.71 | Train_Loss: 0.37, Train_Metric: 0.87
Epoch 00003: reducing learning rate of group 0 to 1.0000e-03.

Validation: 0it [00:00, ?it/s]

Epoch 4: Val_Loss: 0.50, Val_Metric: 0.86 |

INFO:pytorch_lightning.callbacks.early_stopping:Metric val_metric improved by
0.055 >= min_delta = 0.0. New best score: 0.855

Train_Loss: 0.30, Train_Metric: 0.90

Validation: 0it [00:00, ?it/s]

Epoch 5: Val_Loss: 1.67, Val_Metric: 0.47 | Train_Loss: 0.29, Train_Metric: 0.91
Epoch 00005: reducing learning rate of group 0 to 5.0000e-04.

Validation: 0it [00:00, ?it/s]

Epoch 6: Val_Loss: 0.71, Val_Metric: 0.77 | Train_Loss: 0.25, Train_Metric: 0.92
Epoch 00006: reducing learning rate of group 0 to 2.5000e-04.

Validation: 0it [00:00, ?it/s]

Epoch 7: Val_Loss: 0.30, Val_Metric: 0.91 |

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

Train_Loss: 0.21, Train_Metric: 0.93

Validation: 0it [00:00, ?it/s]

Epoch 8: Val_Loss: 0.55, Val_Metric: 0.82 | Train_Loss: 0.20, Train_Metric: 0.94
Epoch 00008: reducing learning rate of group 0 to 1.2500e-04.

Validation: 0it [00:00, ?it/s]

Epoch 9: Val_Loss: 0.30, Val_Metric: 0.90 | Train_Loss: 0.17, Train_Metric: 0.95
Epoch 00009: reducing learning rate of group 0 to 6.2500e-05.

Validation: 0it [00:00, ?it/s]

Epoch 10: Val_Loss: 0.21, Val_Metric: 0.93 |
INFO:pytorch_lightning.callbacks.early_stopping:Metric val_metric improved by
0.019 >= min_delta = 0.0. New best score: 0.928
INFO:pytorch_lightning.utilities.rank_zero:`Trainer.fit` stopped:
`max_epochs=10` reached.
```

```
Train_Loss: 0.15, Train_Metric: 0.96
```

[22]:
```python
file = f"{trainer.logger.log_dir}/metrics.csv"
plot_losses_acc(file)
```



[23]:
```python
ckpt_path = trainer.checkpoint_callback.best_model_path
train_acc = trainer.validate(dataloaders=dm.train_dataloader(),␣
  ↪ckpt_path=ckpt_path, verbose=False)
valid_acc = trainer.validate(dataloaders=dm.val_dataloader(),␣
  ↪ckpt_path=ckpt_path, verbose=False)
print(f"Train Accuracy: {train_acc[0]['val_metric']*100:0.2f}")
print(f"Validation Accuracy: {valid_acc[0]['val_metric']*100:0.2f}")
wandb.finish()
```

```
INFO:pytorch_lightning.utilities.rank_zero:Restoring states from the checkpoint
path at /content/drive/MyDrive/data/models/dl_fall_2023/fmnist/oct-
31/logs/epoch=9-step=3750-v3.ckpt
INFO:pytorch_lightning.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES:
[0]
INFO:pytorch_lightning.utilities.rank_zero:Loaded model weights from the
checkpoint at /content/drive/MyDrive/data/models/dl_fall_2023/fmnist/oct-
31/logs/epoch=9-step=3750-v3.ckpt
/usr/local/lib/python3.10/dist-
packages/pytorch_lightning/trainer/connectors/data_connector.py:490:
PossibleUserWarning: Your `val_dataloader`'s sampler has shuffling enabled, it
```

```
is strongly recommended that you turn shuffling off for val/test dataloaders.
  rank_zero_warn(
/usr/local/lib/python3.10/dist-
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(
```

```
Validation: 0it [00:00, ?it/s]
```

```
Epoch 11: Val_Loss: 0.14, Val_Metric: 0.96 |
```

```
INFO:pytorch_lightning.utilities.rank_zero:Restoring states from the checkpoint
path at /content/drive/MyDrive/data/models/dl_fall_2023/fmnist/oct-
31/logs/epoch=9-step=3750-v3.ckpt
INFO:pytorch_lightning.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES:
[0]
INFO:pytorch_lightning.utilities.rank_zero:Loaded model weights from the
checkpoint at /content/drive/MyDrive/data/models/dl_fall_2023/fmnist/oct-
31/logs/epoch=9-step=3750-v3.ckpt
```

```
Validation: 0it [00:00, ?it/s]
```

```
Epoch 11: Val_Loss: 0.21, Val_Metric: 0.93 | Train Accuracy: 96.32
Validation Accuracy: 92.81
```

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

```
VBox(children=(Label(value='0.006 MB of 0.006 MB uploaded (0.000 MB␣
 ↪deduped)\r'), FloatProgress(value=1.0, max…
```

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

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

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