PyTorch Lightning Workshop

Devin Hajjari, Ben Taylor, and Seyed Alireza Vaezi

November 29, 2021

PyTorch Lightning

- A lightweight deep learning framework built on PyTorch for researchers
  - Created by William Falcon
- “PyTorch Lightning is just organized PyTorch” - (PyTorch Lightning Github)

What does Lightning do?

- **Improved readability**
  - Separates engineering code from research
  - Minimizes boilerplate code
    - (e.g. epochs, dataset looping, model.eval())
- **Easy to implement hardware acceleration (GPUs and TPUs)**
  - Offers 40 advanced features for scaling and research
- **Logging and model performance**
  - Now standalone package –> TorchMetrics
  - General and domain specific metrics of evaluation.
  - Compatible with Weights & Biases (WandB)

Improved readability

1. The `LightningModule() ~ nn.Module`
    a. nn is the base class for neural network modules in PyTorch
    b. The `LightningModule` extends `torch.nn.Module` functionality with built-in methods
        i. Self-contained.
        ii. Where most of your code will be.
2. Do not need to specify:
    a. `Model.eval(), Model.train(), # number of epochs`

Hardware acceleration

Limited code changes are needed for implementation with GPUs, multi-node, TPUs, change max epochs.

```
# Initiate training
Trainer = Trainer()
```

```
# Use 4 GPUs
Trainer = Trainer(gpus=4)
```

```
# Limit epochs to 10, use 48 GPUs (4 * 12)
Trainer = Trainer(max_epochs = 10, gpus = 4, num_nodes = 12)
```

```
# Use 8 TPU cores
Trainer = Trainer(tpu_cores=8)
```

Logging and model performance

```
print(f'Epoch {epoch + 1}', end = ', ')
print(f'Validation loss : {torch.tensor(losses).mean():.2f}', end = ', ')
print(f'Validation accuracy: {torch.tensor(accuracies).mean():.2f}')
```

```
-----------------------------------------------------------------
(returns)
```

```
Epoch 1, Training loss: 0.21, Training accuracy: 0.94
Epoch 1, Validation loss : 0.20, Validation accuracy: 0.94
```

Lightning Logs: Implementing logs

PyTorch Lightning includes logging functionality that you define within your model `LightningModule()`

```
def validation_step(self, batch, batch_idx):
    x, y = batch
    logits = self(x)
    loss = F.nll_loss(logits, y)
    return loss
```

```
def validation_step(self, batch, batch_idx):
    x, y = batch
    logits = self(x)
    loss = F.nll_loss(logits, y)
    preds = torch.argmax(logits, dim=1)
    acc = accuracy(preds, y)
    self.log("val_loss", loss, prog_bar=True)
    self.log("val_acc", acc, prog_bar=True)
    return loss
```

Lightning Logs: Resource view

```
trainer = Trainer(max_epochs=3, gpus=[0])# Init trainer specifying which GPU
trainer.fit(model, dm)
```

```
(returns)
```

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name  | Type       | Params
-------------------------------------
0 | model | Sequential | 55.1 K
-------------------------------------
55.1 K    Trainable params
0         Non-trainable params
55.1 K    Total params
0.220     Total estimated model params size (MB)
```

```
preds = torch.argmax(logits, dim=1)
acc = accuracy(preds, y)
self.log("val_loss", loss, prog_bar=True)
self.log("val_acc", acc, prog_bar=True)
```

Lightning Logs: Progress bar

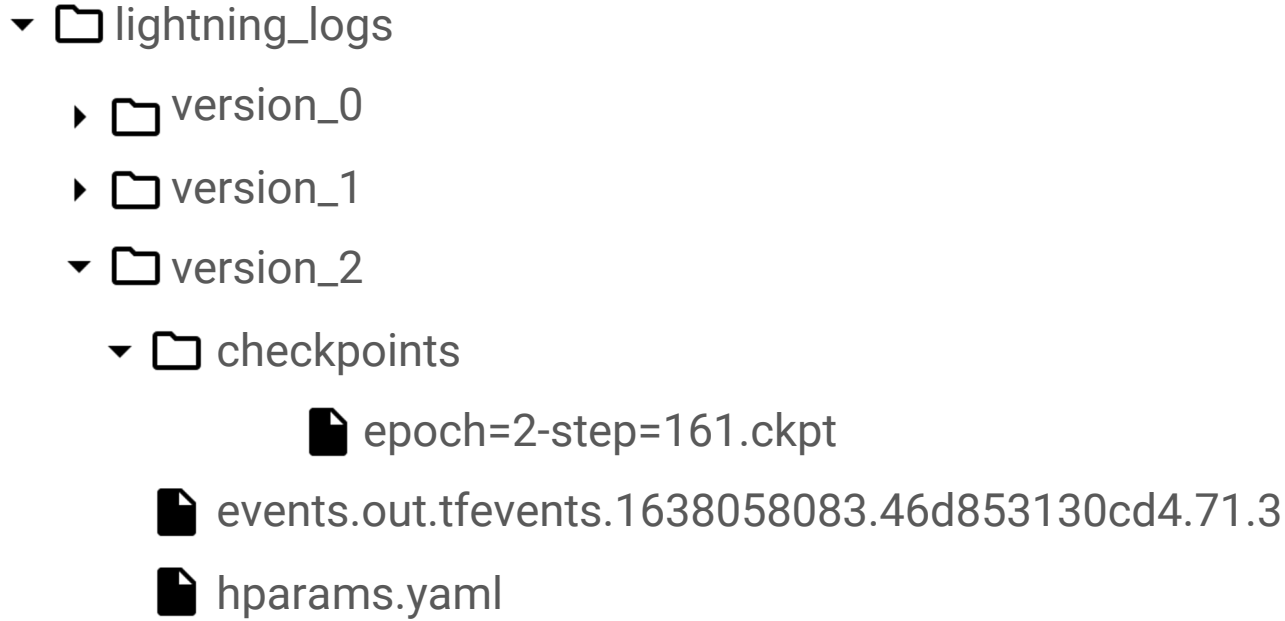
```
preds = torch.argmax(logits, dim=1)
acc = accuracy(preds, y)
self.log("val_loss", loss, prog_bar=True)
self.log("val_acc", acc, prog_bar=True)
```

```
trainer = Trainer(max_epochs=3, gpus=[0])# Init trainer specifying which GPU
trainer.fit(model, dm)
```

```
--------------------------------------------------------------------------------
(returns)
```

Epoch 2: 100%  59/59 [00:07<00:00, 7.51it/s, loss=0.558, v_num=3, val_loss=0.451, val_acc=0.883]

Lightning Logs: Model outputs



Lightning: Getting started

Via Colab:

```
! pip install pytorch_lightning
```

suppress long install print out:

```
%%capture
! pip install pytorch_lightning
```

Local install via Conda:

```
conda install -c conda-forge pytorch-lightning torchmetrics
conda install -c pytorch pytorch torchvision torchaudio
```

Demos

And now some demos...