Compile and Train a Binary Classification Trainer Model with the SST2 Dataset for Single-Node Single-GPU Training

SageMaker Training Compiler Overview

SageMaker Training Compiler is a capability of SageMaker that makes these hard-to-implement optimizations to reduce training time on GPU instances. The compiler optimizes DL models to accelerate training by more efficiently using SageMaker machine learning (ML) GPU instances. SageMaker Training Compiler is available at no additional charge within SageMaker and can help reduce total billable time as it accelerates training.

SageMaker Training Compiler is integrated into the AWS Deep Learning Containers (DLCs). Using the SageMaker Training Compiler enabled AWS DLCs, you can compile and optimize training jobs on GPU instances with minimal changes to your code. Bring your deep learning models to SageMaker and enable SageMaker Training Compiler to accelerate the speed of your training job on SageMaker ML instances for accelerated computing.

For more information, see [SageMaker Training Compiler (https://docs.aws.amazon.com/sagemaker/latest/dg/training-compiler.html)](https://docs.aws.amazon.com/sagemaker/latest/dg/training-compiler.html) in the *Amazon SageMaker Developer Guide*.

Introduction

In this demo, you'll use Hugging Face's `transformers` and `datasets` libraries with Amazon SageMaker Training Compiler to train the `RoBERTa` model on the `Stanford Sentiment Treebank v2 (SST2)` dataset. To get started, we need to set up the environment with a few prerequisite steps, for permissions, configurations, and so on.

NOTE: You can run this demo in SageMaker Studio, SageMaker notebook instances, or your local machine with AWS CLI set up. If using SageMaker Studio or SageMaker notebook instances, make sure you choose one of the PyTorch-based kernels, `Python 3 (PyTorch x.y Python 3.x CPU Optimized)` or `conda_pytorch_p36` respectively.

NOTE: This notebook uses two `ml.p3.2xlarge` instances that have single GPU. If you don't have enough quota, see [Request a service quota increase for SageMaker resources (https://docs.aws.amazon.com/sagemaker/latest/dg/regions-quotas.html#service-limit-increase-request-procedure)](https://docs.aws.amazon.com/sagemaker/latest/dg/regions-quotas.html#service-limit-increase-request-procedure).

Development Environment

Installation

This example notebook requires the **SageMaker Python SDK v2.108.0** and **transformers v4.21**.

In [1]:
```
!pip install "sagemaker>=2.108.0" botocore boto3 awscli "torch==1.1
1.0" --upgrade
```

```
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuro
n.amazonaws.com
Requirement already satisfied: sagemaker>=2.108.0 in /home/ec2-user/
anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (2.109.0)
Requirement already satisfied: botocore in /home/ec2-user/anaconda3/
envs/pytorch_p38/lib/python3.8/site-packages (1.27.72)
Requirement already satisfied: boto3 in /home/ec2-user/anaconda3/env
s/pytorch_p38/lib/python3.8/site-packages (1.24.72)
Requirement already satisfied: awscli in /home/ec2-user/anaconda3/en
vs/pytorch_p38/lib/python3.8/site-packages (1.25.73)
Requirement already satisfied: torch==1.11.0 in /home/ec2-user/anaco
nda3/envs/pytorch_p38/lib/python3.8/site-packages (1.11.0)
Requirement already satisfied: typing-extensions in /home/ec2-user/a
naconda3/envs/pytorch_p38/lib/python3.8/site-packages (from torch==
1.11.0) (4.3.0)
Requirement already satisfied: pathos in /home/ec2-user/anaconda3/en
vs/pytorch_p38/lib/python3.8/site-packages (from sagemaker>=2.108.0)
(0.2.8)
Requirement already satisfied: numpy<2.0,>=1.9.0 in /home/ec2-user/a
naconda3/envs/pytorch_p38/lib/python3.8/site-packages (from sagemake
r>=2.108.0) (1.21.2)
Requirement already satisfied: protobuf<4.0,>=3.1 in /home/ec2-user/
anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from sagemak
er>=2.108.0) (3.20.1)
Requirement already satisfied: google-pasta in /home/ec2-user/anacon
da3/envs/pytorch_p38/lib/python3.8/site-packages (from sagemaker>=2.
108.0) (0.2.0)
Requirement already satisfied: attrs<22,>=20.3.0 in /home/ec2-user/a
naconda3/envs/pytorch_p38/lib/python3.8/site-packages (from sagemake
r>=2.108.0) (21.2.0)
Requirement already satisfied: protobuf3-to-dict<1.0,>=0.1.5 in /hom
e/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (f
rom sagemaker>=2.108.0) (0.1.5)
Requirement already satisfied: packaging>=20.0 in /home/ec2-user/ana
conda3/envs/pytorch_p38/lib/python3.8/site-packages (from sagemaker>
=2.108.0) (21.3)
Requirement already satisfied: pandas in /home/ec2-user/anaconda3/en
vs/pytorch_p38/lib/python3.8/site-packages (from sagemaker>=2.108.0)
(1.3.4)
Requirement already satisfied: importlib-metadata<5.0,>=1.4.0 in /ho
me/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages
(from sagemaker>=2.108.0) (4.8.2)
Requirement already satisfied: smdebug-rulesconfig==1.0.1 in /home/e
c2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from
sagemaker>=2.108.0) (1.0.1)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /home/ec2-us
er/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from boto
core) (1.26.8)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /home/
ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (fro
m botocore) (2.8.2)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /home/ec2-u
ser/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from bot
ocore) (0.10.0)
```

```
Requirement already satisfied: s3transfer<0.7.0,>=0.6.0 in /home/ec2
-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from b
oto3) (0.6.0)
Requirement already satisfied: colorama<0.4.5,>=0.2.5 in /home/ec2-u
ser/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from aws
cli) (0.4.3)
Requirement already satisfied: docutils<0.17,>=0.10 in /home/ec2-use
r/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from awscl
i) (0.15.2)
Requirement already satisfied: PyYAML<5.5,>=3.10 in /home/ec2-user/a
naconda3/envs/pytorch_p38/lib/python3.8/site-packages (from awscli)
(5.4.1)
Requirement already satisfied: rsa<4.8,>=3.1.2 in /home/ec2-user/ana
conda3/envs/pytorch_p38/lib/python3.8/site-packages (from awscli)
(4.7.2)
Requirement already satisfied: zipp>=0.5 in /home/ec2-user/anaconda3
/envs/pytorch_p38/lib/python3.8/site-packages (from importlib-metada
ta<5.0,>=1.4.0->sagemaker>=2.108.0) (3.6.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /home/ec2
-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from p
ackaging>=20.0->sagemaker>=2.108.0) (3.0.6)
Requirement already satisfied: six in /home/ec2-user/anaconda3/envs/
pytorch_p38/lib/python3.8/site-packages (from protobuf3-to-dict<1.0,
>=0.1.5->sagemaker>=2.108.0) (1.16.0)
Requirement already satisfied: pyasn1>=0.1.3 in /home/ec2-user/anaco
nda3/envs/pytorch_p38/lib/python3.8/site-packages (from rsa<4.8,>=3.
1.2->awscli) (0.4.8)
Requirement already satisfied: pytz>=2017.3 in /home/ec2-user/anacon
da3/envs/pytorch_p38/lib/python3.8/site-packages (from pandas->sagem
aker>=2.108.0) (2021.3)
Requirement already satisfied: ppft>=1.6.6.4 in /home/ec2-user/anaco
nda3/envs/pytorch_p38/lib/python3.8/site-packages (from pathos->sage
maker>=2.108.0) (1.6.6.4)
Requirement already satisfied: pox>=0.3.0 in /home/ec2-user/anaconda
3/envs/pytorch_p38/lib/python3.8/site-packages (from pathos->sagemak
er>=2.108.0) (0.3.0)
Requirement already satisfied: multiprocess>=0.70.12 in /home/ec2-us
er/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from path
os->sagemaker>=2.108.0) (0.70.12.2)
Requirement already satisfied: dill>=0.3.4 in /home/ec2-user/anacond
a3/envs/pytorch_p38/lib/python3.8/site-packages (from pathos->sagema
ker>=2.108.0) (0.3.4)
WARNING: You are using pip version 22.0.4; however, version 22.2.2 i
s available.
You should consider upgrading via the '/home/ec2-user/anaconda3/envs
/pytorch_p38/bin/python -m pip install --upgrade pip' command.
```

In [2]:
```
!pip install -U "transformers==4.21.1" datasets --upgrade
```

```
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuro
n.amazonaws.com
Collecting transformers==4.21.1
  Using cached transformers-4.21.1-py3-none-any.whl (4.7 MB)
Requirement already satisfied: datasets in /home/ec2-user/anaconda3/
envs/pytorch_p38/lib/python3.8/site-packages (2.4.0)
Requirement already satisfied: tqdm>=4.27 in /home/ec2-user/anaconda
3/envs/pytorch_p38/lib/python3.8/site-packages (from transformers==
4.21.1) (4.62.3)
Requirement already satisfied: requests in /home/ec2-user/anaconda3/
envs/pytorch_p38/lib/python3.8/site-packages (from transformers==4.2
1.1) (2.26.0)
Requirement already satisfied: regex!=2019.12.17 in /home/ec2-user/a
naconda3/envs/pytorch_p38/lib/python3.8/site-packages (from transfor
mers==4.21.1) (2021.11.10)
Requirement already satisfied: packaging>=20.0 in /home/ec2-user/ana
conda3/envs/pytorch_p38/lib/python3.8/site-packages (from transforme
rs==4.21.1) (21.3)
Requirement already satisfied: tokenizers!=0.11.3,<0.13,>=0.11.1 in
/home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-package
s (from transformers==4.21.1) (0.12.1)
Requirement already satisfied: numpy>=1.17 in /home/ec2-user/anacond
a3/envs/pytorch_p38/lib/python3.8/site-packages (from transformers==
4.21.1) (1.21.2)
Requirement already satisfied: filelock in /home/ec2-user/anaconda3/
envs/pytorch_p38/lib/python3.8/site-packages (from transformers==4.2
1.1) (3.4.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.1.0 in /home/
ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (fro
m transformers==4.21.1) (0.9.0)
Requirement already satisfied: pyyaml>=5.1 in /home/ec2-user/anacond
a3/envs/pytorch_p38/lib/python3.8/site-packages (from transformers==
4.21.1) (5.4.1)
Requirement already satisfied: xxhash in /home/ec2-user/anaconda3/en
vs/pytorch_p38/lib/python3.8/site-packages (from datasets) (3.0.0)
Requirement already satisfied: fsspec[http]>=2021.11.1 in /home/ec2-
user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from da
tasets) (2021.11.1)
Requirement already satisfied: pandas in /home/ec2-user/anaconda3/en
vs/pytorch_p38/lib/python3.8/site-packages (from datasets) (1.3.4)
Requirement already satisfied: pyarrow>=6.0.0 in /home/ec2-user/anac
onda3/envs/pytorch_p38/lib/python3.8/site-packages (from datasets)
(7.0.0)
Requirement already satisfied: aiohttp in /home/ec2-user/anaconda3/e
nvs/pytorch_p38/lib/python3.8/site-packages (from datasets) (3.8.1)
Requirement already satisfied: dill<0.3.6 in /home/ec2-user/anaconda
3/envs/pytorch_p38/lib/python3.8/site-packages (from datasets) (0.3.
4)
Requirement already satisfied: multiprocess in /home/ec2-user/anacon
da3/envs/pytorch_p38/lib/python3.8/site-packages (from datasets) (0.
70.12.2)
Requirement already satisfied: responses<0.19 in /home/ec2-user/anac
onda3/envs/pytorch_p38/lib/python3.8/site-packages (from datasets)
(0.18.0)
```

```
Requirement already satisfied: typing-extensions>=3.7.4.3 in /home/e
c2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from
huggingface-hub<1.0,>=0.1.0->transformers==4.21.1) (4.3.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /home/ec2
-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from p
ackaging>=20.0->transformers==4.21.1) (3.0.6)
Requirement already satisfied: charset-normalizer~=2.0.0 in /home/ec
2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from
requests->transformers==4.21.1) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /home/ec2-user/
anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from request
s->transformers==4.21.1) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in /home/ec2-user/anacon
da3/envs/pytorch_p38/lib/python3.8/site-packages (from requests->tra
nsformers==4.21.1) (3.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/ec2-us
er/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from requ
ests->transformers==4.21.1) (1.26.8)
Requirement already satisfied: frozenlist>=1.1.1 in /home/ec2-user/a
naconda3/envs/pytorch_p38/lib/python3.8/site-packages (from aiohttp-
>datasets) (1.2.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /home/ec2-user
/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from aiohtt
p->datasets) (5.2.0)
Requirement already satisfied: aiosignal>=1.1.2 in /home/ec2-user/an
aconda3/envs/pytorch_p38/lib/python3.8/site-packages (from aiohttp->
datasets) (1.2.0)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /home/
ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (fro
m aiohttp->datasets) (4.0.1)
Requirement already satisfied: attrs>=17.3.0 in /home/ec2-user/anaco
nda3/envs/pytorch_p38/lib/python3.8/site-packages (from aiohttp->dat
asets) (21.2.0)
Requirement already satisfied: yarl<2.0,>=1.0 in /home/ec2-user/anac
onda3/envs/pytorch_p38/lib/python3.8/site-packages (from aiohttp->da
tasets) (1.7.2)
Requirement already satisfied: python-dateutil>=2.7.3 in /home/ec2-u
ser/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from pan
das->datasets) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /home/ec2-user/anacon
da3/envs/pytorch_p38/lib/python3.8/site-packages (from pandas->datas
ets) (2021.3)
Requirement already satisfied: six>=1.5 in /home/ec2-user/anaconda3/
envs/pytorch_p38/lib/python3.8/site-packages (from python-dateutil>=
2.7.3->pandas->datasets) (1.16.0)
Installing collected packages: transformers
  Attempting uninstall: transformers
    Found existing installation: transformers 4.21.0
    Uninstalling transformers-4.21.0:
      Successfully uninstalled transformers-4.21.0
Successfully installed transformers-4.21.1
WARNING: You are using pip version 22.0.4; however, version 22.2.2 i
s available.
You should consider upgrading via the '/home/ec2-user/anaconda3/envs
```

In [3]:
```
import botocore
import boto3
import sagemaker
import transformers
import pandas as pd

print(f"sagemaker: {sagemaker.__version__}")
print(f"transformers: {transformers.__version__}")
```

```
sagemaker: 2.109.0
transformers: 4.21.1
```

Copy and run the following code if you need to upgrade `ipywidgets` for `datasets` library and restart kernel. This is only needed when prepocessing is done in the notebook.

```
%%capture
import IPython
!conda install -c conda-forge ipywidgets -y
# has to restart kernel for the updates to be applied
IPython.Application.instance().kernel.do_shutdown(True)
```

SageMaker environment

In [4]:
```
import sagemaker

sess = sagemaker.Session()

# SageMaker session bucket -> used for uploading data, models and l
ogs
# SageMaker will automatically create this bucket if it does not ex
ist
sagemaker_session_bucket = None
if sagemaker_session_bucket is None and sess is not None:
    # set to default bucket if a bucket name is not given
    sagemaker_session_bucket = sess.default_bucket()

role = sagemaker.get_execution_role()
sess = sagemaker.Session(default_bucket=sagemaker_session_bucket)

print(f"sagemaker role arn: {role}")
print(f"sagemaker bucket: {sess.default_bucket()}")
print(f"sagemaker session region: {sess.boto_region_name}")
```

```
sagemaker role arn: arn:aws:iam::875423407011:role/AdminRole
sagemaker bucket: sagemaker-us-west-2-875423407011
sagemaker session region: us-west-2
```

Loading the SST dataset

When using the 🤗 [Datasets library (https://github.com/huggingface/datasets)](https://github.com/huggingface/datasets), datasets can be downloaded directly with the following `datasets.load_dataset()` method:

```
from datasets import load_dataset
load_dataset('dataset_name')
```

If you'd like to try other training datasets later, you can simply use this method.

For this example notebook, we prepared the `SST2` dataset in the public SageMaker sample file S3 bucket. The following code cells show how you can directly load the dataset and convert to a `HuggingFace DatasetDict`.

Preprocessing

We download and preprocess the `SST2` dataset from the `s3://sagemaker-sample-files/datasets` bucket. After preprocessing, we'll upload the dataset to the `sagemaker_session_bucket`, which will be used as a data channel for the training job.

Tokenization

In [5]:
```
from datasets import Dataset
from transformers import AutoTokenizer
import pandas as pd

# tokenizer used in preprocessing
tokenizer_name = "roberta-base"

# s3 key prefix for the data
s3_prefix = "samples/datasets/sst2"

# Download the SST2 data from s3
!curl https://sagemaker-sample-files.s3.amazonaws.com/datasets/text
/SST2/sst2.test > ./sst2.test
!curl https://sagemaker-sample-files.s3.amazonaws.com/datasets/text
/SST2/sst2.train > ./sst2.train
!curl https://sagemaker-sample-files.s3.amazonaws.com/datasets/text
/SST2/sst2.val > ./sst2.val
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Ti
me  Current
                                 Dload  Upload   Total   Spent    Le
ft  Speed
100  189k  100  189k    0     0   404k      0 --:--:-- --:--:--
--:--:--  404k
  % Total    % Received % Xferd  Average Speed   Time    Time     Ti
me  Current
                                 Dload  Upload   Total   Spent    Le
ft  Speed
100 3716k  100 3716k    0     0  4548k      0 --:--:-- --:--:--
--:--:-- 4543k
  % Total    % Received % Xferd  Average Speed   Time    Time     Ti
me  Current
                                 Dload  Upload   Total   Spent    Le
ft  Speed
100 94916  100 94916    0     0   217k      0 --:--:-- --:--:--
--:--:--  218k
```

In [6]:
```
# download tokenizer
tokenizer = AutoTokenizer.from_pretrained(tokenizer_name)

# tokenizer helper function
def tokenize(batch):
    return tokenizer(batch["text"], padding="max_length", truncatio
n=True)


# load dataset
test_df = pd.read_csv("sst2.test", sep="delimiter", header=None, en
gine="python", names=["line"])
train_df = pd.read_csv("sst2.train", sep="delimiter", header=None,
engine="python", names=["line"])

test_df[["label", "text"]] = test_df["line"].str.split(" ", 1, expa
nd=True)
train_df[["label", "text"]] = train_df["line"].str.split(" ", 1, ex
pand=True)

test_df.drop("line", axis=1, inplace=True)
train_df.drop("line", axis=1, inplace=True)

test_df["label"] = pd.to_numeric(test_df["label"], downcast="intege
r")
train_df["label"] = pd.to_numeric(train_df["label"], downcast="inte
ger")

train_dataset = Dataset.from_pandas(train_df)
test_dataset = Dataset.from_pandas(test_df)

# tokenize dataset
train_dataset = train_dataset.map(tokenize, batched=True)
test_dataset = test_dataset.map(tokenize, batched=True)

# set format for pytorch
train_dataset = train_dataset.rename_column("label", "labels")
train_dataset.set_format("torch", columns=["input_ids", "attention_
mask", "labels"])

test_dataset = test_dataset.rename_column("label", "labels")
test_dataset.set_format("torch", columns=["input_ids", "attention_m
ask", "labels"])
```

Uploading data to `sagemaker_session_bucket`

We are going to use the new `FileSystem` [integration (https://huggingface.co/docs/datasets/filesystems.html)](https://huggingface.co/docs/datasets/filesystems.html) to upload our preprocessed dataset to S3.

In [7]:
```
import botocore
from datasets.filesystems import S3FileSystem

s3 = S3FileSystem()

# save train_dataset to s3
training_input_path = f"s3://{sess.default_bucket()}/{s3_prefix}/tr
ain"
train_dataset.save_to_disk(training_input_path, fs=s3)

# save test_dataset to s3
test_input_path = f"s3://{sess.default_bucket()}/{s3_prefix}/test"
test_dataset.save_to_disk(test_input_path, fs=s3)
```

SageMaker Training Job

To create a SageMaker training job, we use a `HuggingFace/PyTorch` estimator. Using the estimator, you can define which fine-tuning script should SageMaker use through `entry_point`, which `instance_type` to use for training, which `hyperparameters` to pass, and so on.

When a SageMaker training job starts, SageMaker takes care of starting and managing all the required machine learning instances, picks up the `HuggingFace` Deep Learning Container, uploads your training script, and downloads the data from `sagemaker_session_bucket` into the container at `/opt/ml/input/data`.

In the following section, you learn how to set up two versions of the SageMaker `HuggingFace/PyTorch` estimator, a native one without the compiler and an optimized one with the compiler.

Training Setup

Set up an option for fine-tuning or full training. `FINE_TUNING = 1` is for fine-tuning and it will use `fine_tune_with_huggingface.py`. `FINE_TUNING = 0` is for full training and it will use `full_train_roberta_with_huggingface.py`.

In [8]:
```
# Here we configure the training job. Please configure the appropri
ate options below:

# Fine tuning trains a pre-trained model on a different dataset whe
reas full training trains the model from scratch.
FINE_TUNING = 1
FULL_TRAINING = not FINE_TUNING

# Fine tuning is typically faster and is done for fewer epochs
EPOCHS = 7 if FINE_TUNING else 100

TRAINING_SCRIPT = (
    "fine_tune_with_huggingface.py" if FINE_TUNING else "full_train
_roberta_with_huggingface.py"
)

# SageMaker Training Compiler currently only supports training on G
PU
# Select Instance type for training
INSTANCE_TYPE = "ml.p3.2xlarge"
```

Training with Native PyTorch

The `train_batch_size` in the following code cell is the maximum batch that can fit into the memory of the `ml.p3.2xlarge` instance. If you change the model, instance type, sequence length, and other parameters, you need to do some experiments to find the largest batch size that will fit into GPU memory.

In [9]:
```
from sagemaker.pytorch import PyTorch

# hyperparameters, which are passed into the training job
hyperparameters = {"epochs": EPOCHS, "train_batch_size": 18, "model
_name": "roberta-base"}
# The original LR was set for a batch of 32. Here we are scaling le
arning rate with batch size.
hyperparameters["learning_rate"] = float("5e-5") / 32 * hyperparame
ters["train_batch_size"]

# If checkpointing is enabled with higher epoch numbers, your disk
requirements will increase as well
volume_size = 60 + 2 * hyperparameters["epochs"]
```

In [10]:
```
# configure the training job
native_estimator = PyTorch(
    entry_point=TRAINING_SCRIPT,
    source_dir="./scripts",
    instance_type=INSTANCE_TYPE,
    instance_count=1,
    role=role,
    py_version="py38",
    transformers_version="4.21.1",
    framework_version="1.11.0",
    volume_size=volume_size,
    hyperparameters=hyperparameters,
    disable_profiler=True,
    debugger_hook_config=False,
)

# start training with our uploaded datasets as input
native_estimator.fit({"train": training_input_path, "test": test_in
put_path}, wait=False)

# The name of the training job.
native_estimator.latest_training_job.name
```

Out[10]:
```
'pytorch-training-2022-09-13-23-14-37-376'
```

Training with Optimized PyTorch

Compilation through Training Compiler changes the memory footprint of the model. Most commonly, this manifests as a reduction in memory utilization and a consequent increase in the largest batch size that can fit on the GPU. Note that if you want to change the batch size, you must adjust the learning rate appropriately.

Note: We recommend you to turn the SageMaker Debugger's profiling and debugging tools off when you use compilation to avoid additional overheads.

In [11]:
```
# With SageMaker Training Compiler enabled we are able to fit a lar
ger batch into memory.
hyperparameters["train_batch_size"] = 24
# The original LR was set for a batch of 32. Here we are scaling le
arning rate with batch size.
hyperparameters["learning_rate"] = float("5e-5") / 32 * hyperparame
ters["train_batch_size"]

# If checkpointing is enabled with higher epoch numbers, your disk
requirements will increase as well
volume_size = 60 + 2 * hyperparameters["epochs"]

from sagemaker.huggingface import HuggingFace, TrainingCompilerConf
ig
```

In [12]:
```
# configure the training job
optimized_estimator = HuggingFace(
    entry_point=TRAINING_SCRIPT,
    compiler_config=TrainingCompilerConfig(),
    source_dir="./scripts",
    instance_type=INSTANCE_TYPE,
    instance_count=1,
    role=role,
    py_version="py38",
    transformers_version="4.21.1",
    pytorch_version="1.11.0",
    volume_size=volume_size,
    hyperparameters=hyperparameters,
    disable_profiler=True,
    debugger_hook_config=False,
)

# start training with our uploaded datasets as input
optimized_estimator.fit({"train": training_input_path, "test": test
_input_path}, wait=False)

# The name of the training job
optimized_estimator.latest_training_job.name
```

Out[12]:
```
'huggingface-pytorch-trcomp-training-2022-09-13-23-14-38-165'
```

Wait for training jobs to complete

In [29]:
```
waiter = native_estimator.sagemaker_session.sagemaker_client.get_wa
iter(
    "training_job_completed_or_stopped"
)
waiter.wait(TrainingJobName=native_estimator.latest_training_job.na
me)
waiter = optimized_estimator.sagemaker_session.sagemaker_client.get
_waiter(
    "training_job_completed_or_stopped"
)
waiter.wait(TrainingJobName=optimized_estimator.latest_training_jo
b.name)
```

Analysis

Load information and logs of the training job *without* SageMaker Training Compiler

In [30]:
```
# container image used for native training job
print(f"container image used for training job: \n{native_estimator.
image_uri}\n")

# s3 uri where the native trained model is located
print(f"s3 uri where the trained model is located: \n{native_estima
tor.model_data}\n")

# latest training job name for this estimator
print(
    f"latest training job name for this estimator: \n{native_estima
tor.latest_training_job.name}\n"
)
```

```
container image used for training job:
None

s3 uri where the trained model is located:
s3://sagemaker-us-west-2-875423407011/pytorch-training-2022-09-13-23
-14-37-376/output/model.tar.gz

latest training job name for this estimator:
pytorch-training-2022-09-13-23-14-37-376
```

In [31]:
```
%%capture native

# access the logs of the native training job
native_estimator.sagemaker_session.logs_for_job(native_estimator.la
test_training_job.name)
```

Note: If the estimator object is no longer available due to a kernel break or refresh, you need to directly use the training job name and manually attach the training job to a new HuggingFace estimator. For example:

```
huggingface_estimator = HuggingFace.attach("your_huggingface_training_job_
name")
```

Load information and logs of the training job *with* SageMaker Training Compiler

In [32]:
```
# container image used for optimized training job
print(f"container image used for training job: \n{optimized_estimat
or.image_uri}\n")

# s3 uri where the optimized trained model is located
print(f"s3 uri where the trained model is located: \n{optimized_est
imator.model_data}\n")

# latest training job name for this estimator
print(
    f"latest training job name for this estimator: \n{optimized_est
imator.latest_training_job.name}\n"
)
```

```
container image used for training job:
None

s3 uri where the trained model is located:
s3://sagemaker-us-west-2-875423407011/huggingface-pytorch-trcomp-tra
ining-2022-09-13-23-14-38-165/output/model.tar.gz

latest training job name for this estimator:
huggingface-pytorch-trcomp-training-2022-09-13-23-14-38-165
```

In [33]:
```
%%capture optimized

# access the logs of the optimized training job
optimized_estimator.sagemaker_session.logs_for_job(optimized_estima
tor.latest_training_job.name)
```

Note: If the estimator object is no longer available due to a kernel break or refresh, you need to directly use the training job name and manually attach the training job to a new HuggingFace estimator. For example:

```
optimized_estimator = HuggingFace.attach("your_compiled_huggingface_traini
ng_job_name")
```

Create helper functions for analysis

In [34]:
```
from ast import literal_eval
from collections import defaultdict
from matplotlib import pyplot as plt


def _summarize(captured):
    final = []
    for line in captured.stdout.split("\n"):
        cleaned = line.strip()
        if "{" in cleaned and "}" in cleaned:
            final.append(cleaned[cleaned.index("{") : cleaned.index
("}") + 1])
    return final


def make_sense(string):
    try:
        return literal_eval(string)
    except:
        pass


def summarize(summary):
    final = {"train": [], "eval": [], "summary": {}}
    for line in summary:
        interpretation = make_sense(line)
        if interpretation:
            if "loss" in interpretation:
                final["train"].append(interpretation)
            elif "eval_loss" in interpretation:
                final["eval"].append(interpretation)
            elif "train_runtime" in interpretation:
                final["summary"].update(interpretation)
    return final
```

Plot and compare throughput of compiled training and native training

Visualize average throughput as reported by HuggingFace and see potential savings.

In [35]:
```
# collect the average throughput as reported by HF for the native t
raining job
n = summarize(_summarize(native))
native_throughput = n["summary"]["train_samples_per_second"]

# collect the average throughput as reported by HF for the SageMake
r Training Compiler enhanced training job
o = summarize(_summarize(optimized))
optimized_throughput = o["summary"]["train_samples_per_second"]

# Calculate speedup from SageMaker Training Compiler
avg_speedup = f"{round((optimized_throughput/native_throughput-1)*1
00)}%"
```

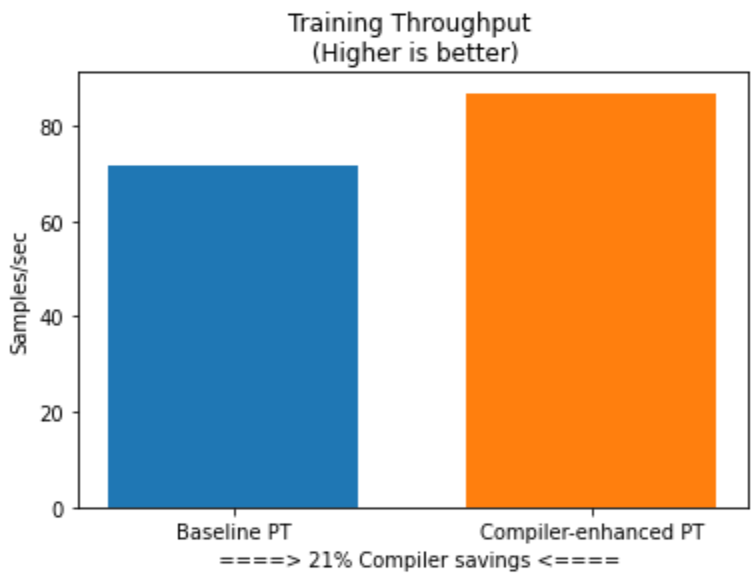
In [36]:
```
%matplotlib inline

plt.title("Training Throughput \n (Higher is better)")
plt.ylabel("Samples/sec")

plt.bar(x=[1], height=native_throughput, label="Baseline PT", width
=0.35)
plt.bar(x=[1.5], height=optimized_throughput, label="Compiler-enhan
ced PT", width=0.35)

plt.xlabel("  ====> {} Compiler savings <====".format(avg_speedup))
plt.xticks(ticks=[1, 1.5], labels=["Baseline PT", "Compiler-enhance
d PT"])
```

Out[36]:
```
([<matplotlib.axis.XTick at 0x7f78f74e9070>,
  <matplotlib.axis.XTick at 0x7f78f74e9b50>],
 [Text(1.0, 0, 'Baseline PT'), Text(1.5, 0, 'Compiler-enhanced PT
')])
```



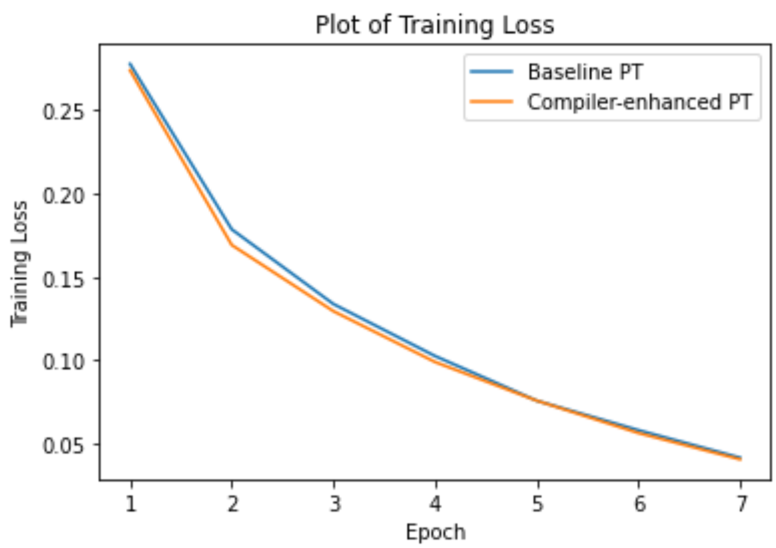
Convergence of Training Loss

SageMaker Training Compiler does not affect the model convergence behavior. Here, we see the decrease in training loss is similar with and without SageMaker Training Compiler

In [37]:
```
vanilla_loss = [i["loss"] for i in n["train"]]
vanilla_epochs = [i["epoch"] for i in n["train"]]
optimized_loss = [i["loss"] for i in o["train"]]
optimized_epochs = [i["epoch"] for i in o["train"]]

plt.title("Plot of Training Loss")
plt.xlabel("Epoch")
plt.ylabel("Training Loss")
plt.plot(vanilla_epochs, vanilla_loss, label="Baseline PT")
plt.plot(optimized_epochs, optimized_loss, label="Compiler-enhanced
PT")
plt.legend()
```

Out[37]:
```
<matplotlib.legend.Legend at 0x7f78f6a30e80>
```



Evaluation Stats

SageMaker Training Compiler does not affect the quality of the model. Here, we compare the evaluation metrics of the models trained with and without SageMaker Training Compiler to verify the same.

In [38]:
```
import pandas as pd

table = pd.DataFrame([n["eval"][-1], o["eval"][-1]], index=["Baseli
ne PT", "Compiler-enhanced PT"])
table.drop(columns=["eval_runtime", "eval_samples_per_second", "epo
ch"])
```

Out[38]:

	eval_loss	eval_accuracy	eval_f1	eval_precision	eval_recall	eval_steps_per_seco
Baseline PT	0.304888	0.938058	0.940032	0.910995	0.970982	3.
Compiler-enhanced PT	0.258519	0.952009	0.952695	0.939262	0.966518	1.

Training Stats

Let's compare various training metrics with and without SageMaker Training Compiler. SageMaker Training Compiler provides an increase in training throughput which translates to a decrease in total training time.

In [39]:
```
pd.DataFrame([n["summary"], o["summary"]], index=["Native", "Optimi
zed"])
```

Out[39]:

	train_runtime	train_samples_per_second	train_steps_per_second	train_loss	epoch
Native	6581.2434	71.634	3.979	0.124132	7.0
Optimized	5424.6973	86.907	3.621	0.120752	7.0

In [40]:
```
# calculate percentage speedup from SageMaker Training Compiler in
terms of total training time reported by HF

speedup = (
    (n["summary"]["train_runtime"] - o["summary"]["train_runtime"])
    * 100
    / n["summary"]["train_runtime"]
)
print(
    f"SageMaker Training Compiler integrated PyTorch is about {int
(speedup)}% faster in terms of total training time as reported by H
F."
)
```

```
SageMaker Training Compiler integrated PyTorch is about 17% faster i
n terms of total training time as reported by HF.
```

Total Billable Time

Finally, the decrease in total training time results in a decrease in the billable seconds from SageMaker

In [41]:
```
def BillableTimeInSeconds(name):
    describe_training_job = (
        optimized_estimator.sagemaker_session.sagemaker_client.desc
ribe_training_job
    )
    details = describe_training_job(TrainingJobName=name)
    return details["BillableTimeInSeconds"]
```

In [42]:
```
Billable = {}
Billable["Native"] = BillableTimeInSeconds(native_estimator.latest_
training_job.name)
Billable["Optimized"] = BillableTimeInSeconds(optimized_estimator.l
atest_training_job.name)
pd.DataFrame(Billable, index=["BillableSecs"])
```

Out[42]:

	Native	Optimized
BillableSecs	7048	6019

In [43]:
```
speedup = (Billable["Native"] - Billable["Optimized"]) * 100 / Bill
able["Native"]
print(f"SageMaker Training Compiler integrated PyTorch was {int(spe
edup)}% faster in summary.")
```

```
SageMaker Training Compiler integrated PyTorch was 14% faster in sum
mary.
```

Clean up

Stop all training jobs launched if the jobs are still running.

In [44]:
```
import boto3

sm = boto3.client("sagemaker")


def stop_training_job(name):
    status = sm.describe_training_job(TrainingJobName=name)["TrainingJobStatus"]
    if status == "InProgress":
        sm.stop_training_job(TrainingJobName=name)


stop_training_job(native_estimator.latest_training_job.name)
stop_training_job(optimized_estimator.latest_training_job.name)
```

Also, to find instructions on cleaning up resources, see [Clean Up (https://docs.aws.amazon.com/sagemaker/latest/dg/ex1-cleanup.html)](https://docs.aws.amazon.com/sagemaker/latest/dg/ex1-cleanup.html) in the *Amazon SageMaker Developer Guide*.

In [ ]:
```

```