Compile and Train a Hugging Face Transformers Trainer Model for Question and Answering with the `SQuAD` dataset

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

This example notebook demonstrates how to compile and fine-tune a question and answering NLP task. We use Hugging Face's `transformers` and `datasets` libraries with Amazon SageMaker Training Compiler to accelerate fine-tuning of a pre-trained transformer model on question and answering. In particular, the pre-trained model will be fine-tuned using the `SQuAD` dataset. To get started, we need to set up the environment with a few prerequisite steps to add 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).

Prepare SageMaker Environment and Permissions

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 s3fs typing
-extensions "torch==1.11.0" --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 (1.27.73)
Requirement already satisfied: boto3 in /home/ec2-user/anaconda3/env
s/pytorch_p38/lib/python3.8/site-packages (1.24.73)
Requirement already satisfied: awscli in /home/ec2-user/anaconda3/en
vs/pytorch_p38/lib/python3.8/site-packages (1.25.74)
Requirement already satisfied: s3fs in /home/ec2-user/anaconda3/envs
/pytorch_p38/lib/python3.8/site-packages (0.4.2)
Collecting s3fs
  Using cached s3fs-2022.8.2-py3-none-any.whl (27 kB)
Requirement already satisfied: typing-extensions in /home/ec2-user/a
naconda3/envs/pytorch_p38/lib/python3.8/site-packages (4.3.0)
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: 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: 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: 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: 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: 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: 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: 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: 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: 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: 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: 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: 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: 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: 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: 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: 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: 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: aiohttp!=4.0.0a0,!=4.0.0a1 in /home/e
c2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from
s3fs) (3.8.1)
Collecting fsspec==2022.8.2
  Using cached fsspec-2022.8.2-py3-none-any.whl (140 kB)
Collecting aiobotocore~=2.4.0
  Using cached aiobotocore-2.4.0-py3-none-any.whl (65 kB)
INFO: pip is looking at multiple versions of typing-extensions to de
termine which version is compatible with other requirements. This co
uld take a while.
Collecting typing-extensions
  Using cached typing_extensions-4.3.0-py3-none-any.whl (25 kB)
INFO: pip is looking at multiple versions of fsspec to determine whi
ch version is compatible with other requirements. This could take a
while.
INFO: pip is looking at multiple versions of <Python from Requires-P
ython> to determine which version is compatible with other requireme
nts. This could take a while.
INFO: pip is looking at multiple versions of s3fs to determine which
version is compatible with other requirements. This could take a whi
le.
Collecting s3fs
  Using cached s3fs-2022.8.1-py3-none-any.whl (27 kB)
Collecting fsspec==2022.8.1
  Using cached fsspec-2022.8.1-py3-none-any.whl (140 kB)
Collecting s3fs
  Using cached s3fs-2022.8.0-py3-none-any.whl (27 kB)
Collecting fsspec==2022.8.0
  Using cached fsspec-2022.8.0-py3-none-any.whl (140 kB)
Collecting s3fs
  Using cached s3fs-2022.7.1-py3-none-any.whl (27 kB)
Collecting fsspec==2022.7.1
  Using cached fsspec-2022.7.1-py3-none-any.whl (141 kB)
Collecting aiobotocore~=2.3.4
  Using cached aiobotocore-2.3.4-py3-none-any.whl (64 kB)
Requirement already satisfied: wrapt>=1.10.10 in /home/ec2-user/anac
```

```
onda3/envs/pytorch_p38/lib/python3.8/site-packages (from aiobotocore
~=2.3.4->s3fs) (1.13.3)
Requirement already satisfied: aioitertools>=0.5.1 in /home/ec2-user
/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from aiobot
ocore~=2.3.4->s3fs) (0.8.0)
Collecting s3fs
  Using cached s3fs-2022.7.0-py3-none-any.whl (27 kB)
Collecting fsspec==2022.7.0
  Using cached fsspec-2022.7.0-py3-none-any.whl (141 kB)
Collecting s3fs
  Using cached s3fs-2022.5.0-py3-none-any.whl (27 kB)
Collecting fsspec==2022.5.0
  Using cached fsspec-2022.5.0-py3-none-any.whl (140 kB)
Requirement already satisfied: aiobotocore~=2.3.0 in /home/ec2-user/
anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (from s3fs)
(2.3.3)
Collecting aiobotocore~=2.3.0
  Using cached aiobotocore-2.3.2.tar.gz (104 kB)
  Preparing metadata (setup.py) ... done
  Using cached aiobotocore-2.3.1.tar.gz (65 kB)
  Preparing metadata (setup.py) ... done
  Using cached aiobotocore-2.3.0.tar.gz (65 kB)
  Preparing metadata (setup.py) ... done
Collecting s3fs
  Using cached s3fs-2022.3.0-py3-none-any.whl (26 kB)
Collecting aiobotocore~=2.2.0
  Using cached aiobotocore-2.2.0.tar.gz (59 kB)
  Preparing metadata (setup.py) ... done
Collecting fsspec==2022.3.0
  Using cached fsspec-2022.3.0-py3-none-any.whl (136 kB)
Collecting s3fs
  Using cached s3fs-2022.2.0-py3-none-any.whl (26 kB)
Collecting fsspec==2022.02.0
  Using cached fsspec-2022.2.0-py3-none-any.whl (134 kB)
Collecting aiobotocore~=2.1.0
  Using cached aiobotocore-2.1.2.tar.gz (58 kB)
  Preparing metadata (setup.py) ... done
  Using cached aiobotocore-2.1.1.tar.gz (57 kB)
  Preparing metadata (setup.py) ... done
  Using cached aiobotocore-2.1.0.tar.gz (54 kB)
  Preparing metadata (setup.py) ... done
INFO: pip is looking at multiple versions of fsspec to determine whi
ch version is compatible with other requirements. This could take a
while.
INFO: pip is looking at multiple versions of <Python from Requires-P
ython> to determine which version is compatible with other requireme
nts. This could take a while.
INFO: pip is looking at multiple versions of s3fs to determine which
version is compatible with other requirements. This could take a whi
le.
Collecting s3fs
  Using cached s3fs-2022.1.0-py3-none-any.whl (25 kB)
Collecting fsspec==2022.01.0
  Using cached fsspec-2022.1.0-py3-none-any.whl (133 kB)
```

```
Collecting s3fs
  Using cached s3fs-2021.11.1-py3-none-any.whl (25 kB)
Requirement already satisfied: fsspec==2021.11.1 in /home/ec2-user/a
naconda3/envs/pytorch_p38/lib/python3.8/site-packages (from s3fs) (2
021.11.1)
Collecting aiobotocore~=2.0.1
  Using cached aiobotocore-2.0.1-py3-none-any.whl
Collecting fsspec==2021.11.1
  Using cached fsspec-2021.11.1-py3-none-any.whl (132 kB)
Collecting s3fs
  Using cached s3fs-2021.11.0-py3-none-any.whl (25 kB)
Collecting aiobotocore~=1.4.1
  Using cached aiobotocore-1.4.2.tar.gz (52 kB)
  Preparing metadata (setup.py) ... done
Collecting fsspec==2021.11.0
  Using cached fsspec-2021.11.0-py3-none-any.whl (132 kB)
Collecting aiobotocore~=1.4.1
  Using cached aiobotocore-1.4.1.tar.gz (52 kB)
  Preparing metadata (setup.py) ... done
Collecting s3fs
  Using cached s3fs-2021.10.1-py3-none-any.whl (26 kB)
Collecting fsspec==2021.10.1
  Using cached fsspec-2021.10.1-py3-none-any.whl (125 kB)
INFO: This is taking longer than usual. You might need to provide th
e dependency resolver with stricter constraints to reduce runtime. S
ee https://pip.pypa.io/warnings/backtracking for guidance. If you wa
nt to abort this run, press Ctrl + C.
Collecting s3fs
  Using cached s3fs-2021.10.0-py3-none-any.whl (26 kB)
Collecting fsspec==2021.10.0
  Using cached fsspec-2021.10.0-py3-none-any.whl (125 kB)
INFO: This is taking longer than usual. You might need to provide th
e dependency resolver with stricter constraints to reduce runtime. S
ee https://pip.pypa.io/warnings/backtracking for guidance. If you wa
nt to abort this run, press Ctrl + C.
INFO: This is taking longer than usual. You might need to provide th
e dependency resolver with stricter constraints to reduce runtime. S
ee https://pip.pypa.io/warnings/backtracking for guidance. If you wa
nt to abort this run, press Ctrl + C.
Collecting s3fs
  Using cached s3fs-2021.9.0-py3-none-any.whl (26 kB)
Collecting fsspec==2021.09.0
  Using cached fsspec-2021.9.0-py3-none-any.whl (123 kB)
Collecting s3fs
  Using cached s3fs-2021.8.1-py3-none-any.whl (26 kB)
Collecting fsspec==2021.08.1
  Using cached fsspec-2021.8.1-py3-none-any.whl (119 kB)
Collecting aiobotocore~=1.4.0
  Using cached aiobotocore-1.4.0.tar.gz (51 kB)
  Preparing metadata (setup.py) ... done
Collecting s3fs
  Using cached s3fs-2021.8.0-py3-none-any.whl (26 kB)
Collecting fsspec==2021.07.0
  Using cached fsspec-2021.7.0-py3-none-any.whl (118 kB)
```

```
Collecting s3fs
  Using cached s3fs-2021.7.0-py3-none-any.whl (25 kB)
Collecting aiobotocore>=1.0.1
  Using cached aiobotocore-2.0.0.tar.gz (52 kB)
  Preparing metadata (setup.py) ... done
  Using cached aiobotocore-1.3.3.tar.gz (50 kB)
  Preparing metadata (setup.py) ... done
  Using cached aiobotocore-1.3.2.tar.gz (49 kB)
  Preparing metadata (setup.py) ... done
  Using cached aiobotocore-1.3.1.tar.gz (48 kB)
  Preparing metadata (setup.py) ... done
  Using cached aiobotocore-1.3.0.tar.gz (48 kB)
  Preparing metadata (setup.py) ... done
  Using cached aiobotocore-1.2.2.tar.gz (48 kB)
  Preparing metadata (setup.py) ... done
  Using cached aiobotocore-1.2.1.tar.gz (48 kB)
  Preparing metadata (setup.py) ... done
  Using cached aiobotocore-1.2.0.tar.gz (47 kB)
  Preparing metadata (setup.py) ... done
  Using cached aiobotocore-1.1.2-py3-none-any.whl (45 kB)
  Using cached aiobotocore-1.1.1-py3-none-any.whl (45 kB)
  Using cached aiobotocore-1.1.0-py3-none-any.whl (43 kB)
  Using cached aiobotocore-1.0.7-py3-none-any.whl (42 kB)
  Using cached aiobotocore-1.0.6-py3-none-any.whl (42 kB)
  Using cached aiobotocore-1.0.5-py3-none-any.whl (42 kB)
  Using cached aiobotocore-1.0.4-py3-none-any.whl (41 kB)
  Using cached aiobotocore-1.0.3-py3-none-any.whl (40 kB)
  Using cached aiobotocore-1.0.2-py3-none-any.whl (40 kB)
  Using cached aiobotocore-1.0.1-py3-none-any.whl (40 kB)
Collecting s3fs
  Using cached s3fs-2021.6.1-py3-none-any.whl (25 kB)
Collecting fsspec==2021.06.1
  Using cached fsspec-2021.6.1-py3-none-any.whl (115 kB)
Collecting s3fs
  Using cached s3fs-2021.6.0-py3-none-any.whl (24 kB)
Collecting fsspec==2021.06.0
  Using cached fsspec-2021.6.0-py3-none-any.whl (114 kB)
Collecting s3fs
  Using cached s3fs-2021.5.0-py3-none-any.whl (24 kB)
Collecting fsspec==2021.05.0
  Using cached fsspec-2021.5.0-py3-none-any.whl (111 kB)
Collecting s3fs
  Using cached s3fs-2021.4.0-py3-none-any.whl (23 kB)
Collecting fsspec==2021.04.0
  Using cached fsspec-2021.4.0-py3-none-any.whl (108 kB)
Collecting s3fs
  Using cached s3fs-0.6.0-py3-none-any.whl (23 kB)
  Using cached s3fs-0.5.2-py3-none-any.whl (22 kB)
  Using cached s3fs-0.5.1-py3-none-any.whl (21 kB)
  Using cached s3fs-0.5.0-py3-none-any.whl (21 kB)
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
```

In [2]:
```
!pip install "transformers==4.21" datasets --upgrade
```

```
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuro
n.amazonaws.com
Requirement already satisfied: transformers==4.21 in /home/ec2-user/
anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (4.21.0)
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) (4.62.3)
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) (0.9.0)
Requirement already satisfied: requests in /home/ec2-user/anaconda3/
envs/pytorch_p38/lib/python3.8/site-packages (from transformers==4.2
1) (2.26.0)
Requirement already satisfied: filelock in /home/ec2-user/anaconda3/
envs/pytorch_p38/lib/python3.8/site-packages (from transformers==4.2
1) (3.4.0)
Requirement already satisfied: numpy>=1.17 in /home/ec2-user/anacond
a3/envs/pytorch_p38/lib/python3.8/site-packages (from transformers==
4.21) (1.21.2)
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) (2021.11.10)
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) (0.12.1)
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) (21.3)
Requirement already satisfied: pyyaml>=5.1 in /home/ec2-user/anacond
a3/envs/pytorch_p38/lib/python3.8/site-packages (from transformers==
4.21) (5.4.1)
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: 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: 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: 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: pandas in /home/ec2-user/anaconda3/en
vs/pytorch_p38/lib/python3.8/site-packages (from datasets) (1.3.4)
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: xxhash in /home/ec2-user/anaconda3/en
vs/pytorch_p38/lib/python3.8/site-packages (from datasets) (3.0.0)
```

```
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) (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) (3.0.6)
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) (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) (3.1)
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) (2.0.7)
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) (1.26.8)
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: 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: 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: 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: 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)
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 [3]:
```
import botocore
import boto3
import sagemaker
import transformers

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

```
sagemaker: 2.109.0
transformers: 4.21.0
```

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

Note: If you are going to use SageMaker in a local environment. You need access to an IAM Role with the required permissions for SageMaker. To learn more, see [SageMaker Roles (https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-roles.html)](https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-roles.html).

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
ists
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 `SQuAD` dataset

When using the 🤗 Datasets library (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 `SQuAD v1.1 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`.

NOTE: The `SQuAD` dataset (https://rajpurkar.github.io/SQuAD-explorer/) is under the CC BY-SA 4.0 license terms (https://creativecommons.org/licenses/by-sa/4.0/).

In [5]:
```
import pandas as pd
import numpy as np
import json
from datasets import Dataset
from datasets import DatasetDict
from datasets.filesystems import S3FileSystem
```

In [6]:
```
pd.__version__
```

Out[6]: '1.3.4'

In [7]:
```
# helper function to grab the dataset and load into DatasetDict
import urllib.request


def make_split(split):
    if split == "train":
        file = "https://sagemaker-sample-files.s3.amazonaws.com/dat
asets/text/squad/train-v1.1.json"
    elif split == "test":
        file = "https://sagemaker-sample-files.s3.amazonaws.com/dat
asets/text/squad/dev-v1.1.json"
    with urllib.request.urlopen(file) as f:
        squad = json.load(f)
        data = []
        for article in squad["data"]:
            title = article.get("title", "")
            for paragraph in article["paragraphs"]:
                context = paragraph["context"]  # do not strip lead
ing blank spaces GH-2585
                for qa in paragraph["qas"]:
                    answer_starts = [answer["answer_start"] for ans
wer in qa["answers"]]
                    answers = [answer["text"] for answer in qa["ans
wers"]]
                    # Features currently used are "context", "quest
ion", and "answers".
                    # Others are extracted here for the ease of fut
ure expansions.
                    data.append(
                        {
                            "title": title,
                            "context": context,
                            "question": qa["question"],
                            "id": qa["id"],
                            "answers": {
                                "answer_start": answer_starts,
                                "text": answers,
                            },
                        }
                    )
        df = pd.DataFrame(data)
        return Dataset.from_pandas(df)


train = make_split("train")
test = make_split("test")

datasets = DatasetDict()
datasets["train"] = train
datasets["validation"] = test
datasets
```

Out[7]:
```
DatasetDict({
    train: Dataset({
        features: ['title', 'context', 'question', 'id', 'answers'],
        num_rows: 87599
    })
    validation: Dataset({
        features: ['title', 'context', 'question', 'id', 'answers'],
        num_rows: 10570
    })
})
```

We will slice off 15,000 training samples and 1500 test samples.

In [8]:
```
datasets["train"] = datasets["train"].select(range(15000))
datasets["validation"] = datasets["validation"].select(range(1500))
```

The `datasets` object itself is `DatasetDict` (https://huggingface.co/docs/datasets/package_reference/main_classes.html#datasetdict), which contains one key for the training, validation and test set.

Preprocessing

Before we can feed those texts to the Trainer model, we need to preprocess them. This can be done by a 🤗 Transformers `Tokenizer` which (as the name indicates) tokenizes the input texts (including converting the tokens to their corresponding IDs in the pretrained vocabulary) and put them into a format the model expects, as well as generate other inputs that the model requires.

To do this, we instantiate a tokenizer using the `AutoTokenizer.from_pretrained` method, which will ensure that:

- We get a tokenizer that corresponds to the model architecture we want to use.
- We download the vocabulary used when pretraining this specific checkpoint.

That vocabulary will be cached, so it's not downloaded again when you run the cell.

In [9]:
```
model_checkpoint = "albert-base-v2"

from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
```

The following assertion ensures that our tokenizer is a fast tokenizer (backed by Rust) from the 🤗 Tokenizers library. Those fast tokenizers are available for almost all models, and we will need some of the special features they have for our preprocessing.

In [10]:
```
import transformers

assert isinstance(tokenizer, transformers.PreTrainedTokenizerFast)
```

You can check which type of models have a fast tokenizer available and which don't on the [big table of models (https://huggingface.co/transformers/index.html#bigtable)](https://huggingface.co/transformers/index.html#bigtable).

In [11]:
```
max_length = 384  # The maximum length of a feature (question and c
ontext)
doc_stride = (
    128  # The authorized overlap between two parts of the context
when splitting it is needed.
)
```

We need to add padding to the right which is specific to the model:

In [12]:
```
pad_on_right = tokenizer.padding_side == "right"
```

Now, let's put everything together in one function that we will apply to our training set. In the case of impossible answers (the answer is in another feature given by an example with a long context), we set the `cls` index for both the start and end position. We could also simply discard those examples from the training set if the flag `allow_impossible_answers` is `False`. Because the preprocessing is already complex enough as it is, we've kept is simple for this part.

In [13]:
```
def prepare_train_features(examples):
    # Some of the questions have lots of whitespace on the left, wh
ich is not useful and will make the
    # truncation of the context fail (the tokenized question will t
ake a lots of space). So we remove that
    # left whitespace
    examples["question"] = [q.lstrip() for q in examples["questio
n"]]

    # Tokenize our examples with truncation and padding, but keep t
he overflows using a stride. This results
    # in one example possibly giving several features when a contex
t is long, each of those features having a
    # context that overlaps a bit the context of the previous featu
re.
    tokenized_examples = tokenizer(
        examples["question" if pad_on_right else "context"],
        examples["context" if pad_on_right else "question"],
        truncation="only_second" if pad_on_right else "only_first",
        max_length=max_length,
        stride=doc_stride,
        return_overflowing_tokens=True,
        return_offsets_mapping=True,
        padding="max_length",
    )

    # Since one example might give us several features if it has a
long context, we need a map from a feature to
    # its corresponding example. This key gives us just that.
    sample_mapping = tokenized_examples.pop("overflow_to_sample_map
ping")
    # The offset mappings will give us a map from token to characte
r position in the original context. This will
    # help us compute the start_positions and end_positions.
    offset_mapping = tokenized_examples.pop("offset_mapping")

    # Let's label those examples!
    tokenized_examples["start_positions"] = []
    tokenized_examples["end_positions"] = []

    for i, offsets in enumerate(offset_mapping):
        # We will label impossible answers with the index of the CL
S token.
        input_ids = tokenized_examples["input_ids"][i]
        cls_index = input_ids.index(tokenizer.cls_token_id)

        # Grab the sequence corresponding to that example (to know
what is the context and what is the question).
        sequence_ids = tokenized_examples.sequence_ids(i)

        # One example can give several spans, this is the index of
the example containing this span of text.
        sample_index = sample_mapping[i]
        answers = examples["answers"][sample_index]
```

```
        # If no answers are given, set the cls_index as answer.
        if len(answers["answer_start"]) == 0:
            tokenized_examples["start_positions"].append(cls_index)
            tokenized_examples["end_positions"].append(cls_index)
        else:
            # Start/end character index of the answer in the text.
            start_char = answers["answer_start"][0]
            end_char = start_char + len(answers["text"][0])

            # Start token index of the current span in the text.
            token_start_index = 0
            while sequence_ids[token_start_index] != (1 if pad_on_r
ight else 0):
                token_start_index += 1

            # End token index of the current span in the text.
            token_end_index = len(input_ids) - 1
            while sequence_ids[token_end_index] != (1 if pad_on_rig
ht else 0):
                token_end_index -= 1

            # Detect if the answer is out of the span (in which cas
e this feature is labeled with the CLS index).
            if not (
                offsets[token_start_index][0] <= start_char
                and offsets[token_end_index][1] >= end_char
            ):
                tokenized_examples["start_positions"].append(cls_in
dex)
                tokenized_examples["end_positions"].append(cls_inde
x)
            else:
                # Otherwise move the token_start_index and token_en
d_index to the two ends of the answer.
                # Note: we could go after the last offset if the an
swer is the last word (edge case).
                while (
                    token_start_index < len(offsets) and offsets[to
ken_start_index][0] <= start_char
                ):
                    token_start_index += 1
                tokenized_examples["start_positions"].append(token_
start_index - 1)
                while offsets[token_end_index][1] >= end_char:
                    token_end_index -= 1
                tokenized_examples["end_positions"].append(token_en
d_index + 1)

    return tokenized_examples
```

This function works with one or several examples. In the case of several examples, the tokenizer will return a list of lists for each key:

In [14]:
```
features = prepare_train_features(datasets["train"][:5])
```

To apply this function on all the sentences (or pairs of sentences) in our dataset, we just use the `map` method of our `dataset` object we created earlier. This will apply the function on all the elements of all the splits in `dataset`, so our training, validation, and testing data will be preprocessed in one single command. Since our preprocessing changes the number of samples, we need to remove the old columns when applying it.

In [15]:
```
tokenized_datasets = datasets.map(
    prepare_train_features, batched=True, remove_columns=datasets["
train"].column_names
)
```

Before we kick off our SageMaker training job we need to transfer our dataset to S3, so the training job can download it from S3.

In [16]:
```
train_dataset = tokenized_datasets["train"]
eval_dataset = tokenized_datasets["validation"]

train_dataset.set_format(
    "torch", columns=["attention_mask", "end_positions", "input_id
s", "start_positions"]
)
eval_dataset.set_format(
    "torch", columns=["attention_mask", "end_positions", "input_id
s", "start_positions"]
)
```

In [17]:
```
import botocore
from datasets.filesystems import S3FileSystem

s3 = S3FileSystem()

s3_prefix = "samples/datasets/squad"

# 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
eval_input_path = f"s3://{sess.default_bucket()}/{s3_prefix}/eval"
eval_dataset.save_to_disk(eval_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 with Native PyTorch

Below, we run a native PyTorch training job with the `PyTorch` estimator on a `ml.p3.2xlarge` instance.

We run a batch size of 28 on our native training job and 52 on our Training Compiler training job to make an apple to apple comparison. These batch sizes along with the max_length variable get us close to 100% GPU memory utilization.

We recommend using the tested batch size that's provided at [Tested Models (https://docs.aws.amazon.com/sagemaker/latest/dg/training-compiler-support.html#training-compiler-tested-models)](https://docs.aws.amazon.com/sagemaker/latest/dg/training-compiler-support.html#training-compiler-tested-models) in the *SageMaker Training Compiler Developer Guide*.

`GPU MEM`

In [18]:
```
from sagemaker.pytorch import PyTorch

batch_size_native = 28
learning_rate_native = float("3e-5") / 32 * batch_size_native

# hyperparameters, which are passed into the training job
hyperparameters = {
    "epochs": 20,
    "train_batch_size": batch_size_native,
    "learning_rate": learning_rate_native,
    "model_name": "albert-base-v2",
    "n_gpus": 1,
    "output_dir": "/opt/ml/model",
}

# If checkpointing is enabled with higher epoch numbers
# your disk requirements should be increased as well
volume_size = 200
```

In [19]:
```
native_estimator = PyTorch(
    entry_point="qa_trainer_huggingface.py",
    source_dir="./scripts",
    instance_type="ml.p3.2xlarge",
    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,
)

# starting the train job with our uploaded datasets as input
native_estimator.fit({"train": training_input_path, "test": eval_input_path}, wait=False)

# The name of the training job. You might need to note this down in case you lose connection to your notebook.
native_estimator.latest_training_job.name
```

Out[19]:
```
'pytorch-training-2022-09-15-18-41-35-064'
```

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.

We use the tested batch size that's provided at [Tested Models (https://docs.aws.amazon.com/sagemaker/latest/dg/training-compiler-support.html#training-compiler-tested-models)](https://docs.aws.amazon.com/sagemaker/latest/dg/training-compiler-support.html#training-compiler-tested-models) in the *SageMaker Training Compiler Developer Guide*.

In [20]:
```
from sagemaker.huggingface import HuggingFace, TrainingCompilerConf
ig
# an updated max batch size that can fit into GPU memory with compi
ler
batch_size = 52

# update the global learning rate
learning_rate = learning_rate_native / batch_size_native * batch_si
ze

# hyperparameters, which are passed into the training job
hyperparameters = {
    "epochs": 20,
    "train_batch_size": batch_size,
    "learning_rate": learning_rate,
    "model_name": "albert-base-v2",
    "n_gpus": 1,
    "output_dir": "/opt/ml/model",
}

# If checkpointing is enabled with higher epoch numbers
# your disk requirements should be increased as well
volume_size = 200
```

In [21]:
```
optimized_estimator = HuggingFace(
    entry_point="qa_trainer_huggingface.py",
    compiler_config=TrainingCompilerConfig(),
    source_dir="./scripts",
    instance_type="ml.p3.2xlarge",
    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,
)

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

# The name of the training job. You might need to note this down in
case you lose connection to your notebook.
optimized_estimator.latest_training_job.name
```

Out[21]:
```
'huggingface-pytorch-trcomp-training-2022-09-15-18-41-35-867'
```

In [22]:
```
# Wait for training jobs to complete.

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 [23]:
```
# 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-2022-09-15-18
-41-35-064/output/model.tar.gz

latest training job name for this estimator:
pytorch-training-2022-09-15-18-41-35-064
```

In [24]:
```
%%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 native estimator. For example:

```
native_estimator = native.attach("your_native_training_job_name")
```

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

In [25]:
```
# 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-2022-09-15-18-41-35-867/output/model.tar.gz

latest training job name for this estimator:
huggingface-pytorch-trcomp-training-2022-09-15-18-41-35-867
```

In [26]:
```
%%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 native estimator. For example:

```
optimized_est = native.attach("your_optimized_native_training_job_name")
```

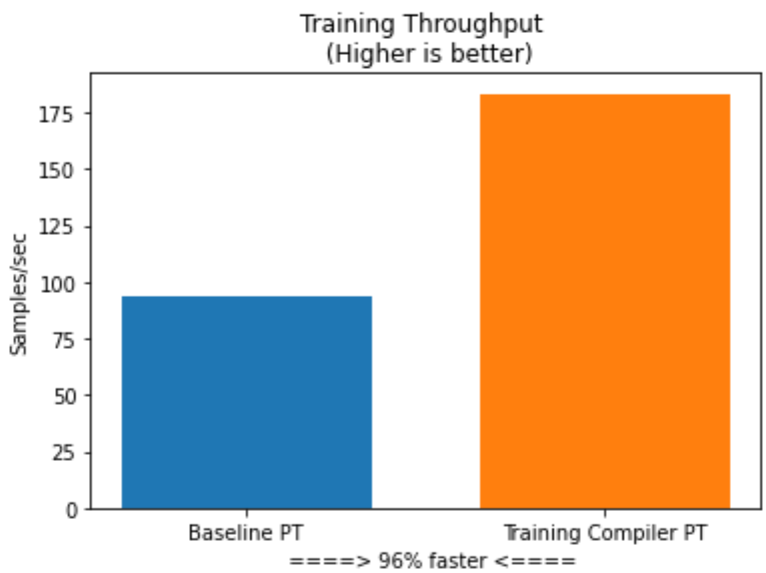
Create helper functions for analysis

In [27]:
```
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
```

Training Throughput Plot

The following script creates a plot that compares the throughput (number_of_samples/second) of the two training jobs with and without SageMaker Training Compiler.

In [28]:
```
n = summarize(_summarize(native))
native_throughput = n["summary"]["train_samples_per_second"]

o = summarize(_summarize(optimized))
optimized_throughput = o["summary"]["train_samples_per_second"]

avg_speedup = f"{round((optimized_throughput/native_throughput-1)*1
00)}%"

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

plt.bar(x=[1], height=native_throughput, width=0.35)
plt.bar(x=[1.5], height=optimized_throughput, width=0.35)

plt.xlabel("  ====> {} faster <====".format(avg_speedup))
plt.xticks(ticks=[1, 1.5], labels=["Baseline PT", "Training Compile
r PT"])
plt.show()
```



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 [29]:
```
import pandas as pd

pd.DataFrame([n["summary"], o["summary"]], index=["Native", "Optimized"])
```

Out[29]:

	train_runtime	train_samples_per_second	train_steps_per_second	train_loss	epoch
Native	3249.207	93.389	3.336	0.160402	20.0
Optimized	1655.076	183.339	3.529	0.171578	20.0

In [30]:
```
# 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 HF."
)
```

```
SageMaker Training Compiler integrated PyTorch is about 49% faster in terms of total training time as reported by HF.
```

Billable Time

The following script creates a plot that compares the billable time of the two training jobs with and without SageMaker Training Compiler.

In [31]:
```
def BillableTimeInSeconds(name):
    describe_training_job = (
        optimized_estimator.sagemaker_session.sagemaker_client.describe_training_job
    )
    details = describe_training_job(TrainingJobName=name)
    return details["BillableTimeInSeconds"]
```

In [32]:
```
Billable = {}
Billable["Native"] = BillableTimeInSeconds(native_estimator.latest_training_job.name)
Billable["Optimized"] = BillableTimeInSeconds(optimized_estimator.latest_training_job.name)
pd.DataFrame(Billable, index=["BillableSecs"])
```

Out[32]:

	Native	Optimized
BillableSecs	3612	2141

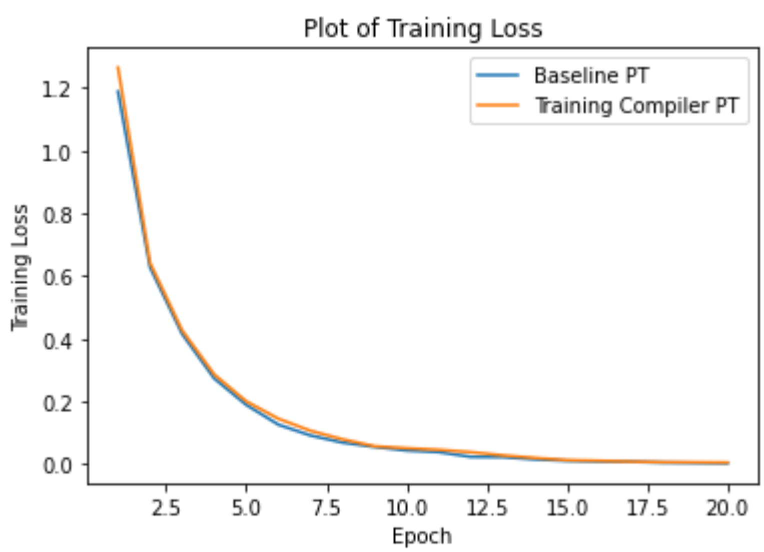
In [33]:
```
speedup = (Billable["Native"] - Billable["Optimized"]) * 100 / Bill
able["Native"]
print(f"SageMaker Training Compiler integrated PyTorch was {int(spe
edup)}% faster in summary.")
```

```
SageMaker Training Compiler integrated PyTorch was 40% faster in sum
mary.
```

Convergence of Training Loss

The following script creates a plot that compares the loss function of the two training jobs with and without SageMaker Training Compiler.

In [34]:
```
native_loss = [i["loss"] for i in n["train"]]
native_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(native_epochs, native_loss, label="Baseline PT")
plt.plot(optimized_epochs, optimized_loss, label="Training Compiler
PT")
plt.legend()
plt.show()
```



Conclusion

In this example, we fine-tuned an [ALBERT model (https://huggingface.co/albert-base-v2)](https://huggingface.co/albert-base-v2) (`albert-base-v2`) with the `SQuAD` dataset and compared a native training job with a SageMaker Training Compiler training job. The Training Compiler job has `93% higher throughput` and `38% quicker training` time while training loss was equal with the native PyTorch training job.

Clean up

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

In [36]:
```
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*.