```
import torch
torch.cuda.is_available()
```

```
True
```

```
import os
import re
import json
import random
import string
from dataclasses import dataclass
from typing import Dict, List, Union, Optional

import torch
import torchaudio
import librosa
import evaluate
from datasets import load_dataset, Audio, DatasetDict


# Audio parameters
TARGET_SAMPLING_RATE = 16000

# Training output dir
OUTPUT_DIR = "Ed-168/wav2vec2-large-xls-r-300m-hi"

# Training hyperparameters (tune for your budget)
BATCH_SIZE          = 1
GRAD_ACCUM          = 16
LEARNING_RATE = 3e-5    # Good starting LR for ASR, tune lower if model is unstable
NUM_TRAIN_EPOCHS    = 40
EVAL_STRATEGY       = "steps"
EVAL_STEPS          = 1000   # Evaluate less frequently to save memory
SAVE_STEPS          = 1000   # Save less frequently to reduce disk I/O
LOGGING_STEPS       = 50
WARMUP_RATIO        = 0.05
FP16                = torch.cuda.is_available()      # Enable mixed precision


# # If you want to push to the Hub, set these:
# PUSH_TO_HUB = True
# HF_REPO_ID = "Ed-168/Fine-tuned-wav2vec2-BERT-indian-languages"  # e.g. "username/w2vbert-hi-ctc-cv17"
```

```
from datasets import load_dataset

common_voice_train = load_dataset("mozilla-foundation/common_voice_16_0", "hi", split="train+validation")
common_voice_test = load_dataset("mozilla-foundation/common_voice_16_0", "hi", split="test")
print(common_voice_train)
print(common_voice_test)
```

```
Dataset({
    features: ['client_id', 'path', 'audio', 'sentence', 'up_votes', 'down_votes', 'age', 'gender', 'accent', 'locale', 'segment',
    num_rows: 7084
})
Dataset({
    features: ['client_id', 'path', 'audio', 'sentence', 'up_votes', 'down_votes', 'age', 'gender', 'accent', 'locale', 'segment',
    num_rows: 3107
})
```

```
len(common_voice_train)
```

```
7084
```

```
NUM_TRAIN_SAMPLES = 1500
NUM_TEST_SAMPLES =750

common_voice_train = common_voice_train.select(range(NUM_TRAIN_SAMPLES))
common_voice_test = common_voice_test.select(range(NUM_TEST_SAMPLES))
```

```
common_voice_train = common_voice_train.remove_columns(["accent", "age", "client_id", "down_votes", "gender", "locale", "segment", "up_v
common_voice_test = common_voice_test.remove_columns(["accent", "age", "client_id", "down_votes", "gender", "locale", "segment", "up_vo
```

```
def display_samples(data):
    for i in range(10):
        print(f"{i+1} {data[i]['sentence']}")
display_samples(common_voice_train)
```

```
1 हमने उसका जन्मदिन मनाया।
2 साउथ दिल्ली नगर निगम सख्त, शॉपिंग मॉल के बाहर नहीं दिखेंगे होर्डिंग
3 उत्तर कोरिया ने अमेरिका को दी हमले की धमकी
4 अगले कमरे में अनेक रोमन मूर्तियाँ हैं।
5 तुम ने टॉम को कहाँ भेज दिया?
6 सर्दोयों के आने से दिन छोटे होते जाते हैं।
7 मुझे और वक़्त दो।
8 कंगना के वक़ील ने कहा, पुलिस ने किसी लैपटॉप की डिमांड नहीं की है
9 क्या सवाल है!
10 वह अच्छा राजा था।
```

```
import re

chars_to_ignore_regex = r"[\"\'\(\)\[\]\{\}\<\>\–\-\-\–\–\-\–\–\.\,\?\!\:\;\।\d\@\#\$\%\^\&\*\+\=\_\\\/\|~`]+"

def normalize_text(batch):
    text = batch["sentence"]
    text = text.lower()
    text = re.sub(chars_to_ignore_regex, " ", text)
    text = re.sub(r"\s+", " ", text).strip()
    batch["sentence"] = text
    return batch

common_voice_train = common_voice_train.map(normalize_text)
common_voice_test = common_voice_test.map(normalize_text)
display_samples(common_voice_train)
```

```
Map:   0%|          | 0/1500 [00:00<?, ? examples/s]
Map:   0%|          | 0/750 [00:00<?, ? examples/s]
1 हमने उसका जन्मदिन मनाया
2 साउथ दिल्ली नगर निगम सख्त शॉपिंग मॉल के बाहर नहीं दिखेंगे होर्डिंग
3 उत्तर कोरिया ने अमेरिका को दी हमले की धमकी
4 अगले कमरे में अनेक रोमन मूर्तियाँ हैं
5 तुम ने टॉम को कहाँ भेज दिया
6 सर्दोयों के आने से दिन छोटे होते जाते हैं
7 मुझे और वक़्त दो
8 कंगना के वकील ने कहा पुलिस ने किसी लैपटॉप की डिमांड नहीं की है
9 क्या सवाल है
10 वह अच्छा राजा था
```

```
import os
import json

def extract_all_chars(batch):
    all_text = " ".join(batch["sentence"])
    return {"all_text": [all_text]}

vocabs = common_voice_train.map(
    extract_all_chars, batched=True, batch_size=-1, remove_columns=common_voice_train.column_names
)
all_text = " ".join(vocabs["all_text"])
vocab_list = sorted(list(set(list(all_text))))

# Remove the space from the set; we'll add a dedicated word_delimiter_token later.
if " " in vocab_list:
    vocab_list.remove(" ")

# Remove English letters (latin script)
vocab_list = [c for c in vocab_list if not (c >= 'a' and c <= 'z') and not (c >= 'A' and c <= 'Z')]

# Build vocab dict
vocab_dict = {v: k for k, v in enumerate(vocab_list)}
vocab_dict["|"] = len(vocab_dict)  # word delimiter
vocab_dict["[UNK]"] = len(vocab_dict)
vocab_dict["[PAD]"] = len(vocab_dict)

print("Vocab size:", len(vocab_dict))
print("Sample of vocab keys:", list(vocab_dict.keys())[:60])

# Save vocab to disk
os.makedirs(OUTPUT_DIR, exist_ok=True)
vocab_path = os.path.join(OUTPUT_DIR, "vocab.json")
with open(vocab_path, "w", encoding="utf-8") as f:
    json.dump(vocab_dict, f, ensure_ascii=False, indent=2)
print("Saved vocab to:", vocab_path)
```

```
Map:   0%|          | 0/1500 [00:00<?, ? examples/s]
Vocab size: 73
Sample of vocab keys: ['ँ', 'ं', 'ः', 'अ', 'आ', 'इ', 'ई', 'उ', 'ऊ', 'ऋ', 'ए', 'ऐ', 'ऑ', 'ओ', 'औ', 'क', 'ख', 'ग', 'घ', 'च', '
Saved vocab to: Ed-168/wav2vec2-large-xls-r-300m-hi\vocab.json
```

```
from transformers import Wav2Vec2CTCTokenizer
tokenizer = Wav2Vec2CTCTokenizer.from_pretrained(
    OUTPUT_DIR,
    unk_token="[UNK]",
    pad_token="[PAD]",
    word_delimiter_token="|"
)
```

```
from transformers import Wav2Vec2FeatureExtractor

feature_extractor = Wav2Vec2FeatureExtractor(feature_size=1, sampling_rate=16000, padding_value=0.0, do_normalize=True, return_attention
```

```
from transformers import Wav2Vec2Processor

processor = Wav2Vec2Processor(feature_extractor=feature_extractor, tokenizer=tokenizer)
```

```
common_voice_train = common_voice_train.cast_column("audio", Audio(sampling_rate=16_000))
common_voice_test = common_voice_test.cast_column("audio", Audio(sampling_rate=16_000))
print(common_voice_train[0]["audio"])
```

```
{'path': 'C:\\Users\\EDWIN\\.cache\\huggingface\\datasets\\downloads\\extracted\\2c6cd998a8800b56f2fb15d7259927dbca1bd0fa2f05133309
        -1.31181878e-07,  2.62807589e-07,  4.76284185e-08]), 'sampling_rate': 16000}
```

```
import IPython.display as ipd
import numpy as np
import random

rand_int = random.randint(0, len(common_voice_train)-1)

print(common_voice_train[rand_int]["sentence"])
ipd.Audio(data=common_voice_train[rand_int]["audio"]["array"], autoplay=True, rate=16000)
```

```
दूसरा मकान खरीदने पर कितना मिलता है इनकम टैक्स में फायदा
```

0:00 / 0:05

```
rand_int = random.randint(0, len(common_voice_train)-1)

print("Target text:", common_voice_train[rand_int]["sentence"])
print("Input array shape:", common_voice_train[rand_int]["audio"]["array"].shape)
print("Sampling rate:", common_voice_train[rand_int]["audio"]["sampling_rate"])
```

```
Target text: कहाँ गई हो
Input array shape: (42048,)
Sampling rate: 16000
```

```
def prepare_dataset(batch):
    audio = batch["audio"]

    # batched output is "un-batched"
    batch["input_values"] = processor(audio["array"], sampling_rate=audio["sampling_rate"]).input_values[0]
    batch["input_length"] = len(batch["input_values"])

    with processor.as_target_processor():
        batch["labels"] = processor(batch["sentence"]).input_ids
    return batch
```

```
common_voice_train = common_voice_train.map(prepare_dataset, remove_columns=common_voice_train.column_names)
common_voice_test = common_voice_test.map(prepare_dataset, remove_columns=common_voice_test.column_names)
```

```
Map:   0%|          | 0/1500 [00:00<?, ? examples/s]
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\transformers\models\wav2vec2\processing_wav2vec2.p
  warnings.warn(
Map:   0%|          | 0/750 [00:00<?, ? examples/s]
```

```
# max_input_length_in_sec = 5.0
# common_voice_train = common_voice_train.filter(lambda x: x < max_input_length_in_sec * processor.feature_extractor.sampling_rate, inpu
```

```
len(common_voice_train)
```

```
1500
```

```
# import torch

# from dataclasses import dataclass, field
# from typing import Any, Dict, List, Optional, Union

@dataclass
class DataCollatorCTCWithPadding:
    processor: Wav2Vec2Processor
    padding: Union[bool, str] = True

    def __call__(self, features: List[Dict[str, Union[List[int], torch.Tensor]]]) -> Dict[str, torch.Tensor]:
        # split inputs and labels since they have to be of different lengths and need
        # different padding methods
        input_features = [{"input_values": feature["input_values"]} for feature in features]
        label_features = [{"input_ids": feature["labels"]} for feature in features]

        batch = self.processor.pad(
            input_features,
            padding=self.padding,
            return_tensors="pt",
        )
        with self.processor.as_target_processor():
            labels_batch = self.processor.pad(
                label_features,
                padding=self.padding,
                return_tensors="pt",
            )

        # replace padding with -100 to ignore loss correctly
        labels = labels_batch["input_ids"].masked_fill(labels_batch.attention_mask.ne(1), -100)

        batch["labels"] = labels

        return batch
```

```
data_collator = DataCollatorCTCWithPadding(processor=processor, padding=True)
```

```
wer_metric = evaluate.load("wer")

def compute_metrics(pred):
    # pred.predictions is float logits of shape (batch, time, vocab_size)
    pred_logits = pred.predictions
    pred_ids = torch.from_numpy(pred_logits).argmax(-1)

    # Decode predictions and references
    pred_str = processor.batch_decode(pred_ids, skip_special_tokens=True)
    # Replace -100 with pad_token_id for decoding refs
    label_ids = pred.label_ids
    label_ids[label_ids == -100] = processor.tokenizer.pad_token_id
    label_str = processor.batch_decode(label_ids, group_tokens=False)

    wer = wer_metric.compute(predictions=pred_str, references=label_str)
    return {"wer": wer}
```

```
from transformers import Wav2Vec2ForCTC

model = Wav2Vec2ForCTC.from_pretrained(
    "facebook/wav2vec2-xls-r-300m",
    attention_dropout=0.0,
    hidden_dropout=0.0,
    feat_proj_dropout=0.0,
    mask_time_prob=0.05,
    layerdrop=0.0,
    ctc_loss_reduction="mean",
    pad_token_id=processor.tokenizer.pad_token_id,
    vocab_size=len(processor.tokenizer),
)
```

```
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at facebook/wav2vec2-xls-r-300m and are newly initiali
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
model.freeze_feature_extractor()
```

```
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\transformers\models\wav2vec2\modeling_wav2vec2.py
  warnings.warn(
```

```
repo_name = "Ed-168/wav2vec2-large-xls-r-300m-hi"
```

```
from transformers import TrainingArguments

training_args = TrainingArguments(
  output_dir=repo_name,
  group_by_length=True,
  per_device_train_batch_size=16,
  gradient_accumulation_steps=2,
  num_train_epochs=40,
  gradient_checkpointing=True,
  fp16=True,
  save_steps=400,
  eval_steps=400,
  logging_steps=50,
  learning_rate=3e-4,
  warmup_steps=500,
  save_total_limit=2,
  push_to_hub=False,
)
```

```
from transformers import Trainer

trainer = Trainer(
    model=model,
    data_collator=data_collator,
    args=training_args,
    compute_metrics=compute_metrics,
    train_dataset=common_voice_train,
    eval_dataset=common_voice_test,
    tokenizer=processor.feature_extractor,
)
```

```
C:\Users\EDWIN\AppData\Local\Temp\ipykernel_27200\2066899627.py:3: FutureWarning: `tokenizer` is deprecated and will be removed in v
  trainer = Trainer(
```

```
trainer.train()
```

```
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\torch\utils\checkpoint.py:85: UserWarning: None of
  warnings.warn(
```

[1880/1880 12:00:35, Epoch 40/40]

Step	Training Loss
50	17.862000
100	7.304300
150	4.795900
200	3.652600
250	3.460900
300	3.414800
350	3.353800
400	2.742000
450	1.242600
500	0.779900
550	0.573600
600	0.442200
650	0.352000
700	0.290900
750	0.241100
800	0.217400
850	0.184800
900	0.168400
950	0.152100
1000	0.136200
1050	0.126600
1100	0.119400
1150	0.119800
1200	0.107900
1250	0.104900
1300	0.091900
1350	0.087300
1400	0.086400
1450	0.079400
1500	0.072500
1550	0.069500
1600	0.068600
1650	0.061700
1700	0.061600
1750	0.057700
1800	0.058600
1850	0.057800

```
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\transformers\models\wav2vec2\processing_wav2vec2.p
  warnings.warn(
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\torch\utils\checkpoint.py:85: UserWarning: None of
  warnings.warn(
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\transformers\models\wav2vec2\processing_wav2vec2.p
  warnings.warn(
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\torch\utils\checkpoint.py:85: UserWarning: None of
  warnings.warn(
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\transformers\models\wav2vec2\processing_wav2vec2.p
  warnings.warn(
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\torch\utils\checkpoint.py:85: UserWarning: None of
  warnings.warn(
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\transformers\models\wav2vec2\processing_wav2vec2.p
  warnings.warn(
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\torch\utils\checkpoint.py:85: UserWarning: None of
  warnings.warn(
TrainOutput(global_step=1880, training_loss=1.4051658075540623, metrics={'train_runtime': 43248.0009, 'train_samples_per_second':
1.387, 'train_steps_per_second': 0.043, 'total_flos': 9.181862684461875e+18, 'train_loss': 1.4051658075540623, 'epoch': 40.0})
```

```
from transformers import Wav2Vec2ForCTC, Wav2Vec2Processor
repo_name = "Ed-168/wav2vec2-large-xls-r-300m-hi/checkpoint-1880"
model = Wav2Vec2ForCTC.from_pretrained(repo_name).to("cuda")
processor = Wav2Vec2Processor.from_pretrained(repo_name)
```

```
from datasets import load_dataset

common_voice_train = load_dataset("mozilla-foundation/common_voice_16_0", "hi", split="train+validation")
common_voice_test = load_dataset("mozilla-foundation/common_voice_16_0", "hi", split="test")
print(common_voice_train)
print(common_voice_test)
```

```
Dataset({
    features: ['client_id', 'path', 'audio', 'sentence', 'up_votes', 'down_votes', 'age', 'gender', 'accent', 'locale', 'segment',
    num_rows: 7084
})
Dataset({
    features: ['client_id', 'path', 'audio', 'sentence', 'up_votes', 'down_votes', 'age', 'gender', 'accent', 'locale', 'segment',
    num_rows: 3107
})
```

```
NUM_TRAIN_SAMPLES = 1500
NUM_TEST_SAMPLES =750

common_voice_train = common_voice_train.select(range(NUM_TRAIN_SAMPLES))
common_voice_test = common_voice_test.select(range(NUM_TEST_SAMPLES))
```

```
common_voice_train = common_voice_train.remove_columns(["accent", "age", "client_id", "down_votes", "gender", "locale", "segment", "up_v
common_voice_test = common_voice_test.remove_columns(["accent", "age", "client_id", "down_votes", "gender", "locale", "segment", "up_vot
```

```
import re

chars_to_ignore_regex = r"[\"\'\(\)\[\]\{\}\<\>\—\-\-\–\—\-\—\.\,\?\!\:\;\।\d\@\#\$\%\^\&\*\+\=\_\\\/\|~`]+"

def normalize_text(batch):
    text = batch["sentence"]
    text = text.lower()
    text = re.sub(chars_to_ignore_regex, " ", text)
    text = re.sub(r"\s+", " ", text).strip()
    batch["sentence"] = text
    return batch

common_voice_train = common_voice_train.map(normalize_text)
common_voice_test = common_voice_test.map(normalize_text)
```

```
from transformers import Wav2Vec2CTCTokenizer
tokenizer = Wav2Vec2CTCTokenizer.from_pretrained(
    repo_name,
    unk_token="[UNK]",
    pad_token="[PAD]",
    word_delimiter_token="|"
)
```

```
from transformers import Wav2Vec2FeatureExtractor

feature_extractor = Wav2Vec2FeatureExtractor(feature_size=1, sampling_rate=16000, padding_value=0.0, do_normalize=True, return_attentior
```

```
from datasets import Audio
common_voice_train = common_voice_train.cast_column("audio", Audio(sampling_rate=16_000))
common_voice_test = common_voice_test.cast_column("audio", Audio(sampling_rate=16_000))
print(common_voice_train[0]["audio"])
```

```
{'path': 'C:\\Users\\EDWIN\\.cache\\huggingface\\datasets\\downloads\\extracted\\2c6cd998a8800b56f2fb15d7259927dbca1bd0fa2f05133309
       -1.31181878e-07,  2.62807589e-07,  4.76284185e-08]), 'sampling_rate': 16000}
```

```
def prepare_dataset(batch):
    audio = batch["audio"]

    # batched output is "un-batched"
    batch["input_values"] = processor(audio["array"], sampling_rate=audio["sampling_rate"]).input_values[0]
    batch["input_length"] = len(batch["input_values"])

    with processor.as_target_processor():
        batch["labels"] = processor(batch["sentence"]).input_ids
    return batch
```

```
#common_voice_train = common_voice_train.map(prepare_dataset, remove_columns=common_voice_train.column_names)
common_voice_test = common_voice_test.map(prepare_dataset, remove_columns=common_voice_test.column_names)
```

```
Map:   0%|          | 0/750 [00:00<?, ? examples/s]
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\transformers\models\wav2vec2\processing_wav2vec2.p
  warnings.warn(
```

```
from datasets import load_dataset
import soundfile as sf
import torch

# Load only a small sample from the Common Voice Hindi test set
# common_voice_test = load_dataset("mozilla-foundation/common_voice_16_0", "hi", split="test")

input_dict = processor(common_voice_test[1]["input_values"], return_tensors="pt", padding=True)

logits = model(input_dict.input_values.to("cuda")).logits

pred_ids = torch.argmax(logits, dim=-1)[0]
```

```
It is strongly recommended to pass the `sampling_rate` argument to `Wav2Vec2FeatureExtractor()`. Failing to do so can result in sile
```

```
print("Prediction:")
print(processor.decode(pred_ids))

common_voice_test_prediction = load_dataset("mozilla-foundation/common_voice_16_0", "hi", split="test")
print("\nReference:")
print(common_voice_test_prediction[1]["sentence"])
```

```
Prediction:
आबढ़ामपूर्ढ में अखिलेश बाटेंगे लैपटॉक का लॉलीपॉट

Reference:
अब रामपुर में अखिलेश बांटेंगे लैपटॉप का 'लॉलीपॉप'
```

```
repo_name = "Ed-168/wav2vec2-large-xls-r-300m-hi"
tokenizer.push_to_hub(repo_name)
tokenizer.save_pretrained(repo_name)
```

```
README.md: 0.00B [00:00, ?B/s]
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\huggingface_hub\file_download.py:143: UserWarning
To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an administrator. In order to activat
  warnings.warn(message)
('Ed-168/wav2vec2-large-xls-r-300m-hi\\tokenizer_config.json',
 'Ed-168/wav2vec2-large-xls-r-300m-hi\\special_tokens_map.json',
 'Ed-168/wav2vec2-large-xls-r-300m-hi\\vocab.json',
 'Ed-168/wav2vec2-large-xls-r-300m-hi\\added_tokens.json')
```

```
repo_name = "Ed-168/wav2vec2-large-xls-r-300m-hi"
model.push_to_hub(repo_name)
processor.push_to_hub(repo_name)
```

```
Processing Files (0 / 0)                : |          |  0.00B /  0.00B
New Data Upload                         : |          |  0.00B /  0.00B
  ...\Temp\tmpgb9j8hra\model.safetensors:   4%|3         | 50.4MB / 1.26GB
README.md: 0.00B [00:00, ?B/s]
c:\Users\EDWIN\OneDrive\Documents\GitHub\Multilingual-ASR\.venv\lib\site-packages\huggingface_hub\file_download.py:143: UserWarning
```