```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/dri
```

```
pip install datasets
```

```
Requirement already satisfied: datasets in /usr/local/lib/python3.11/dist-packages (3.5.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from datasets) (3.1
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from datasets) (
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.11/dist-packages (from dataset
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from data
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from datasets) (2.2.2
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.11/dist-packages (from datase
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.11/dist-packages (from datasets)
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-packages (from datasets) (3.5.0
Requirement already satisfied: multiprocess<0.70.17 in /usr/local/lib/python3.11/dist-packages (from da
Requirement already satisfied: fsspec<=2024.12.0,>=2023.1.0 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from datasets) (3.11
Requirement already satisfied: huggingface-hub>=0.24.0 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from datasets) (24
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from datasets) (
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohtt
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aioht
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aic
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohtt
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from aioht
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requ
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requ
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->da
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateuti
```

```
import os
import torch
import torch.nn as nn
import numpy as np
import pandas as pd
from datasets import load_dataset, Dataset, Audio
from transformers import Wav2Vec2ForSequenceClassification, Wav2Vec2FeatureExtractor
from transformers import TrainingArguments, Trainer
from sklearn.metrics import accuracy_score, precision_recall_fscore_support
```

```
real_audio_dir = "/content/drive/MyDrive/mini_bonafide"
fake_audio_dir = "/content/drive/MyDrive/mini_spoof"
```

```
def load_audio_dataset(real_audio_dir, fake_audio_dir):
    data = {
        'audio': [],
        'label': []
    }

    for filename in os.listdir(real_audio_dir):
        if filename.endswith(('.wav', '.mp3', '.flac')):
            data['audio'].append(os.path.join(real_audio_dir, filename))
            data['label'].append(0)
    for filename in os.listdir(fake_audio_dir):
        if filename.endswith(('.wav', '.mp3', '.flac')):
            data['audio'].append(os.path.join(fake_audio_dir, filename))
            data['label'].append(1)

    df = pd.DataFrame(data)

    train_df = df.sample(frac=0.8, random_state=42)
    val_df = df.drop(train_df.index)

    train_dataset = Dataset.from_pandas(train_df)
    val_dataset = Dataset.from_pandas(val_df)

    train_dataset = train_dataset.cast_column("audio", Audio(sampling_rate=16000))
    val_dataset = val_dataset.cast_column("audio", Audio(sampling_rate=16000))

    return train_dataset, val_dataset

train_dataset, val_dataset = load_audio_dataset(real_audio_dir, fake_audio_dir)
```

```
model_checkpoint = "facebook/wav2vec2-base"
feature_extractor = Wav2Vec2FeatureExtractor.from_pretrained(model_checkpoint)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
preprocessor_config.json: 100%                                  159/159 [00:00<00:00, 15.7kB/s]
```

```
def preprocess_function(examples):
    audio_arrays = [x["array"] for x in examples["audio"]]

    max_duration_in_seconds = 10
    max_length = feature_extractor.sampling_rate * max_duration_in_seconds
    processed_arrays = []

    for audio in audio_arrays:
        if len(audio) > max_length:
            processed_arrays.append(audio[:max_length])
        else:
            padding = np.zeros(max_length - len(audio), dtype=np.float32)
            processed_arrays.append(np.concatenate([audio, padding]))

    inputs = feature_extractor(
        processed_arrays,
        sampling_rate=feature_extractor.sampling_rate,
        padding="max_length",
        max_length=max_length,
        truncation=True,
        return_tensors="pt"
    )

    inputs["labels"] = examples["label"]
    return inputs

train_dataset = train_dataset.map(preprocess_function, batched=True)
val_dataset = val_dataset.map(preprocess_function, batched=True)
```

```
Map: 100%                                        176/176 [01:39<00:00,  1.76 examples/s]

Map: 100%                                        44/44 [00:19<00:00,  2.30 examples/s]
```

```
num_labels = 2
model = Wav2Vec2ForSequenceClassification.from_pretrained(
    model_checkpoint,
    num_labels=num_labels,
)

for param in model.wav2vec2.feature_extractor.parameters():
    param.requires_grad = False
```

```
config.json: 100%                                 1.84k/1.84k [00:00<00:00, 35.9kB/s]
/usr/local/lib/python3.11/dist-packages/transformers/configuration_utils.py:315: UserWarning: Passing `
  warnings.warn(
pytorch_model.bin: 100%                                 380M/380M [00:04<00:00, 88.7MB/s]
Some weights of Wav2Vec2ForSequenceClassification were not initialized from the model checkpoint at fac
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inf
```

```
def compute_metrics(pred):
    predictions = np.argmax(pred.predictions, axis=1)
    labels = pred.label_ids
    precision, recall, f1, _ = precision_recall_fscore_support(labels, predictions, average='binary')
    accuracy = accuracy_score(labels, predictions)
    return {
        'accuracy': accuracy,
        'f1': f1,
        'precision': precision,
        'recall': recall
    }
```

```
os.environ["WANDB_DISABLED"] = "true"
training_args = TrainingArguments(
    output_dir="./wav2vec2-finetuned-deepfake-detection",
    evaluation_strategy="epoch",
    save_strategy="epoch",
    learning_rate=3e-5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=10,
    weight_decay=0.01,
    load_best_model_at_end=True,
    metric_for_best_model="f1",
    push_to_hub=False,
)
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    compute_metrics=compute_metrics,
)
trainer.train()
trainer.save_model("./wav2vec2-finetuned-deepfake-detection")
```

```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611: FutureWarning: `evaluation_
  warnings.warn(
Using the `WANDB_DISABLED` environment variable is deprecated and will be removed in v5. Use the --repc
```

[220/220 10:35, Epoch 10/10]

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	No log	0.569532	0.818182	0.764706	1.000000	0.619048
2	No log	0.257916	0.954545	0.950000	1.000000	0.904762
3	No log	0.280652	0.909091	0.894737	1.000000	0.809524
4	No log	0.029720	1.000000	1.000000	1.000000	1.000000
5	No log	0.046004	0.977273	0.976744	0.954545	1.000000
6	No log	0.104494	0.977273	0.976744	0.954545	1.000000
7	No log	0.084310	0.977273	0.976744	0.954545	1.000000
8	No log	0.034408	0.977273	0.976744	0.954545	1.000000
9	No log	0.015373	1.000000	1.000000	1.000000	1.000000
10	No log	0.012653	1.000000	1.000000	1.000000	1.000000

```python
def predict_audio(audio_path, model, feature_extractor):

    max_duration_in_seconds = 10

    dataset = Dataset.from_dict({"audio": [audio_path]})
    dataset = dataset.cast_column("audio", Audio(sampling_rate=16000))
    audio = dataset[0]["audio"]

    inputs = feature_extractor(
        audio["array"],
        sampling_rate=audio["sampling_rate"],
        padding="max_length",
        max_length=feature_extractor.sampling_rate * max_duration_in_seconds,
        truncation=True,
        return_tensors="pt"
    )

    with torch.no_grad():
        logits = model(**inputs).logits

    probabilities = torch.softmax(logits, dim=1).cpu().numpy()[0]
    predicted_class = np.argmax(probabilities)

    labels = ["Real", "Deepfake"]
    return {
        "prediction": labels[predicted_class],
        "confidence": float(probabilities[predicted_class]),
        "probabilities": {labels[i]: float(prob) for i, prob in enumerate(probabilities)}
    }
```

```python
model = Wav2Vec2ForSequenceClassification.from_pretrained("./wav2vec2-finetuned-deepfake-detection")
feature_extractor = Wav2Vec2FeatureExtractor.from_pretrained(model_checkpoint)
result = predict_audio("/content/drive/MyDrive/D_0000406645.flac", model, feature_extractor)
print(result)
```

```
{'prediction': 'Real', 'confidence': 0.9749166369438171, 'probabilities': {'Real': 0.9749166369438171,
```

D_2362 D_0000406645 M - - - - bonafide bonafide -

from ASVspoof5.dev.track_1.tsv file