

```

# %%capture
# #!unzip Datasets.zip

# from google.colab import drive

# # Mount the Google Drive
# drive.mount('/content/drive')

# %%capture
# !pip install datasets
# !pip install transformers
# !pip install librosa
# !pip install jiwer
# !pip install evaluate

import os
import datasets
import pandas as pd
from sklearn.model_selection import train_test_split
from datasets import Dataset

# Set paths
csv_path = "/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/final.csv"
audio_folder = "/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/final-w"

# Load the CSV
df = pd.read_csv(csv_path)
df = pd.read_csv(csv_path)
# Ensure the column names match
df.columns = ["Filename", "Transcription"] # Rename columns if needed

# Append '.wav' to the file names
df['Filename'] = df['Filename'].apply(lambda x: f"{x}.wav")

# Add full paths to the audio files
df['file_path'] = df['Filename'].apply(lambda x: os.path.join(audio_folder, x))

# Verify that all audio files exist
missing_files = df[~df['file_path'].apply(os.path.exists)]
if not missing_files.empty:
    print("The following audio files are missing:")
    print(missing_files)
    raise FileNotFoundError("Some audio files listed in the CSV are missing in the folder.")

# Split into train (27) and test (3)
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)

# Save splits to CSV for reference
train_csv_path = "train_split.csv"
test_csv_path = "test_split.csv"
train_df.to_csv(train_csv_path, index=False)
test_df.to_csv(test_csv_path, index=False)

# Convert to HuggingFace Dataset format
train_dataset = Dataset.from_pandas(train_df)
test_dataset = Dataset.from_pandas(test_df)

# Save HuggingFace datasets
train_dataset_path = "train_dataset"
test_dataset_path = "test_dataset"
train_dataset.save_to_disk(train_dataset_path)
test_dataset.save_to_disk(test_dataset_path)

# Output
print(f"Train set saved to: {train_csv_path} and {train_dataset_path}")
print(f"Test set saved to: {test_csv_path} and {test_dataset_path}")

↗ /home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please
from .autonotebook import tqdm as notebook_tqdm
Saving the dataset (1/1 shards): 100%|██████████| 2600/2600 [00:00<00:00, 726044.63 examples/s]
Saving the dataset (1/1 shards): 100%|██████████| 650/650 [00:00<00:00, 357687.96 examples/s]Train set saved to: train_s
Test set saved to: test_split.csv and test_dataset

from datasets import load_from_disk

train_dataset = load_from_disk("train_dataset")
test_dataset = load_from_disk("test_dataset")

```

```
print(train_dataset)
print(test_dataset)
```

```

/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/tqdm/auto.py:21: TqdmWarning: IPProgress not found. Please
from .autonotebook import tqdm as notebook_tqdm
Dataset({
  features: ['Filename', 'Transcription', 'file_path', '__index_level_0__'],
  num_rows: 2600
})
Dataset({
  features: ['Filename', 'Transcription', 'file_path', '__index_level_0__'],
  num_rows: 650
})

```

```

from datasets import ClassLabel
import random
import pandas as pd
from IPython.display import display, HTML

```

```

def show_random_elements(dataset, num_examples=10):
    assert num_examples <= len(dataset), "Can't pick more elements than there are in the dataset."
    picks = []
    for _ in range(num_examples):
        pick = random.randint(0, len(dataset)-1)
        while pick in picks:
            pick = random.randint(0, len(dataset)-1)
        picks.append(pick)

    df = pd.DataFrame(dataset[picks])
    display(HTML(df.to_html()))

```

```
show_random_elements(train_dataset)
```

	Filename	Transcription	file_path	__index_level_0__
0	8140244.wav	گیتھو بیگان ز بوزن وول چٹھ حاران گڑھان	/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/final-waves/8140244.wav	243
1	ishrat2-01_13.wav	وہ زبانی چٹھ ترجمہ تہ تلخیص کرچہ انہ آہ	/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/final-waves/ishrat2-01_13.wav	1863
2	jhon-02_234.wav	ہز آسٹک دعویٰ چٹھ کہی یاہ مسلمان رد کورمت	/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/final-waves/jhon-02_234.wav	2697
3	guddy1-08_13.wav	وہی اگر یہ تاپچہ یہ گاش نصیبہ آہ	/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/final-waves/guddy1-08_13.wav	1223
4	jhon-01.1_77.wav	ترو نہ تہ پٹن سٹک شہل آدن گام میاں کہی چٹھ سائہ فہم لورک اصل پرستان سائین گامین ہند رون ہز کوڈ چٹھ وے	/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/final-waves/jhon-01.1_77.wav	2485
5	rafiya-01_59.wav	زویٹھ نہ حرکت کیہ گوی پک صاف شراکھ	/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/final-waves/rafiya-01_59.wav	2976
6	8150220.wav	ہین ہنر ٹھہ بانہ اہمیت چٹھ یہ ز	/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/final-waves/8150220.wav	616
7	ishrat1-	تہ ہ چٹھہ ہکا ہا	/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/final-waves/ishrat1-	1693

If there are any unwanted special characters in the dataset, we can remove them here, since there are none, I am keeping that as it is.

```

def extract_all_chars(batch):
    all_text = " ".join(batch["Transcription"])
    vocab = list(set(all_text))
    return {"vocab": [vocab], "all_text": [all_text]}

```

```

vocab_train = train_dataset.map(extract_all_chars, batched=True, batch_size=-1, keep_in_memory=True, remove_columns=train_dataset.get_column_names())
vocab_test = test_dataset.map(extract_all_chars, batched=True, batch_size=-1, keep_in_memory=True, remove_columns=test_dataset.get_column_names())

```

```

Map: 100%|██████████| 2600/2600 [00:00<00:00, 312263.85 examples/s]
Map: 100%|██████████| 650/650 [00:00<00:00, 250947.86 examples/s]

```

```
vocab_list = list(set(vocab_train["vocab"][0]) | set(vocab_test["vocab"][0]))
```

```

vocab_dict = {v: k for k, v in enumerate(sorted(vocab_list))}
vocab_dict

```

Show hidden output

```

vocab_dict["|"] = vocab_dict[" "]
del vocab_dict[" "]

vocab_dict["[UNK]"] = len(vocab_dict)
vocab_dict["[PAD]"] = len(vocab_dict)
len(vocab_dict)

↩ 60

import json
with open('vocab.json', 'w') as vocab_file:
    json.dump(vocab_dict, vocab_file)

from transformers import Wav2Vec2CTCTokenizer

tokenizer = Wav2Vec2CTCTokenizer.from_pretrained("./", unk_token="[UNK]", pad_token="[PAD]", word_delimiter_token="|", clear

from transformers import Wav2Vec2FeatureExtractor

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

from transformers import Wav2Vec2Processor

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

train_dataset[0]["file_path"]

↩ '/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/final-
waves/farhat-03_21.wav'

Replacing the File Path with Actual Audio.

from datasets import load_from_disk, Audio

# Load datasets
train_dataset = load_from_disk("train_dataset") # Adjust to your actual path
test_dataset = load_from_disk("test_dataset")

# Rename 'file_path' to 'audio'
train_dataset = train_dataset.rename_column("file_path", "audio")
test_dataset = test_dataset.rename_column("file_path", "audio")

# # Cast the 'audio' column to use the Audio feature
train_dataset = train_dataset.cast_column("audio", Audio(sampling_rate=16_000))
test_dataset = test_dataset.cast_column("audio", Audio(sampling_rate=16_000))

# # Drop unnecessary columns if needed
train_dataset = train_dataset.remove_columns(["__index_level_0__"])
test_dataset = test_dataset.remove_columns(["__index_level_0__"])

# # Verify the dataset structure
print(train_dataset)
print(test_dataset)

# # Inspect the first example
print(train_dataset[0])

↩ Dataset({
  features: ['Filename', 'Transcription', 'audio'],
  num_rows: 2600
})
Dataset({
  features: ['Filename', 'Transcription', 'audio'],
  num_rows: 650
})
{'Filename': 'farhat-03_21.wav', 'Transcription': 'زَانِم شاه صَابِن بڈشابس\xa0مرا قِبم پَتِم تِم بِيَنَلِم', 'audio': {'path': '/ho
0.0256958 , 0.02392578}}, 'sampling_rate': 16000}}

#print(test_dataset[0]['audio'])

rand_int = random.randint(0, len(train_dataset))

print("Target text:", train_dataset[rand_int]["Transcription"])
print("Input array shape:", train_dataset[rand_int]["audio"]["array"].shape)
print("Sampling rate:", train_dataset[rand_int]["audio"]["sampling_rate"])

```

```

Target text: يعني لل ماجم تراو شري سنيديس دبانس مننر ز اونگج تم تمو مننر دراو
Input array shape: (121391,)
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"])

    batch["labels"] = processor(text=batch["Transcription"]).input_ids

    return batch

train_dataset = train_dataset.map(prepare_dataset, remove_columns=train_dataset.column_names)
test_dataset = test_dataset.map(prepare_dataset, remove_columns=test_dataset.column_names)

import torch

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

@dataclass
class DataCollatorCTCWithPadding:
    """
    Data collator that will dynamically pad the inputs received.
    Args:
        processor (:class:`~transformers.Wav2Vec2Processor`)
            The processor used for processing the data.
        padding (:obj:`bool`, :obj:`str` or :class:`~transformers.tokenization_utils_base.PaddingStrategy`, `optional`, default: `bool`)
            Select a strategy to pad the returned sequences (according to the model's padding side and padding index)
            among:
            * :obj:`True` or :obj:`'longest'`: Pad to the longest sequence in the batch (or no padding if only a single
              sequence is provided).
            * :obj:`'max_length'`: Pad to a maximum length specified with the argument :obj:`max_length` or to the
              maximum acceptable input length for the model if that argument is not provided.
            * :obj:`False` or :obj:`'do_not_pad'` (default): No padding (i.e., can output a batch with sequences of
              different lengths).
    """

    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)

import evaluate

wer_metric = evaluate.load("wer")

2025-04-05 08:41:28.456292: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register cuFFT factory
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1743822688.465556 27665 cuda_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register factory
E0000 00:00:1743822688.468393 27665 cuda_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register facto

```

2025-04-05 08:41:28.478105: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler fl

```
from evaluate import load
```

```
cer_metric = load("cer")
```

includes both WER and CER

```
def compute_metrics(pred):
    pred_logits = pred.predictions
    pred_ids = np.argmax(pred_logits, axis=-1)

    # Replace padding token (-100) with pad_token_id
    pred.label_ids[pred.label_ids == -100] = processor.tokenizer.pad_token_id

    # Decode predictions and labels to strings
    pred_str = processor.batch_decode(pred_ids)
    label_str = processor.batch_decode(pred.label_ids, group_tokens=False)

    if isinstance(label_str, list):
        if isinstance(pred_str, list) and len(pred_str) == len(label_str):
            for index in random.sample(range(len(label_str)), 3):
                print(f'reference: "{label_str[index]}"')
                print(f'predicted: "{pred_str[index]}"')

        else:
            for index in random.sample(range(len(label_str)), 3):
                print(f'reference: "{label_str[index]}"')
                print(f'predicted: "{pred_str}"')

    # Compute WER
    wer = wer_metric.compute(predictions=pred_str, references=label_str)

    # Compute CER
    cer = cer_metric.compute(predictions=pred_str, references=label_str)

    return {"wer": wer, "cer": cer}
```

```
from transformers import Wav2Vec2ForCTC
```

```
model = Wav2Vec2ForCTC.from_pretrained(
    "facebook/wav2vec2-xl-r-300m",
    #'facebook/wav2vec2-large-xlsr-53',
    attention_dropout=0.05,
    hidden_dropout=0.1,
    feat_proj_dropout=0.1,
    mask_time_prob=0.05,
    layerdrop=0.01377,
    gradient_checkpointing=True,
    ctc_loss_reduction="mean",
    ctc_zero_infinity=True,
    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-xl-r-300m and are ne
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

```
# from transformers import Wav2Vec2ForCTC
```

```
# model = Wav2Vec2ForCTC.from_pretrained(
#     'facebook/wav2vec2-large-xlsr-53',
#     attention_dropout=0.05,
#     activation_dropout=0.1,
#     hidden_dropout=0.1,
#     feat_proj_dropout=0.01249,
#     final_dropout=0.0,
#     mask_time_prob=0.05,
#     mask_time_length=10,
#     mask_feature_prob=0,
#     mask_feature_length=10,
#     layerdrop=0.01377,
#     gradient_checkpointing=True,
```

```

#     ctc_loss_reduction="mean",
#     ctc_zero_infinity=True,
#     bos_token_id=processor.tokenizer.bos_token_id,
#     eos_token_id=processor.tokenizer.eos_token_id,
#     pad_token_id=processor.tokenizer.pad_token_id,
#     vocab_size=len(processor.tokenizer.get_vocab())
# )

model.freeze_feature_encoder()

# import huggingface_hub

# huggingface_hub.login()

#repo_name = "wav2vec2-kashmiri-jhon-data-one"

save_dir = "/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/training_ε

from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir=save_dir,
    group_by_length=True,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    gradient_accumulation_steps=2,
    evaluation_strategy="steps",
    num_train_epochs=30,
    fp16=True,
    save_steps=500,
    eval_steps=500,
    logging_steps=10,
    learning_rate=4e-4,
    warmup_steps=250,
    save_total_limit=2,
    dataloader_num_workers=24
)

🔗 /home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/training_args.py:1594: FutureWarning: `eval
warnings.warn(

# from transformers import TrainingArguments

# training_args = TrainingArguments(
#     output_dir=repo_name,
#     group_by_length=True,
#     per_device_train_batch_size=2,
#     gradient_accumulation_steps=2,
#     eval_strategy="steps",
#     num_train_epochs=20,
#     gradient_checkpointing=True,
#     fp16=True,
#     save_steps=20,
#     eval_steps=20,
#     logging_steps=40,
#     learning_rate=3e-4,
#     warmup_steps=50,
#     save_total_limit=2,
#     push_to_hub=True,
# )

# import numpy as np
# from transformers import Trainer
# trainer = Trainer(
#     model=model,
#     data_collator=data_collator,
#     args=training_args,
#     compute_metrics=compute_metrics,
#     train_dataset=train_dataset,
#     eval_dataset=test_dataset,
#     tokenizer=processor.feature_extractor,
# )

import numpy as np
from transformers import Trainer

# Assuming processor is an instance of Wav2Vec2Processor (or similar for your model)

```


```
print("step1")
train_result = trainer.train()
print("step2")


metrics = train_result.metrics
print("step3")
max_train_samples = len(train_dataset)
metrics["train_samples"] = min(max_train_samples, len(train_dataset))
print("step4")
trainer.save_model()
print("model created!")
trainer.log_metrics("train", metrics)
trainer.save_metrics("train", metrics)
trainer.save_state()
```

[4860/4860 7:19:40, Epoch 29/30]

Step	Training Loss	Validation Loss	Wer	Cer
500	3.331200	3.327143	1.000000	1.000000
1000	1.248900	1.150660	0.881418	0.331099



```
trainer.evaluate() # Evaluate the model on the test dataset
```

 `Trainer.tokenizer` is now deprecated. You should use `Trainer.processing_class` instead.
`Trainer.tokenizer` is now deprecated. You should use `Trainer.processing_class` instead.

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/torch/utils/data/dataloader.py:624: UserWarning: This DataLoader will create 160 worker processes on CUDA device 0. This requires that your compute mode be set to 'spawn' in the torch config. Please see the following documentation for more details: https://pytorch.org/docs/stable/notes/cuda.html#torch-config-compute-mode. (Triggered internally at /home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/torch/cuda/__init__.py:226.)`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`


 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`


 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`


 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`


 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`


 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`


 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

```
# trainer.push_to_hub()

import torch
import torchaudio
import librosa
import numpy
from transformers import Wav2Vec2ForCTC, Wav2Vec2Processor
from transformers import Wav2Vec2Processor
```

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/tqdm/auto.py:219: TqdmWarning: IPProgress not found. Please`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

 `/home/muzaffar/anaconda3/envs/tf14/lib/python3.11/site-packages/transformers/models/wav2vec2/processing_wav2vec2.py:174: warnings.warn()`

```
model_name_or_path = "/home/muzaffar/Desktop/Research/papers/5-paper Wav2Vec/5. Wave2vec Whisper Paper/KASHMIRI/experiment5/"
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(model_name_or_path, device)
```

```
processor = Wav2Vec2Processor.from_pretrained(model_name_or_path)
model = Wav2Vec2ForCTC.from_pretrained(model_name_or_path).to(device)
```

```
def speech_file_to_array_fn(batch):
    speech_array, sampling_rate = torchaudio.load(batch["file_path"])
    speech_array = speech_array.squeeze().numpy()
    #speech_array = librosa.resample(np.asarray(speech_array), sampling_rate, processor.feature_extractor.sampling_rate)
    speech_array = librosa.resample(y=np.asarray(speech_array), orig_sr=sampling_rate, target_sr=processor.feature_extractor.sampling_rate)

    batch["speech"] = speech_array
    return batch
```

```
def predict(batch):
    features = processor(
        batch["speech"],
        sampling_rate=processor.feature_extractor.sampling_rate,
        return_tensors="pt",
        padding=True
    )

    input_values = features.input_values.to(device)
    #attention_mask = features.attention_mask.to(device)
    attention_mask = features.attention_mask.to(device) if "attention_mask" in features else None

    with torch.no_grad():
        logits = model(input_values, attention_mask=attention_mask).logits
```


9/11

https://colab.research.google.com/drive/1smhDH6U2lscA0sSOfxTM1Y6ehk_UfRIk 10/11

11/11