Processing and classifying images

WORKING WITH HUGGING FACE

Jacob H. Marquez
Lead Data Engineer

datacamp

Pixels in images

Pixels in images

- Images are made of pixels
- Pixels contain info about color and more
- Number of pixels from 1 to infinite

Pixels in images

- Images are made of pixels
- Pixels contain info about color and more
- Number of pixels from 1 to infinite
- More pixels = higher resolution

Importance of pre-processing images

- Models have expectations for the images
- Maintains consistency
- Target specific parts of images

Common pre-processing steps

- Cropping
- Resizing

Common pre-processing steps

Cropping

Common pre-processing steps

Cropping

- Removing unwanted portions of the image

Resizing

- Change dimensions of the image

Common pre-processing steps

Cropping

- Removing unwanted portions of the image

Resizing

- Change dimensions of the image

Transforming images

```
from transformers import image_transforms
```

```
from PIL import Image

original_image = Image.open("my_image.jpeg")
```

```
import numpy as np

image_array = np.array(original_image)
```

```
array([[[43, 32, 14],...,[77, 47, 23]]], dtype=uint8)
```

Cropping an image

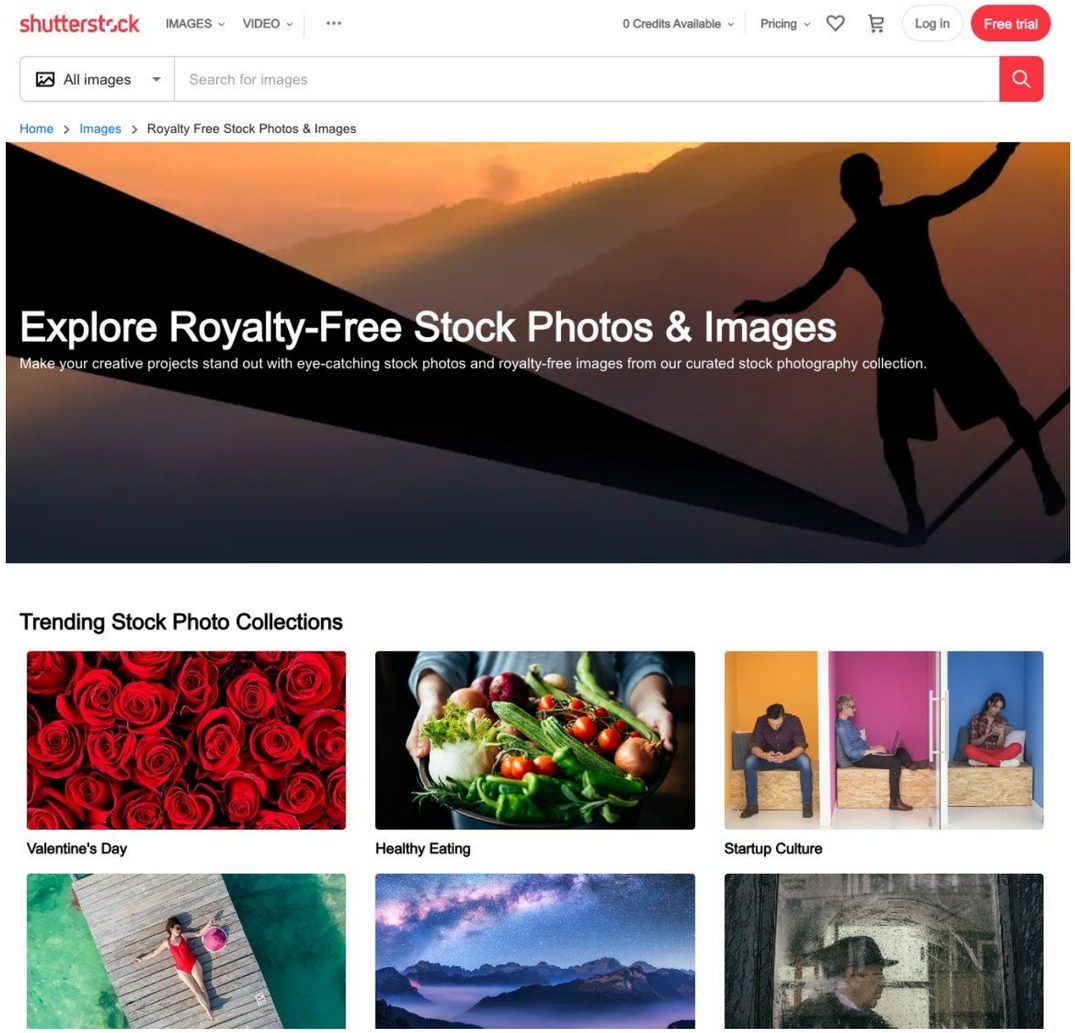
```
cropped_image = image_transforms \
    .center_crop(
      image=np.ndarray,
      size=(1000, 1000)
)
```

Original image

Cropped image

What is image classification?

- The process of assigning labels to an image
- Stock photography



[1] https://www.shutterstock.com/photos

What is image classification?

- The process of assigning labels to an image
- Stock photograph
- Agriculture

[1] https://phys.org/news/2021-03-crop-diseasesthere-app.html

What is image classification?

- The process of assigning labels to an image
- Stock photograph
- Agriculture
- Wildlife monitoring

Image classification pipeline

```
from transformers import pipeline


classifier = pipeline(task="image-classification",
                      model="google/vit-base-patch16-224")
```

Image classification pipeline

```
# http link to image
classifier("https://www.pexels.com/photo/elephants.jpeg")
```

```
# local path to image
classifier("/my_folder/elephants.jpeg")
```

```
# PIL image
from PIL import Image
image = Image.open("/my_folder/elephants.jpeg")
classifier(image)
```

```
classifier(["image1.jpeg", "image2.jpeg"])
```

Top K parameter

```
from transformers import pipeline

classifier = pipeline(task="image-classification",
                      model="google/vit-base-patch16-224")

results = classifier(image, top_k=2)

print(results[0]['label'])
```

```
African elephant, Loxodonta africana
```

Let's practice!

WORKING WITH HUGGING FACE

Question answering and multi-modal tasks

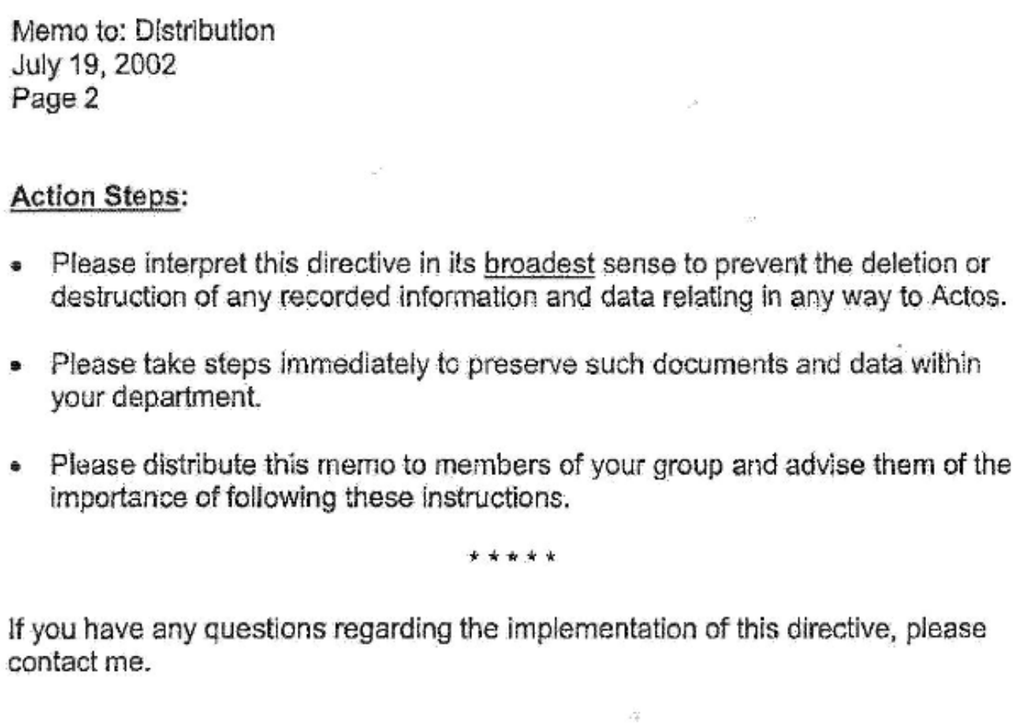
WORKING WITH HUGGING FACE

Jacob H. Marquez
Lead Data Engineer

datacamp

What is document question and answering?

- Answering questions about content of document
- Document is a text-based image
- Question is specific to the document
- Answer can be direct quote or paraphrased response



Memo to: Distribution
July 19, 2002
Page 2

Action Steps:

- Please interpret this directive in its broadest sense to prevent the deletion or destruction of any recorded information and data relating in any way to Actos.
- Please take steps immediately to preserve such documents and data within your department.
- Please distribute this memo to members of your group and advise them of the importance of following these instructions.

* * * * *

If you have any questions regarding the implementation of this directive, please contact me.

Question: "What are the action steps?"

Visual question and answering

- Ask a question of an actual image or video
- Pipeline requires the image and question

Question: "What type of animal is in this picture?"

Use cases: information retrieval

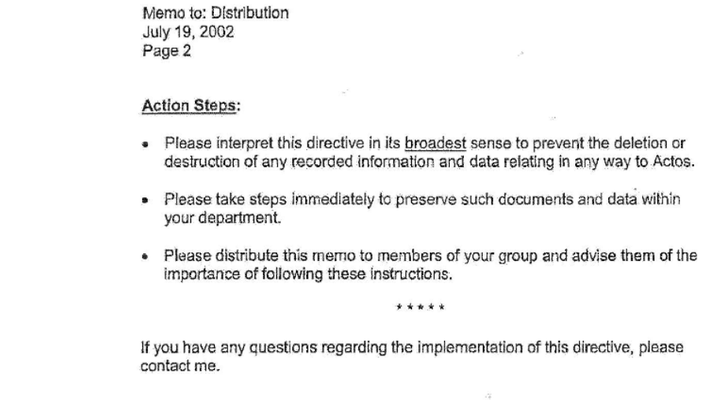
Customer Support

Legal Compliance

Search

Preprocessing for multi-modal tasks

- Multi-modal
- Process each data type
- Tokenization for text
- Resizing for images



Memo to: Distribution
July 19, 2002
Page 2

Action Steps:

- Please interpret this directive in its broadest sense to prevent the deletion or destruction of any recorded information and data relating in any way to Actos.
- Please take steps immediately to preserve such documents and data within your department.
- Please distribute this memo to members of your group and advise them of the importance of following these instructions.

*****

If you have any questions regarding the implementation of this directive, please contact me.

Document Q&A pipeline

```
from transformers import pipeline

dqa = pipeline(
    task="document-question-answering",
    model="naver-clova-ix/donut-base-finetuned-docvqa")
document_image = "memo.jpg"
question_text = "What is this memo about?"

results = dqa(document_image, question_text)
```

Results of document Q&A pipeline

```
print(results)
```

```
{
  "score": 0.789,
  "start": 1,
  "end": 2,
  "answer": "distribution",
  "words": [102]
}
```

- `score` is the probability of the answer
- `answer` is the answer to the question
- `start` is the start word index of the answer
- `end` is the last word index of the answer
- `words` is a list of all indices for each word in the answer

```
dqa(image=image,
    question=question,
    max_answer_len=15)
```

- 79% probability the memo is about distribution

Visual Q&A pipeline

```
from transformers import pipeline

vqa = pipeline(
    task="visual-question-answering",
    model="dandelin/vilt-b32-finetuned-vqa"
    )
```

```
result = vqa(
    image="image.jpeg",
    question="what's the person wearing?")
```

Results of visual Q&A pipeline

```
print(result)
```

```
[
    {'score': 0.9795706272125244,
    'answer': 'hat'
    },
    ...,
    {'score': 0.02153933234512806,
    'answer': 'hoodie'
    }
]
```

- `label` label identified by the model
- `score` probability of the label from the model

Let's practice!

WORKING WITH HUGGING FACE

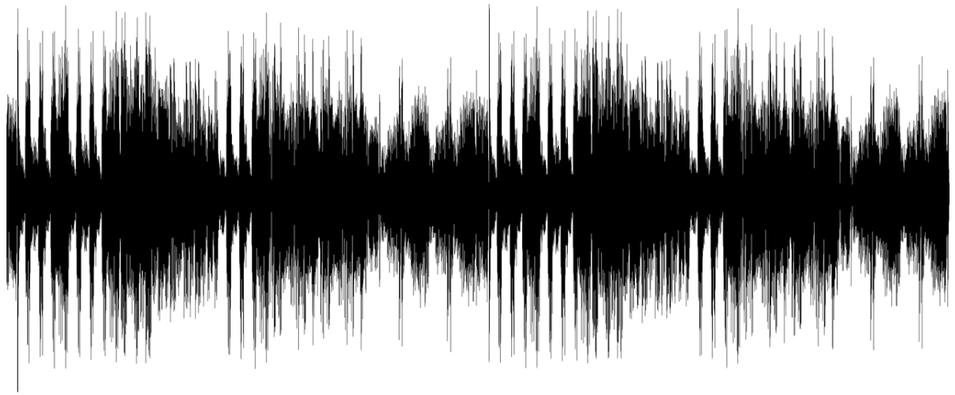
Audio classification

WORKING WITH HUGGING FACE

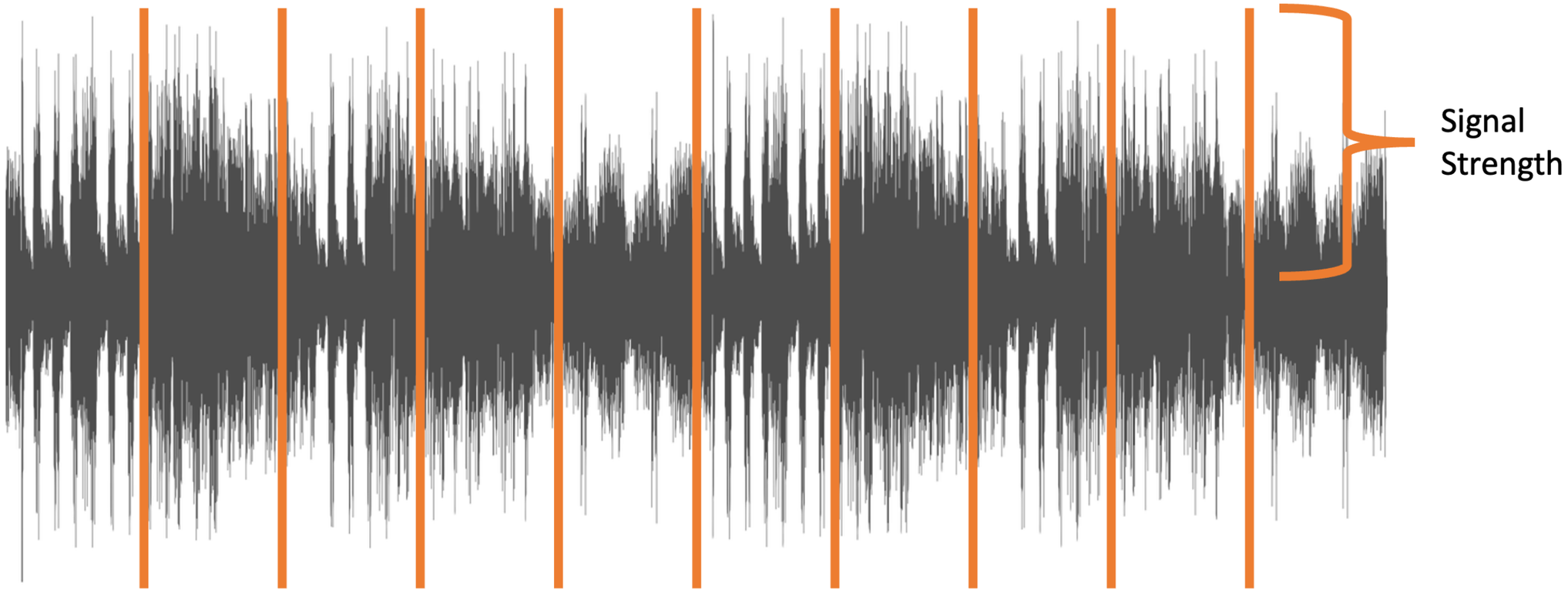
Jacob H. Marquez
Lead Data Engineer

What is audio data?

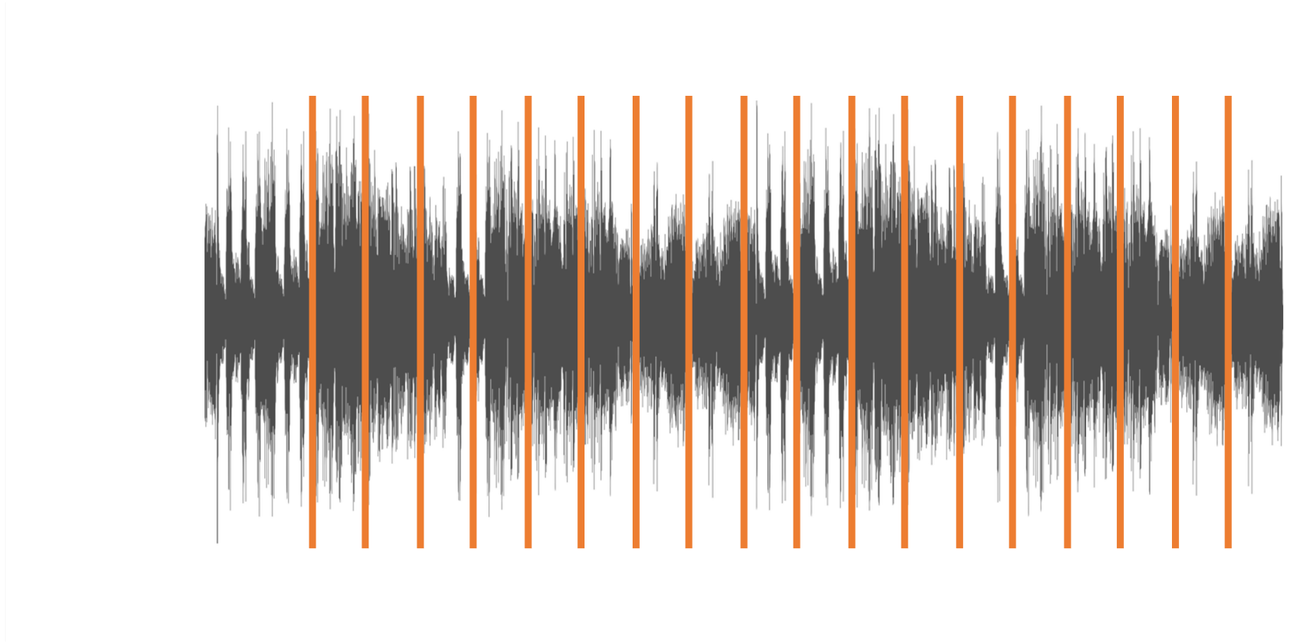
- Continuous signal called a sound wave
- Length and amplitude
- Converted into series of discrete values
- Results in a digital representation
- Conversion helps ML algorithms process and use audio
- Sampling is an important step



The importance of sampling

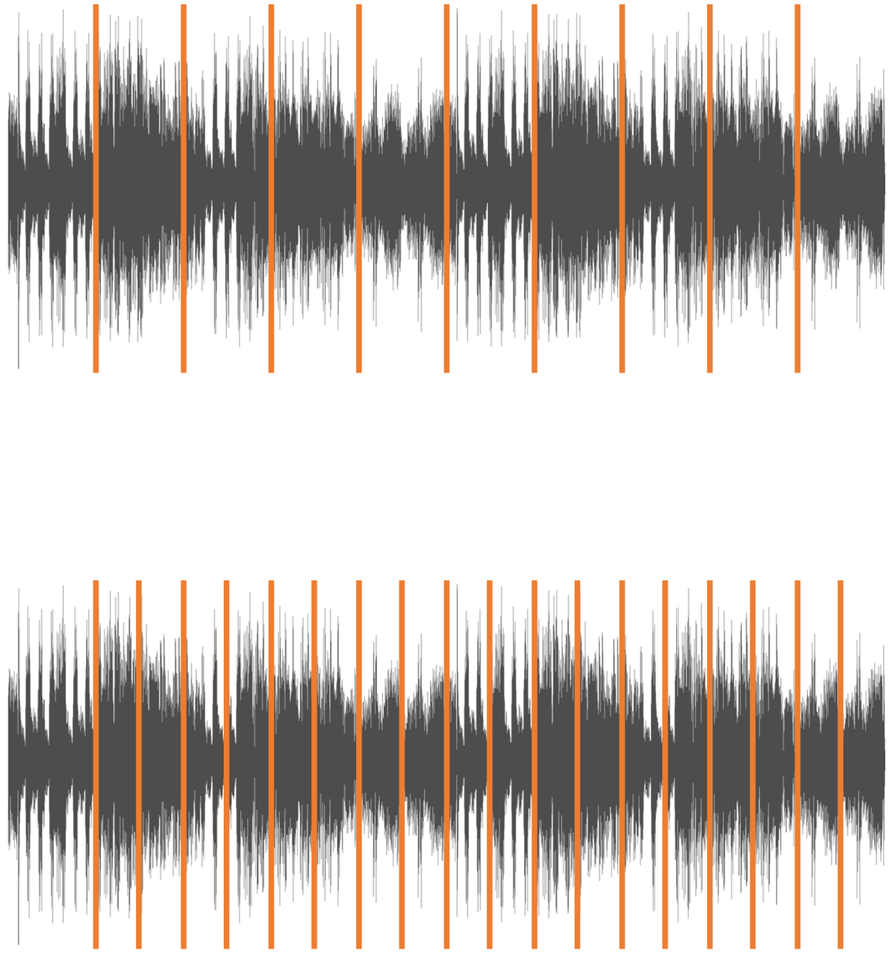


The importance of sampling



- Speech models trained at 16kHz
- Sampling rate specified in the model card

Resampling



- Aligns sample rates across all files
- Ensures consistency
- Helps with processing

Resampling using Hugging Face

```
from datasets import Audio


songs = songs.cast_column("audio", Audio(sampling_rate=16_000))
```

Finding the sampling rate:

```
print(songs[0]["audio"]["sampling_rate"])
```

```
16_000
```

Filtering

Benefits

- Ensure enough audio for inference or training
- Reduce computation by limiting file size

```
import librosa
```

Filtering

```
durations = []

for row in songs["path"]:
    durations.append(librosa.get_duration(path=row))

songs.add_column("duration", durations)

songs = dataset.filter(
    lambda d: d < 10.0, input_columns=["duration"]
)
```

What is audio classification?

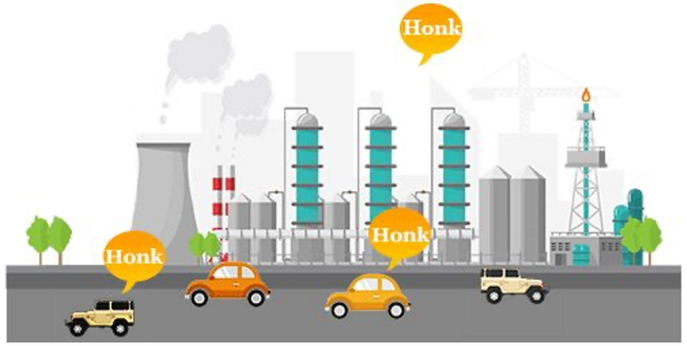
Definition: *process of assigning one or more labels to audio clips based on its content*

Language Identification

What is audio classification?

Definition: *process of assigning one or more labels to audio clips based on its content*

Language Identification

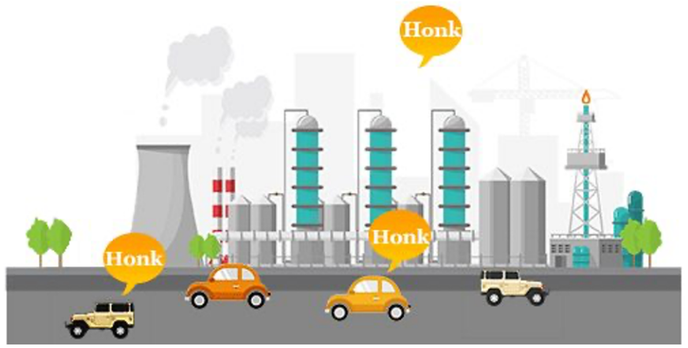


Environmental Sounds

What is audio classification?

Definition: *process of assigning one or more labels to audio clips based on its content*

Language Identification



Environmental Sounds

Speaker Identification

Using Hugging Face pipelines

```
from transformers import pipeline

classifier = pipeline(task="audio-classification",
                      model="superb/wav2vec2-base-superb-ks")

genreClassifier = pipeline(task="audio-classification",
                           model="mtg-upf/discogs-maest-30s-pw-73e-ts")
```

Using Hugging Face pipelines

```
audio = songs[0]['audio']['array']
prediction = genreClassifier(audio)

print(prediction)
```

```
[
{'score': 0.07648807018995285, 'label': 'Non-Music---Field Recording'},
...
{'score': 0.05880315974354744, 'label': 'Electronic---Noise'}
]
```

[1] https://huggingface.co/mtg-upf/discogs-maest-30s-pw-73e-ts

Using Hugging Face pipelines

```
prediction = genreClassifier(audio, top_k=1)
```

```
[
{'score': 0.07648807018995285, 'label': 'Non-Music---Field Recording'}
]
```

Let's practice!

WORKING WITH HUGGING FACE

Automatic speech recognition

WORKING WITH HUGGING FACE

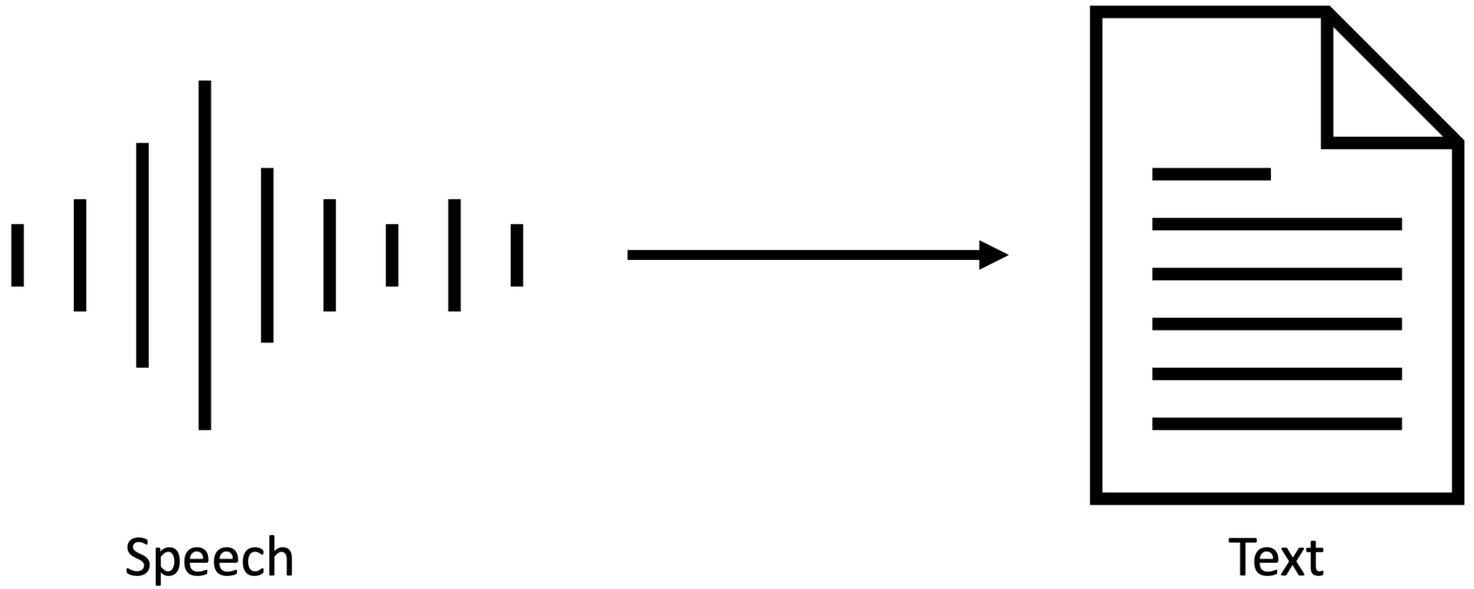
Jacob H. Marquez
Lead Data Engineer

datacamp

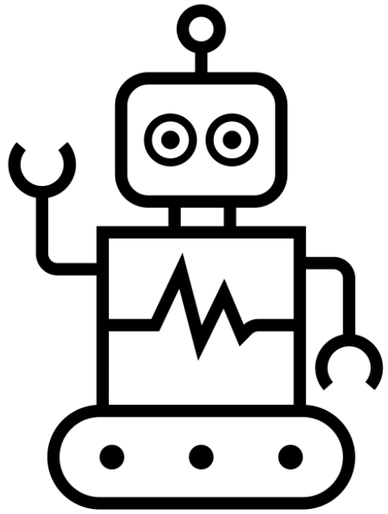
What is automatic speech recognition?

Speech

What is automatic speech recognition?

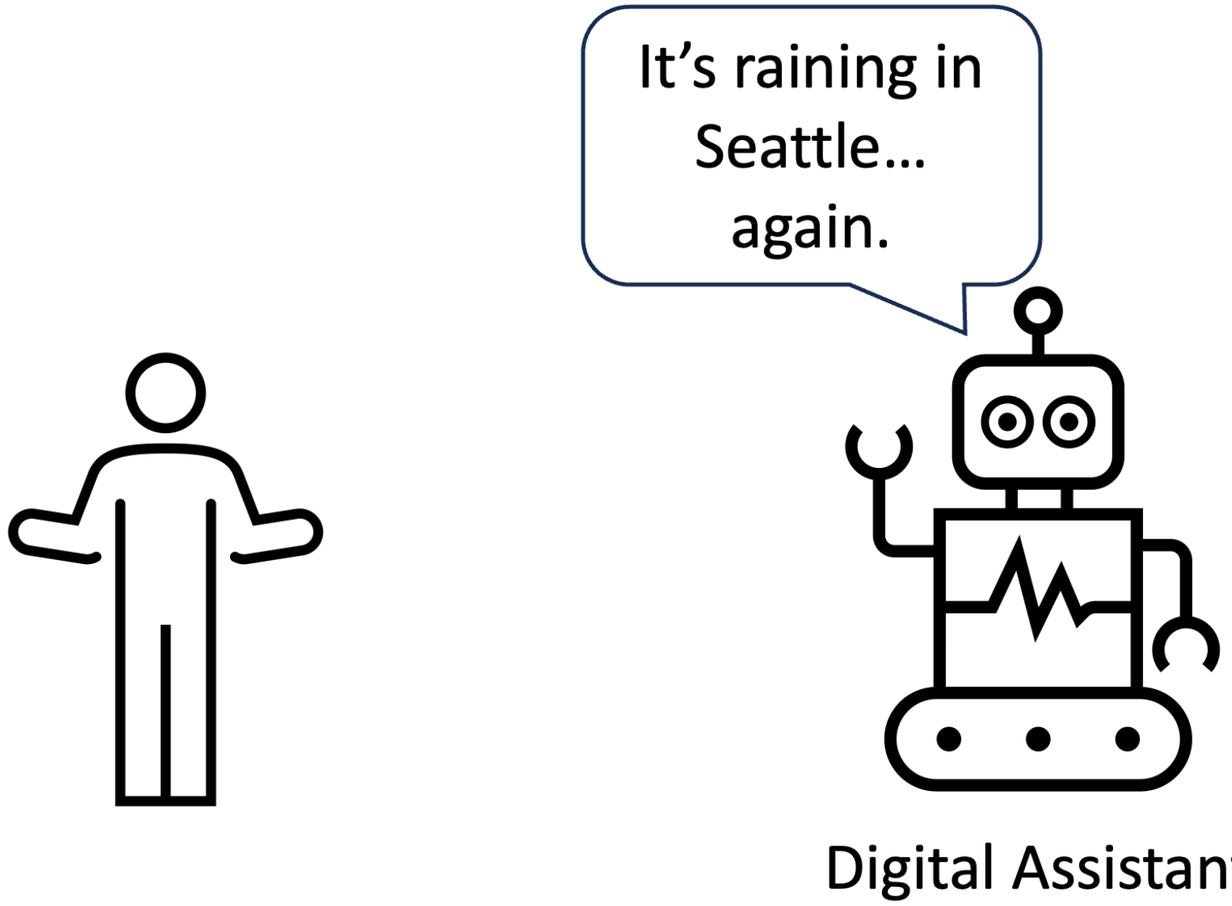


Use cases of ASR



Digital Assistant

Use cases of ASR



Use cases for ASR

- Create transcripts
- Finding relevant documentation and solutions

Use cases for ASR

Models for ASR

Wav2Vec

```
model_id:
"facebook/wav2vec2-base-960h"
```

Models for ASR

`Wav2Vec`

```
model_id:
"facebook/wav2vec2-base-960h"
```

`Whisper`

```
model_id:
”openai/whisper-base"
```

- Whisper performs better with punctuation and casing.

Instantiating a pipeline for ASR

```
transcriber = pipeline(task="automatic-speech-recognition",
                       model="facebook/wav2vec2-base-960h")

# Path to audio file
transcriber("my_audio.wav")

# Numpy array
transcriber(numpy_audio_array)

# Dictionary
transcriber({"sampling_rate" = 16_000,"raw" = "my_audio.wav"})
```

Results from a pipeline

```
sampling_rate = 16_000
dataset = dataset.cast_column("audio", Audio(sampling_rate=sampling_rate))

input = data[0]['audio']['array']

prediction = transcriber(input)
```

```
print(prediction)
```

```
"what game do you want to play"
```

Predicting over a dataset

```
def data():
    for i in range(dataset):
        yield dataset[i]['audio']['array'], dataset[i]['sentence'].lower()

output = []

for audio, sentence in data():
    prediction = transcriber(audio)
    output.append((prediction, sentence))
```

```
[("what a nice black shirt", "what a nice blue shirt"), ...]
```

Evaluating ASR systems

- Word Error Rate (WER)
- Based on Levenshtein Distance
- Metric for the difference between two sequences

$$\frac{(Substitutions + Insertions + Deletions)}{Number of Words Spoken}$$

- Range from 0 to 1
- Smaller value indicates closer similarity

[1] https://en.wikipedia.org/wiki/Levenshtein_distance

Word Error Rate

Correct	Prediction
I love DataCamp courses on AI!	I love DataCamp portraits on hay!

- 2 substitutions required to match to correct

```
2 / 6 = 0.33
```

Computing WER using Hugging Face

```
from evaluate import load

# Instantiate word error rate metric
wer = load("wer")

# Save true sentence as reference
reference = data[0]['sentence']

predictions = "I love DataCamp portraits on hay"
```

[1] https://huggingface.co/spaces/evaluate-metric/wer

Computing WER using Hugging Face

```
# Compute the WER between predictions and reference
wer_score = wer.compute(
  predictions=[prediction],
  references=[reference]
  )

print(wer_score)
```

```
0.33
```

Let's practice!

WORKING WITH HUGGING FACE