Building Real-world ASR Systems

Harisankar Haridas
21 Nov, 2020

Talk outline

Part 1: Basic concepts → State-of-the-art

Break

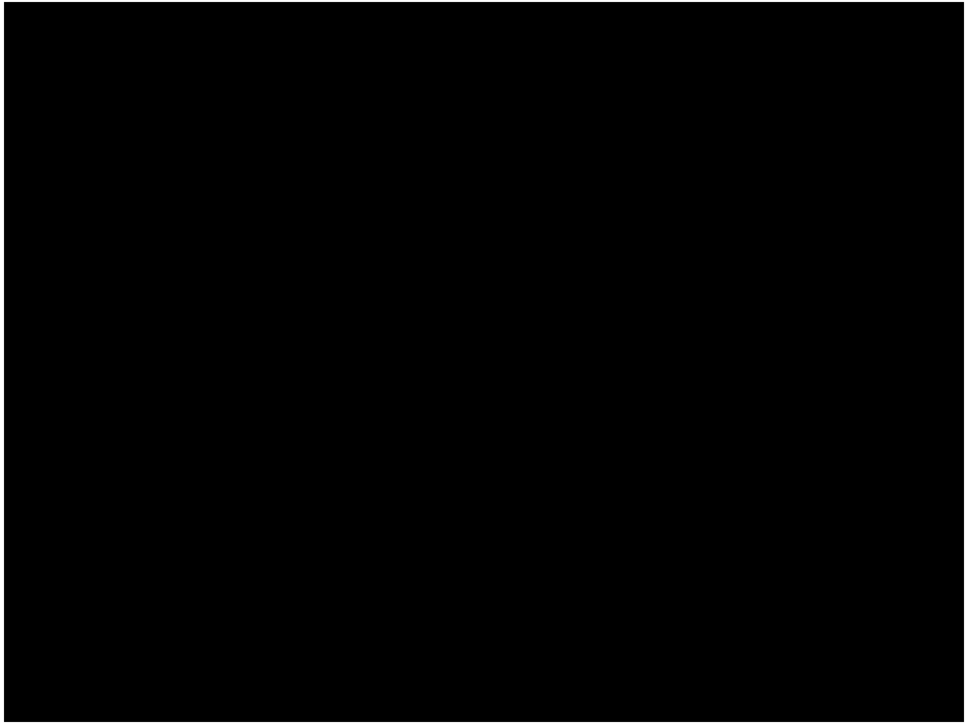
Part 2: Practical challenges and overcoming them

Principle

Build!

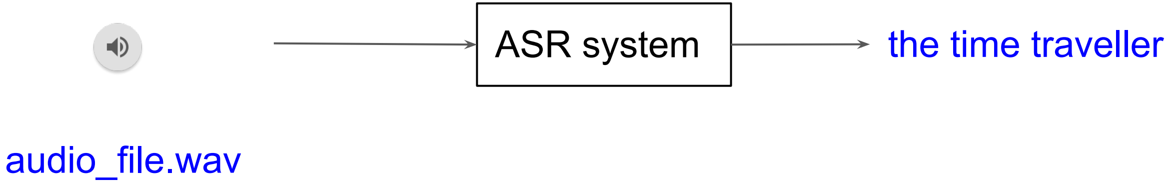
What helps right now // less emphasis on 70+ years history of ASR

Demo

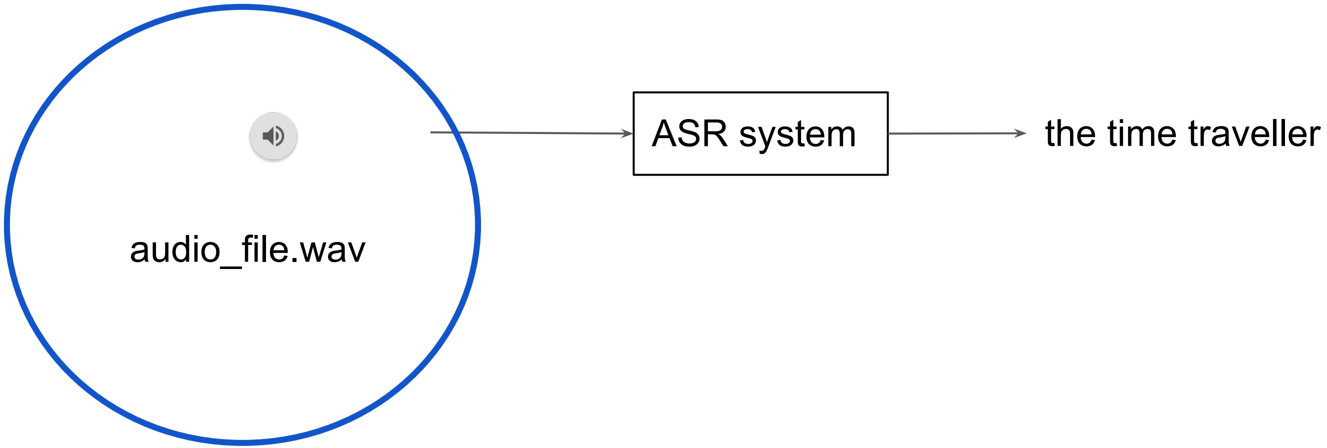


From https://www.flickr.com/photos/8749778@N06/3733627940/

Input-output

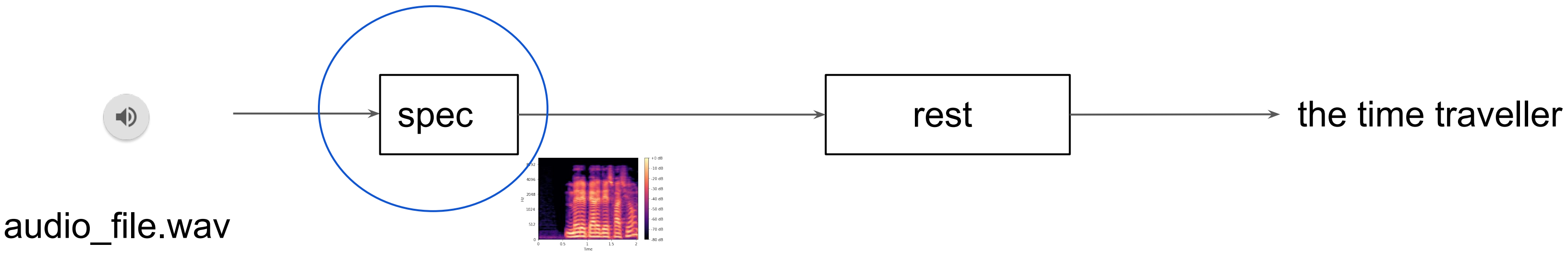


Input-output

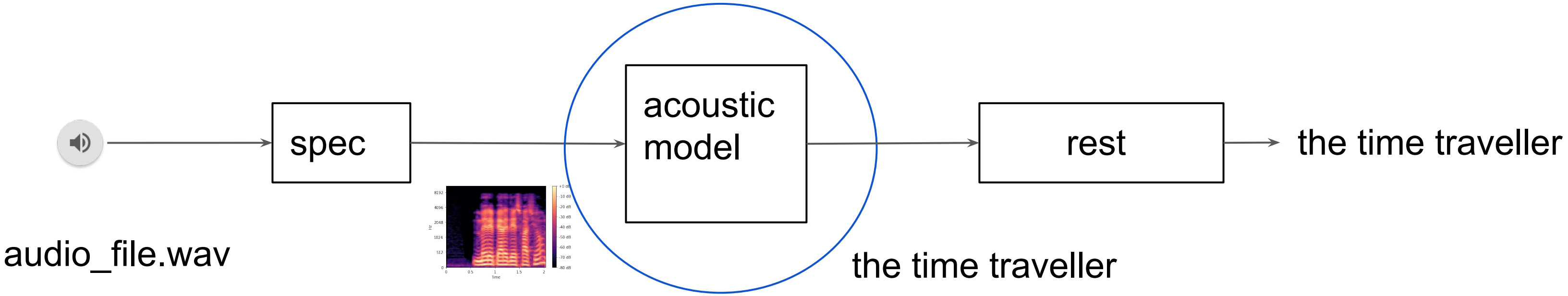


Demo: Notebook

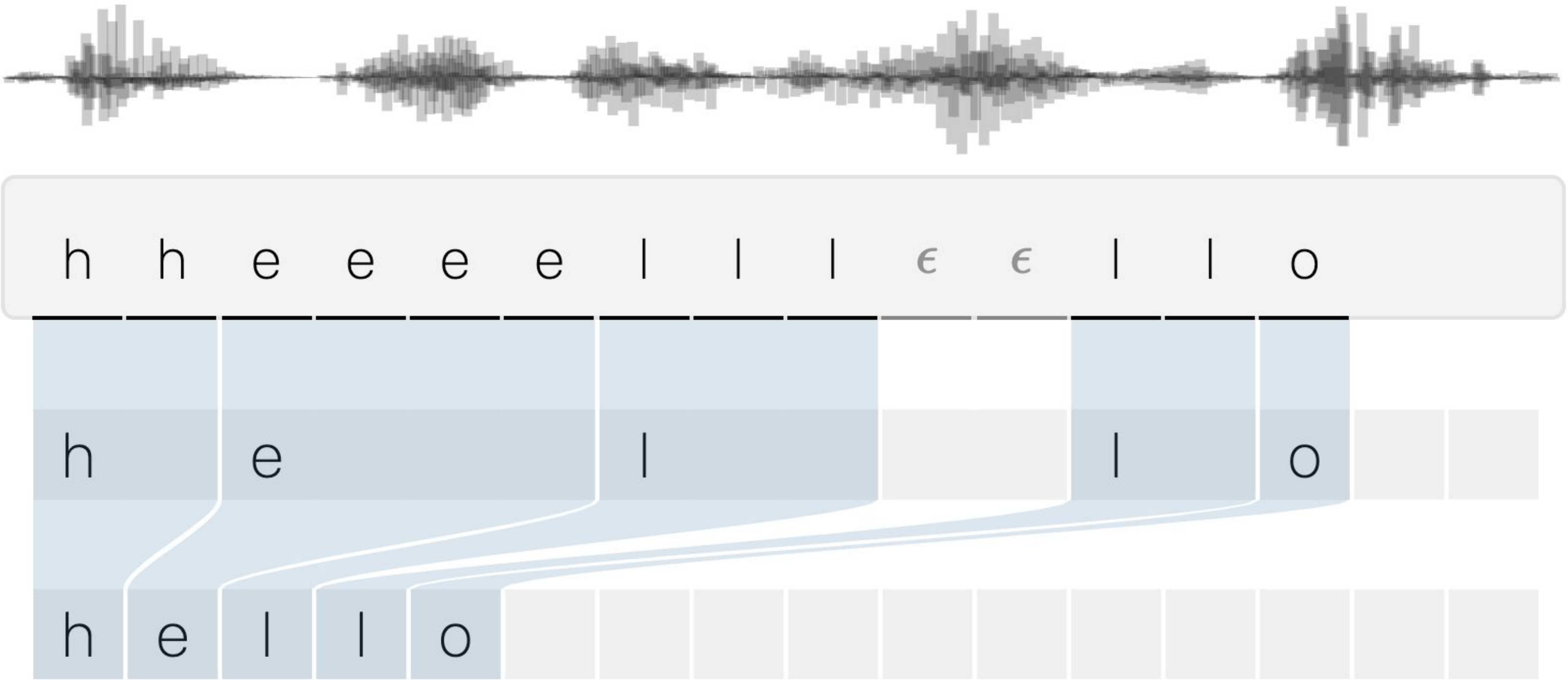
Input-output



Input-output

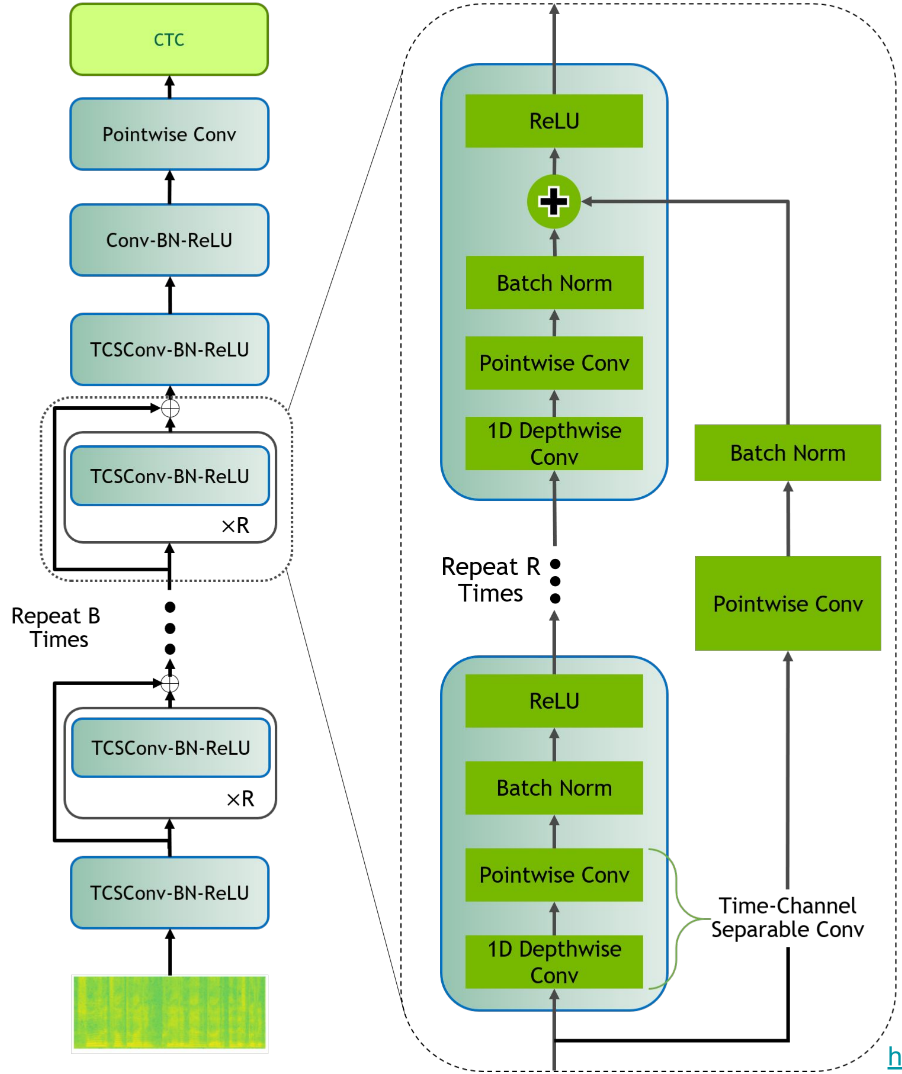


Demo: one char per time step



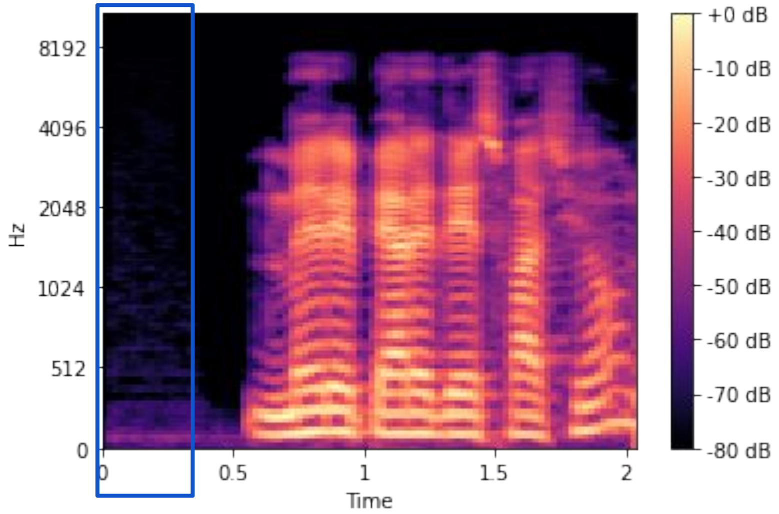
Quartznet acoustic model [2019]

1. 1D time-channel separable convolutions [2014]
2. CTC loss [2006]



https://arxiv.org/abs/1910.10261

(1) 1D time-channel separable convolutions



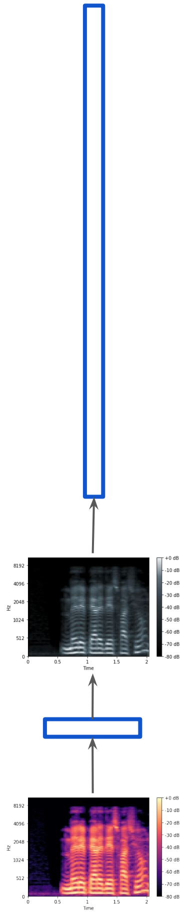
TxC -> 330 M parameters, Jasper [2019]

1 D convolution

Tx1 + Cx1 -> 19 M parameters

Cx1

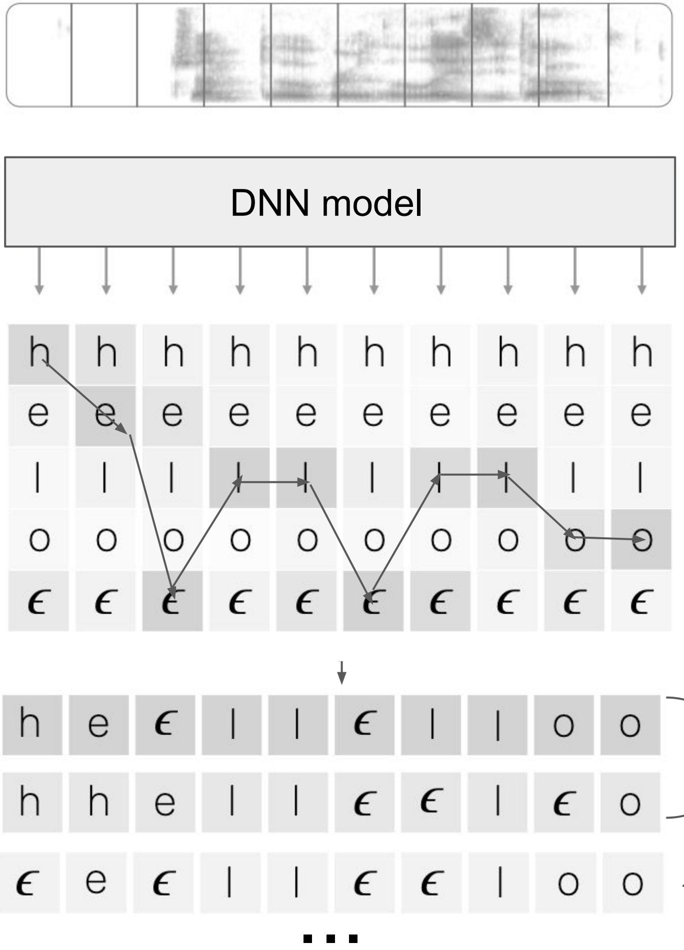
time-wise conv

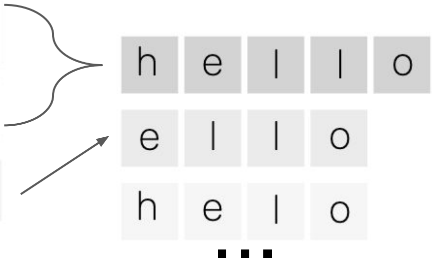


Tx1

channel-wise conv

(2) CTC loss





Loss = - log P ("hello" | audio)

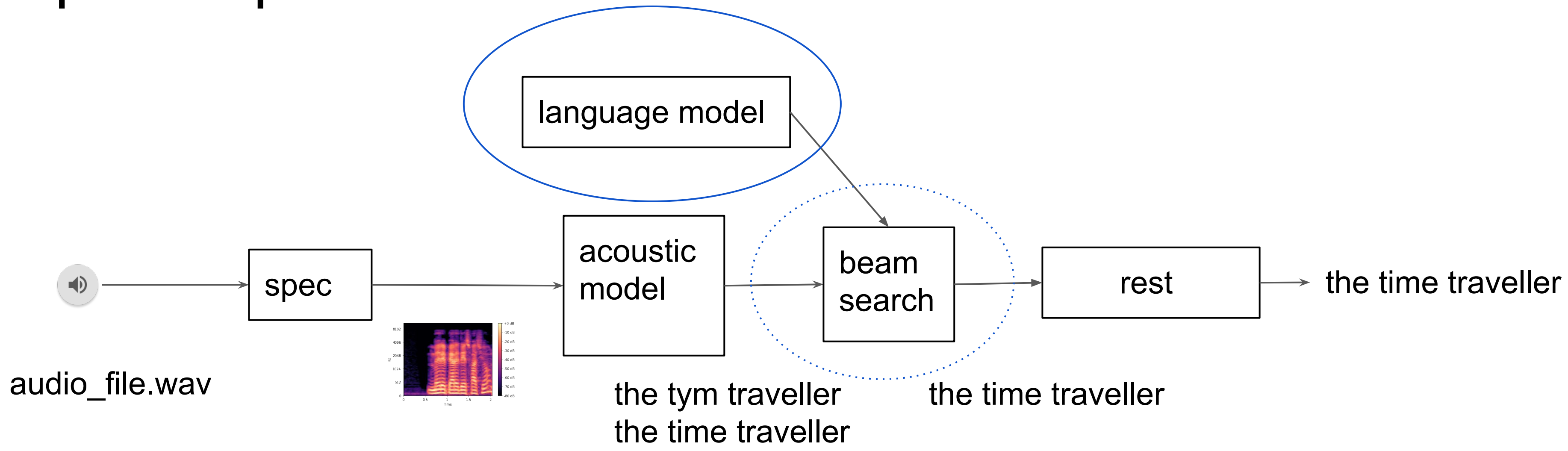
$$P(\text{“hello”} \mid \text{audio}) = \sum_{\text{variations of “hello”}} P(\text{variation of “hello”} \mid \text{audio})$$

P("heεllε..") = p('h') x p('e') x p('ε') x p('l') x …

Dynamic programming-based computation

https://distill.pub/2017/ctc/

Input-output



Need for language model

Training data for acoustic model

- ~1 hour of Audio corresponds to (less than) ~1000 sentences
- Expensive to collect

Tough to distinguish between similar sounding hypotheses

- the tym traveller
- the time traveller

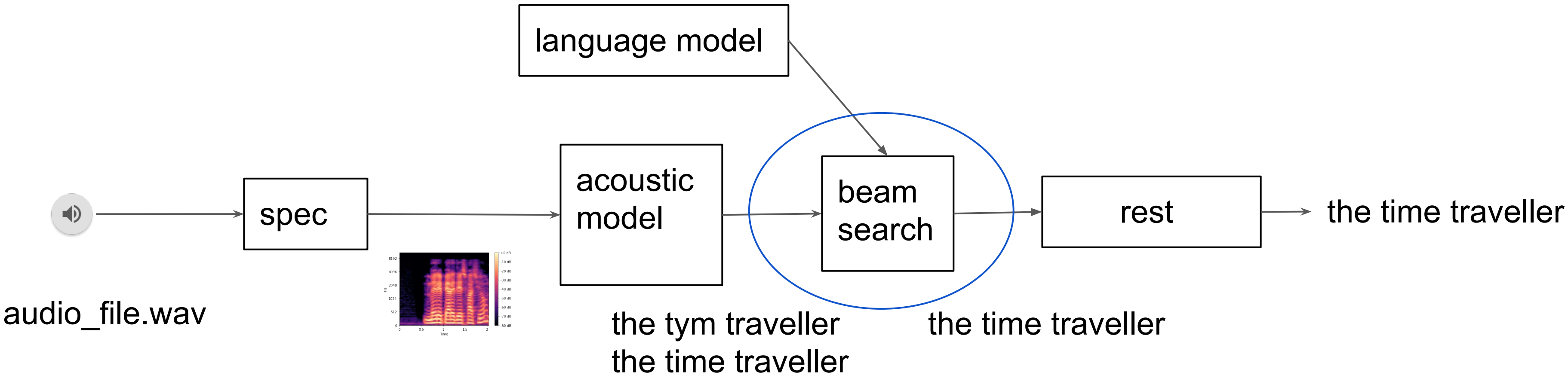
Language models trained only on (millions of lines of) text help here

- According to language model: p("the tym traveller") < p("the time traveller")
- Usually computed through word-combination count-based ngrams

After incorporating language model:

$$\text{Final output} \cong \underset{\text{text} \in \{\text{all possible texts}\}}{\operatorname{argmax}} \; p(\text{text}|\text{audio}) * p(\text{text})$$

Input-output



Beam search

Final output ≅ argmax p(text|audio) * p(text)

text ∈ {all possible texts}

P("hello" | audio, lm) = $\sum$ P (variation of "hello" | audio) x P ("hello")

variations of "hello"   lm

All possible texts = all paths which can be drawn through the matrix

// can result in C^T possibilities (28^88)

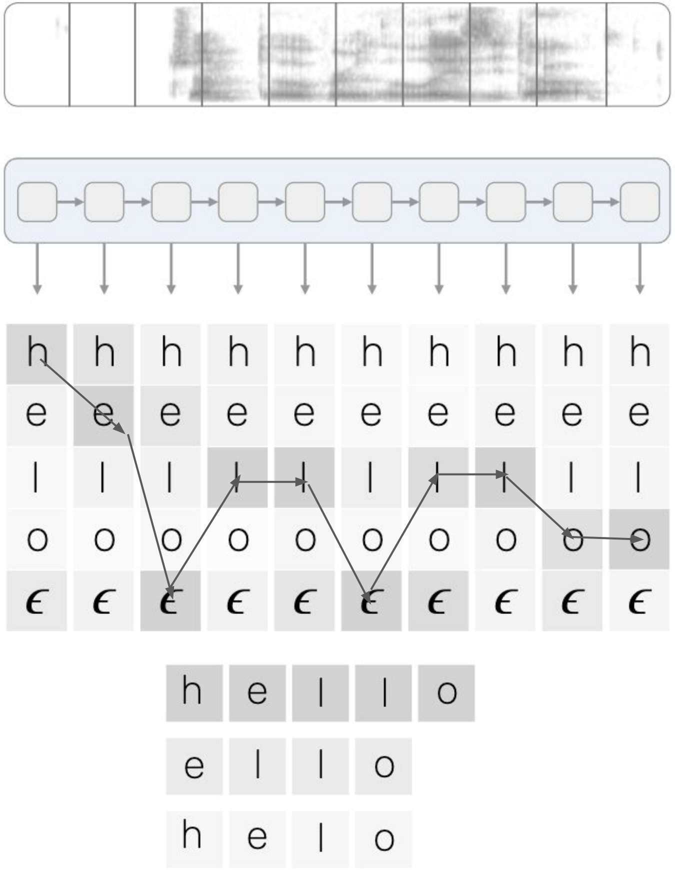
Solution: beam search

Go from left to right in time steps

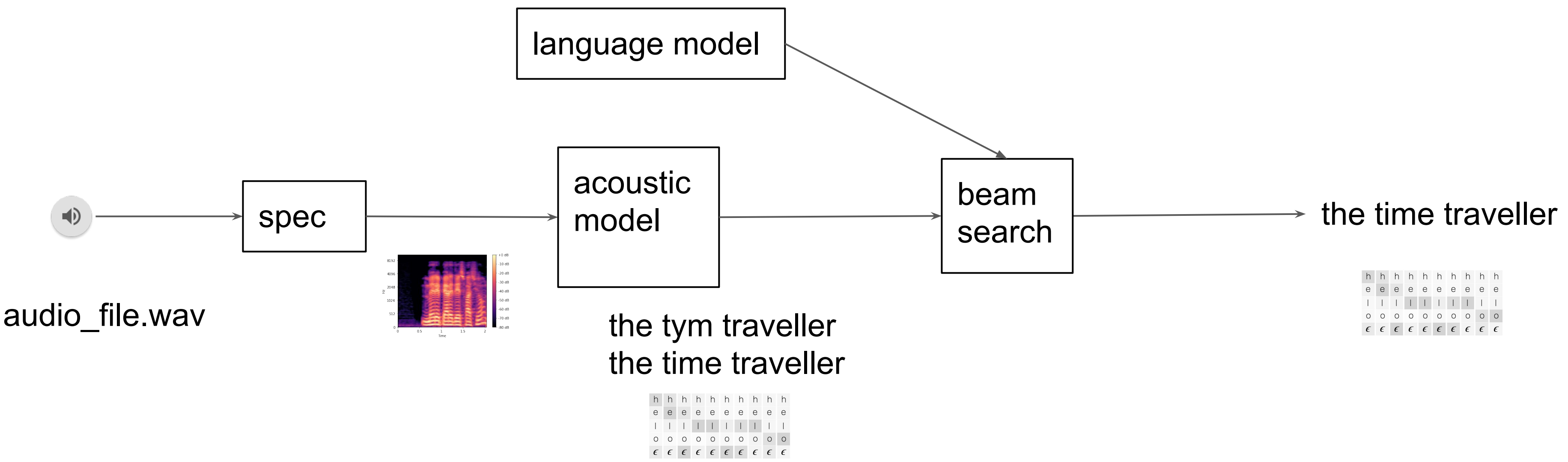
At time step t

Compute the probabilities of seen prefixes till t

Keep only the top-k prefixes for next step



Input-output



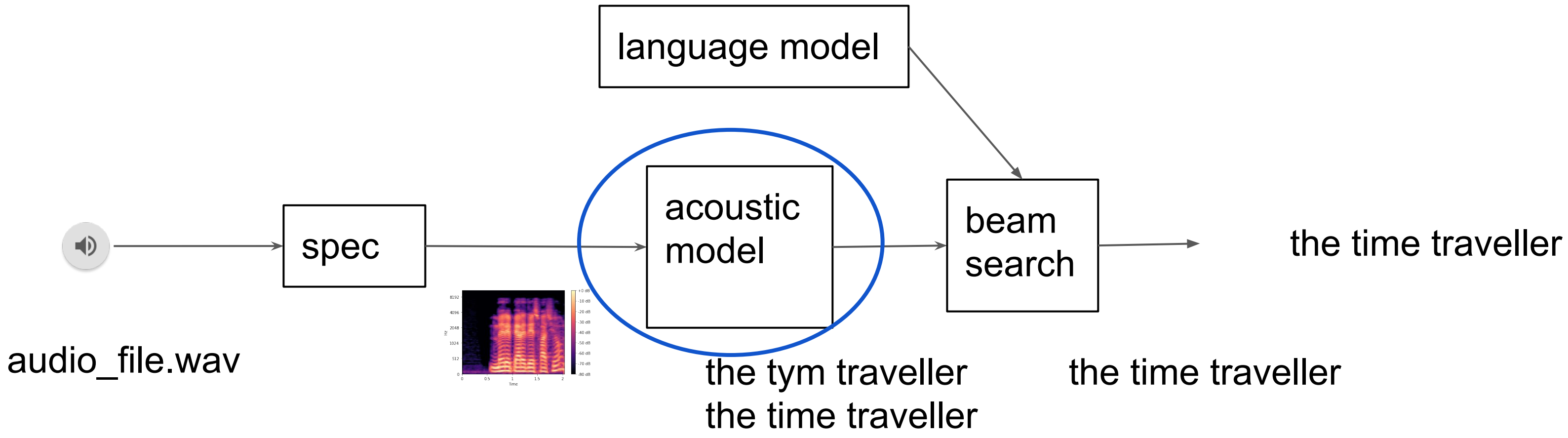
Part 1 - Q&A

Next

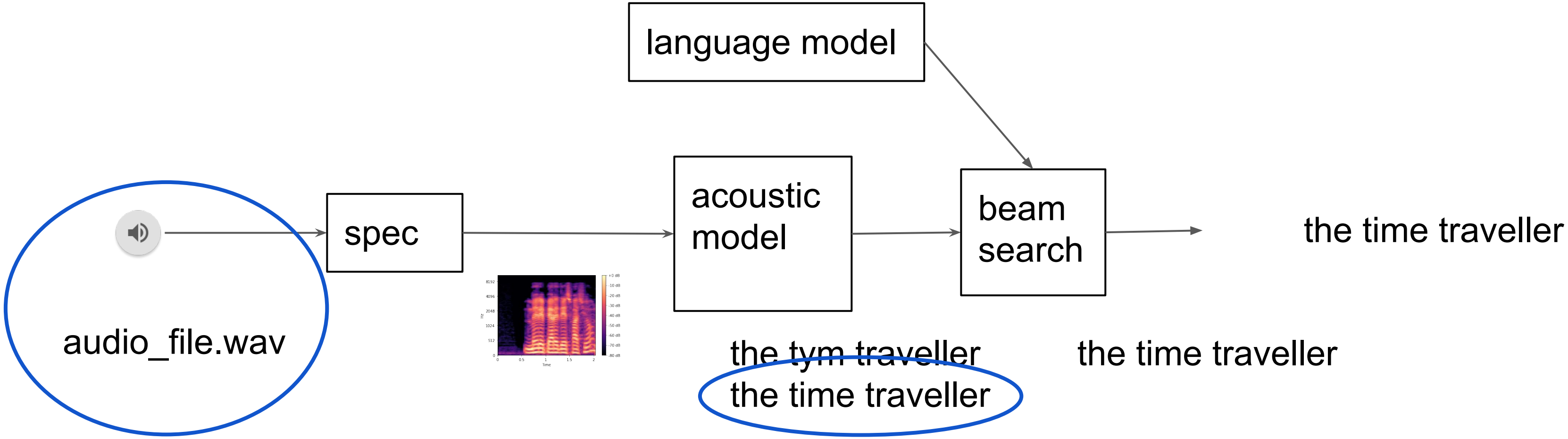
How to address practical challenges:

- Low amount of training data
- Different audio formats and sampling rates (e.g., 8k)
- Short utterances
- Streaming audio
- Diverse segments of audio (e.g., numbers, names)

Input-output



Input-output



Audio-text paired data: English

Baidu [2015] (English, private, read)

12,000 hours

Fraction of Data	Hours	Regular Dev
1%	120	29.23
10%	1200	13.80
20%	2400	11.65
50%	6000	9.51
100%	12000	8.46

Google [2018] (English, private)

162,000 hours

Librispeech (English, public, read)

~1000 hours (from audiobooks)

Audio-text paired data: Hindi

Private

- Google [2019]: 10,000+ hours

Publicly available

- Less than 100 hours

Challenge

- Handle low-resource setting
  - Create audio-text paired data for Hindi
  - ...

Speech data characteristics

Style: read, spontaneous
Length of audio: short, long
Speaker: accent, age, gender, ...
Domain: e.g., financial
Background noise
Audio codec: e.g., ogg/opus, mp3
Sampling rate: e.g., 8k for telecom
Channels: mono vs stereo
Distance: phone microphone, headphone-mic, far field
Code-mixing: e.g., Hinglish
Speed, volume
...

Hindi data collection:

~at least 100s of hours of data to be collected (~100k - ~1M sentences)

ideal: contributors geographically distributed

<demo: common voice donor portal>

    >= 2 upvotes and 0 downvotes

Needs splitting of train, dev, test buckets

    Need disjoint split of {speakers} AND {text}

    speaker → bucket, text → bucket mapping need to be set at data collection time

Speech data characteristics

Style: read, spontaneous
Length of audio: short, long
Speaker: accent, location, age, gender, ...
Domain: e.g., financial, general
Background noise
Audio codec: e.g., ogg/opus, mp3
Sampling rate
Channels: mono vs stereo
Distance: phone microphone, headphone, far field
Code-mixing: e.g., Hinglish
Speed, volume
...

Text corpus for data collection

Keep track of the different segments and their proportions

- domain-specific, general, numbers, alphabets, ...

Search datasets

- https://www.kaggle.com/datasets

Subtitles

- http://opus.nlpl.eu/

Scraping

- https://www.httrack.com/
- Extraction: textract

Clean the text data!

Always track the set of unique characters!

Characters should be within unicode range of the language (demo)

Verbalize: e.g., “4 apples” -> “4 apples”

sparrowhawk* // “$2” → “two dollars”

num2words // “2” -> “two”

Normalize: e.g, क़ → क़

क़ = \u0958, क़ = \u0915\u093c (क and ◌़)

indic_nlp_library

Filter out punctuations, invalid characters

check whether or not to replace with space

Look for anomalies in word and sentence splitting (based on char/word counts)

Speech data characteristics

Style: read, spontaneous
Length of audio: short, long
Speaker: accent, location, age, gender, ...
Domain: e.g., financial
Background noise
Audio codec: e.g., ogg/opus, mp3
Sampling rate
Channels: mono vs stereo
Distance: phone microphone, headphone, far field
Code-mixing: e.g., Hinglish
Speed, volume
...

Handling code-mixing (Hindi-English)

Use a common script: devanagari

“how are you” → “हाउ आर यू” // transliterate

Incorporate English at data collection time

Include transliterated English data in training (from public and private sources)

Librispeech, common voice

Transliteration can be challenging

libindic/indic-trans*

Azure transliterate service

word-wise cache

Speech data characteristics

Style: read, spontaneous

Length of audio: short, long

Speaker: accent, location, age, gender, ...

Domain: e.g., financial

Background noise

Audio codec: e.g., ogg/opus, mp3

Sampling rate

Channels: mono vs stereo

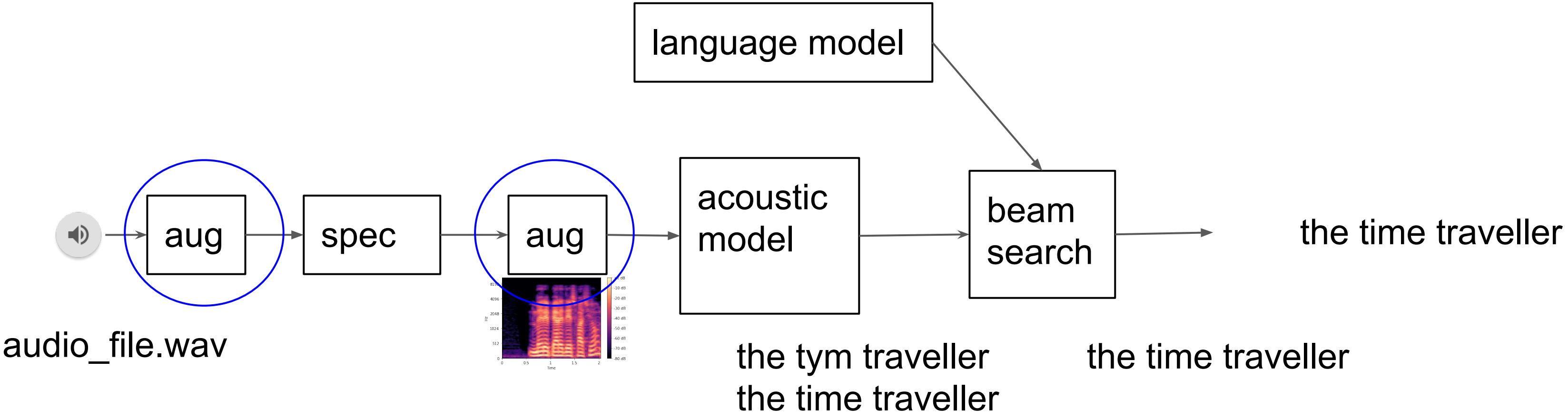
Distance: phone microphone, headphone, far field

Code-mixing: e.g., Hinglish

Speed, volume

...

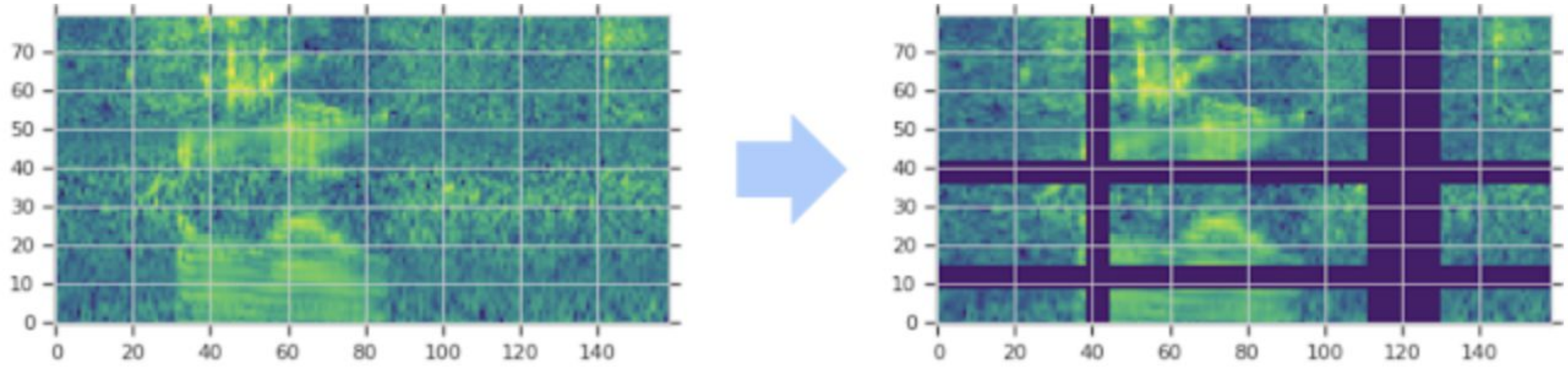
Input-output



Augmentations

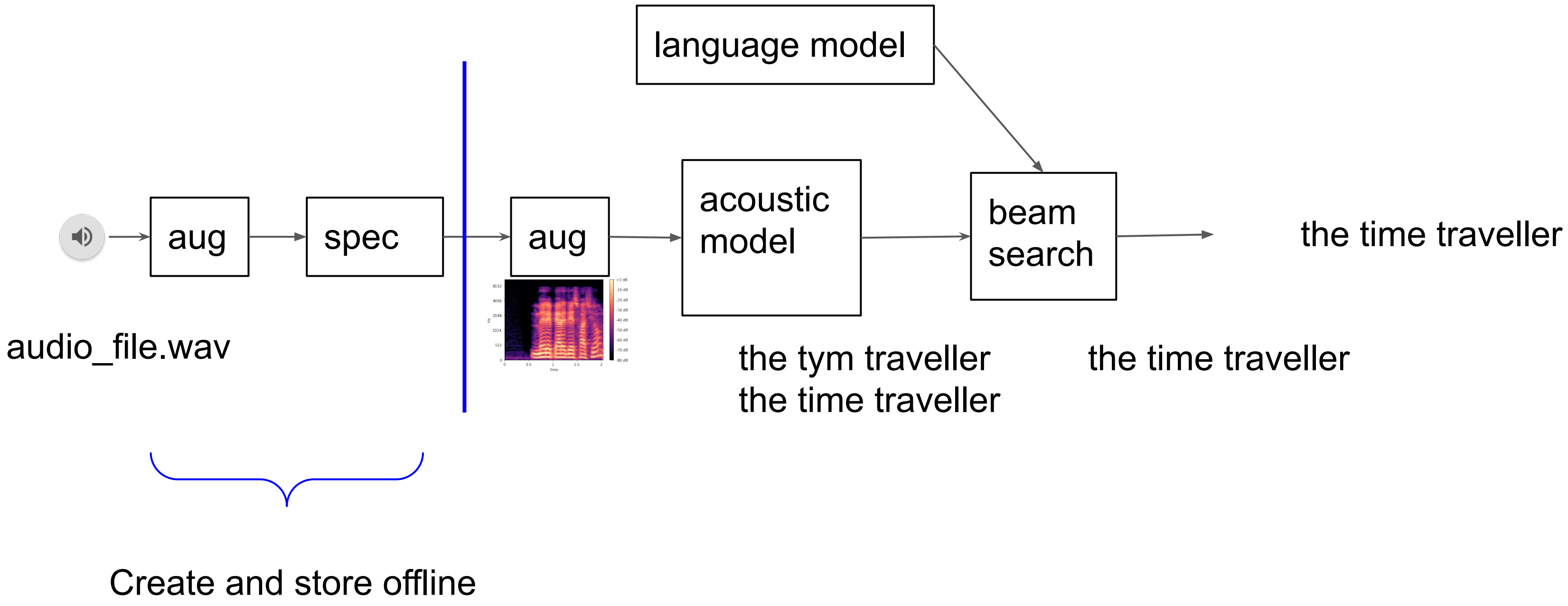
E.g., increase/decrease volume, speed, mix background noise, resample, codec, room simulator, spec augment

Specaugment [2019]



Combine offline augmentation with online specaugment

Input-output



Speech data characteristics

Style: **read**, spontaneous* → *forced alignment**

Length of audio: *short**, **long** → *modeling techniques**

Speaker: *accent, location, age*, gender, …* → diverse contributors

Domain: e.g., financial → with text corpus

Background noise → augment* (* Lombard effect)

Audio codec: e.g., ogg/opus, mp3 → augment

Sampling rate → augment

Channels: mono vs stereo → reduce to mono

Distance: phone microphone, headphone, far field → room simulator

Code-mixing: e.g., Hinglish → add transliterated data

Speed, volume → augment

...

Audio-text paired data: Hindi

Private

- Google [2019]: 10,000+ hours

Publicly available

- Less than 100 hours

Challenge

- Handle low-resource setting
  - Create audio-text paired data for hindi → ~100s of hours (expensive)
  - Extract from public data sources

Extract from external data sources

Youtube subtitles: KTSpeechCrawler

Audio books: Librivox

News bulletins

Movie subtitles

Earnings calls

Commercial datasets

Forced alignment

Forced alignment required for long paired data

- E.g., aeneas [demo]

Need to filter out errors in forced alignment

- Use a v1 ASR model trained on previously available data
  - Filter out alignment output which differ significantly from v1 ASR output
  - Manually verify alignment output

Audio-text paired data: Hindi

Private

Google [2019]: 10,000+ hours

Publicly available

Less than 100 hours

Challenge

Handle low-resource setting

Create audio-text paired data for hindi → ~500 hours (expensive)

Extract from public data sources → ~ 500 hours

Speech data characteristics

Style: **read**, ***spontaneous***** → forced alignment

Length of audio: *short**, **long** → *modeling techniques**

Speaker: *accent, location, age, gender,* … → 1000s of contributors

Domain: e.g., financial → with text corpus

Background noise → augment* (* Lombard effect)

Audio codec: e.g., ogg/opus, mp3 → augment

Sampling rate → augment

Channels: mono vs stereo → reduce to mono

Distance: phone microphone, headphone, far field → room simulator

Code-mixing: e.g., Hinglish → add transliterated data

Speed, volume → augment

...

Transfer learning

Pre-trained English models publicly available

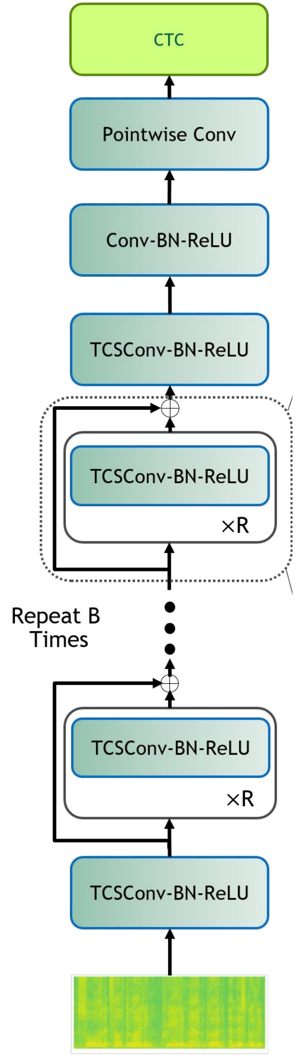
- 12,000 hours (deepspeech 2 from Baidu)
- ~54,000 hours (wav2letter++ from Facebook)

Can we utilize this in addition to the ~1000 hours we have for Hindi ?

Strategy

- Initialize the model parameters based with English
- Fine-tune using Hindi data

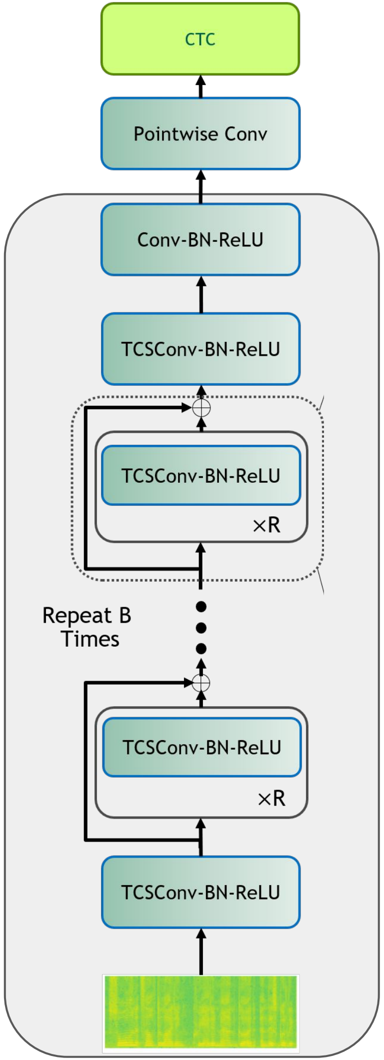
Transfer learning



https://arxiv.org/abs/1910.10261

Transfer learning

1. Freeze all layers except last

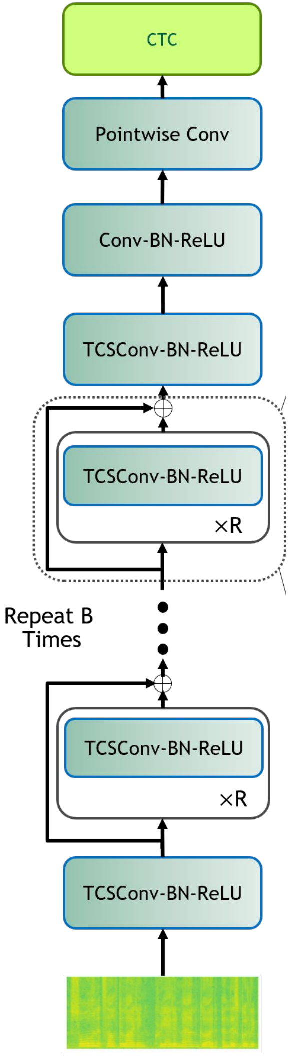


Only the last layer gets trained

https://arxiv.org/abs/1910.10261

Transfer learning

1. Freeze all layers except last
2. Load best checkpoint from phase 1
3. Unfreeze all layers
4. Save best checkpoint from phase 2



All layers get trained

https://arxiv.org/abs/1910.10261

Transfer learning: sample results

Transfer learning from English to German [2020]

Similar results for Eng → Hindi

- ~3-5% accuracy improvement
- Better training stability

Pre-train	Fine-tune	WER
CV-ge	-	23.35
5D	CV-ge	**18.65**

Training time reduces from order of weeks to days

- Faster iterations
- Lower hardware costs

Accuracy benchmarking

Could consider Cloud STT services as a baseline

For code-mixed text, use CER as metric:

E.g, हेलो, हैलो, हलो

Other factors

- Cost
- Privacy
- Support for formats like telephone [web page]

Further challenges

Handling different speech segments

Handling streaming, short audio

Handling different speech segments

Ensure minimum proportion of critical speech segments:

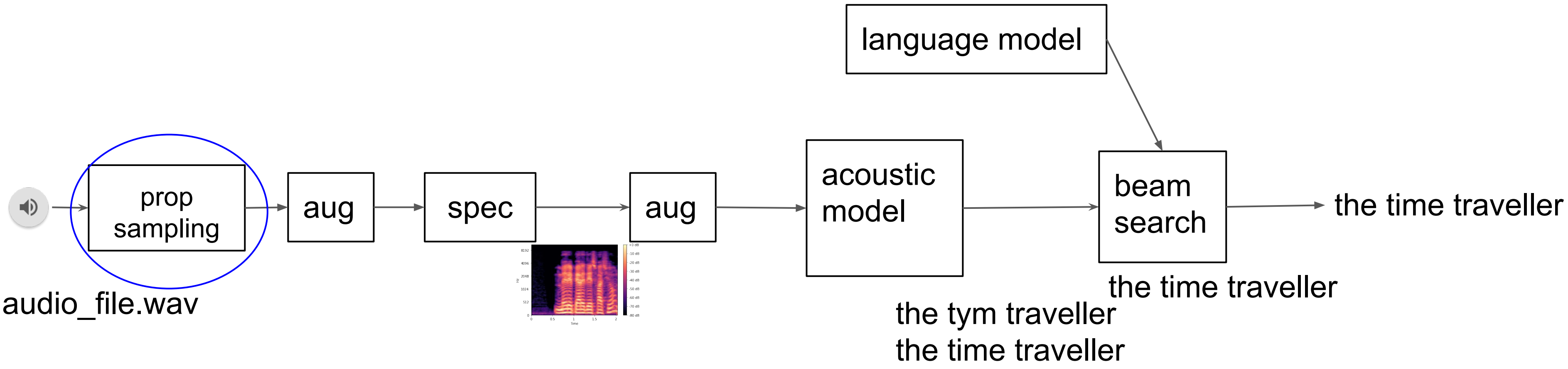
- e.g., domain-specific, numbers, transliterated english..
- e.g., minimum 5% of training set should be domain-specific speech

Proportional sampling

- Perform weighted random sampling of training examples ensuring
  - required proportions of different speech segments
  - (proportions determined based on importance)

Can be incorporated at offline augmentation time

Input-output



Monitor segment-wise val-set performance as training progresses

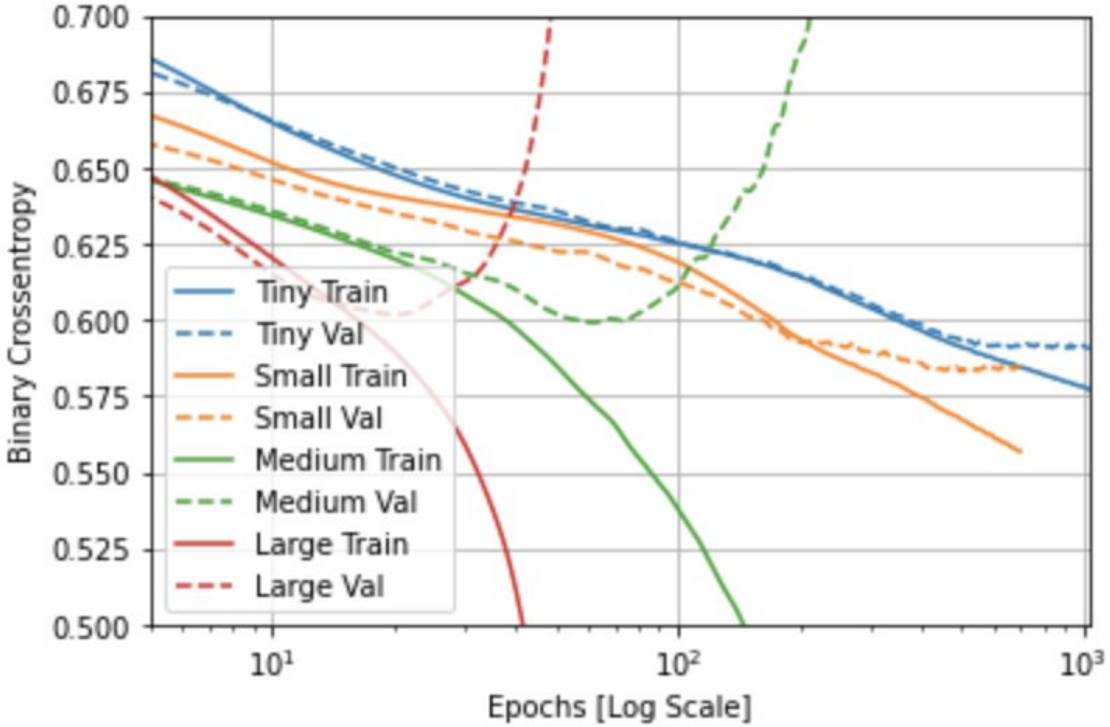


Image only for illustrative purpose

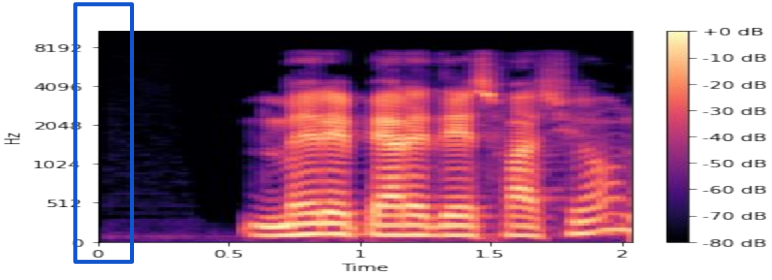
Further challenges

Handling different speech segments

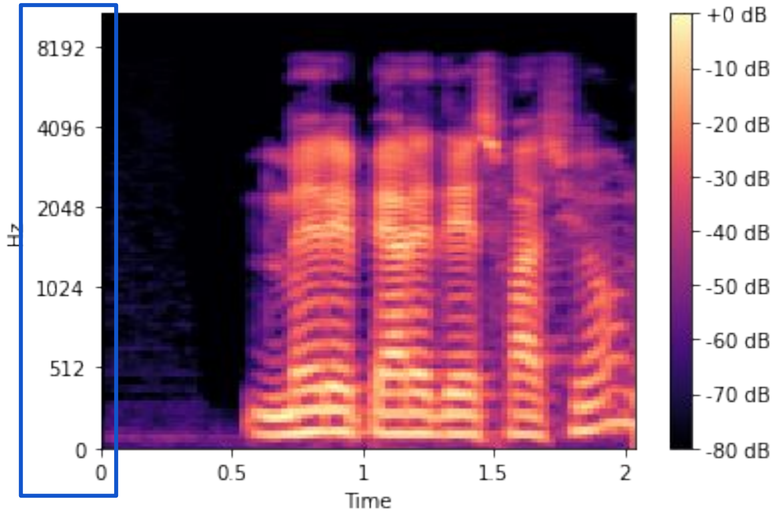
Handling streaming, short audio

Model should be able to work with limited temporal context

Rules out (basic versions of) bi-directional RNNs, attention models*

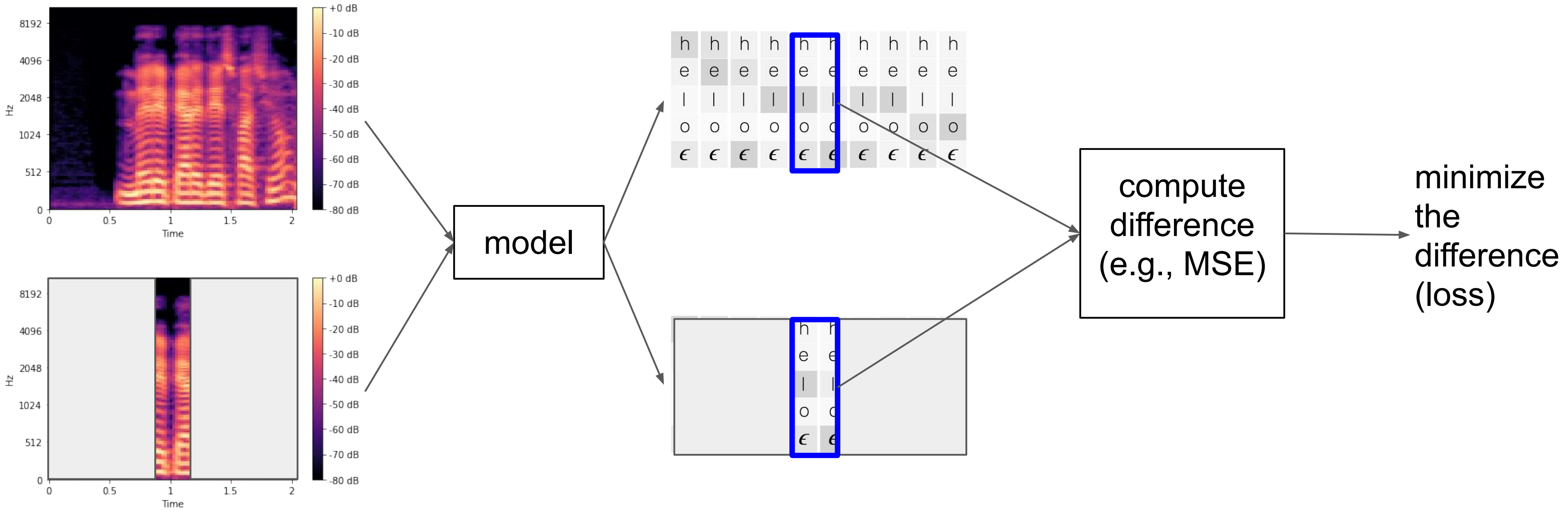


Handling short/streaming audio [fb, 2020]



Solution: add longer padding in the left (left padding of 8 with a conv width of 9)

Handling short/streaming audio



LCI: Learn Context Independence

Error rates on short audios: from ~55% to ~20%

Interesting directions for further R&D

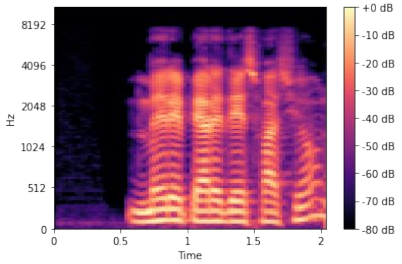
Learn from large scale unsupervised audio (e.g., wav2vec 2.0)

Cost effective training and inference

On-device models, hybrid models

Code-mixing

Deep learning architectures for waveform processing



Other Resources

Keep track of SOTA

Demo: paperswithcode

Note: Librispeech is from audiobooks

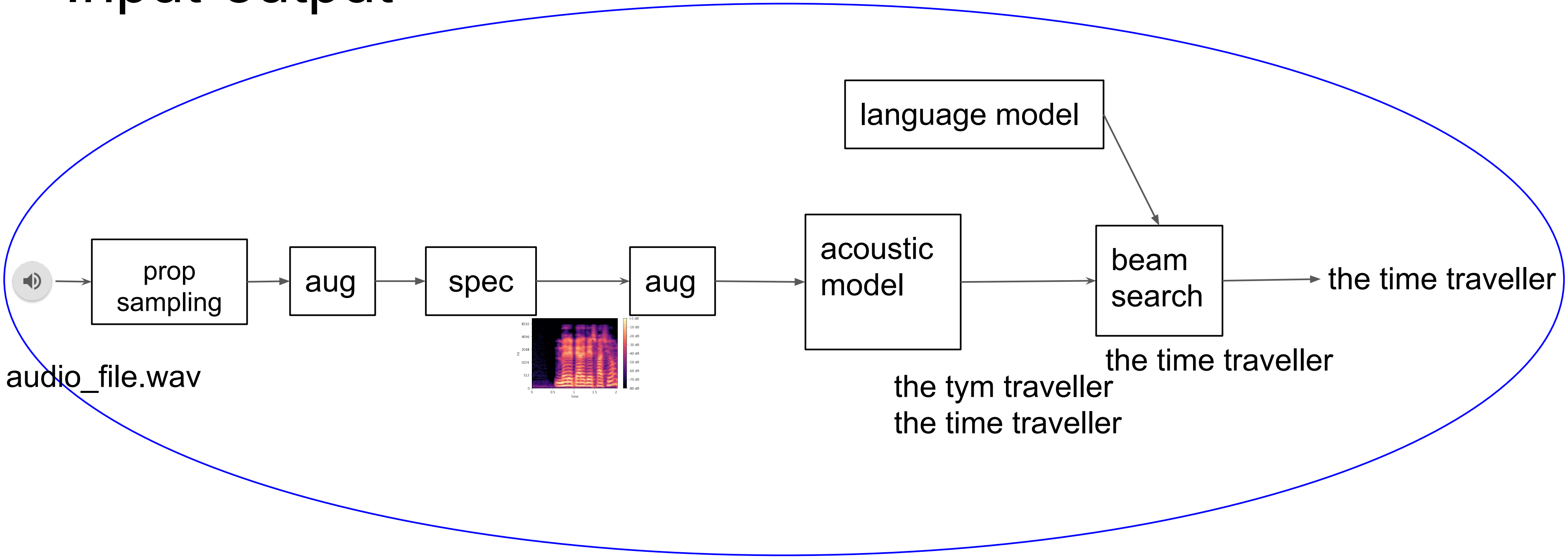
Interesting open-source projects

Nvidia deep learning example: Pytorch-ASR

Baidu Deepspeech (Paddle paddle), 12,000 hours pretrained model

FB wav2letter (Flashlight), ~54,000 hours pretrained model

Input-output



Q&A

THANK YOU

Slides, notebook at https://voicetech.substack.com
Subscribe for more!