Speech Recognition and Deep Learning

Adam Coates
Vinay Rao

Baidu Research
Silicon Valley AI Lab

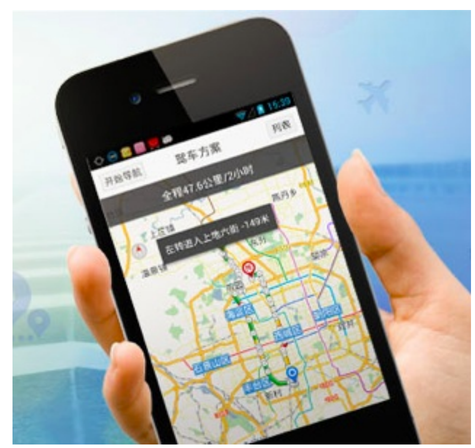
Speech recognition

- Many exciting & valuable applications

Content captioning

Cars / Hands-free interfaces

Home devices

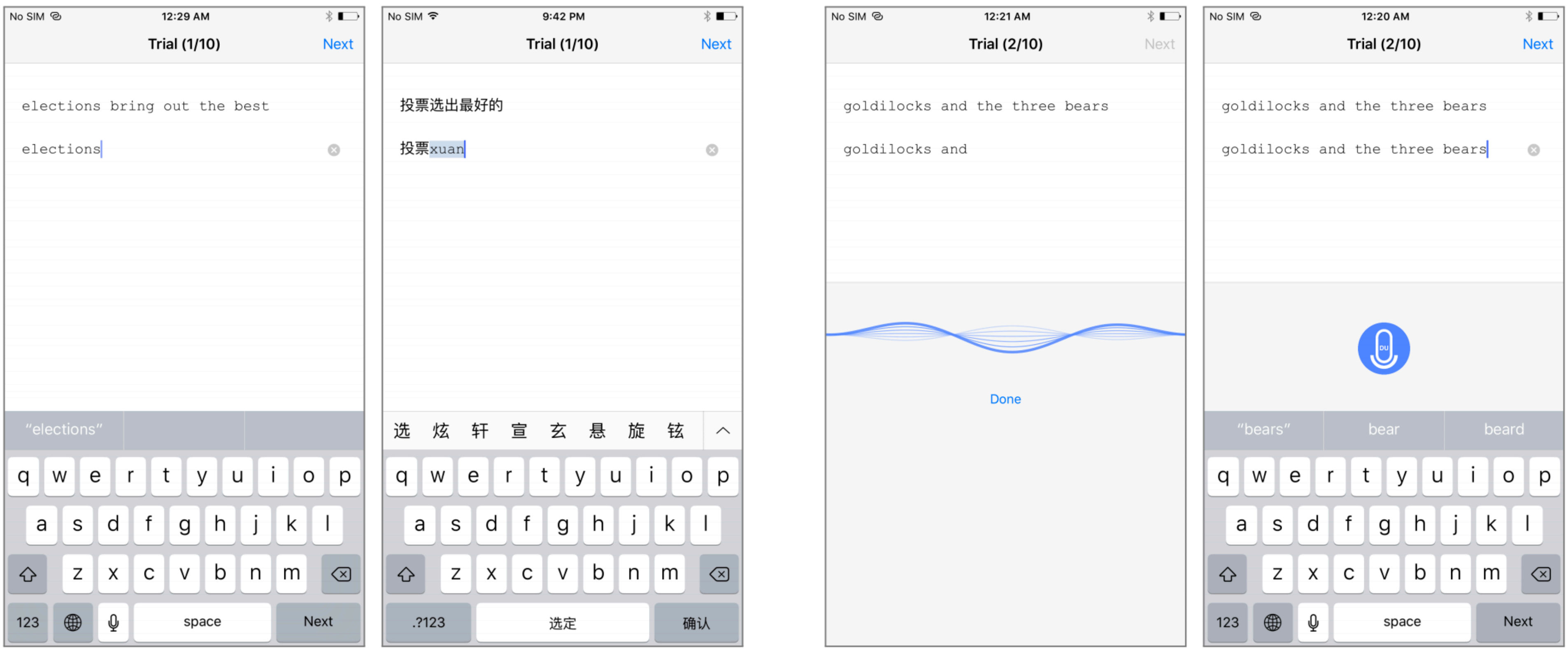


Mobile devices

Speech recognition

- Many exciting & valuable applications

"Speech Is 3x Faster than Typing for English and Mandarin Text Entry on Mobile Devices"



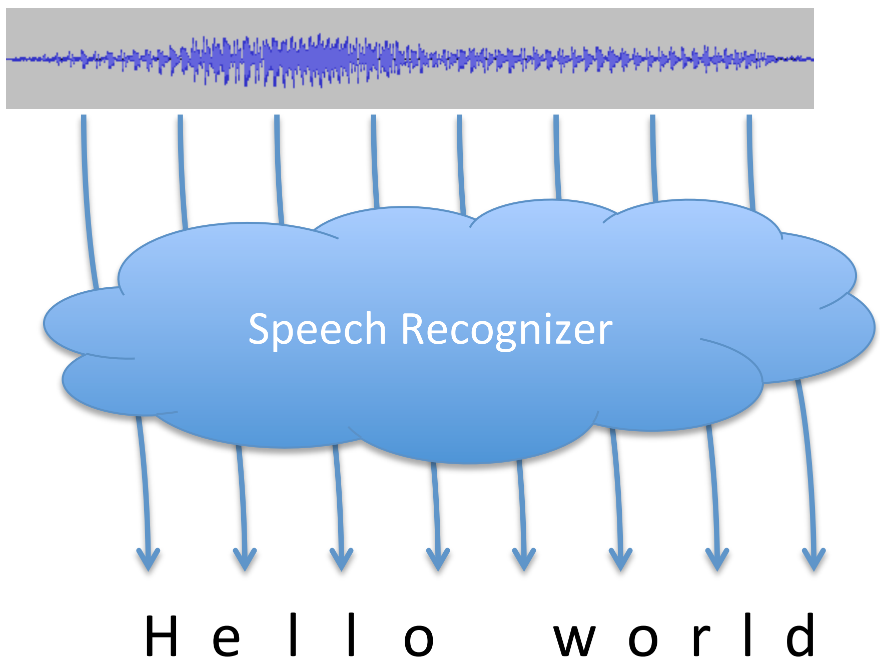
[Ruan et al., 2016]

Speech recognition

- Many components make up a complete speech application:
  - Speech transcription — This lecture
  - Word spotting / trigger word
  - Speaker identification / verification

Speech recognition

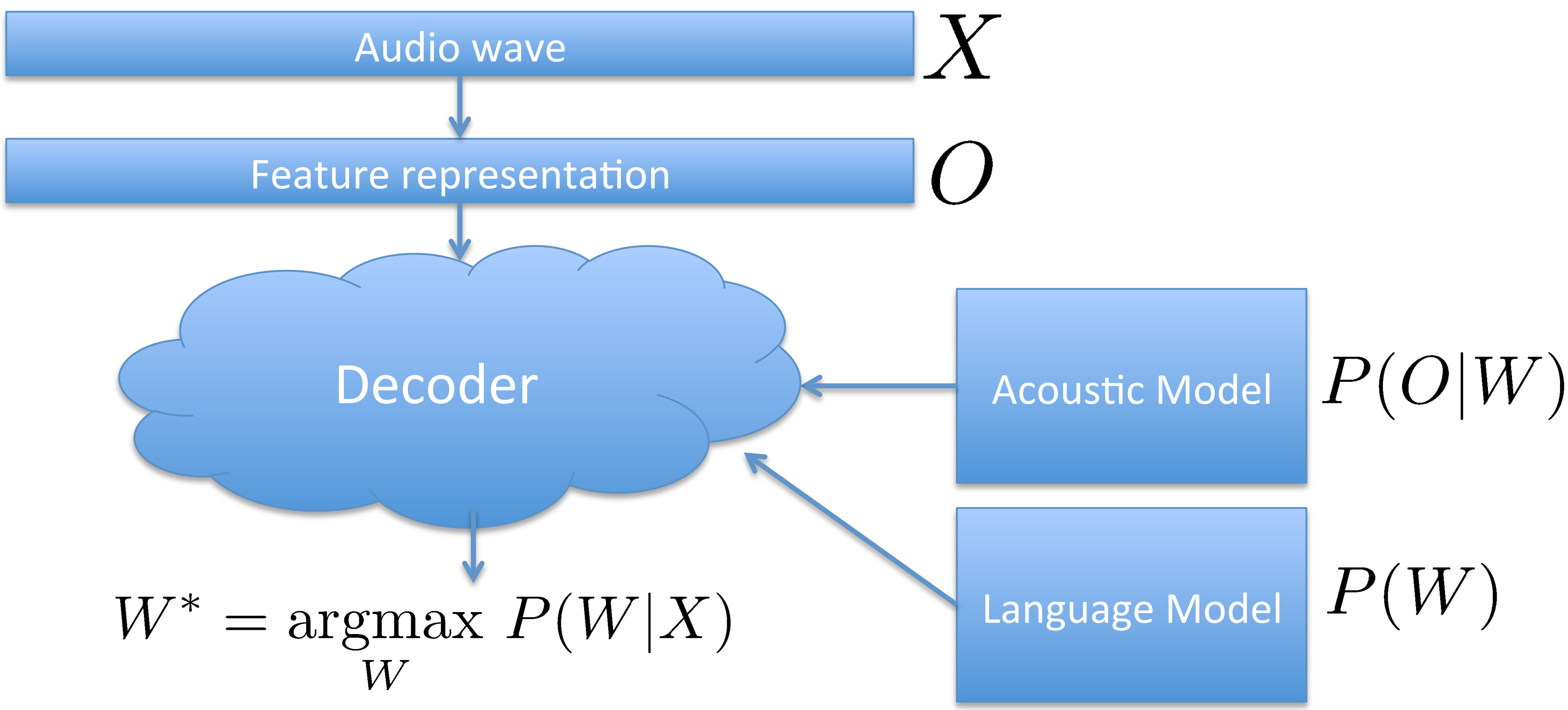
- Given speech audio, generate a transcript.



Important goal of AI: historically hard for machines, easy for people.

Traditional ASR pipeline

- Traditional systems break problem into several key components:



$$
\begin{aligned}
W^* &= \underset{W}{\operatorname{argmax}}\ P(W|X) \\
&= \underset{W}{\operatorname{argmax}}\ P(O|W)P(W)
\end{aligned}
$$

Gales & Young, 2008
Jurafsky & Martin, 2000

Traditional ASR pipeline

- Usually represent words as sequence of "phonemes":

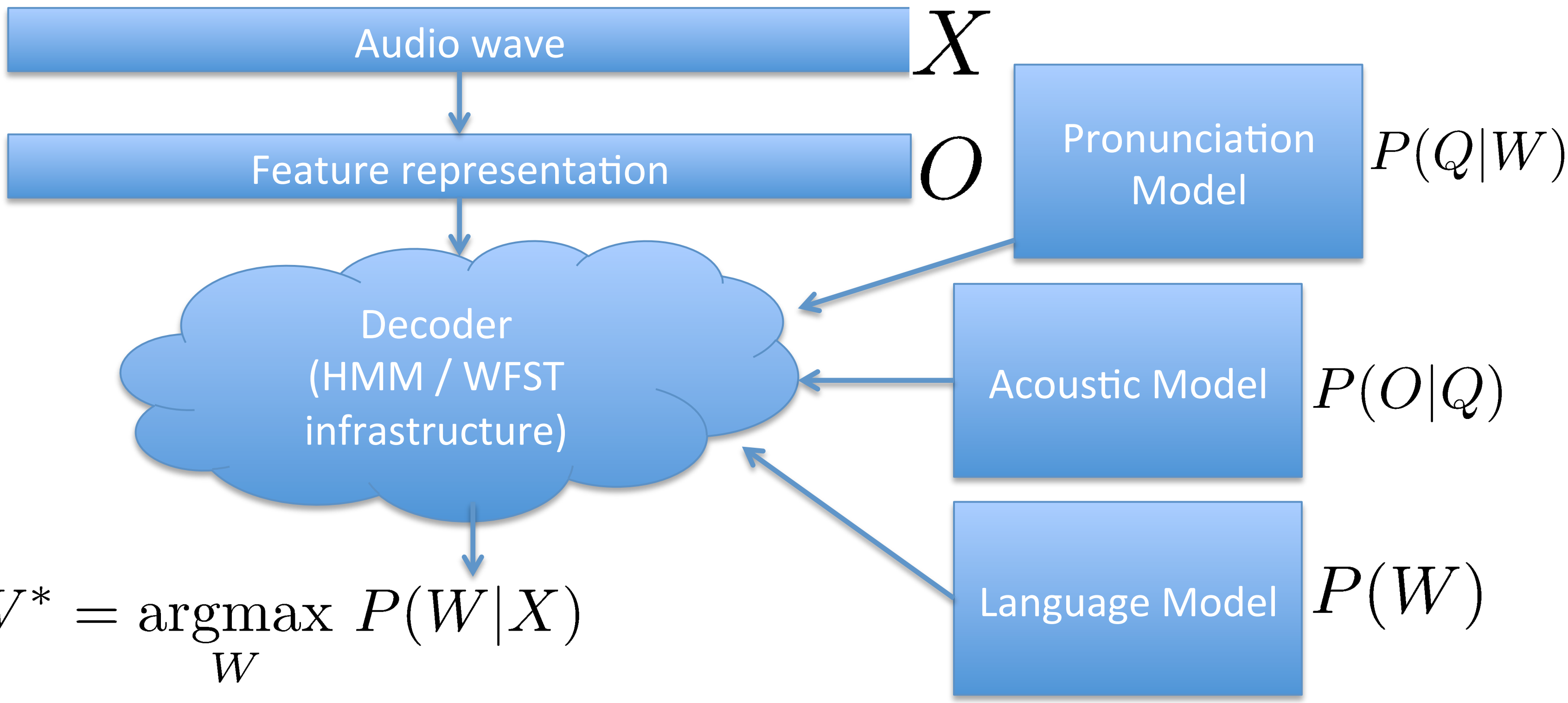
$$w_1 = \text{``hello''} = \text{[HH AH L OW]} = [q_1 q_2 q_3 q_4]$$

- Phonemes are the perceptually distinct units of sound that distinguish words.
  - Quite approximate... but sorta standardized-ish.
  - Some labeled corpora available (e.g., TIMIT)

	Phone Label	Example		Phone Label	Example		Phone Label	Example
1	iy	beet	22	ch	choke	43	en	button
2	ih	bit	23	b	bee	44	eng	Washington

Traditional ASR pipeline

- Traditional systems usually model phoneme sequences instead of words. This necessitates a dictionary or other model to translate.



$$W^* = \operatorname*{argmax}_W P(W|X)$$

$$= \operatorname*{argmax}_W \sum_Q P(O|Q)P(Q|W)P(W)$$

Traditional ASR pipeline

- Traditional pipeline is highly tweak-able, but also hard to get working well.
- Historically, each part of system has own set of challenges.
  - E.g., choosing feature representation.

Deep Learning in ASR

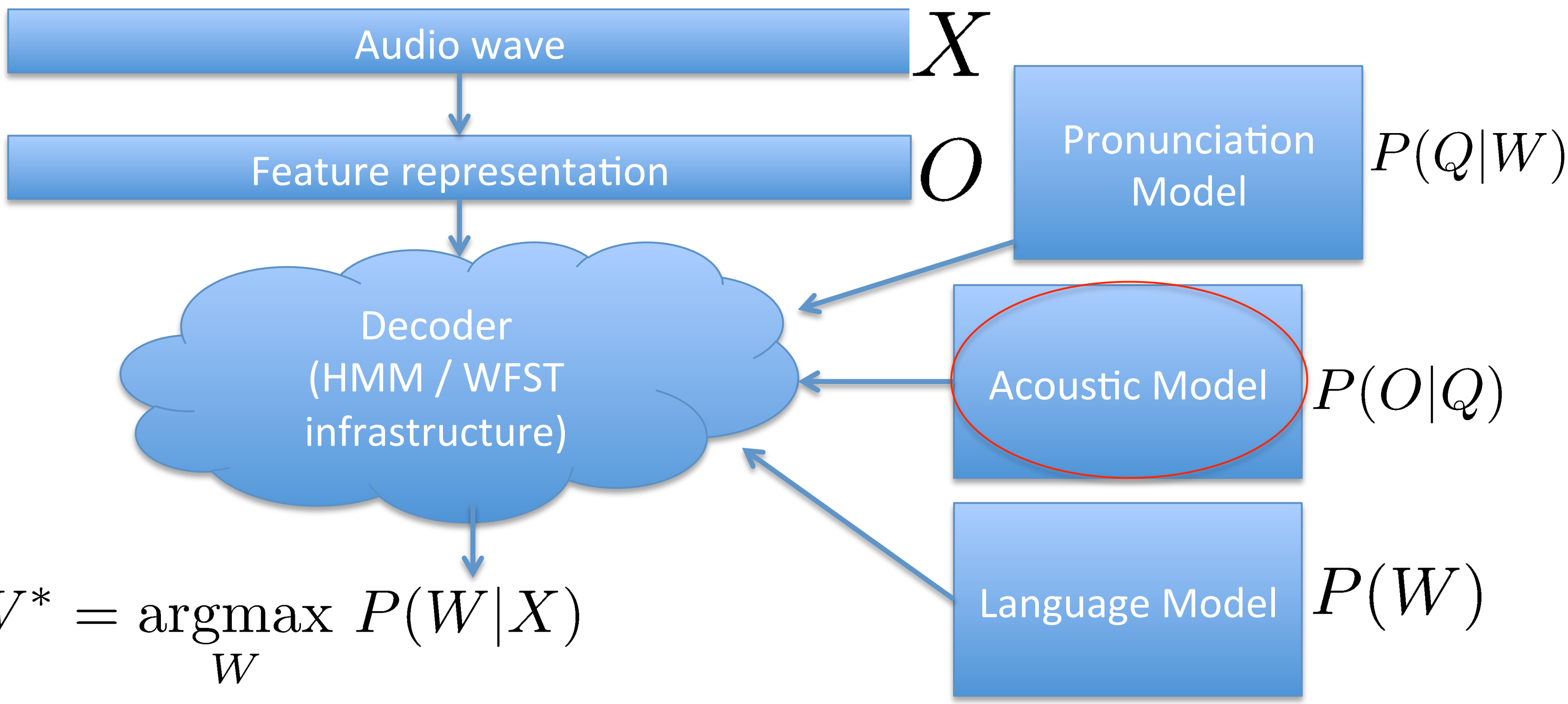
- Where to apply DL to make ASR better?
  - Good start: improve acoustic model $P(O|Q)$

- Introduction of pre-training/DBN:

DBN-HMM	5	from DBN-HMM	Triphone Senones	yes	71.8%	69.6%
ML GMM-HMM baseline					62.9%	60.4%
MMI GMM-HMM baseline					65.1%	62.8%
MPE GMM-HMM baseline					65.5%	63.8%
ML GMM-HMM baseline 2100 hours of training data (transcription is 90% accurate)					-	62.9% [13]

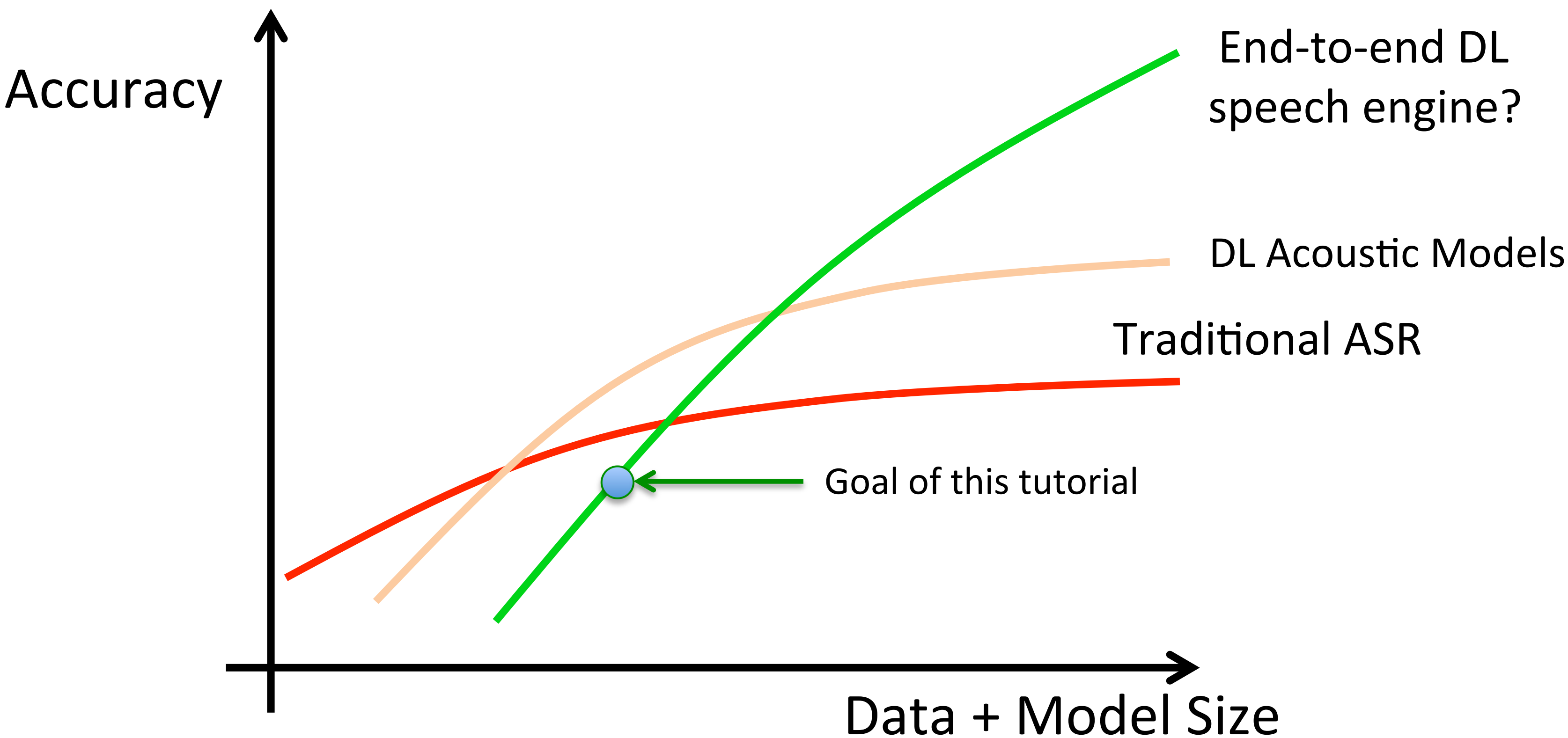
[Dahl et al., ICASSP 2011]

Traditional ASR pipeline

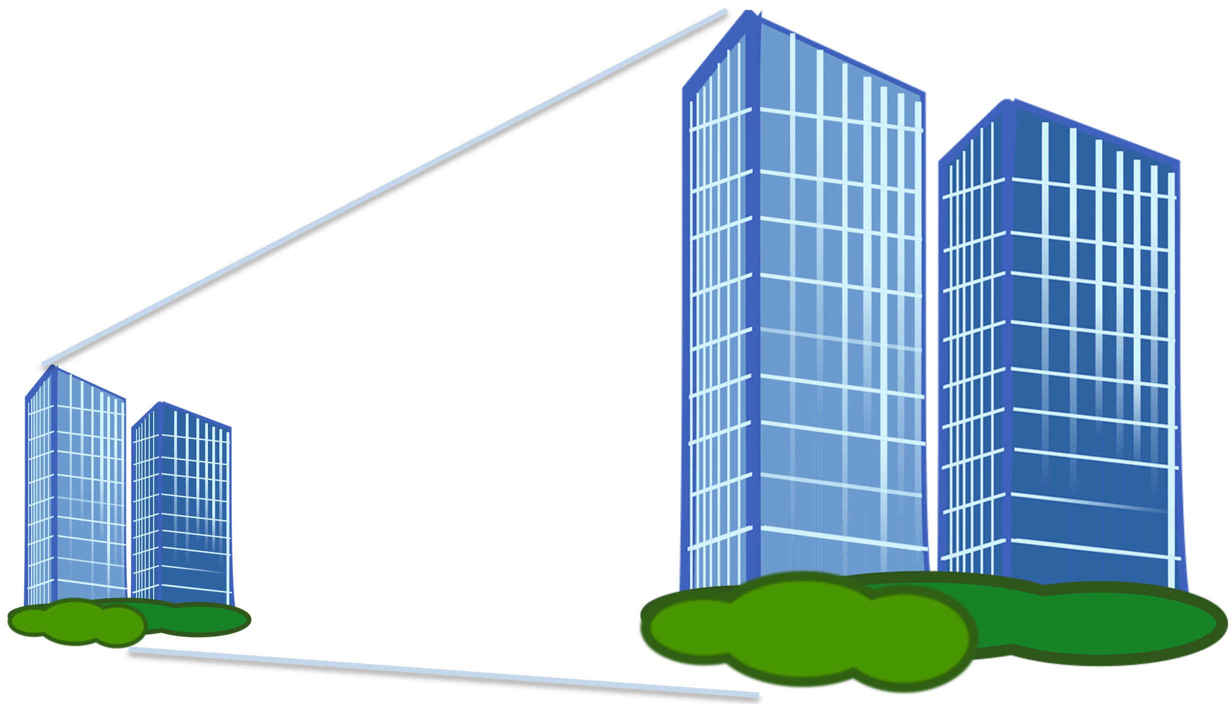


$$W^* = \operatorname*{argmax}_W P(W|X)$$

$$= \operatorname*{argmax}_W \sum_Q P(O|Q)P(Q|W)P(W)$$


Accuracy
End-to-end DL
speech engine?
DL Acoustic Models
Traditional ASR
Goal of this tutorial
Data + Model Size

Scale model



Starter code: github.com/baidu-research/ba-dls-deepspeech

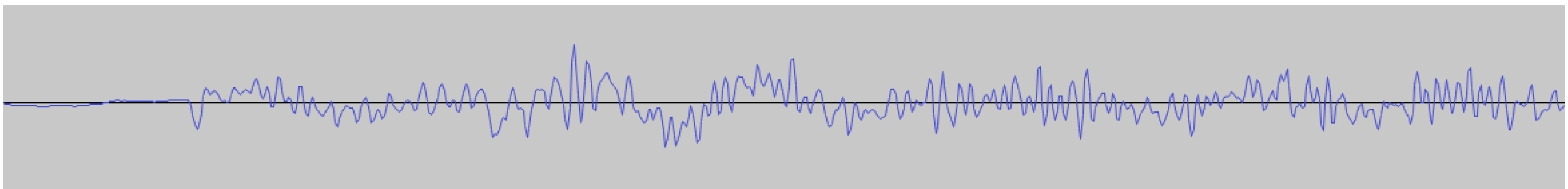
Get a simple max-decoded pipeline running.

Outline

- DL speech pipeline walkthrough
  - Preprocessing
  - CTC
  - Training
  - Decoding & language models
- Scaling up!
  - Data
  - Systems
- Production / reality

Raw audio

- Simple 1D signal:



Typical sample rates for speech: 8KHz, 16KHz.
Each sample typically 8-bit or 16-bit.

- 1D vector: $X = [x_1 x_2 \ldots]$

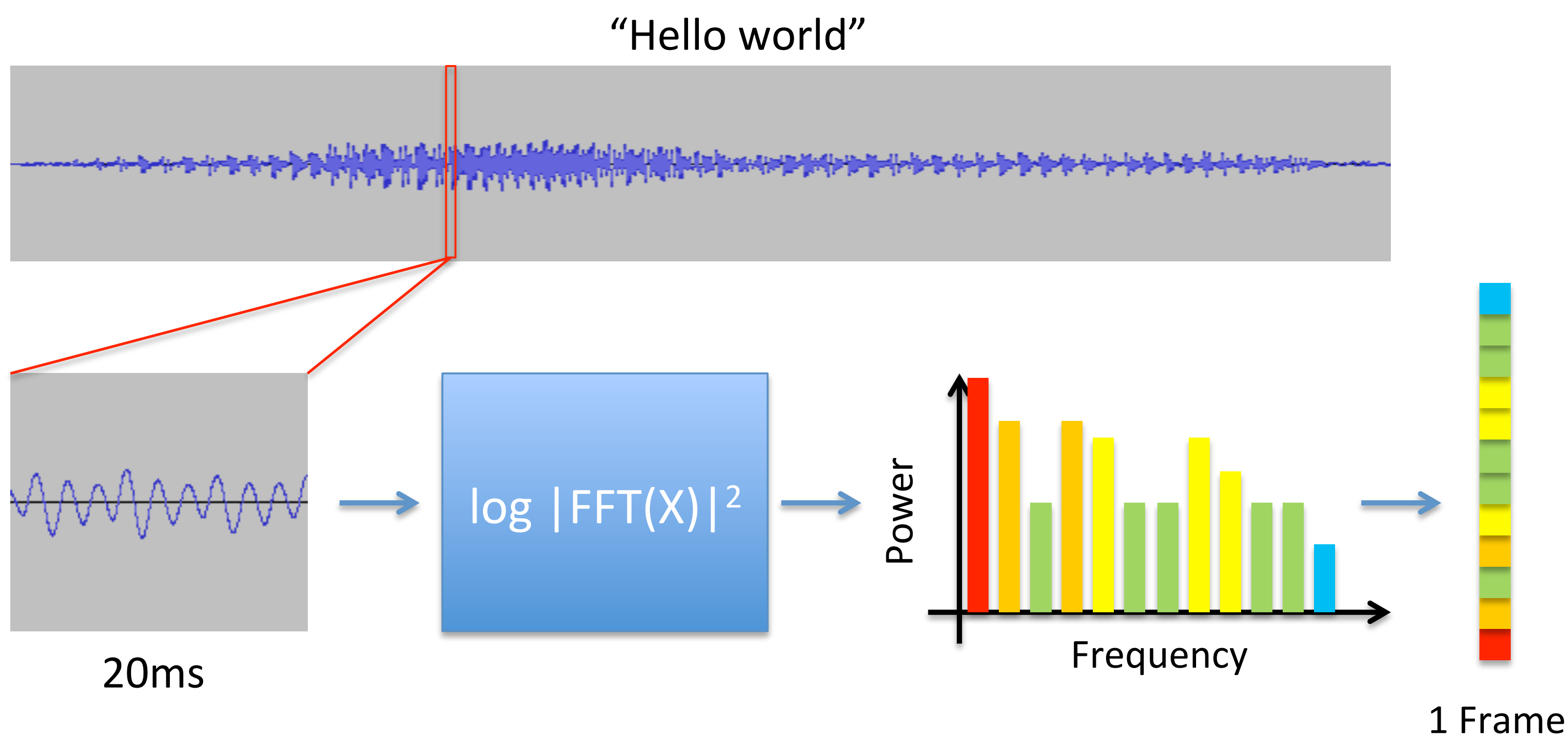
Pre-processing

- Two ways to start:
  - Minimally pre-process (e.g., simple spectrogram).
    - We'll use this.
  - Train model from raw audio wave.
    - It works!

    See, e.g., Sainath et al., Interspeech 2015

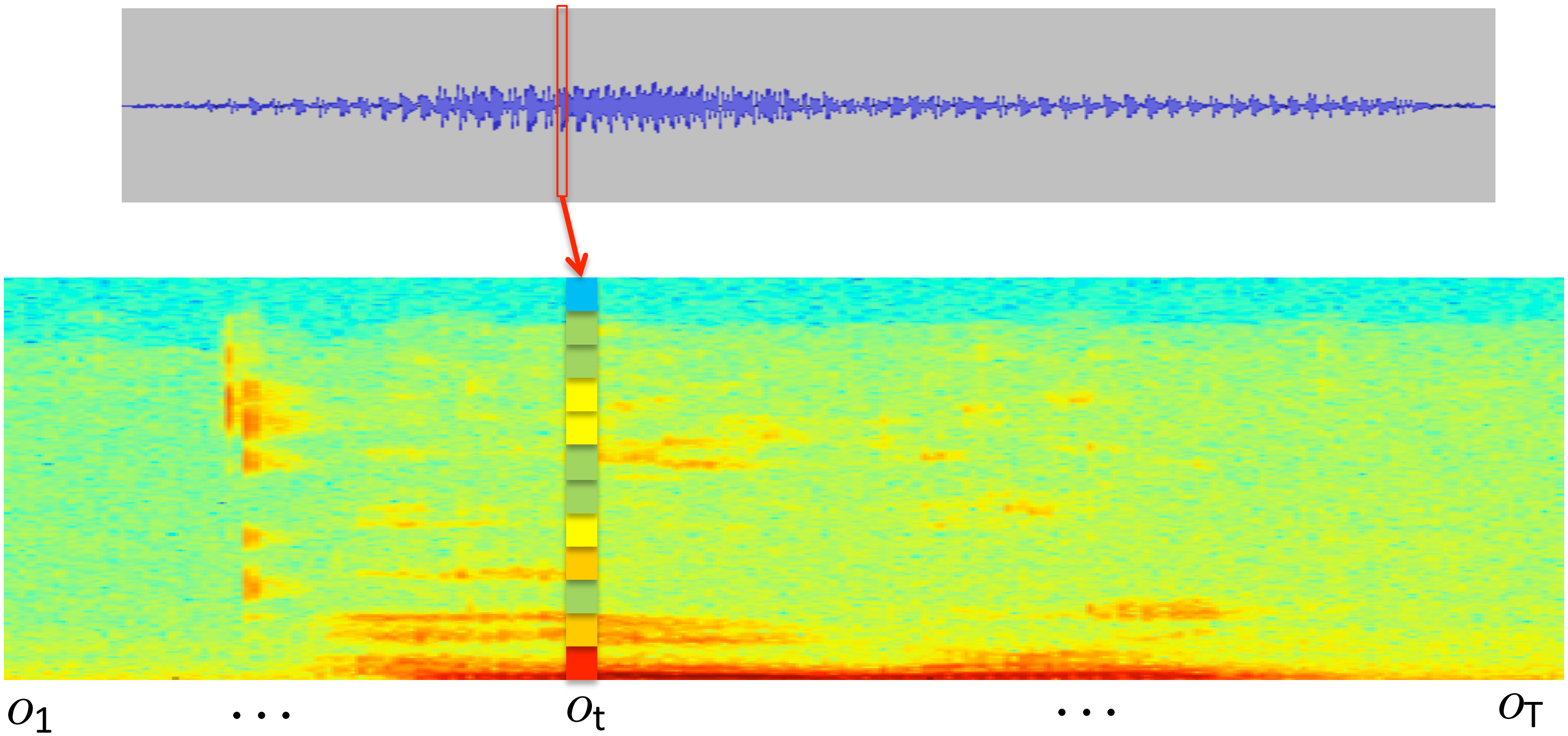
Spectrogram

- Take a small window (e.g., 20ms) of waveform.
  - Compute FFT and take magnitude. (i.e., power)
  - Describes frequency content in local window.



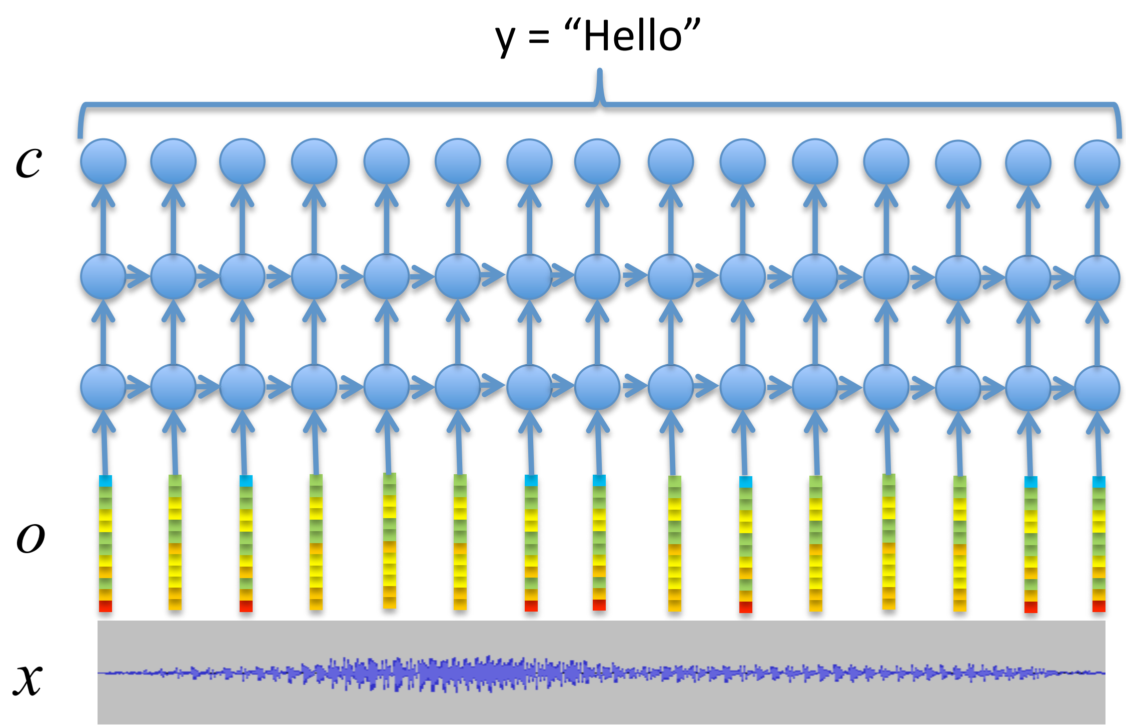
Spectrogram

- Concatenate frames from adjacent windows to form "spectrogram".



Acoustic Model

- Goal: create a neural network (DNN/RNN) from which we can extract transcription, y.
  - Train from labeled pairs (x, y^*)



Acoustic model

- Main issue: length(x) != length(y)
  - Don't know how symbols in y map to frames of audio.
  - Traditionally, try to bootstrap alignment – painful!

- Multiple ways to resolve:
  - Use attention, sequence to sequence models, etc. [Chan et al., 2015; Bahdanau et al., 2015]
  - Connectionist Temporal Classification [Graves et al., 2006]

Connectionist Temporal Classification (CTC)

- Basic idea:
  1. RNN output neurons c encode distribution over symbols. Note length(c) == length(x).

     For phoneme-based model: $c \in \{AA, AE, AX, ..., ER1, \text{blank}\}$

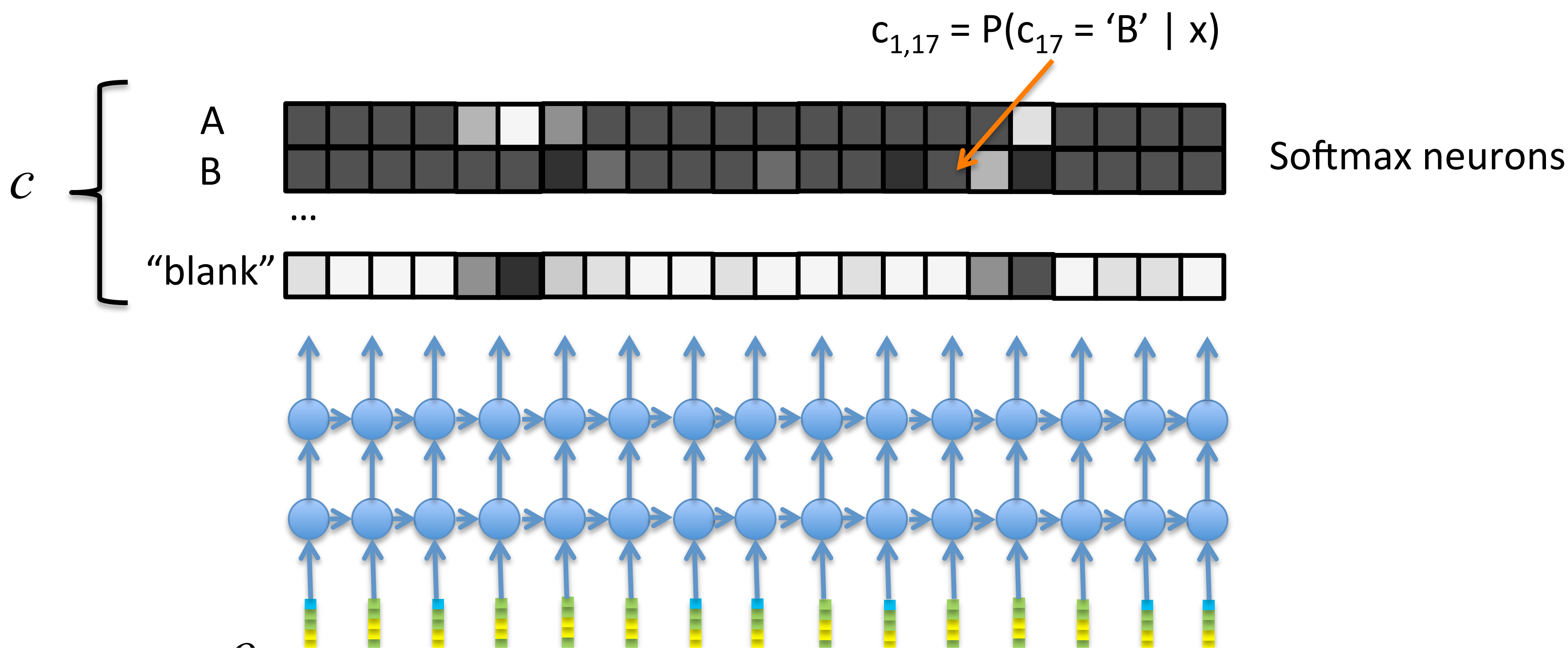
     For grapheme-based model: $c \in \{A, B, C, D, ..., Z, \text{blank}, \text{space}\}$

  2. Define a mapping $\beta(c) \rightarrow y$.
  3. Maximize likelihood of y* under this model.

Connectionist Temporal Classification (CTC)

1. RNN output neurons c encode distribution over symbols. Note length(c) == length(x).

For grapheme-based model: $c \in \{A, B, C, D, ..., Z, \text{blank}, \text{space}\}$



Connectionist Temporal Classification (CTC)

1. RNN output neurons c encode distribution over symbols. Note length(c) == length(x).

For grapheme-based model: $c \in \{A, B, C, D, ..., Z, \text{blank}, \text{space}\}$

- Output neurons define distribution over whole character sequences c assuming independence:

$$P(c|x) \equiv \prod_{i=1}^{N} P(c_i|x)$$

$$P(c = \text{HHH\_E\_\_LL\_LO\_\_\_}|x) = P(c_1 = \text{H}|x)P(c_2 = \text{H}|x) \cdots P(c_{15} = \text{blank}|x)$$

Connectionist Temporal Classification (CTC)

2. Define a mapping $\beta(c) \rightarrow y$.
   - Given a specific character sequence c, squeeze out duplicates + blanks to yield transcription:

$$y = \beta(c) = \beta(\text{HHH\_E\_\_LL\_LO\_\_\_}) = \text{“HELLO”}$$

Connectionist Temporal Classification (CTC)

- Mapping implies a distribution over possible *transcriptions* y:

$P(c\|x) = \{$	0.1	HHH_E__LL_LO___	"HELLO" v.1
	0.02	HH__E__LL_LO___	"HELLO" v.2
	0.01	HHH_E__L_L_OH__	"HELL OH"
	0.01	HHH_EE_LL_L_O__	"HELLO" v.3
	…	YY__E__LL_LO_W_	"YELLOW"

$$P(y|x) = \sum_{c:\beta(c)=y} P(c|x)$$

$$P(\text{“HELLO”}) = 0.1 + 0.02 + 0.01 + \ldots$$

Connectionist Temporal Classification (CTC)

3. Update network parameters θ to maximize likelihood of correct label $y*$:

$$
\begin{aligned}
\theta^\star &= \arg\max_\theta \sum_i \log P(y^{\star(i)}|x^{(i)}) \\
&= \arg\max_\theta \sum_i \log \sum_{c:\beta(c)=y^{\star(i)}} P(c|x^{(i)})
\end{aligned}
$$

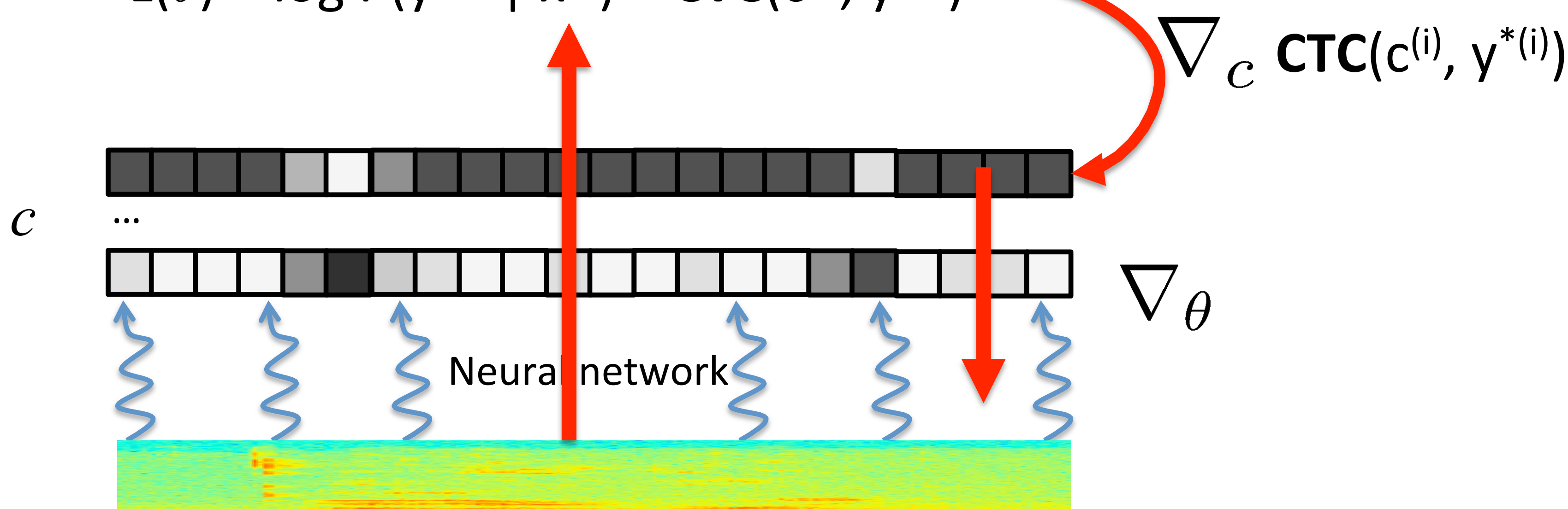
- [Graves et al., 2006] provides an efficient dynamic programming algorithm to compute the inner summation and its gradient.

Connectionist Temporal Classification (CTC)

- Use usual gradient descent methods to optimize.
  Tune entire network with backpropagation.
  - Given network outputs, many off-the-shelf packages to compute CTC loss (likelihood) from c and y^*, and gradient w.r.t. c.
    - Warp CTC: github.com/baidu-research/warp-ctc
    - Stanford CTC: github.com/amaas/stanford-ctc
    - Tensorflow: tf.nn.ctc_loss

Efficient OTS software

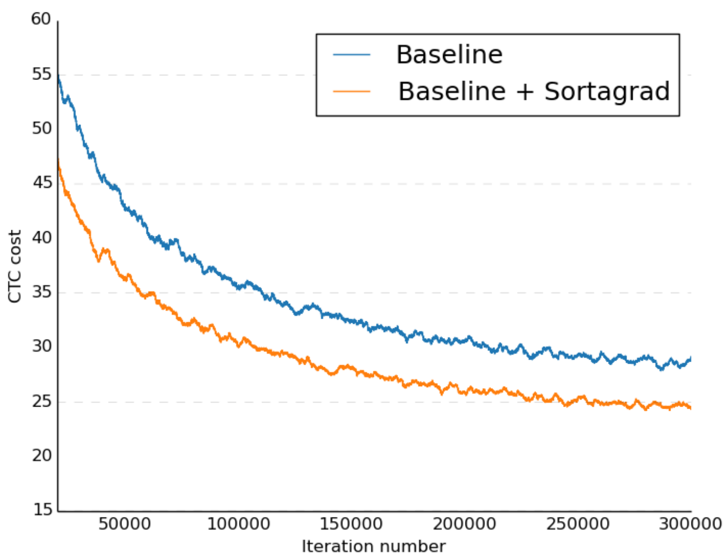
$$L(\theta) = \log P(y^{*(i)} \mid x^{(i)}) = \mathbf{CTC}(c^{(i)}, y^{*(i)})$$



Training tricks

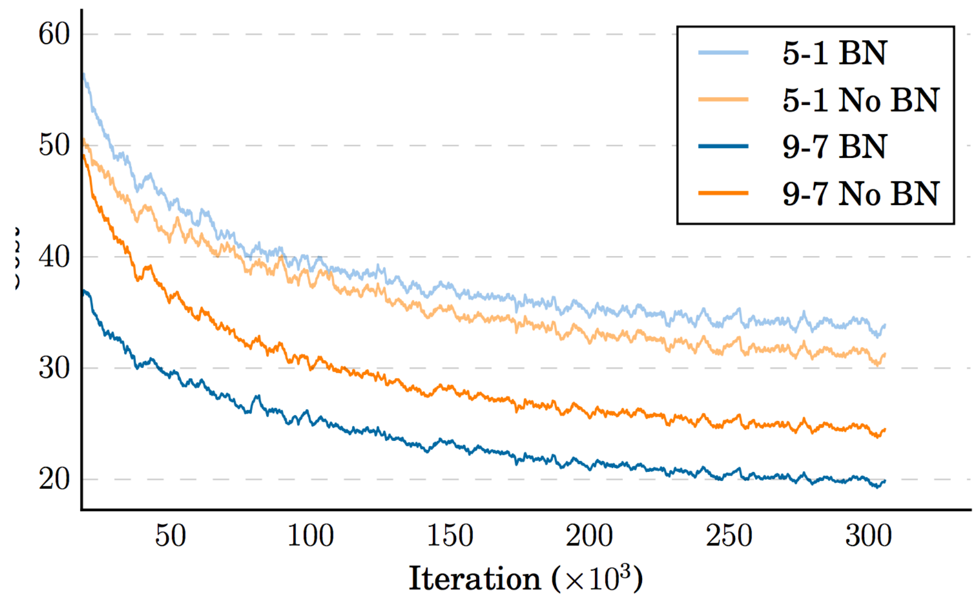
- Getting RNN to train well is tricky.

"SortaGrad": order utterances by length during first epoch.



See ["Curriculum Learning", Bengio et al., ICML 2009]

Batch normalization

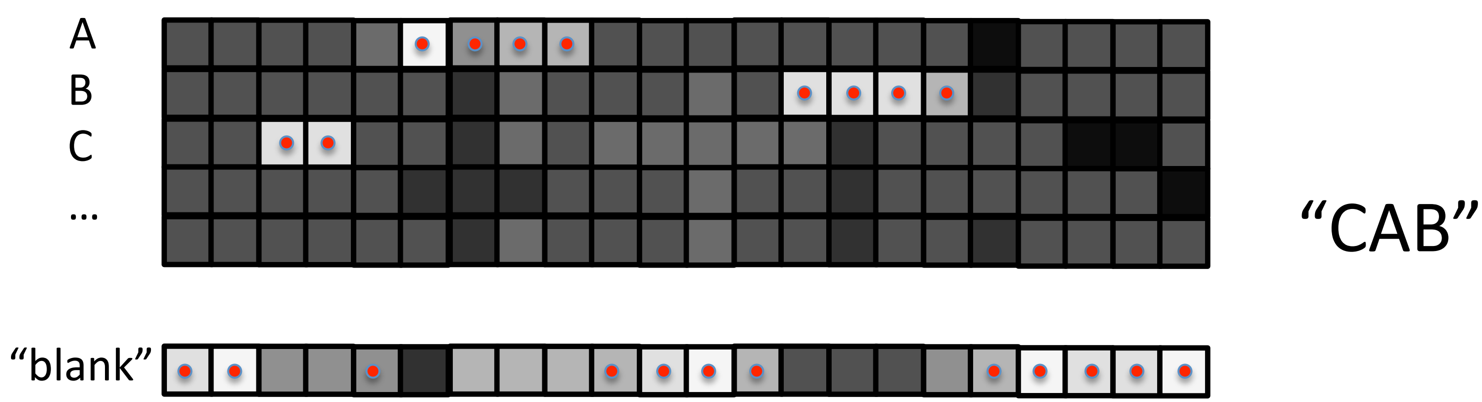


See [Ioffe & Szegedy, 2015]

Decoding

- Network outputs P(c|x). How do we find most likely transcription from P(y|x)?
- Simple (approximate) solution:

$$\beta\left(\arg\max_{c} \; P(c|x)\right)$$



- Often terrible, but a useful diagnostic to "eyeball" models.

Example

- Wall Street Journal:
  - https://catalog.ldc.upenn.edu/ldc93s6a
  - Reading WSJ articles.

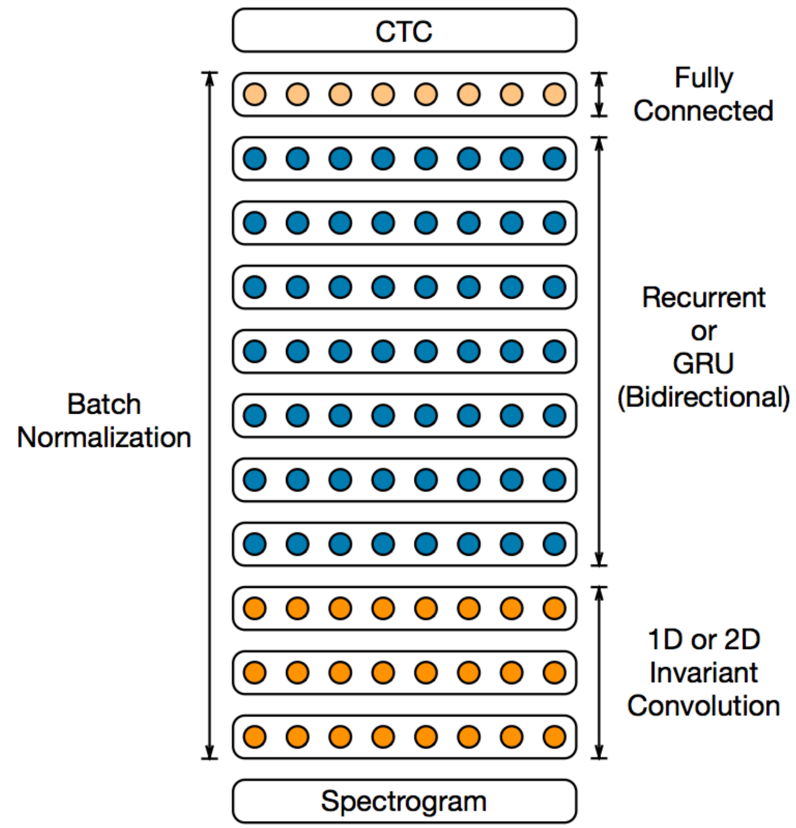
Free alternative: LibriSpeech

- http://www.openslr.org/12/ [Panayotov et al., ICASSP 2015]
- Read speech from public domain audiobooks.

Example

- RNN to predict graphemes (26 characters + space + blank):
  - Spectrograms as input.
  - 1 layer of convolutional filters.
  - 3 layers of Gated Recurrent Units.
    - 1000 neurons per layer.
  - 1 full-connected layer to predict c.
  - Batch normalization [Ioffe & Szegedy, 2015]
- CTC loss function (Warp-CTC)
- Train with SGD+Nesterov momentum.

Typical model family:



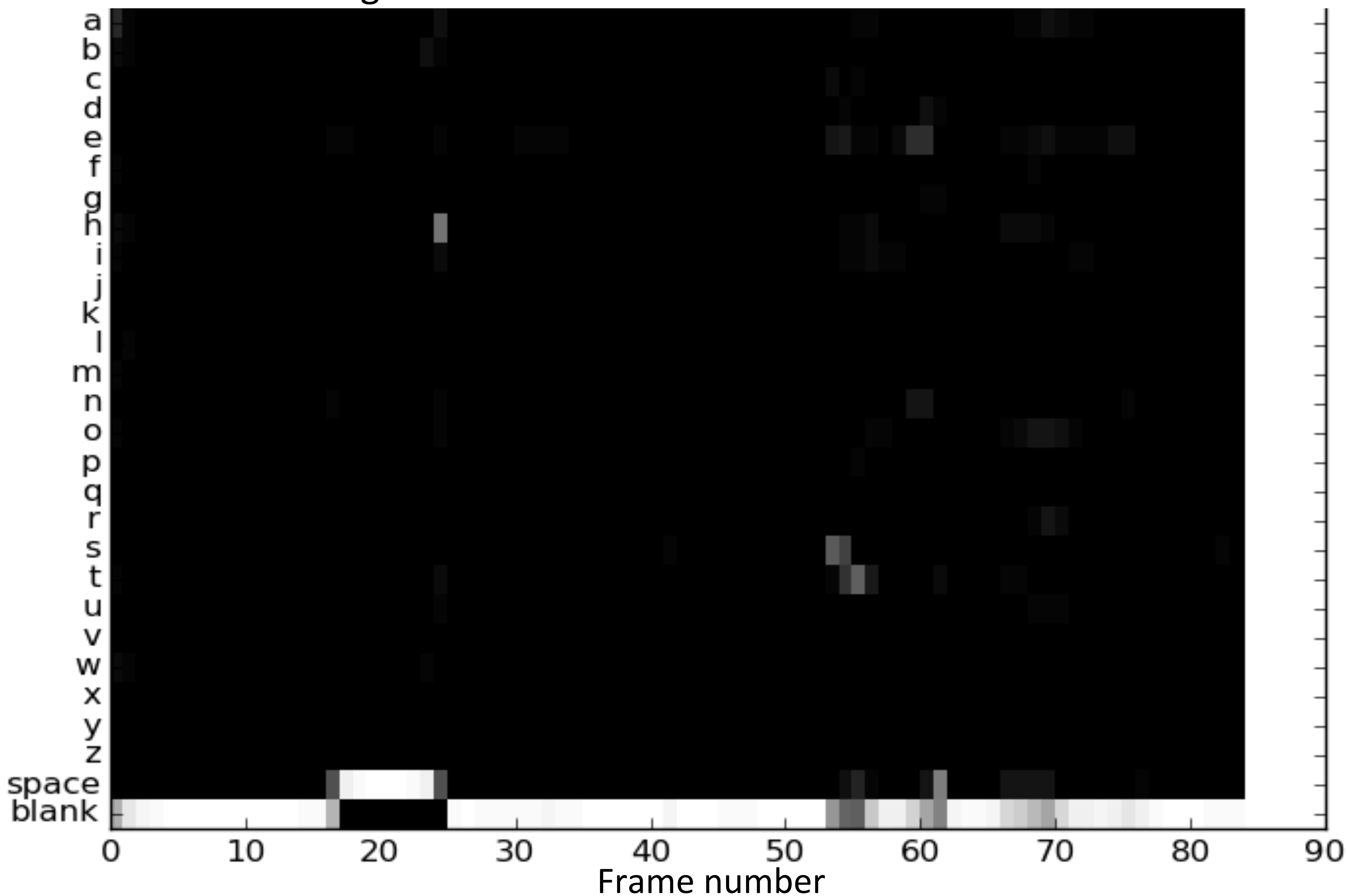
Example

- What happens inside?

Network outputs (c) at Iteration 300

(Thresholding / contrast added for clarity.)

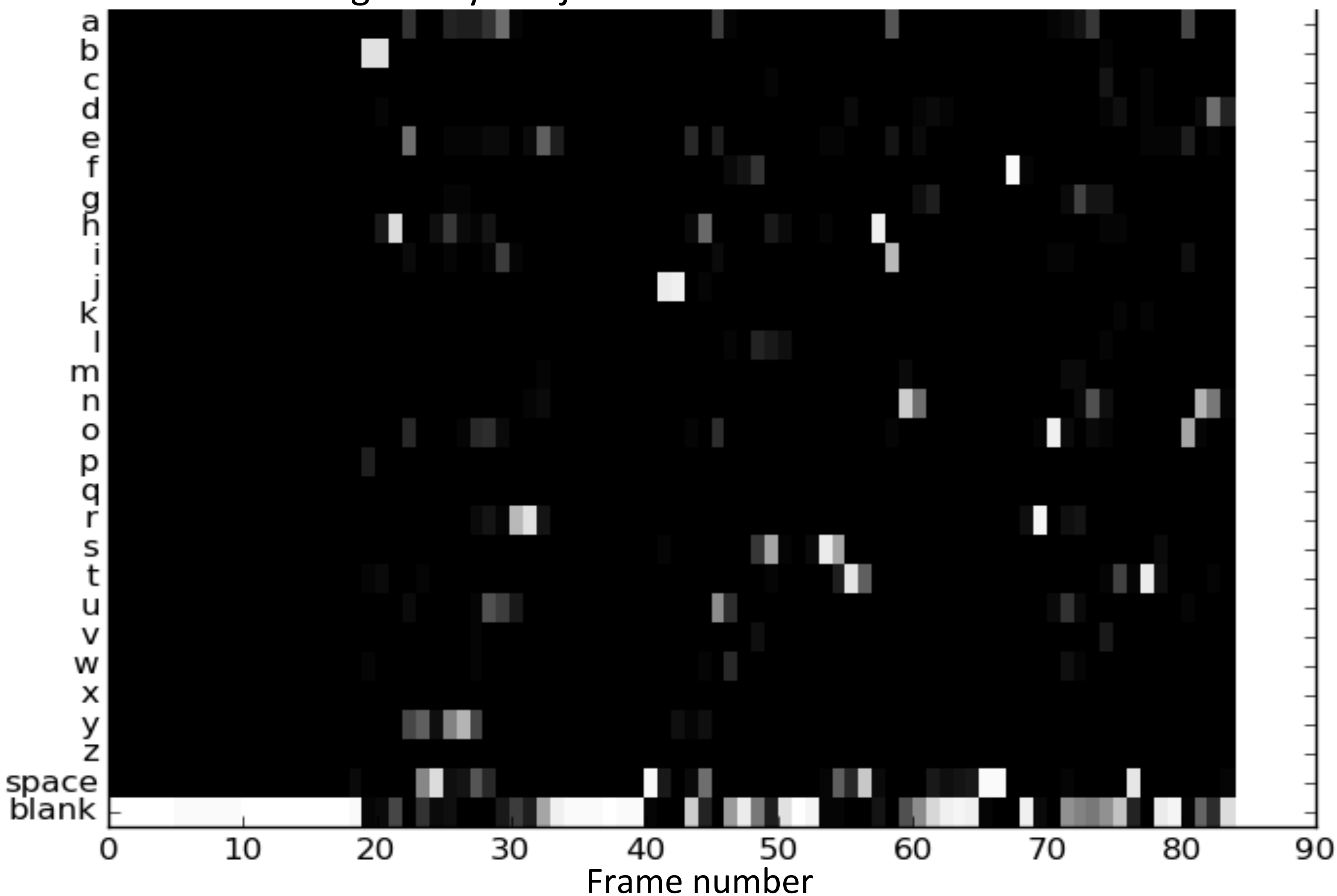
Max decoding: h



Network outputs (c) at Iteration 1500

(Thresholding / contrast added for clarity.)

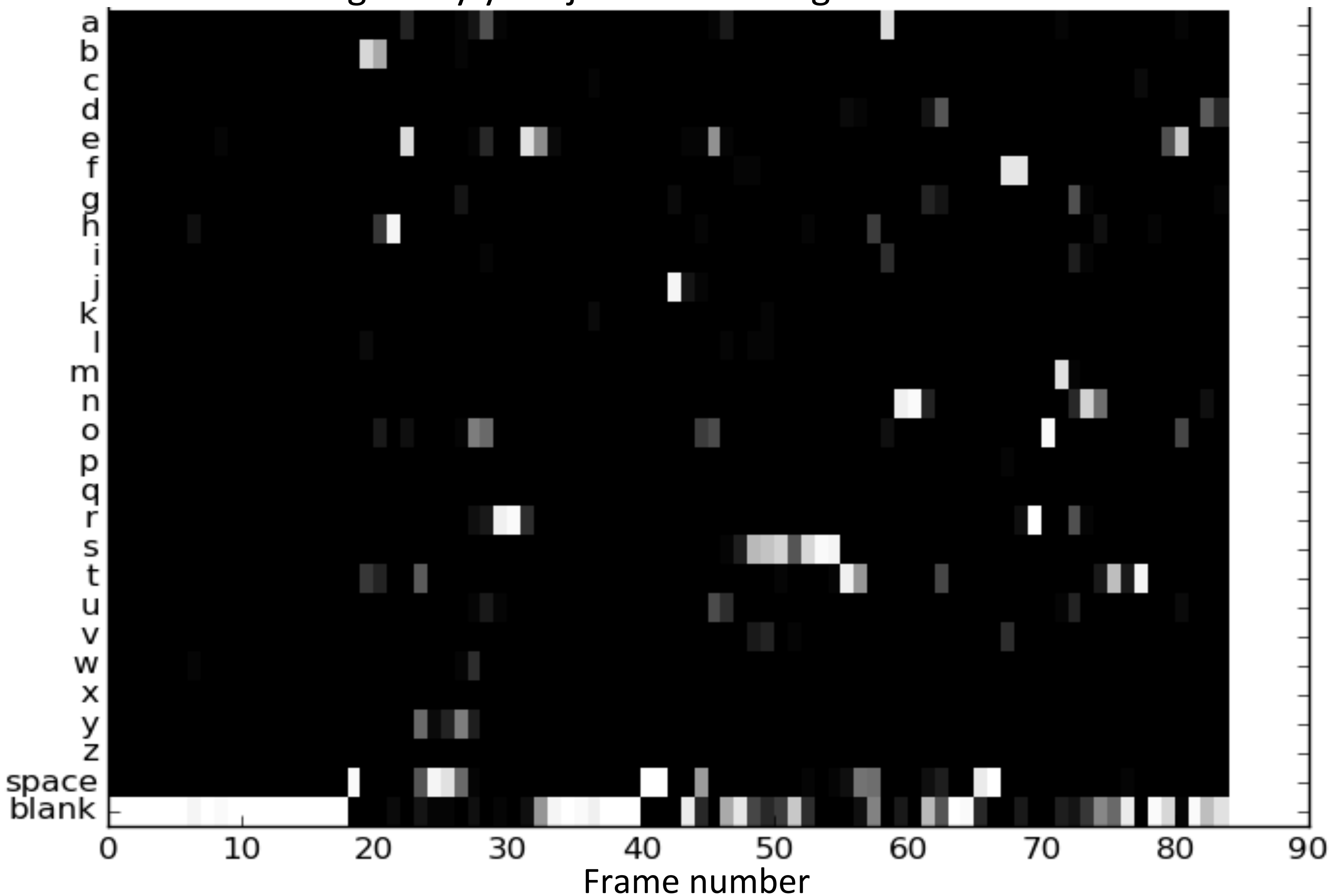
Max decoding: bhe y uar j usst hin fro ton



Network outputs (c) at Iteration 2500

(Thresholding / contrast added for clarity.)

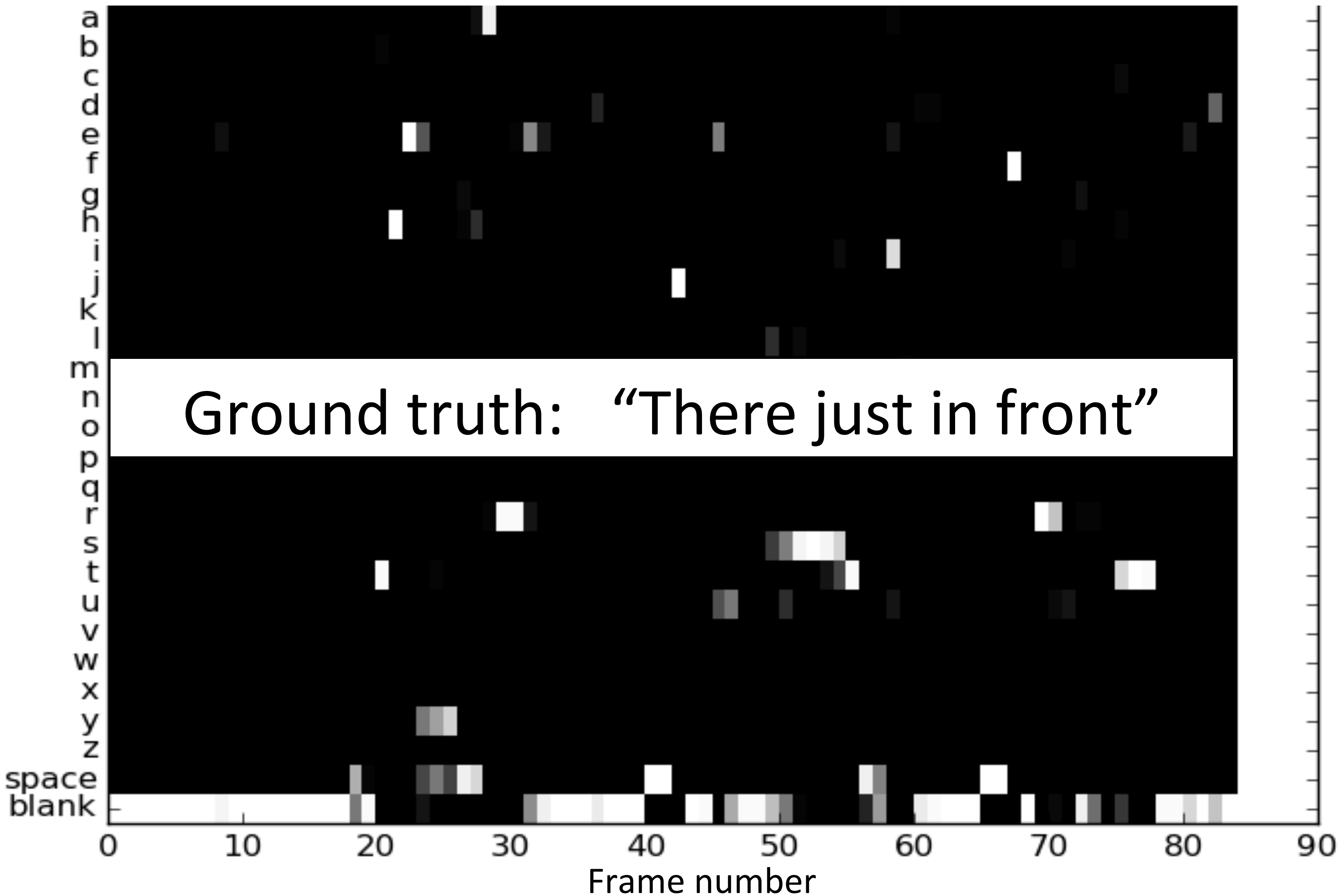
Max decoding: bhey yore j esstand fromgntte



Network outputs (c) at Iteration 5500

(Thresholding / contrast added for clarity.)

Max decoding: they ar jest in front



Max Decoding

- Examples:

Max Decoding
"put pore lotttle thank and sr crits sinpt the atting to them having been turned ef the wal al thes years con"

True Label
"the poor little things cried cynthia think of them having been turned to the wall all these years"

Max Decoding
"that is true baddel gre"

True Label
"that is true badauderie"

Decoding

- Network outputs $P(c|x)$. How do we find most likely transcription from $P(y|x)$?
- No efficient solution in general. Resort to search!
  - See [Graves et al., 2006] for prefix decoding strategy.

Language models

- Even with better decoding, CTC model tends to make spelling + linguistic errors. E.g.:

RNN output
what is the weather like in bostin right now
prime miniter nerenr modi
arther n tickets for the game

From Hannun et al., 2014.

- P(y|x) modeled directly from audio.
  - But not enough audio data to learn complicated spelling and grammatical structure.
  - Only supports small vocabulary.
  - For grapheme models: "Tchaikovsky" problem.

Language models

- Two solutions
  - Fuse acoustic model with language model: P(y)
  - Incorporate linguistic data:
    - Predict phonemes + pronunciation lexicon + LM.

- Possible to train language model from massive *text* corpora.
  - Learn spelling + grammar
  - Greatly expand vocabulary
  - Elevate likely cases ("Tchaikovsky concerto") over unlikely cases ("Try cough ski concerto").

Language models

- Standard approach: n-gram models
  - Simple n-gram models are common, well supported.
    - KenLM: kheafield.com/code/kenlm/
  - Train easily from huge corpora.
  - Quickly update to follow trends in traffic.
  - Fast lookups inside decoding algorithms.

Decoding with LMs

- Given a word-based LM of form $P(w_{t+1} | w_{1:t})$, Hannun et al., 2014 optimize:

$$\arg\max_{w} P(w|x)P(w)^{\alpha}[\text{length}(w)]^{\beta}$$

$P(w|x) = P(y|x)$ for characters that make up w.

α and β are tunable parameters to govern weight of LM and a bonus/penalty for each word.

Decoding with LMs

- Basic strategy: beam search to maximize

$$\arg\max_{w} P(w|x)P(w)^{\alpha}[\text{length}(w)]^{\beta}$$

Start with set of candidate transcript prefixes, A = {}.

For t = 1..T:

For each candidate in A, consider:

1. Add blank; don't change prefix; update probability using AM.
2. Add space to prefix; update probability using LM.
3. Add a character to prefix; update probability using AM.

Add new candidates with updated probabilities to A_{new}.

A := K most probable prefixes in A_{new}.

Decoding with LMs: Examples

RNN output	Decoded Transcription
what is the weather like in bostin right now	what is the weather like in boston right now
prime miniter nerenr modi	prime minister narendra modi
arther n tickets for the game	are there any tickets for the game

From Hannun et al., 2014.

Rescoring

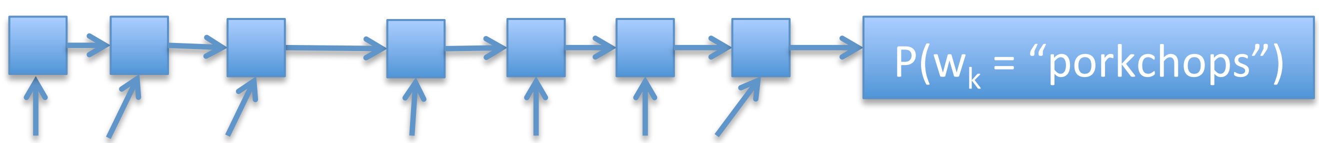
- Another place to plug in DL algorithms:

  Systems usually produce N-best list.

  Use fancier models to "rescore" this list.

Rescoring with Neural LM

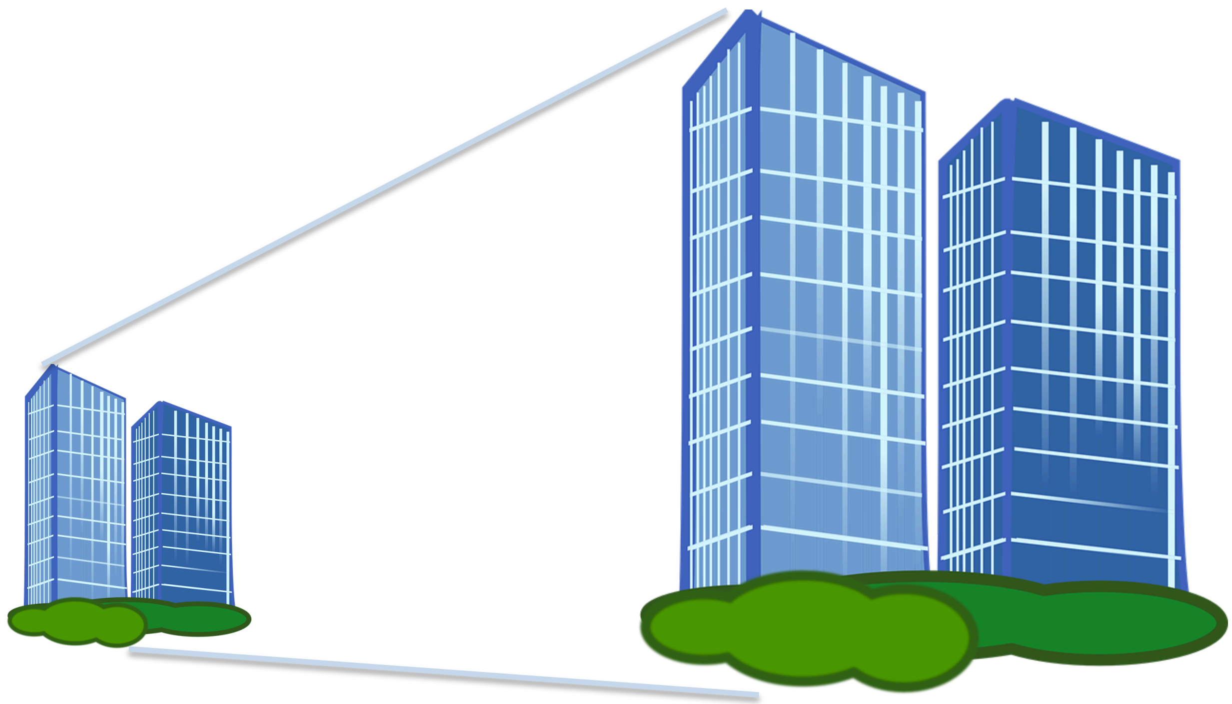
- Example: train neural language model and rescore word transcriptions.
  - Cheap to evaluate $P(w_k | w_{k-1}, w_{k-2}, \ldots, w_1)$ NLM on many sentences.
  - In practice, often combine with N-gram trained from big corpora.



1. (-25.45) I'm a connoisseur looking for wine and porkchops. -24.45
2. (-26.32) I'm a connoisseur looking for wine and port shops. -23.45
3. …
4. …
5. …

SCALING UP

Scale model



- Two main components to scale from “tutorial” to state-of-art accuracy:
  - Data
  - Computing power

Data

- Transcribing speech data isn't cheap, but not prohibitive:
  – Roughly 50¢ to $1 per minute.

- Typical speech benchmarks offer 100s to few 1000s of hours.
  – LibriSpeech (audiobooks)
  – LDC corpora (Fisher, Switchboard, WSJ) ($$)
  – VoxForge

Types of speech data

- Application matters
  - We want to find data that matches our goals.

Styles of speech	Issues	Applications
Read	Disfluency / stuttering	Dictation
Conversational	Noise	Meeting transcription
Spontaneous	Mic quality / #channels	Call centers
Command/control	Far field	Device control
	Reverb / echo	Mobile texting
	Lombard effect	Home / IoT / Cars
	Speaker accents	

Read speech

- Reading is inexpensive way to get more data.

  < $10/hour depending on source

- Disadvantages:
  - Misses inflection/conversational tone
  - Lombard effect
  - Speaker variety sometimes a limitation.

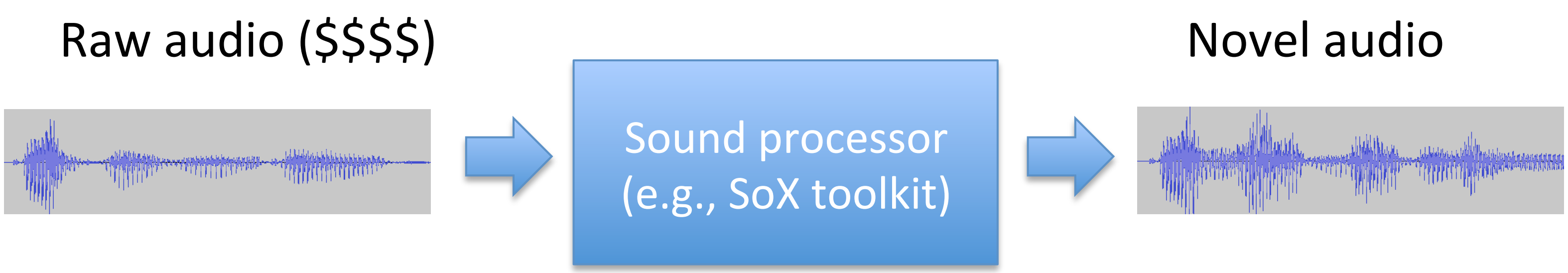
Read speech

- Some tricks we've tried to address these:
  - Elicit "voice acting" by using movie scripts:
  - Elicit Lombard effects by playing loud noise (sometimes via headphones):

[Sanjeev Satheesh]

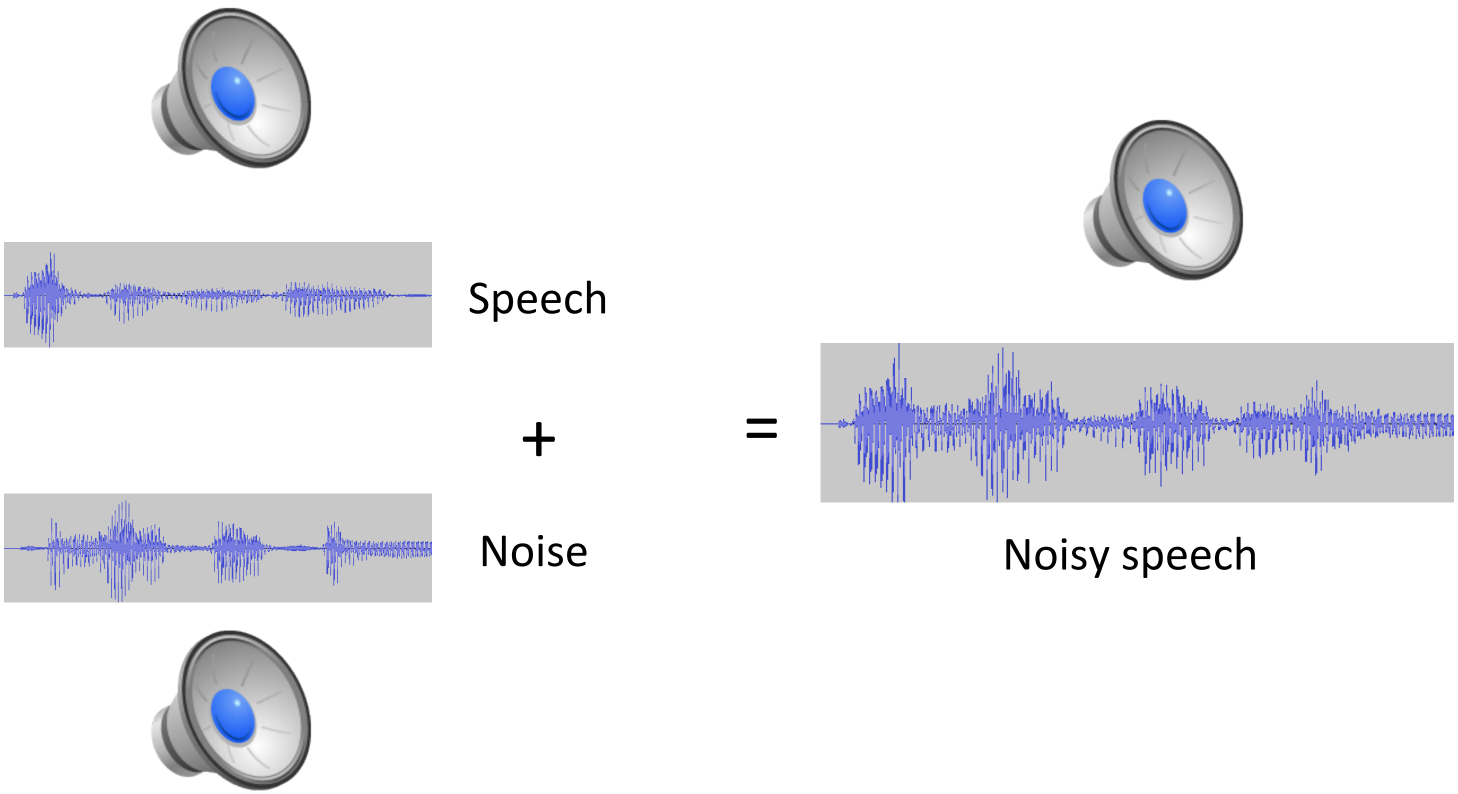
Augmentation

- Many forms of distortion that model should be robust to:
  – Reverb, noise, far field effects, echo, compression artifacts, changes in tempo



http://sox.sourceforge.net/

Example: additive noise



DeepSpeech 2: 10K hours of raw audio -> 100K hours of novel training data

Example: tempo

WSJ example

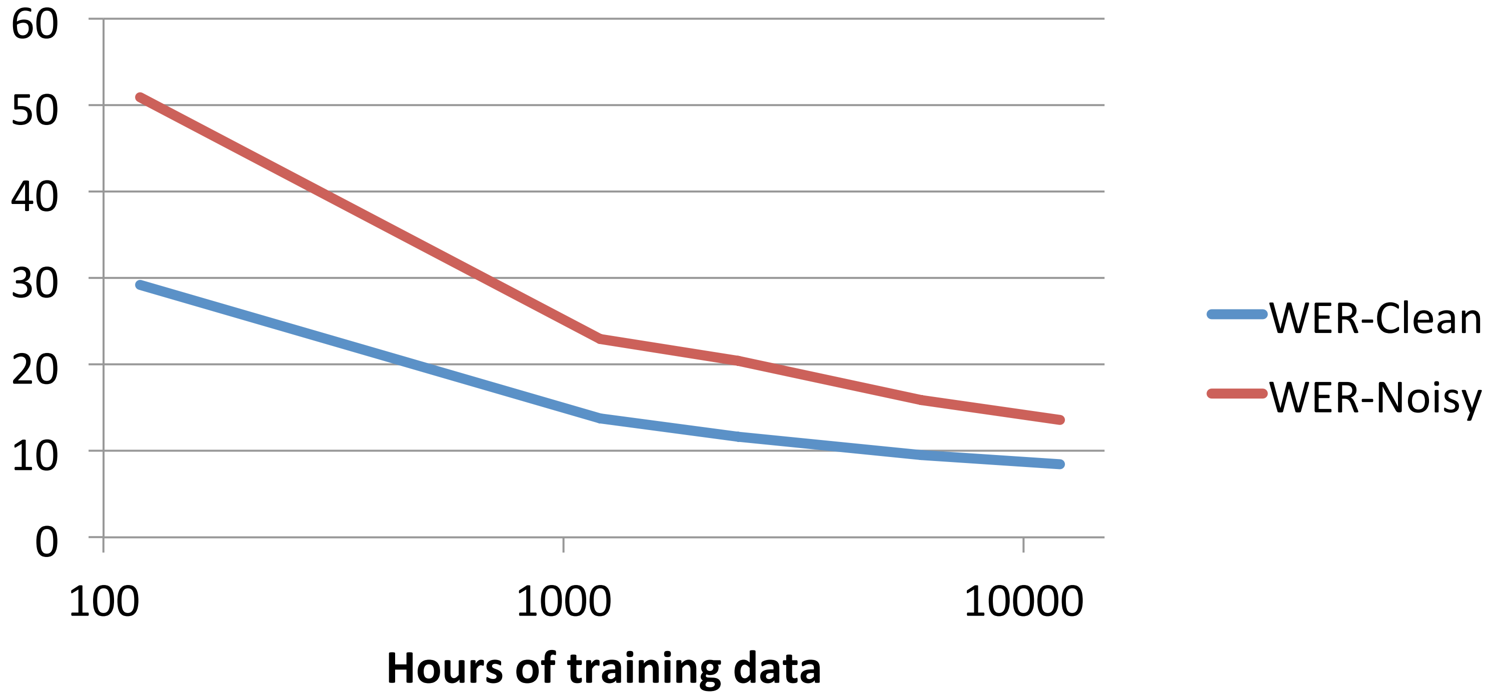
Faster WSJ reader w/ reverb

sox <infile> <outfile> tempo 1.3 reverb

- In general: easier to engineer data pipeline than to engineer recognition pipeline.

Results: DeepSpeech 2

- Steady fall in error rates with new raw data.



(Assuming we have a big enough model!)

Computation

- How big is 1 experiment?

At least:

(# connections)· (# frames) · (# utterances) · (# epochs) · 3 · 2 FLOPs

E.g.: for DS2 with 10K hours of data:

$$100e6 \cdot 100 \cdot 10e6 \cdot 20 \cdot 3 \cdot 2 = 1.2e19 \text{ FLOPs}$$

~30 days with well-optimized code on Titan X.

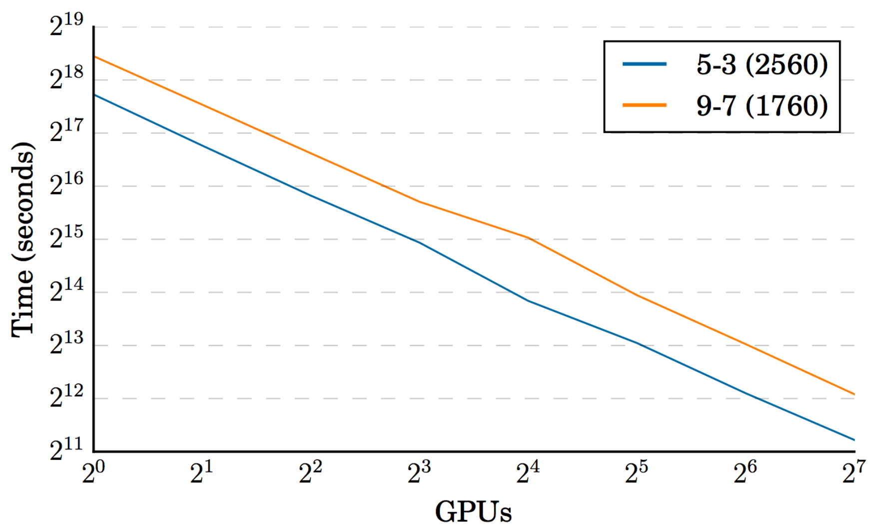
Computation

- Easy: use more GPUs with data parallelism.
  - Minibatches up to 1024 seem useful.
  - Would like ≥64 utterances per GPU.

- Current servers support ~8 Titans.
  - Will get you < 1 week training time.

Computation: multi-node

- Many ways to use more GPUs.
  - At Baidu, use synch. SGD:

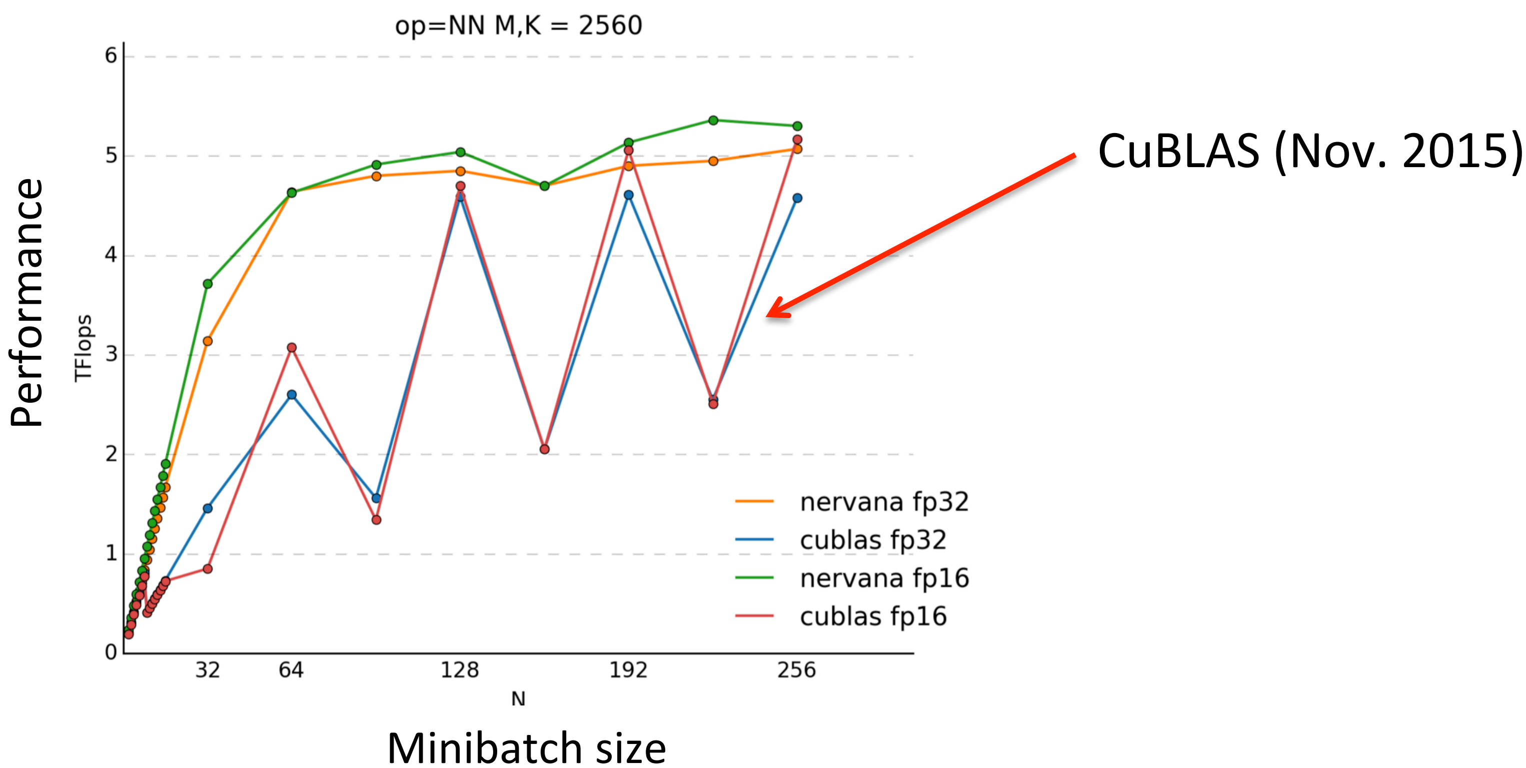


Needed to optimize OpenMPI to achieve efficiency.

  - Other solutions:
    - Async SGD [Dean et al., NIPS 2012]
    - Synch SGD w/ backup workers [Chen et al., ICLR 2016]

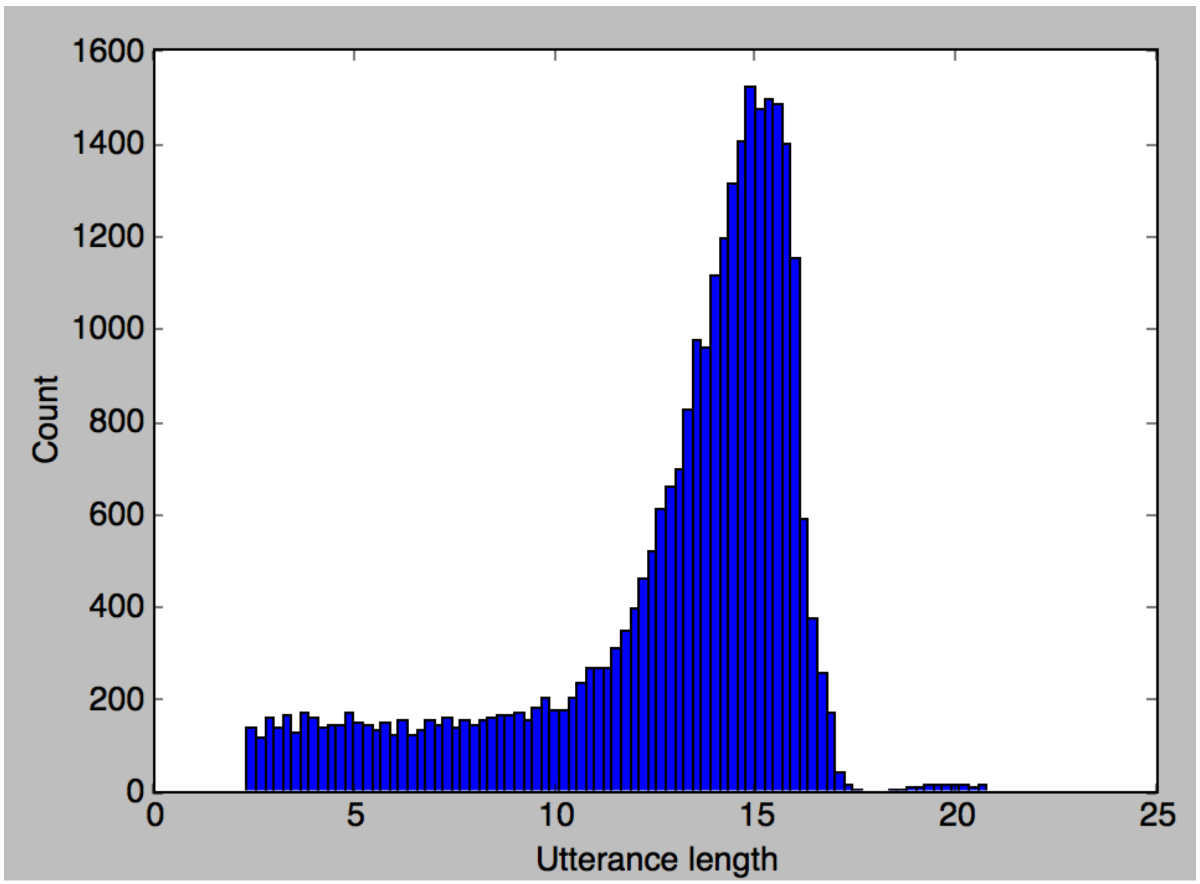
Computation

- Need *optimized* single-GPU code. But a lot of off-the-shelf code has inefficiencies.
- E.g., Watch for bad GEMM sizes:



Computation

- Try to keep similar-length utterances together.

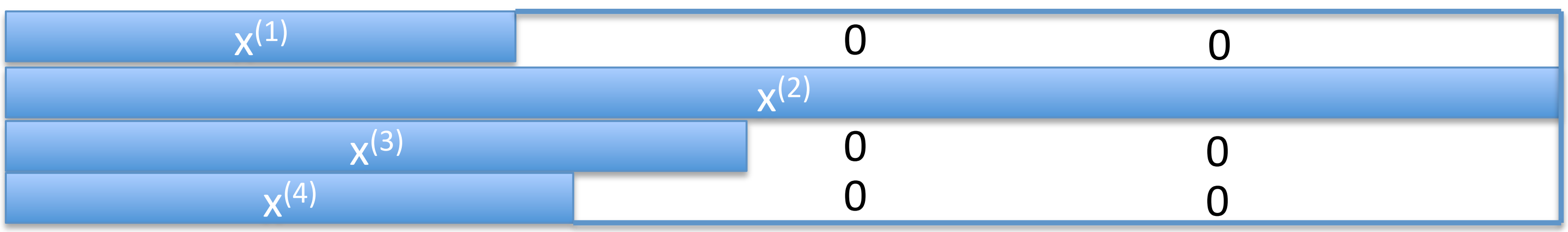


(LibriSpeech clean data.)

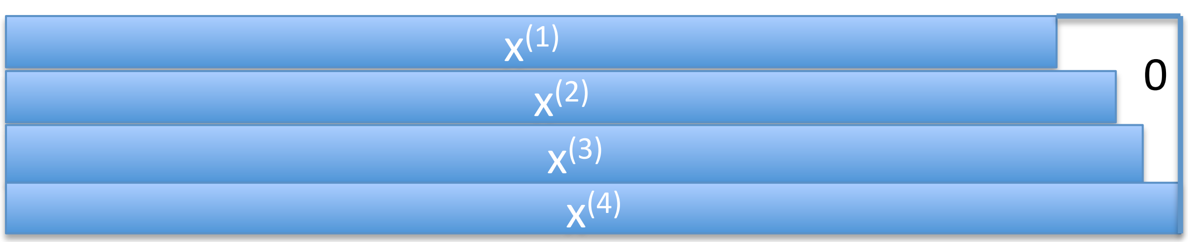
Computation

- Try to keep similar-length utterances together.

Bad minibatch:

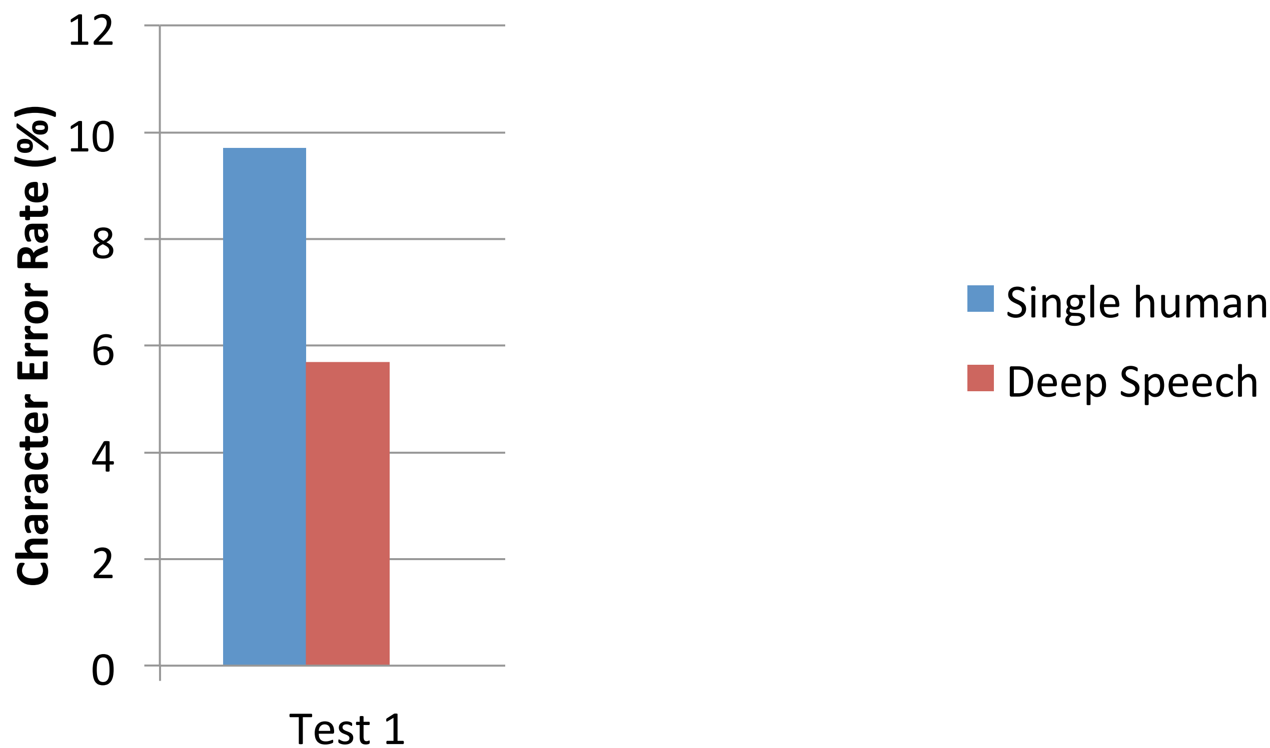


Good minibatch:



Results

- Scaled up models in Mandarin:



Reality check

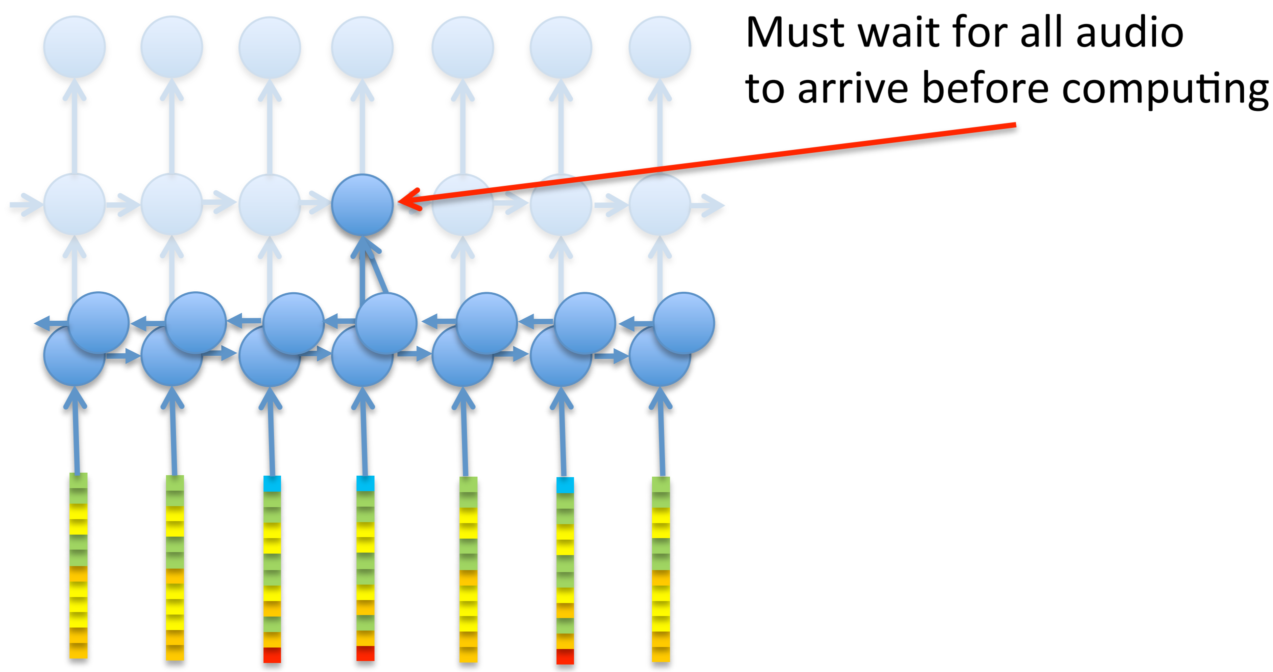
PRODUCTION

Production speech

- So far:
  - Train acoustic + language models.
  - Scale them up.
- But how to serve users?
  - Accuracy is only one measure of performance.
  - Users care about latency.
  - Need to serve economically.

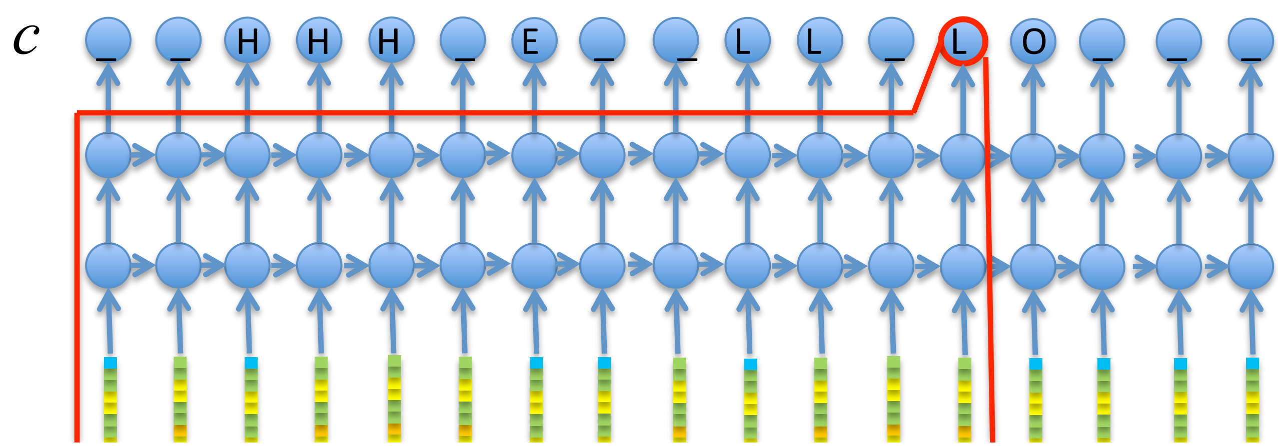
Latency

- Many acoustic model structures hard to serve in practice.
  - E.g., Bi-directional RNNs.



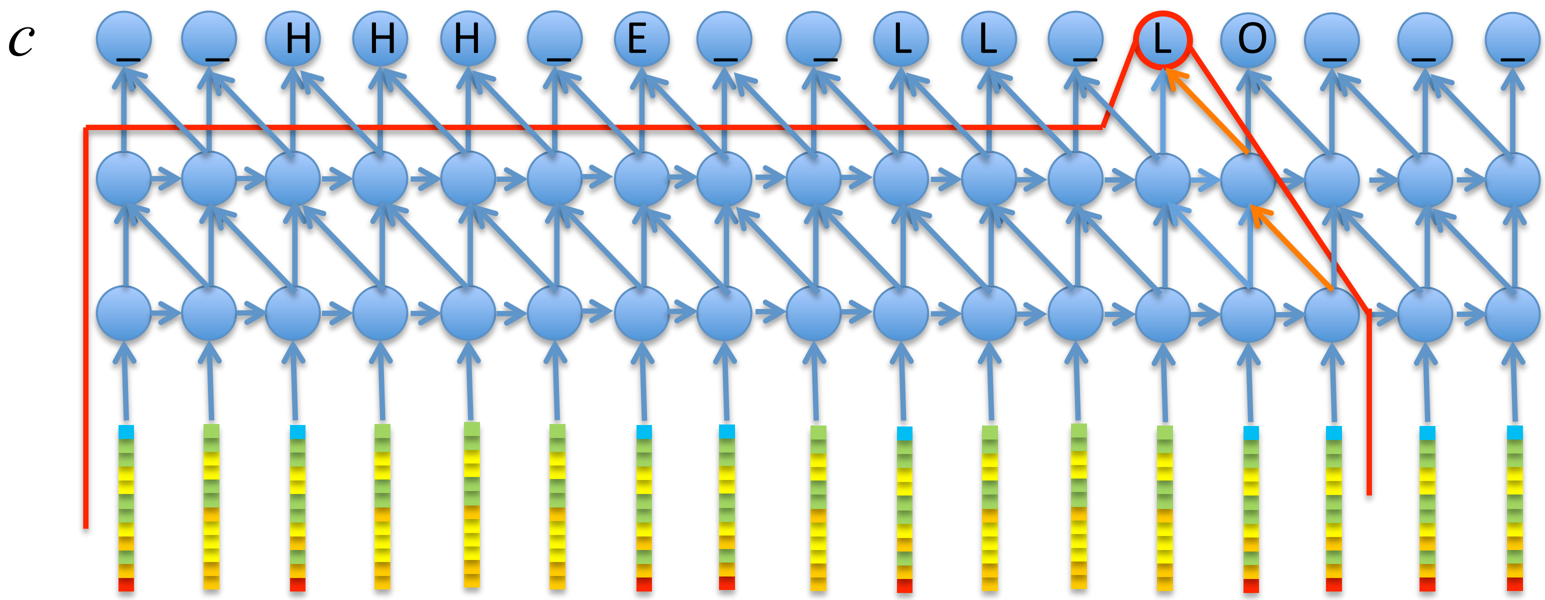
Latency

- Use forward-only RNNs. But:
  - Usually hurts accuracy. Why? Context?
  - CTC could learn to delay output on its own in order to improve accuracy.
    - ➢ In practice, tends to align transcription closely.
    - ➢ This is especially problematic for English letters (spelling!)



Latency

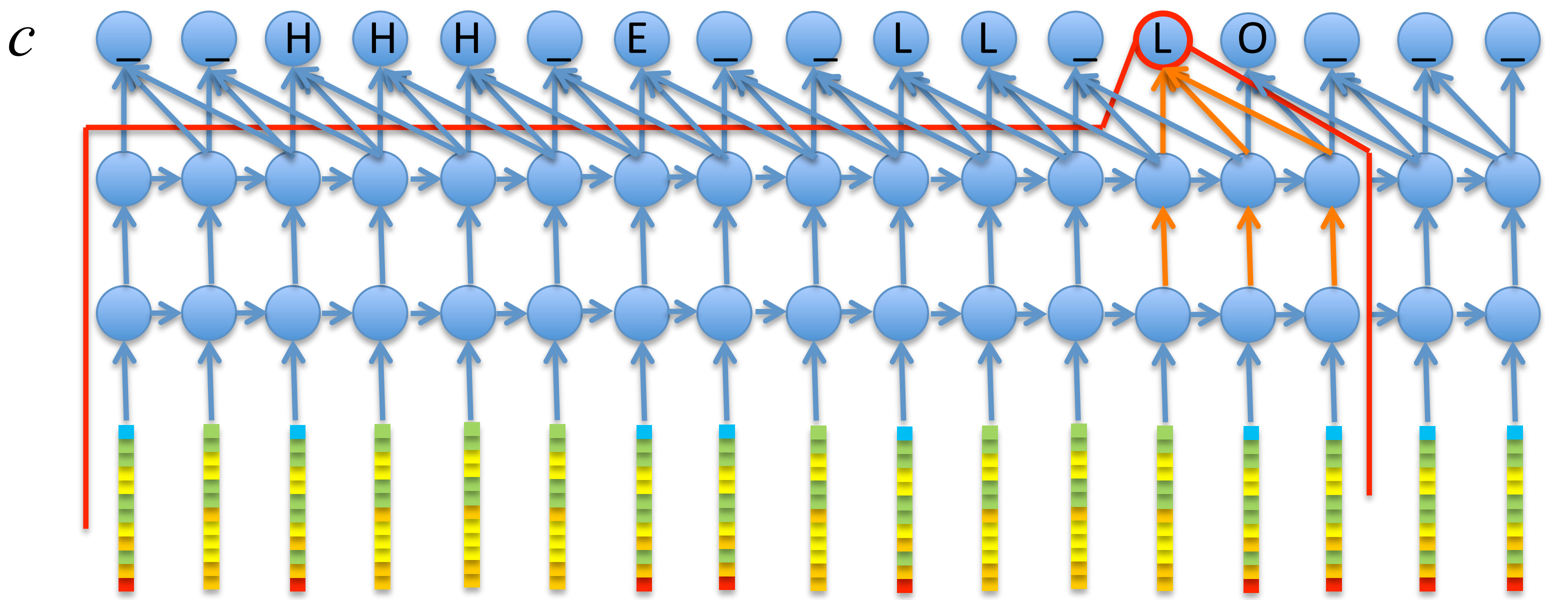
- Fix: bake limited context into model structure.



- Caveat: May need to compute upper layers quickly after sufficient context arrives. Can be easier if context is near top.

Latency

- Fix: Move most of context to the top.

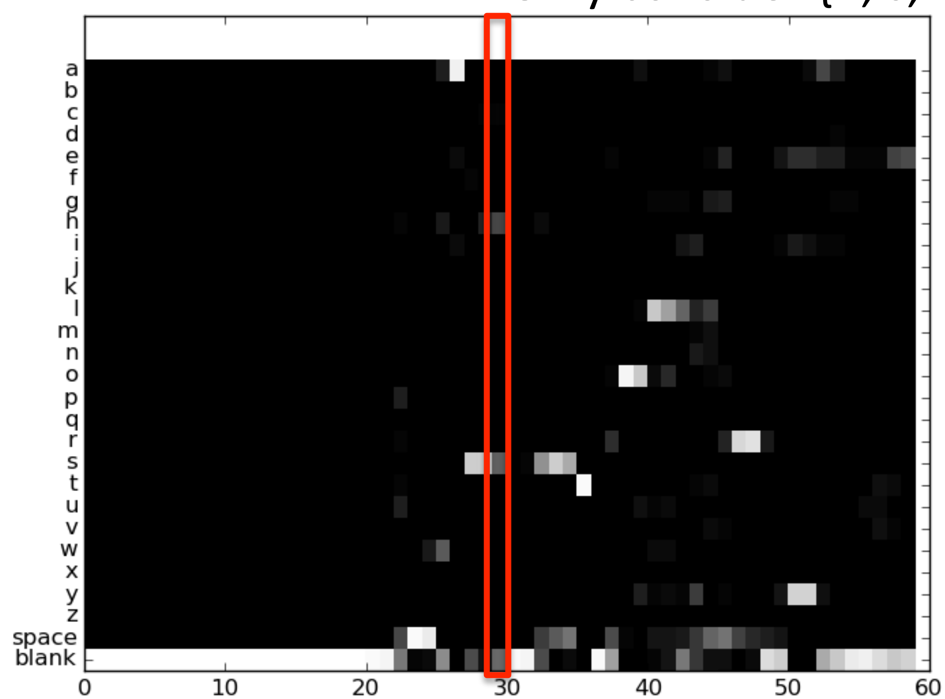


- Can easily compute/recompute top layers online.

Pruning candidates

- For models with many character outputs (e.g., Mandarin), decoding slows down.
  - Trick: only consider top 99% of characters according to CTC.
  - 150x speedup for 5000 char output.

Only consider {h, s, blank}



Throughput

- Large DNN/RNN models hard to deploy on CPUs.
- Large DNN/RNN models run great on GPUs.
  - But only if "batch size" is high enough.
  - ➢ Processing 1 audio stream at a time is inefficient.

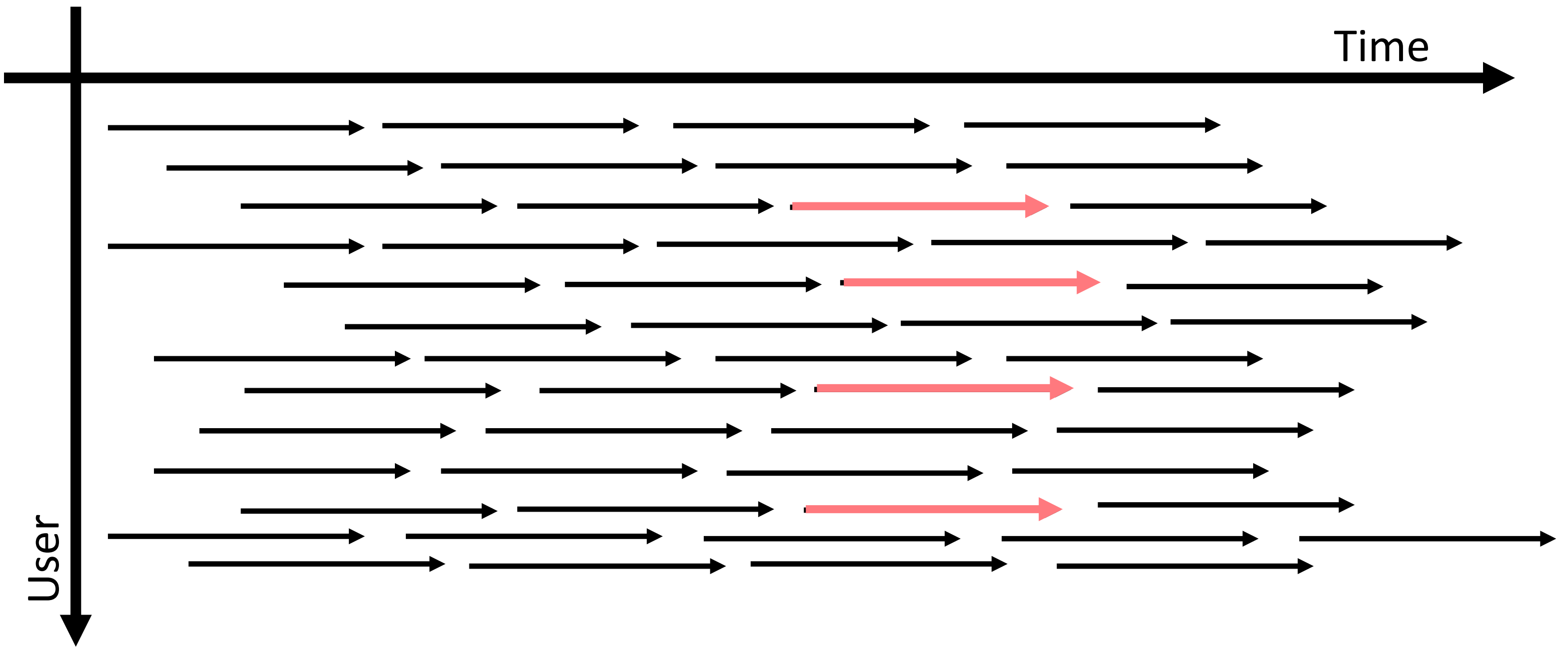
Performance for K1200 GPU:

Batch Size	FLOPs	Throughput
1	0.065 TFLOPs	1x
10	0.31 TFLOPs	5x
32	0.92 TFLOPs	14x

[Chris Fougner]

Throughput

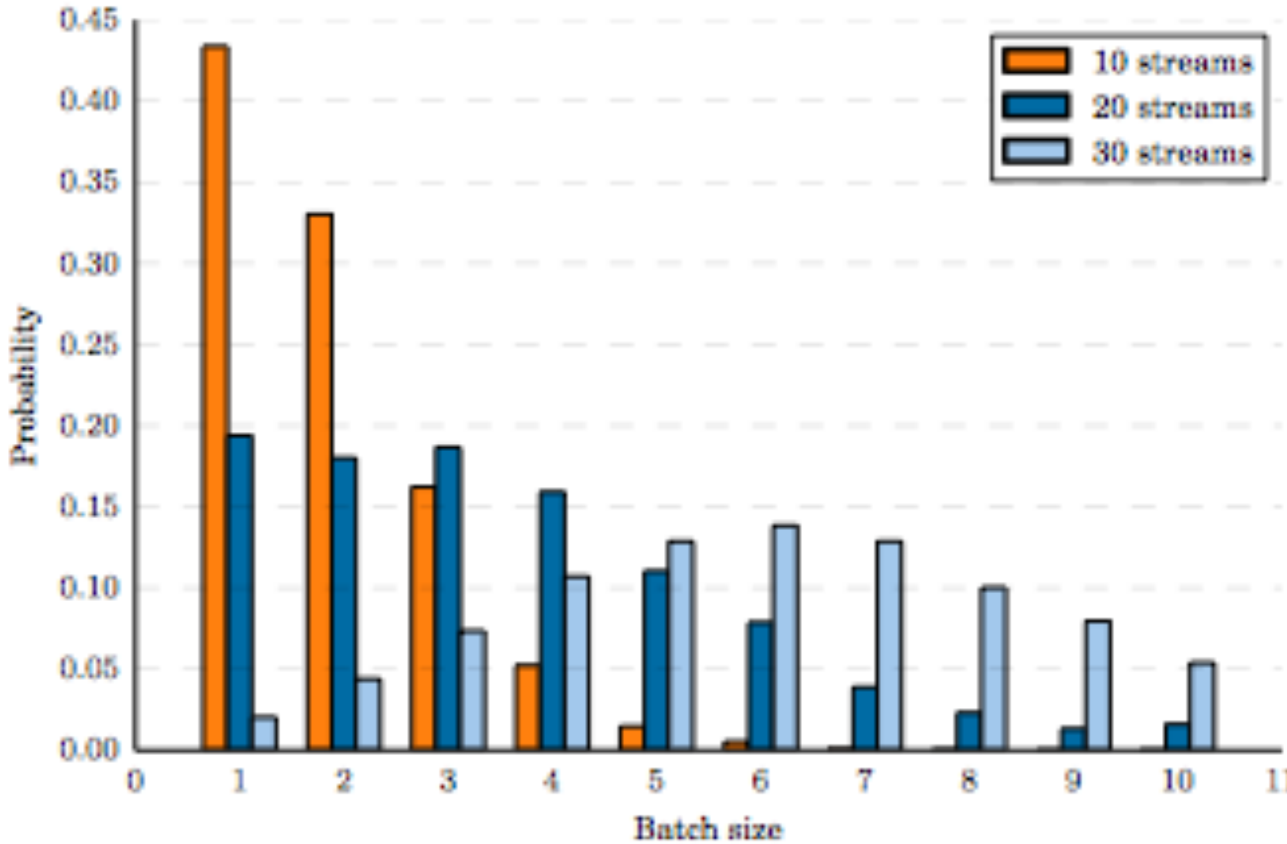
- Batch packets together as data comes in.



- ➢ Arrows represent packet of speech data (e.g., 100ms).
- ➢ Idea: Process packets that arrive at similar times in parallel.

[Chris Fougner]

Throughput



With ~30 concurrent users, few GEMM batches have less than size 4.

Summary

- Deep Learning makes first steps to state-of-art speech system simpler than ever.

- Performance significantly driven by data & models.
  - Focus on scaling data + compute.
  - Try more models, make more progress!

- Mature enough for production.
  - DeepSpeech model is live in Mandarin & English.

Starter code: github.com/baidu-research/ba-dls-deepspeech

References

•Gales and Young. “The Application of Hidden Markov Models in Speech Recognition” Foundations and Trends in Signal Processing, 2008.

•Jurafsky and Martin. “Speech and Language Processing”. Prentice Hall, 2000.

•Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, Yoshua Bengio. “End-to-End Attention-based Large Vocabulary Speech Recognition.” 2015. arxiv.org/abs/1508.04395

•Bourlard and Morgan. “CONNECTIONIST SPEECH RECOGNITION: A Hybrid Approach”. Kluwer Publishing, 1994.

•William Chan, Navdeep Jaitly, Quoc V. Le, Oriol Vinyals. “Listen Attend Spell” 2015. arxiv.org/abs/1508.01211

•Jianmin Chen, Rajat Monga, Samy Bengio, Rafal Jozefowicz, “Revisiting Distributed Synchronous SGD.” ICLR 2016 arXiv:1604.00981

•A Graves, S Fernández, F Gomez, J Schmidhuber. “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks.” ICML, 2006.

•Dahl, Yu, Deng, Acero, “Large Vocabulary Continuous Speech Recognition with Context-Dependent DBN-HMMs”. ICASSP, 2011

•Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q.V. and Ng, A.Y. “Large scale distributed deep networks.” NIPS 2012

•Hannun, Maas, Jurafsky, Ng. “First-Pass Large Vocabulary Continuous Speech Recognition using Bi-Directional Recurrent DNNs” ArXiv: 1408.2873

•Hannun, et al. “Deep Speech: Scaling up end-to-end speech recognition”. ArXiv:1412.5567

•H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech", J. Acoust. Soc. Am., vol. 87, no. 4, pp. 1738-1752, Apr. 1990.

•H. Hermansky and N. Morgan, "RASTA processing of speech", IEEE Trans. on Speech and Audio Proc., vol. 2, no. 4, pp. 578-589, Oct. 1994.

•Vassil Panayotov, Guoguo Chen, Daniel Povey and Sanjeev Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books.” ICASSP 2015

•H. Schwenk, “Continuous space language models”, 2007.