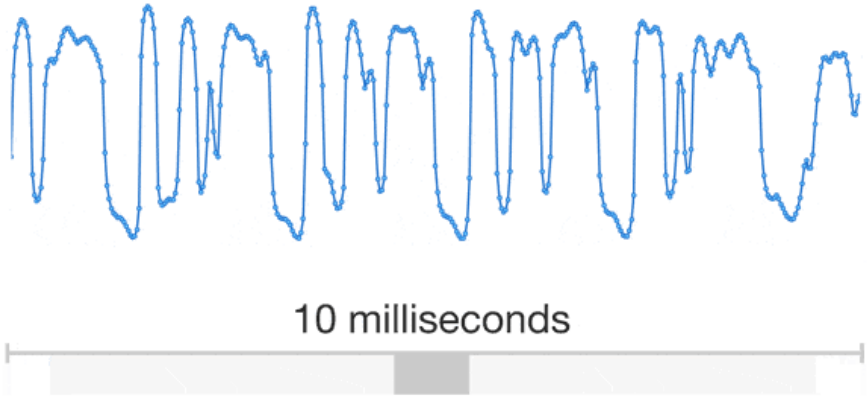
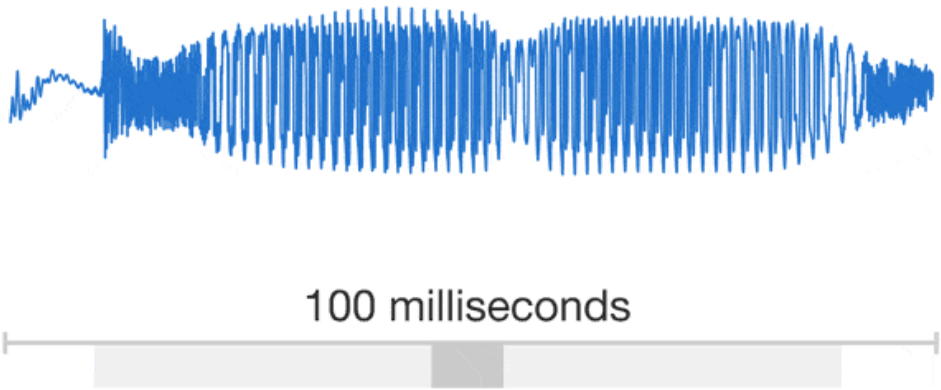
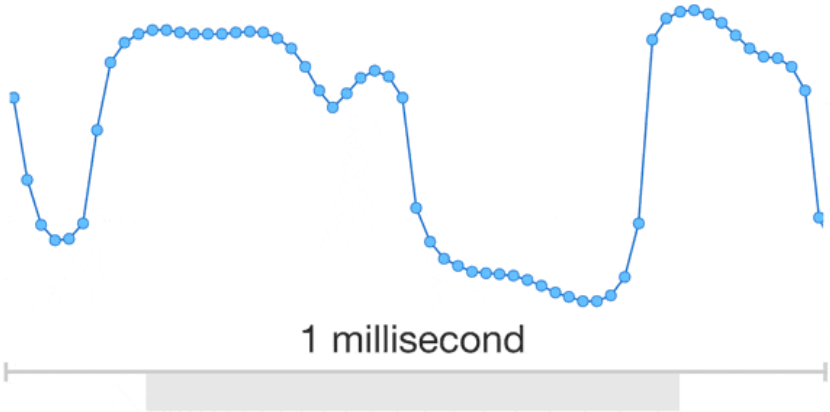
Machine Learning

Audio Applications

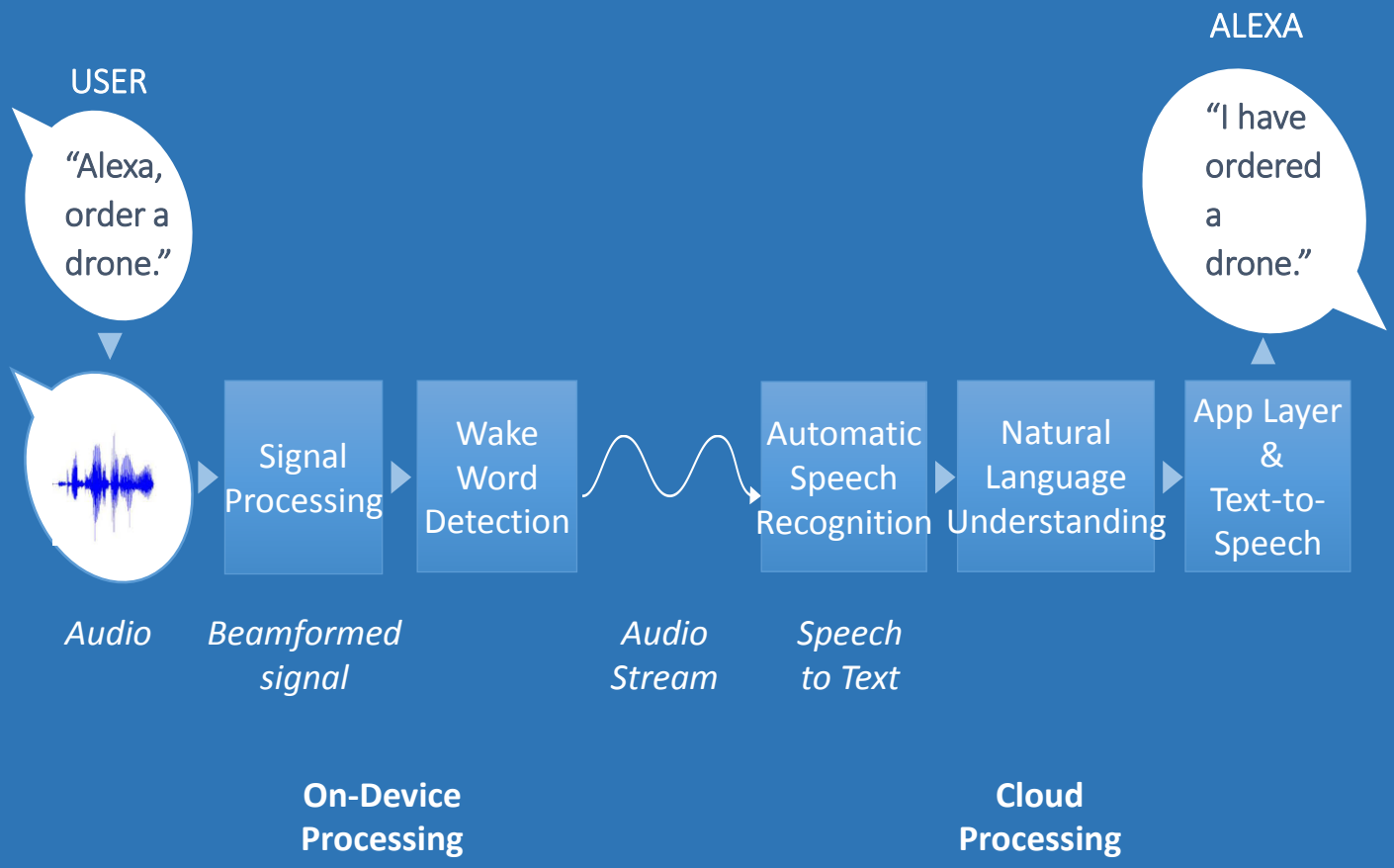
Raw Audio



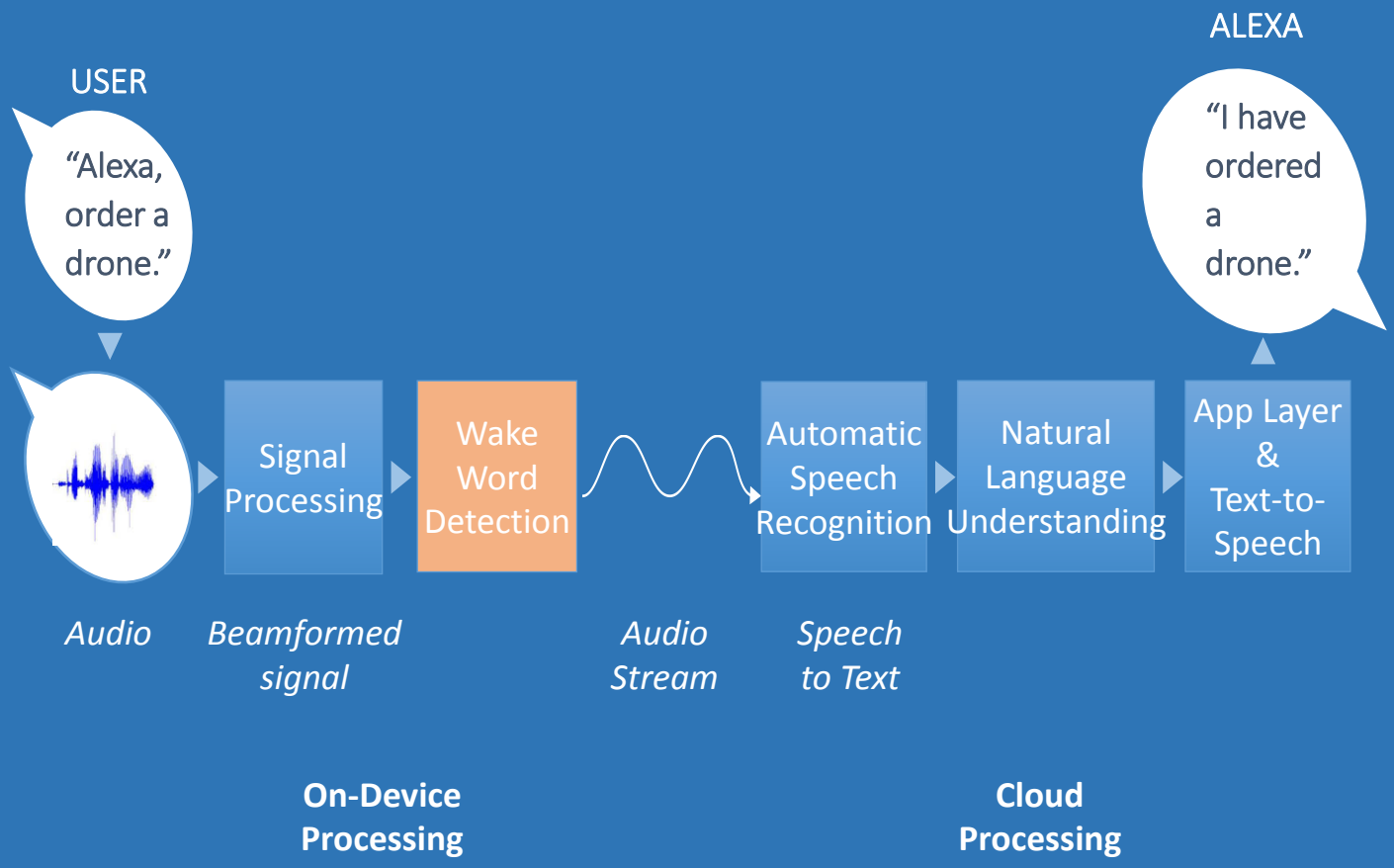




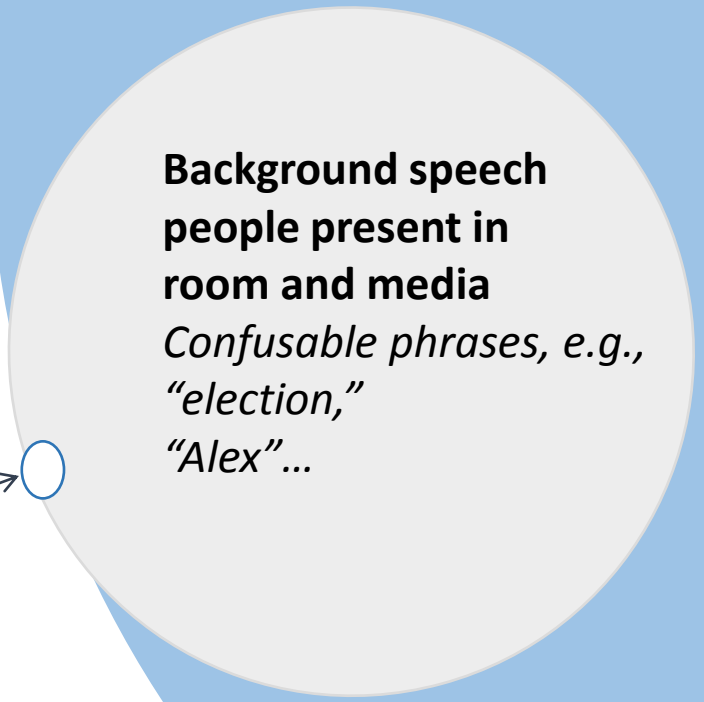
The Alexa Pipeline



The Alexa Pipeline



Problem Space for the Model

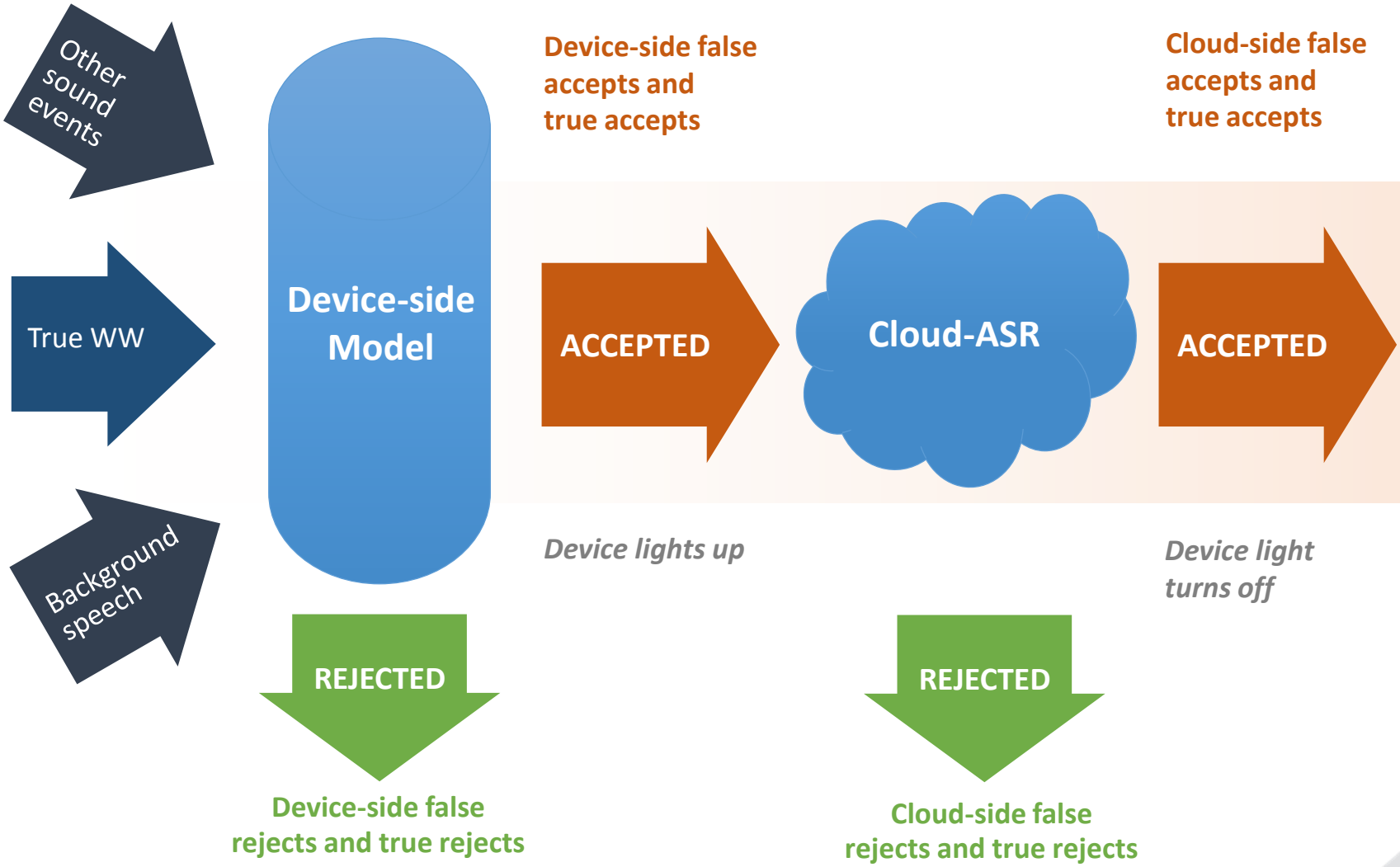


Background speech people present in room and media
Confusable phrases, e.g., "election," "Alex"…

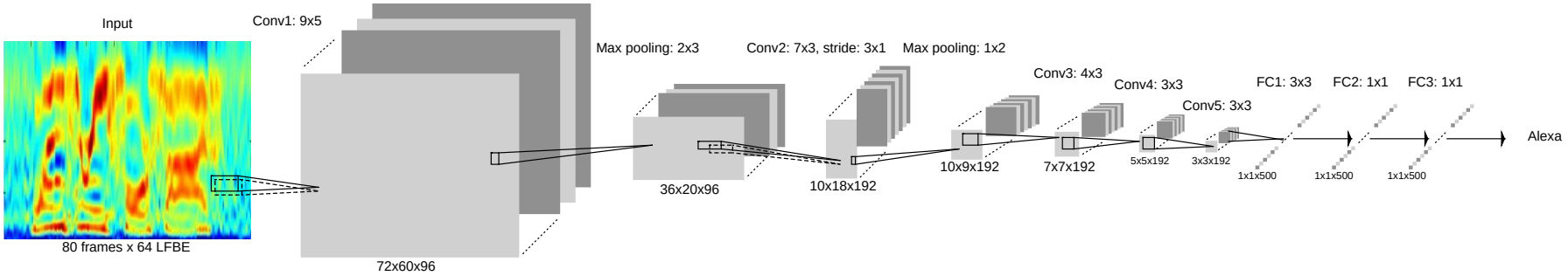
Universe of random sound events
Kitchen noise, office noise, birds…

True wakeword instances
Variation due to distance, gender, accent, mood…

WHAT HAPPENS DURING A DETECTION?



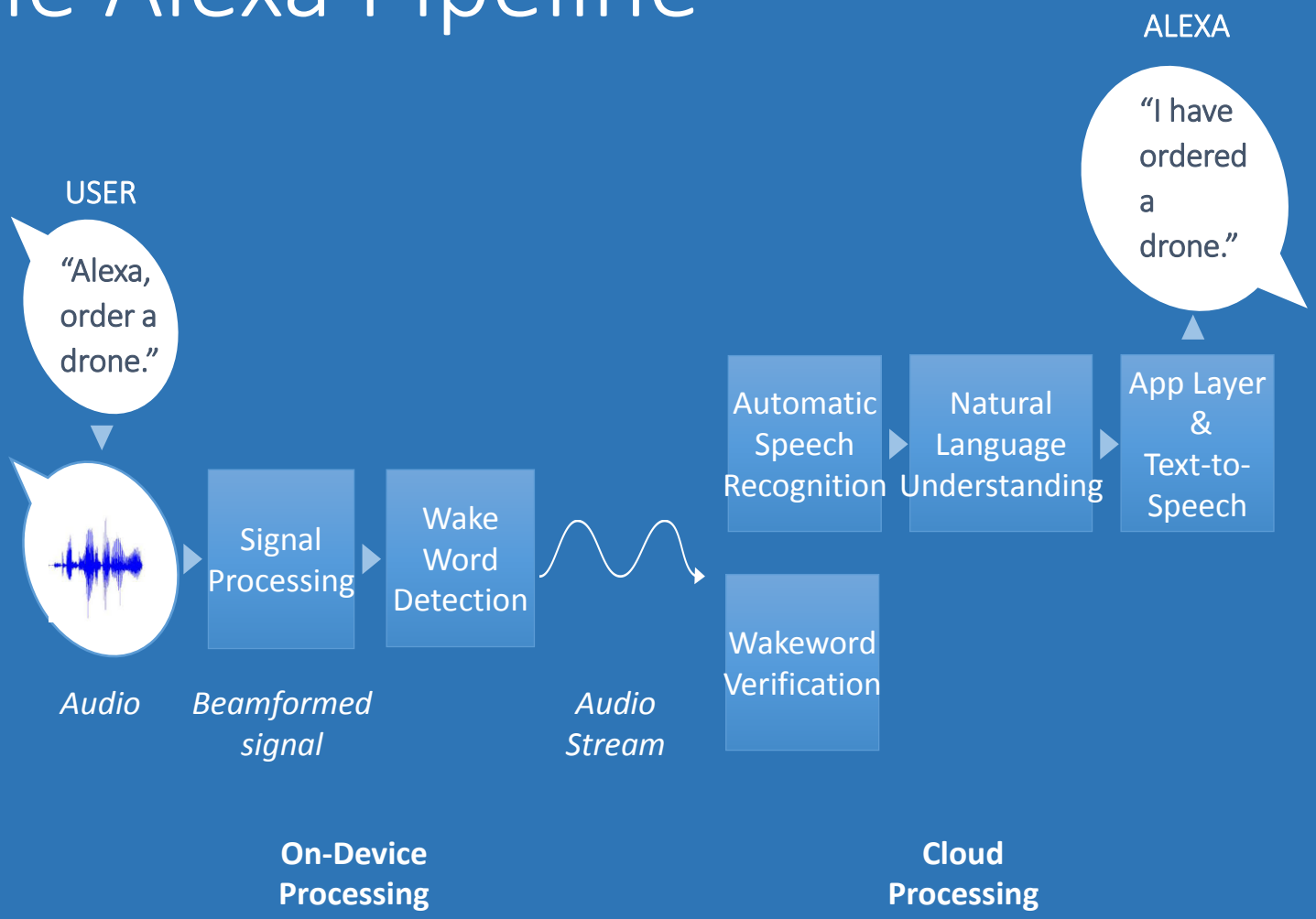
Wakeword Model



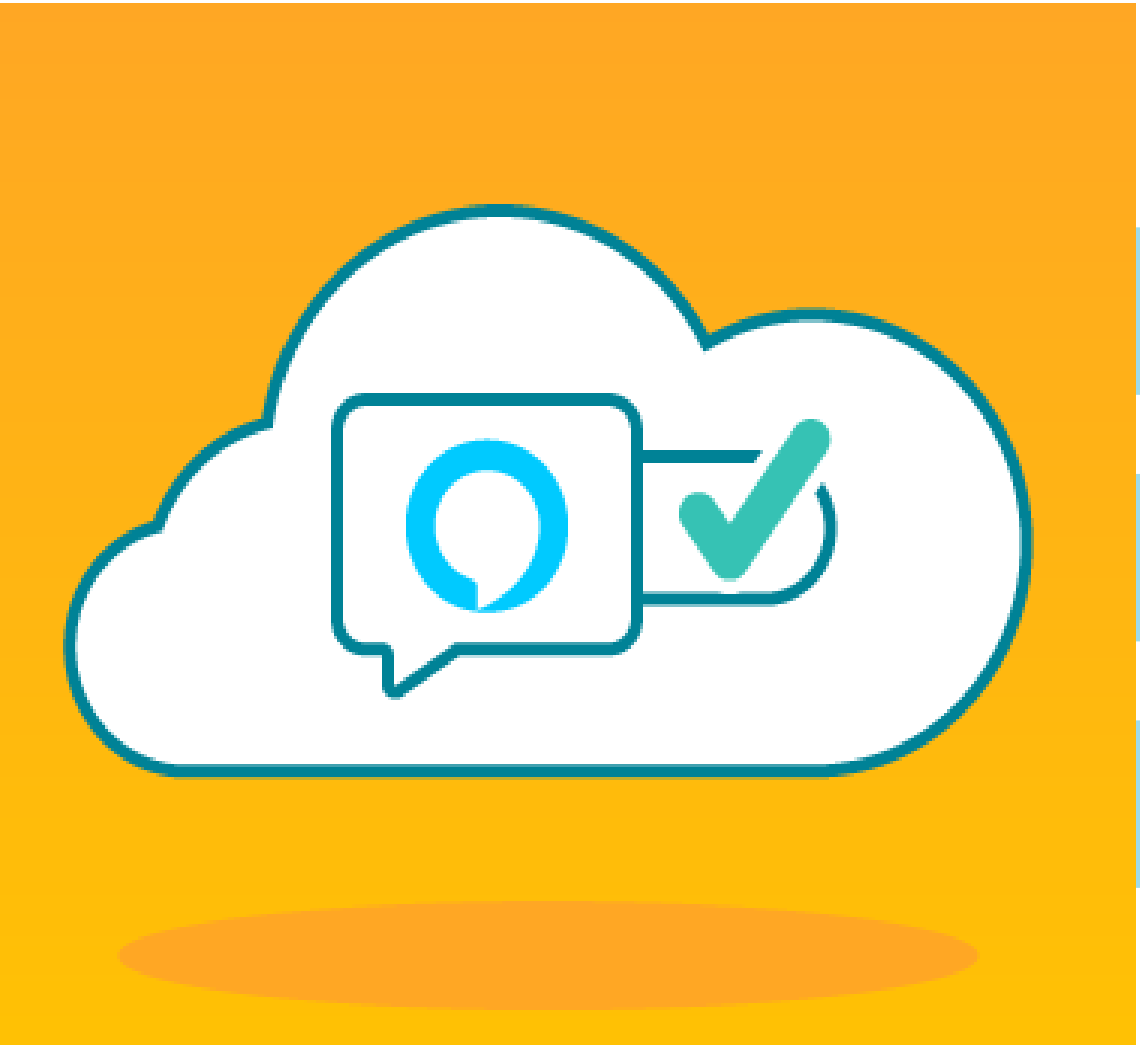
Feature Extraction Tutorial:

http://www.practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

The Alexa Pipeline

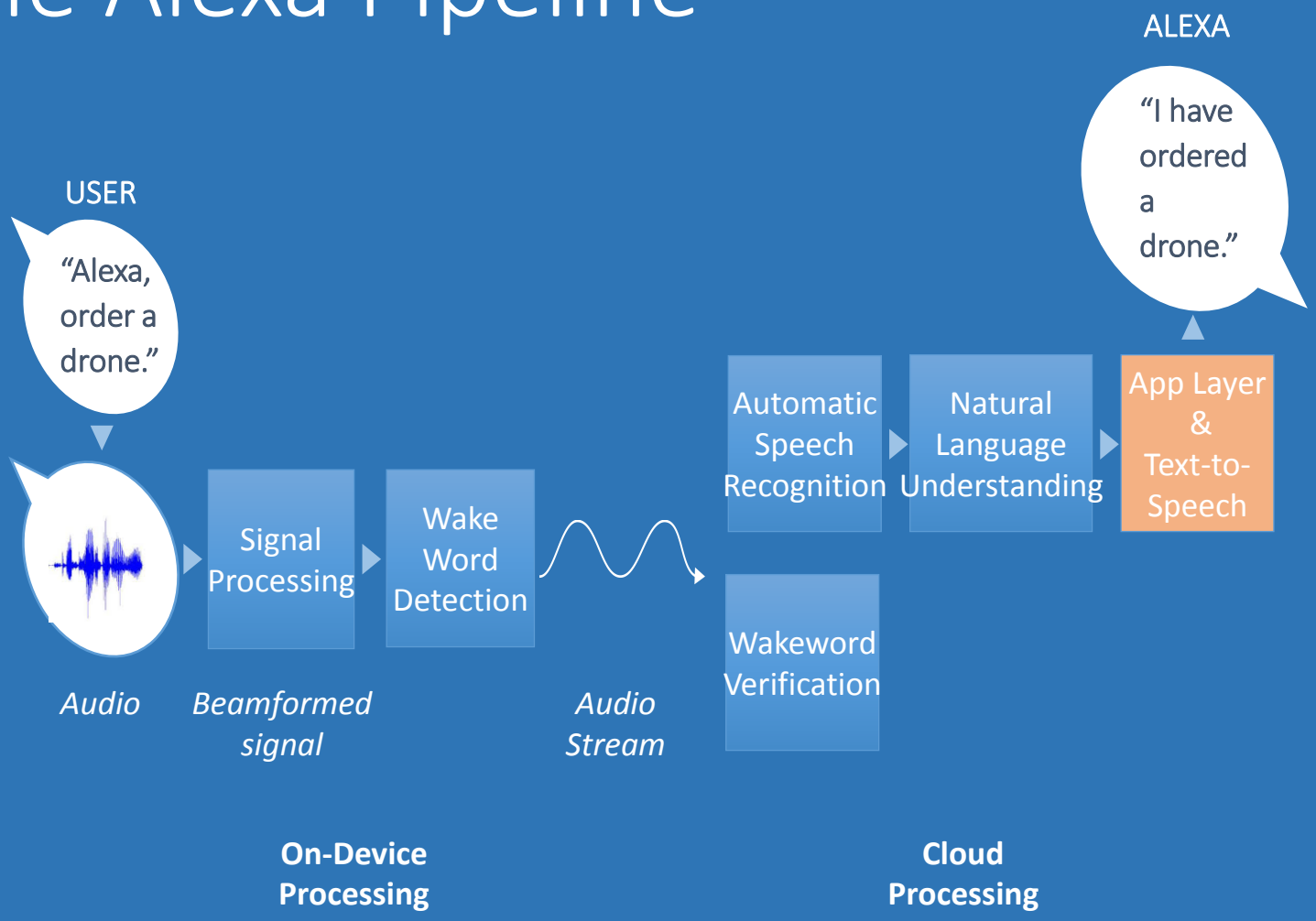


Cloud-Side Verification



- Fewer computational restrictions
- Can detect media false wakes
- Ability to incorporate more context

The Alexa Pipeline



Text to Speech

WaveNet - Overview

- Deep generative network for raw audio
- Based on the PixelRNN architecture
- Probabilistic and Autoregressive
  - Joint probability of the output waveform x is governed by

$$p(x) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1})$$

- Probability distribution modeled as a discrete softmax
- Applications in
  - Speech generation
  - Text to speech
  - Music

Approaches for Generative Networks

- Generative Adversarial Networks (Goodfellow et al. 2014)
  - Training is a tug of war between generator and discriminator
- Variational Autoencoders (Kingma et al. 2013)
  - Use probabilistic graphical models and maximize lower bound on log likelihood
- Autoregressive models (e.g. WaveNet, PixelRNN, van den Oord et al. 2016)
  - Model the data distribution given all previous data points

Background on Raw Audio Generation

- Raw Audio is a lot of data
    - At traditional 44.1 kHz, 16 bits/sample data rate – about 1 megabit per second
- Training is parallelizable
- Sampling is serial
- WaveNet trains on down sampled and compressed audio

Architecture

WaveNet - Overview

- Major architectural components:
  - Dilated – causal Convolutions
  - Gated activation units
  - Residual / skip connections

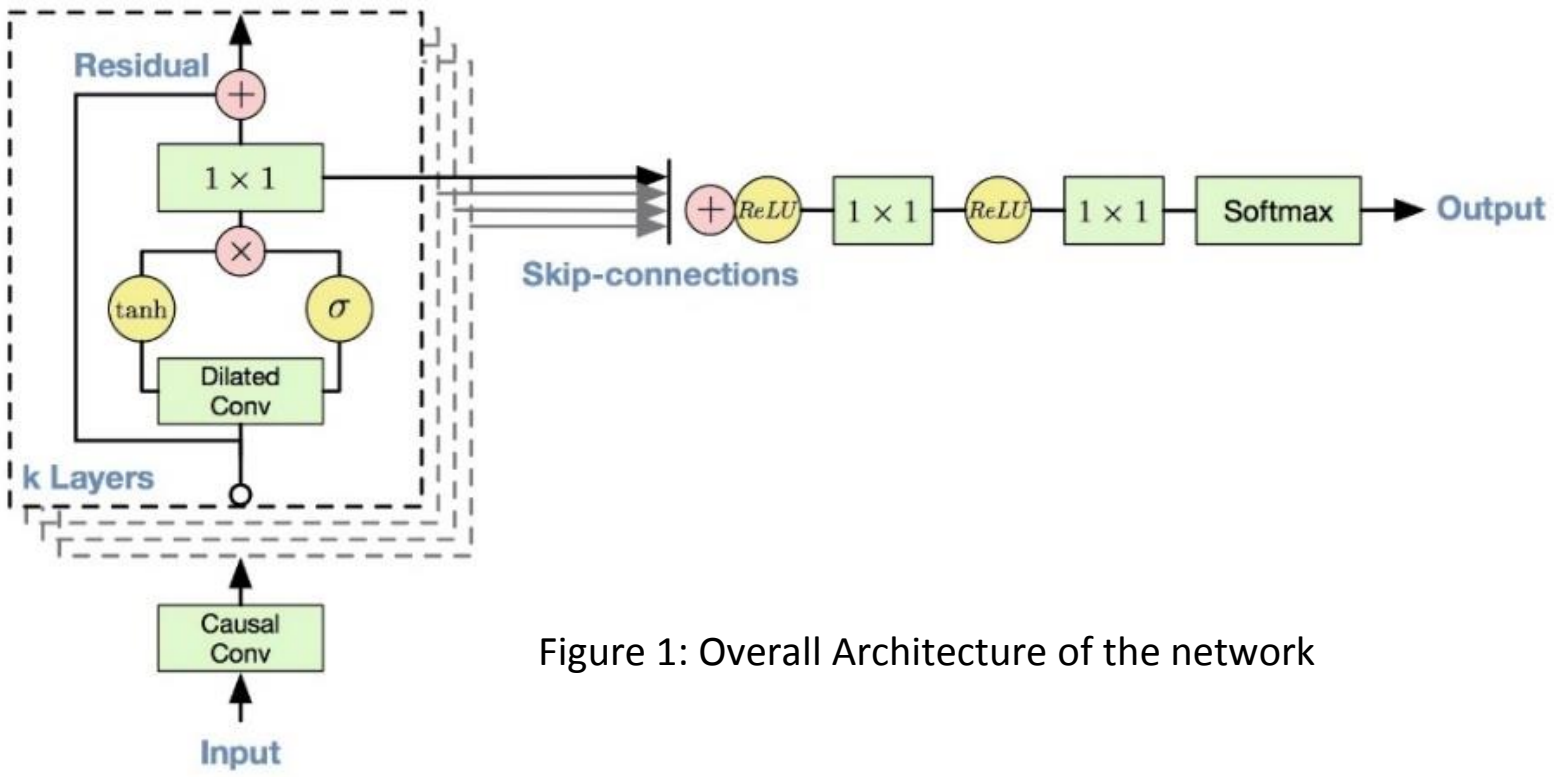


Figure 1: Overall Architecture of the network

But First, Compression!

- 16-bit audio samples would require 65,536 probabilities per sample from the softmax
- Enter mu-law companding transform:

$$f(x_t) = \text{sgn}(x_t) \frac{\ln(1 + \mu|x_t|)}{\ln(1 + \mu)}$$

- Compress to 8 bits using non-linear logarithmic compression (mu = 255)
- Better than a simple linear mapping in terms of preserving perceived quality

Dilated Causal Convolutions

- Causal: Output at t only depends upon values at 1 to t-1
  - Easier to train than RNNs
  - Require many layers
- Dilated:
  - Filter is applied to a field larger than its kernel length
  - Allows for a much higher temporal resolution

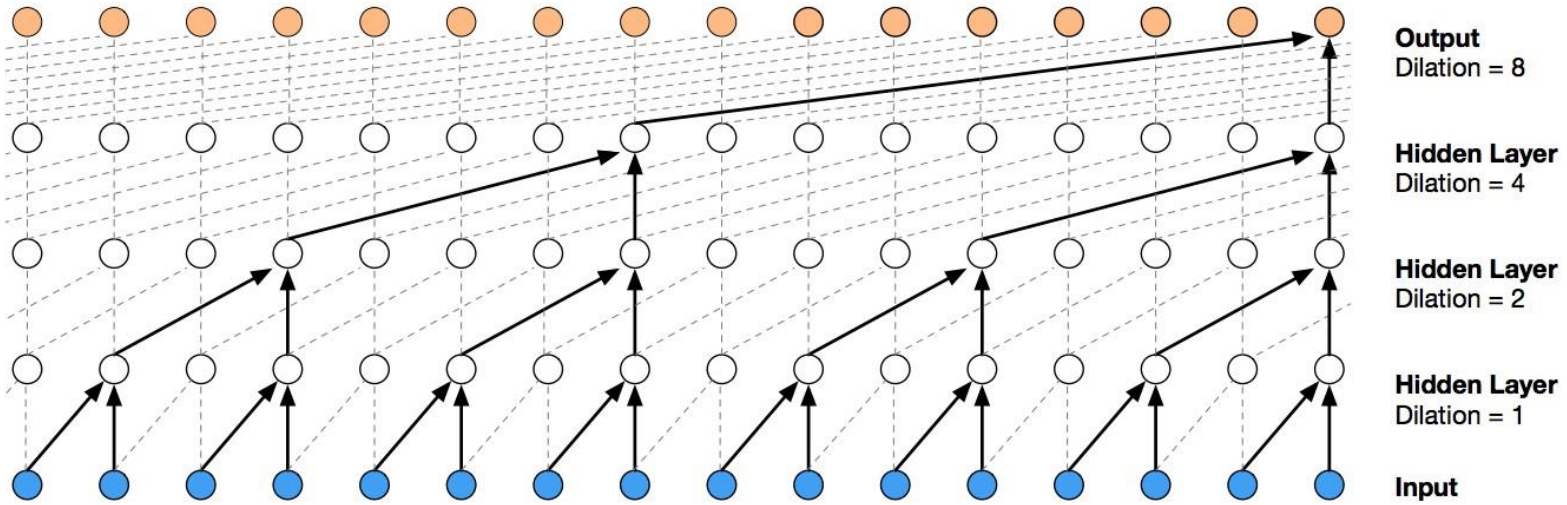


Figure 2: Visual representation of the dilated causal convolutional layers

Dilated Causal Convolutions

- WaveNet uses dilations increasing by a factor of two until a certain limit, and then repeats them:
  - $1, 2, 4, \ldots, 512, 1, 2, 4, \ldots, 512$
- Each 1,2,4,…,512 block has a receptive filed of 1024 samples
- Stacking them exponentially increase temporal resolution
  -

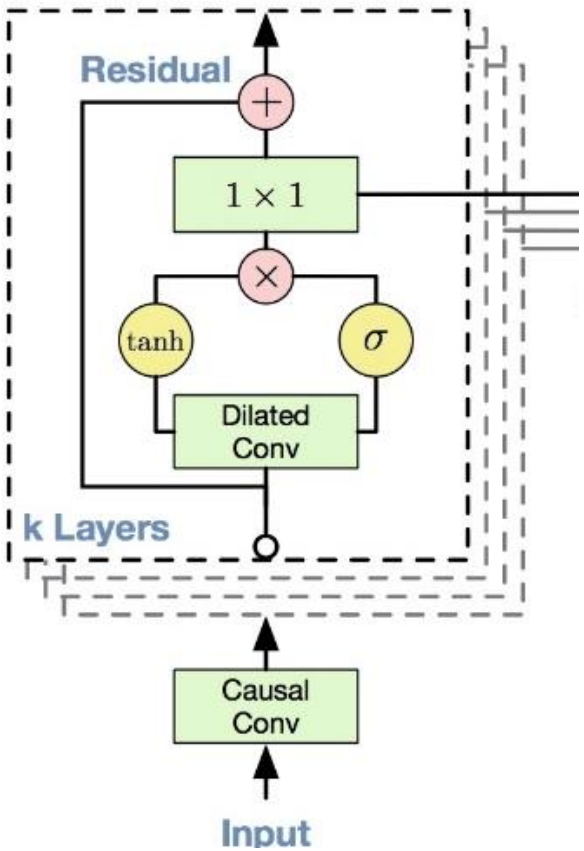


Figure 3: Dilated causal convolutions in the network

Gated Activation Units

- Similar to ones used in PixelRNN[2]

$$z = \tanh(W_{f,k} * x) \odot \sigma(W_{g,k} * x)$$

- layer index
- filter index
- gate index
- learned convolution filter

- Empirically shown to outperform ReLUs

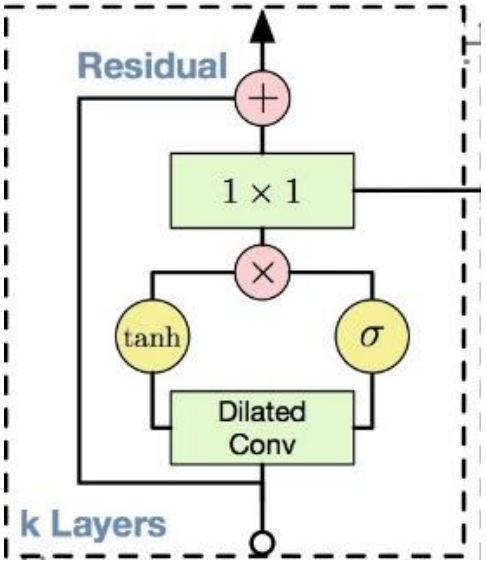


Figure 4: Gated activation units in WaveNet

Residual and Skip Connections

- Makes output of the k^{th} layer a linear combination of the output of $k\text{-}1^{th}$ later and kth layer's gated activation unit
- Makes training of deeper networks more tractable by preventing vanishing gradients
- Using this with skip connections empirically shown to have better performance[2]

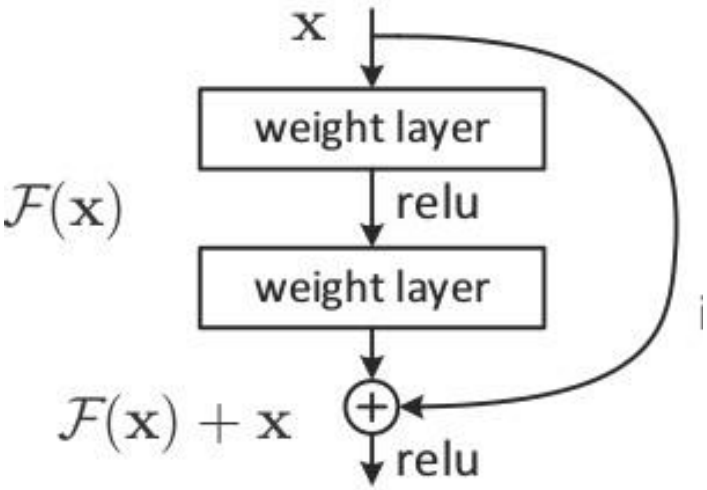
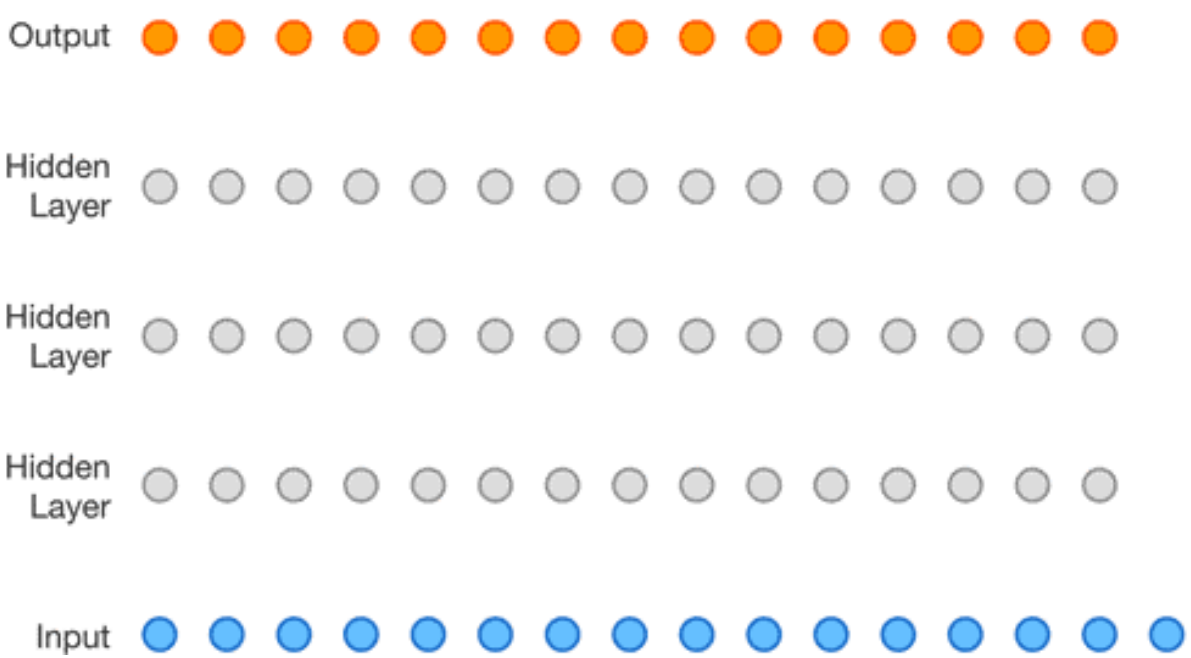


Figure 5: Residual and skip connections in the network

Sampling a Trained WaveNet and Demos



Conditioning with WaveNet

- Given other inputs, the conditional distribution can be modeled

$$p(x|h) = \prod_{t=1}^{T} p(x_t|x_1, ..., x_{t-1}, h)$$

- Useful for generating audio with unique identities
  - Speaker identity
  - Text as extra input for TTS
  - Genres for Music

Global Conditioning

- Condition on a single latent variable that influences output at all time stamps
- So the activation function becomes:

$$z = \tanh(W_{f,k} * x + V_{f,k}^T h) \odot \sigma(W_{g,k} * x + V_{g,k}^T h)$$

- Where $V_{\cdot,k}$ is a learnable linear projection

Local Conditioning

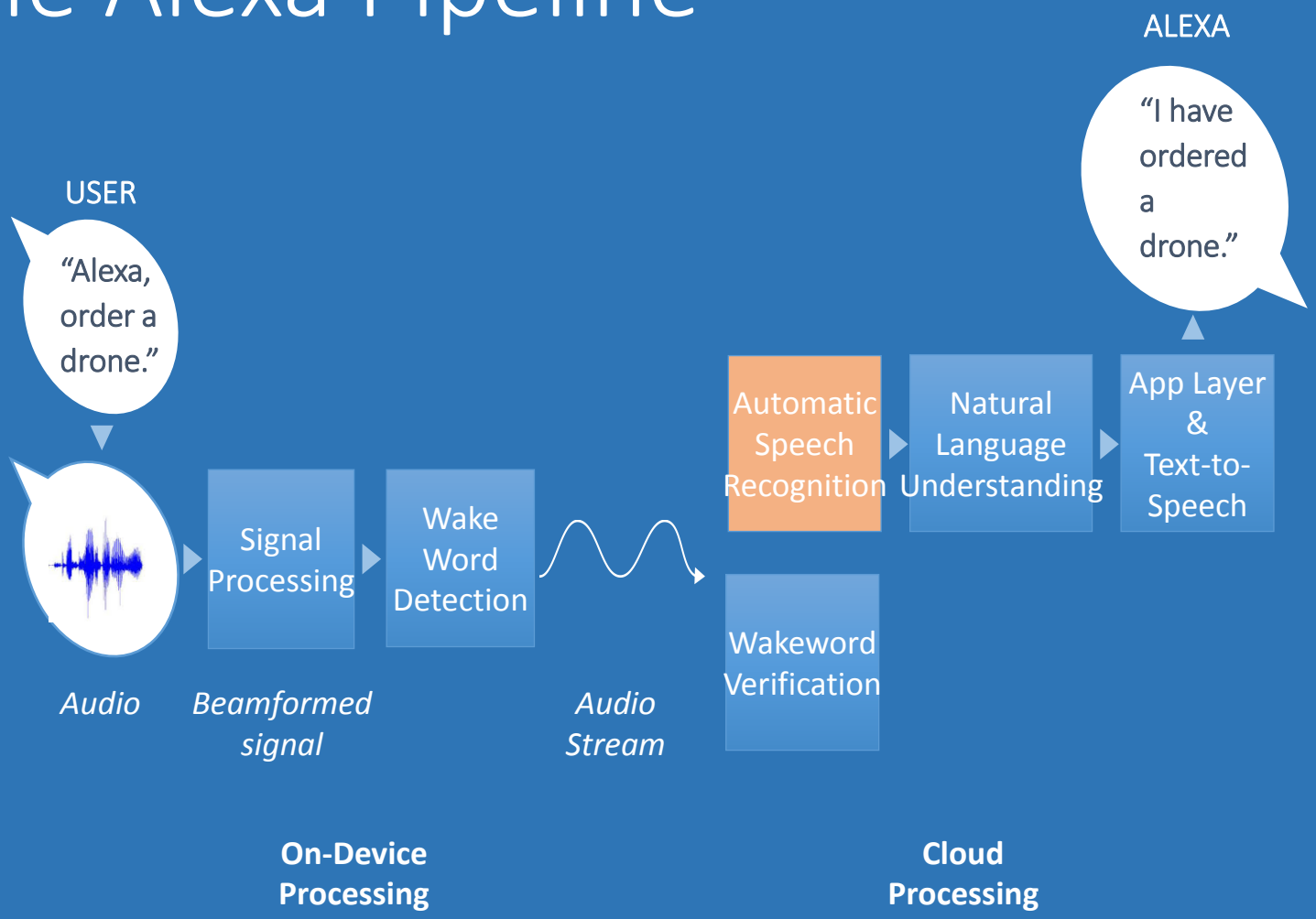
- Condition on a second time series h_t
- Can have a lower sampling rate than x_t
- First transform time series using a transposed convolution network and map to $y = f(h)$
- The activation function now becomes
- $$z = \tanh(W_{f,k} * x + V_{f,k} * y) \odot \sigma(W_{g,k} * x + V_{g,k} * y)$$

References

- WaveNet: A Generative Model for Raw Audio
  - arXiv:1609.03499 [cs.SD]

- Pixel Recurrent Neural Networks
  - arXiv:1601.06759 [cs.CV]

The Alexa Pipeline

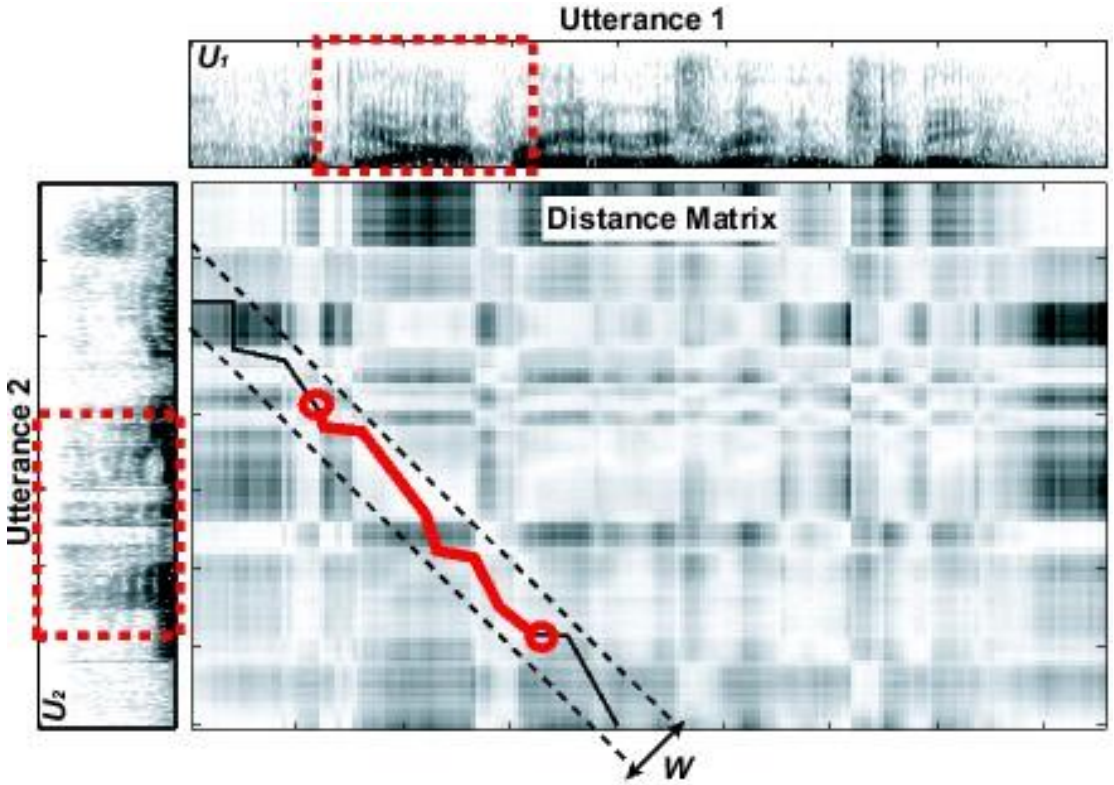


Outline

- History of Automatic Speech Recognition
- Hidden Markov Model (HMM) based ASR
  - Gaussian Mixtures with HMMs
  - Deep Models with HMMs
- End-to-end Deep Models for ASR
  - Connectionist Temporal Classification (CTC)
  - Attention-based Models

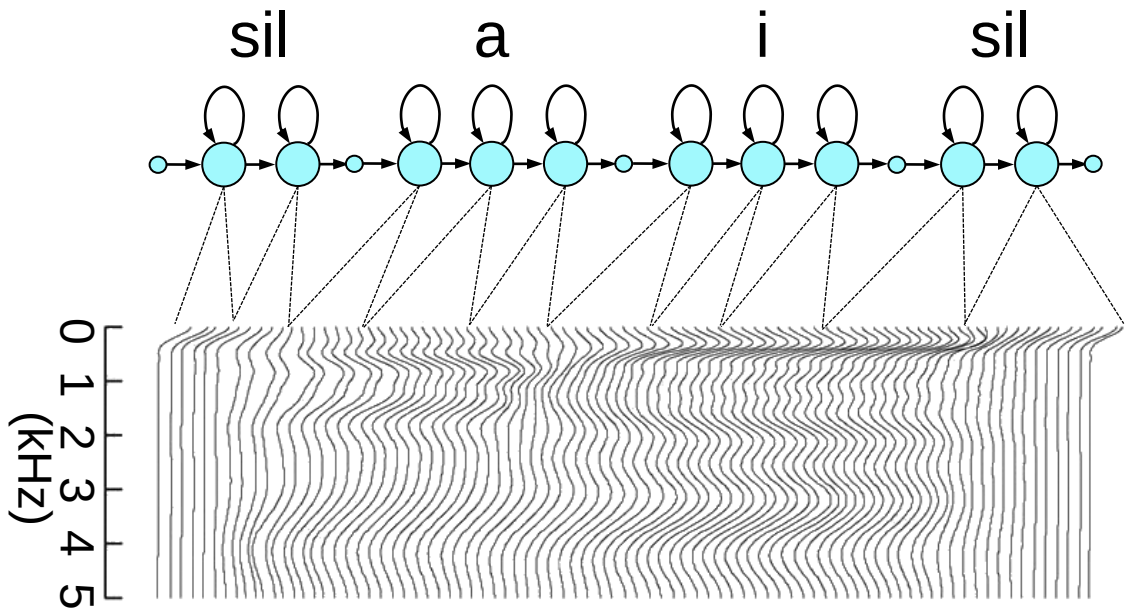
History of ASR

- Early 1970s: Dynamic Time Warping (DTW) to handle time variability
  - Distance measure for spectral variability



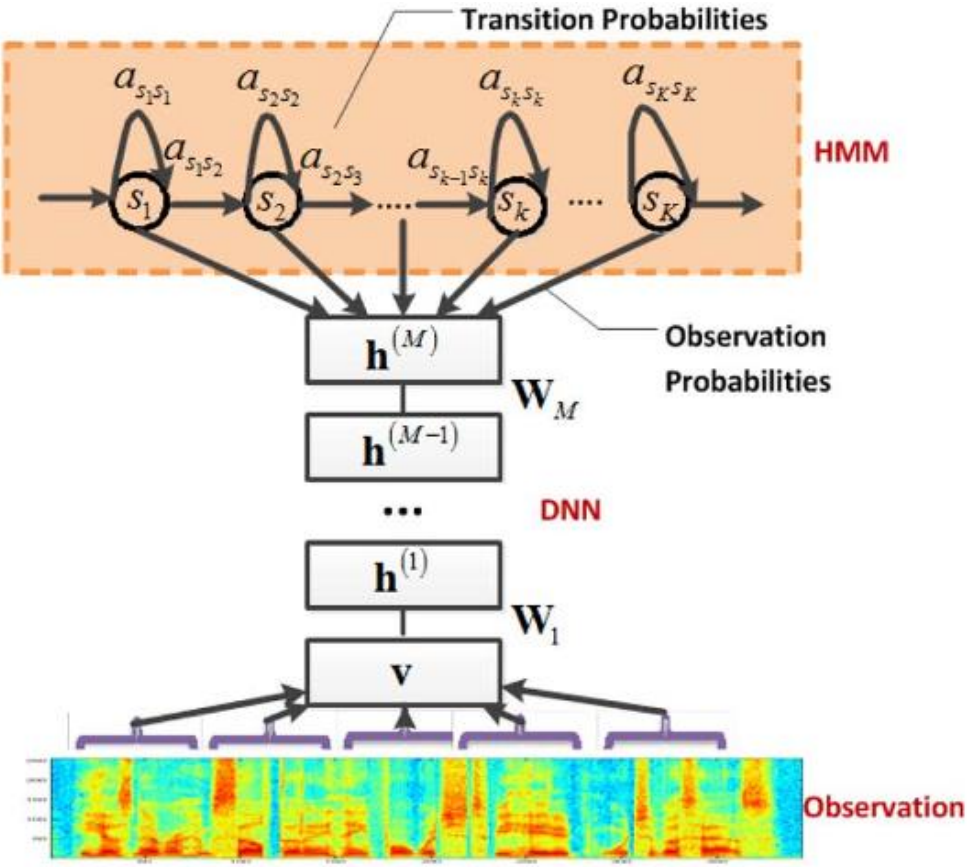
History of ASR

- Mid-Late 1970s: Hidden Markov Models (HMMs) – statistical models of spectral variations, for discrete speech.
- Mid 1980s: HMMs become the dominant technique for all ASR



History of ASR

- 1990s: Large vocabulary continuous dictation
- 2000s: Discriminative training (minimize word/phone error rate)
- 2010s: Deep learning significantly reduce error rate



Goal of ASR

- Find the most likely word sequence *W* which transcribes the speech audio *A*

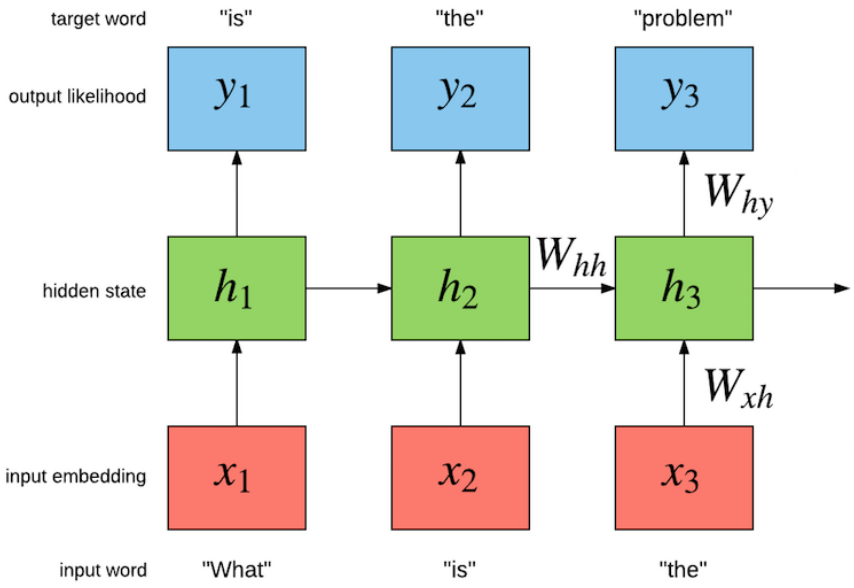
$$\text{argmax}_W p(W|A) = \text{argmax}_W p(A|W)p(W)$$

- Acoustic model: *p(A|W)*
- Language model: *p(W)*
- Training: find parameters of acoustic and language models separately
  - Speech corpus: speech waveform and human-annotated transcriptions
  - Language model: text or other annotated speech data

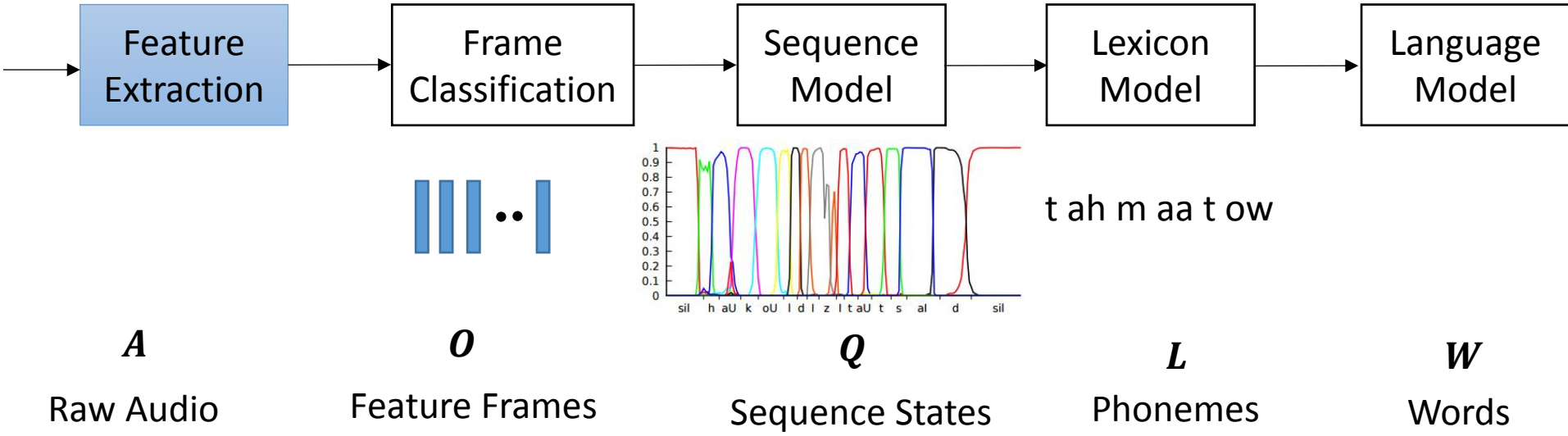
Language Model

- Guide the search algorithm (predict next word given previous words)
- Disambiguate between phrases that are acoustically similar (e.g., great wine vs grey twine)
- Assign probability to a sequence of tokens
- N-gram model:

$$p(w_n|w_1, \ldots, w_{n-1})$$

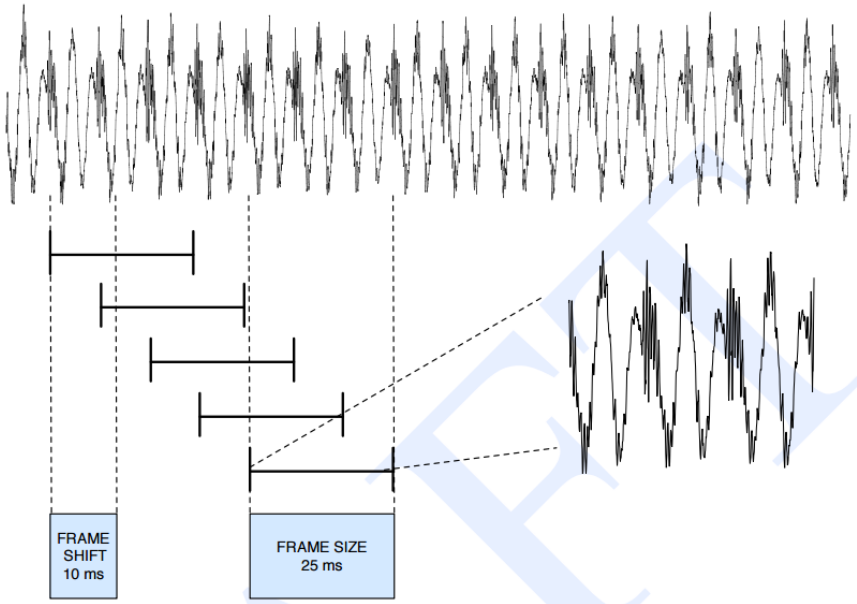


Architecture of Speech Recognition

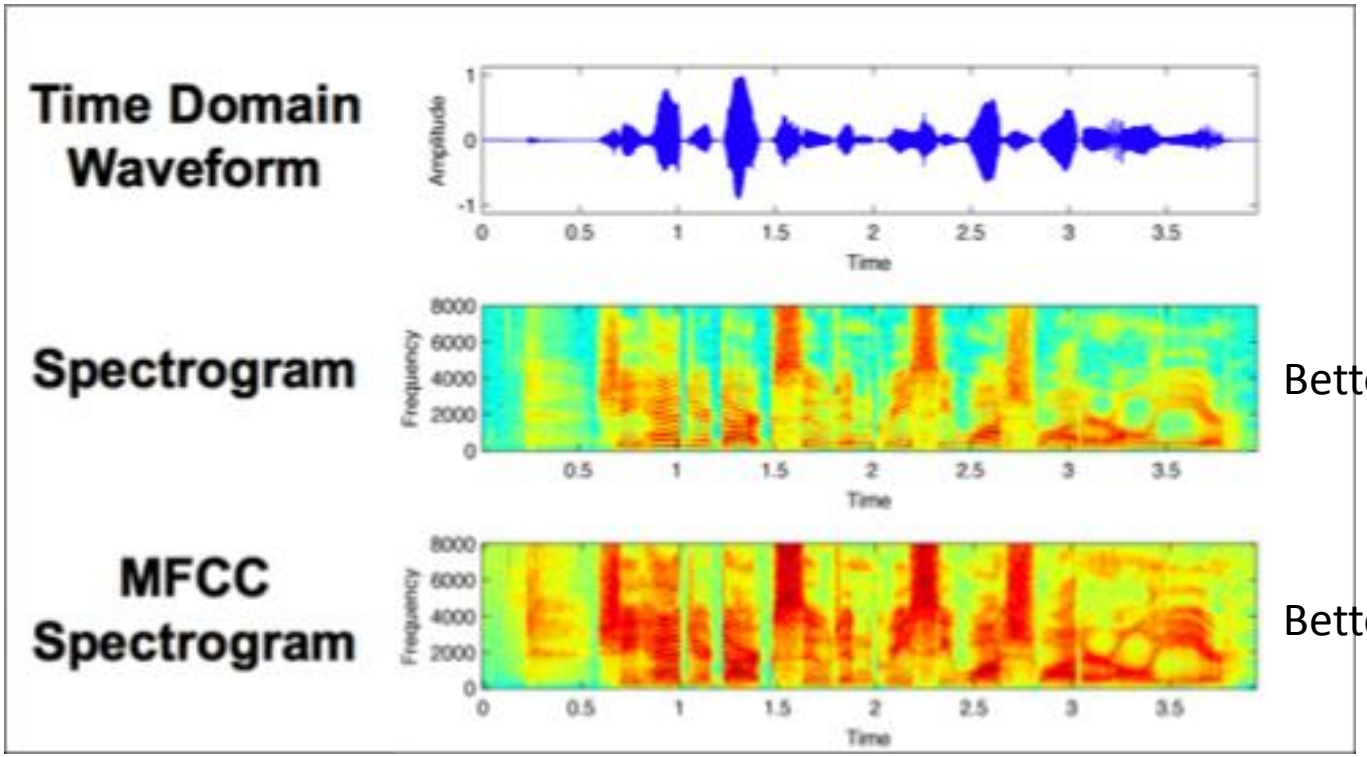


Feature Extraction

- Raw waveforms are transformed into a sequence of feature vectors using signal processing techniques
  - Time domain to frequency domain
  - Feature extraction is deterministic
  - Remove noise and other irrelevant information
- Extracted in 25ms windows with 10ms overlap
- Still useful for deep models



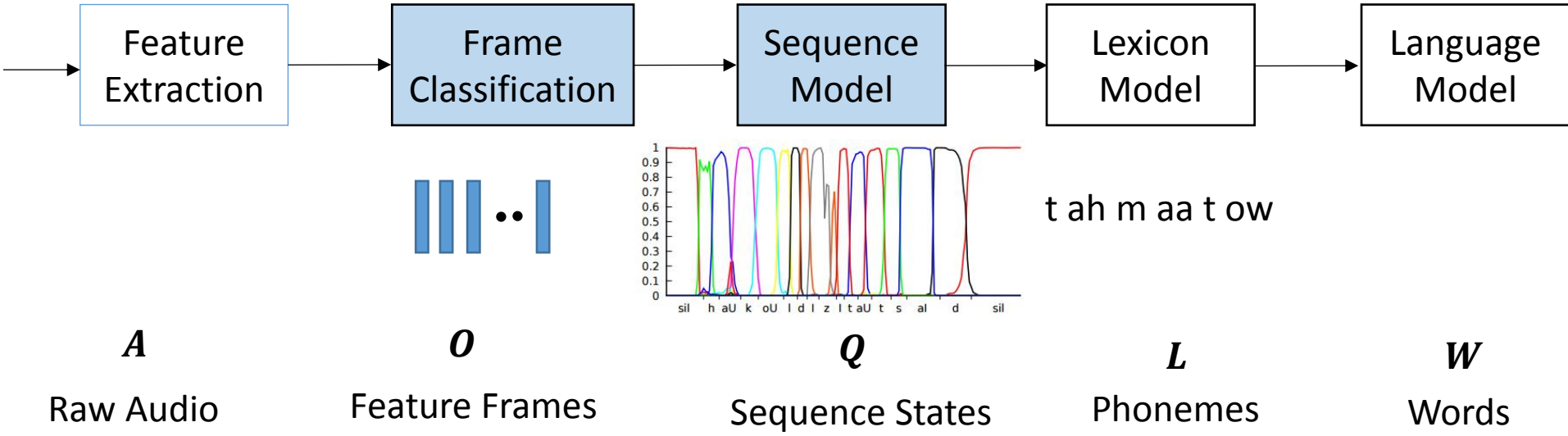
Feature Extraction



Better for deep models

Better for shallow models

Architecture of Speech Recognition



Frame Classification

- Determine the probability that a frame (~25ms overlapping window) belongs to a particular subword unit
- Easiest to think of phoneme categories
  - “Right” = /r/ + /ai/ + /t/
  - Around 40 phonemes in English
- Reality is often a bit more complex
  - The same phoneme may sound different in different contexts
  - Consider /n/ in “ten” vs “tenth”
  - Or /t/ in “tube” vs “suit”
  - Can expand to biphones or triphones
  - Also, in HMMs, there will be 3 states per phone/biphone/triphone

Gaussian Mixture Model

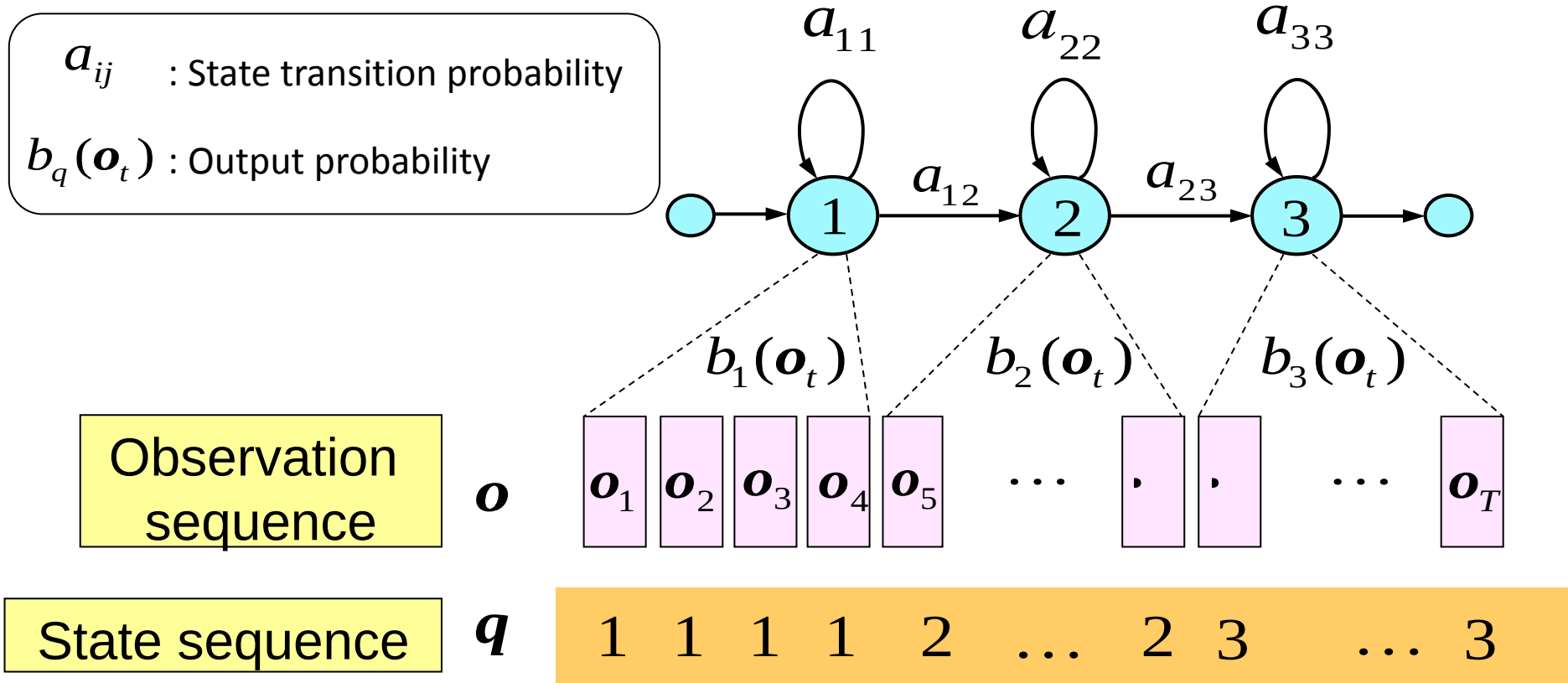
- Use a Gaussian mixture model to determine, for each feature frame, the probability that the frame is a particular phoneme

Neural Net Frame Classification

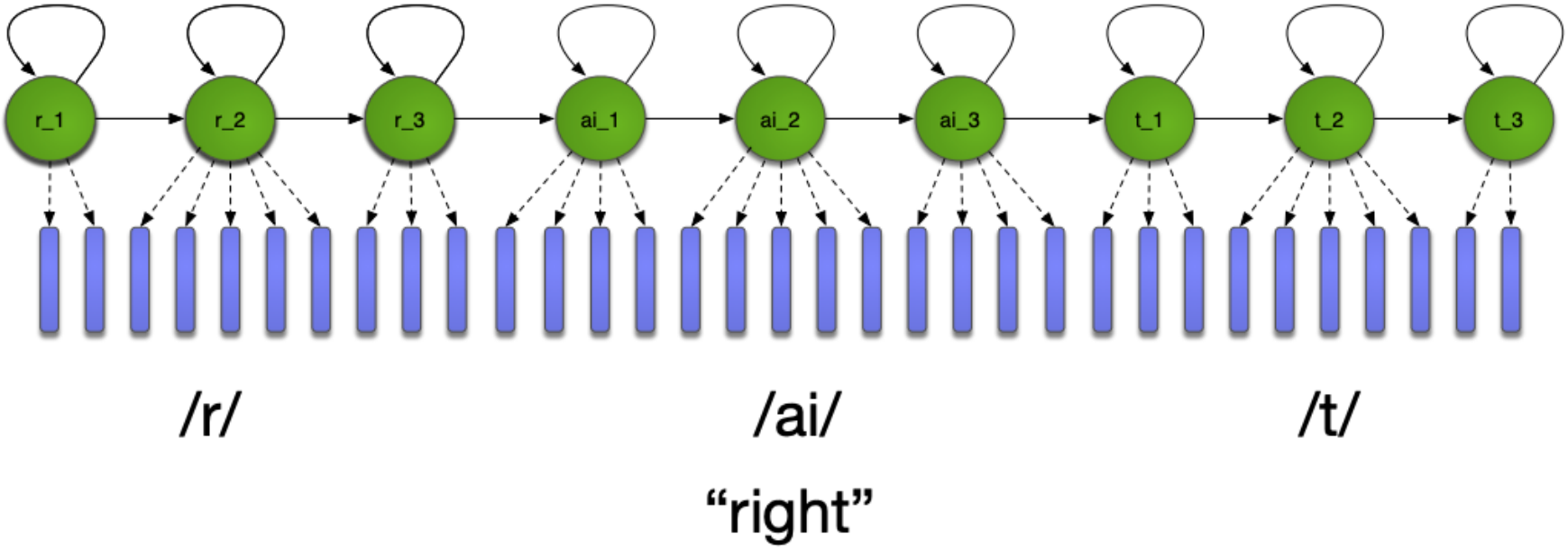
- DNN-HMM models repace the GMM with a neural network
- Input is feature
- Output is softmax over phonemes

Hidden Markov Models

- The Markov chain whose state sequence is unknown



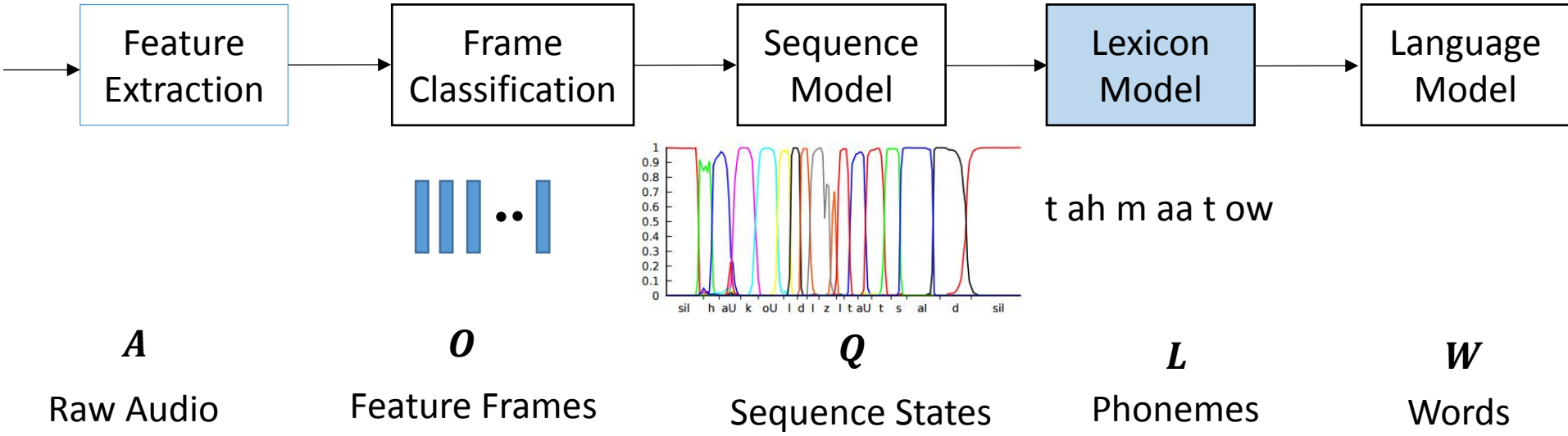
HMM Example



Training a GMM-HMM or DNN-HMM

- LOTS of details here that we will not cover!
- Algorithms for training use dynamic programming to estimate the transition probabilities, train the GMM/DNN, and perform decoding (find the most likely sequence of states given frame inputs)

Architecture of Speech Recognition



Lexical Model

- Bridges the gap between acoustic and language models
- Mapping between words and acoustic units (e.g., phonemes)

Deterministic

Word	Pronunciation
TOMATO	t ah m aa t ow
	t ah m ey t ow
COVERAGE	k ah v er ah jh
	k ah v r ah jh

Probabilistic

Word	Pronunciation	Probability
TOMATO	t ah m aa t ow	0.45
	t ah m ey t ow	0.55
COVERAGE	k ah v er ah jh	0.65
	k ah v r ah jh	0.35

Comparison of GMM and DNN

- Word error rates measured

TASK	HOURS OF TRAINING DATA	DNN-HMM	GMM-HMM WITH SAME DATA	GMM-HMM WITH MORE DATA
SWITCHBOARD (TEST SET 1)	309	18.5	27.4	18.6 (2,000 H)
SWITCHBOARD (TEST SET 2)	309	16.1	23.6	17.1 (2,000 H)
ENGLISH BROADCAST NEWS	50	17.5	18.8	
BING VOICE SEARCH (SENTENCE ERROR RATES)	24	30.4	36.2	
GOOGLE VOICE INPUT	5,870	12.3		16.0 (>> 5,870 H)
YOUTUBE	1,400	47.6	52.3	

G. Hinton, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition. Signal Processing Magazine (2012).

DNN vs GMM

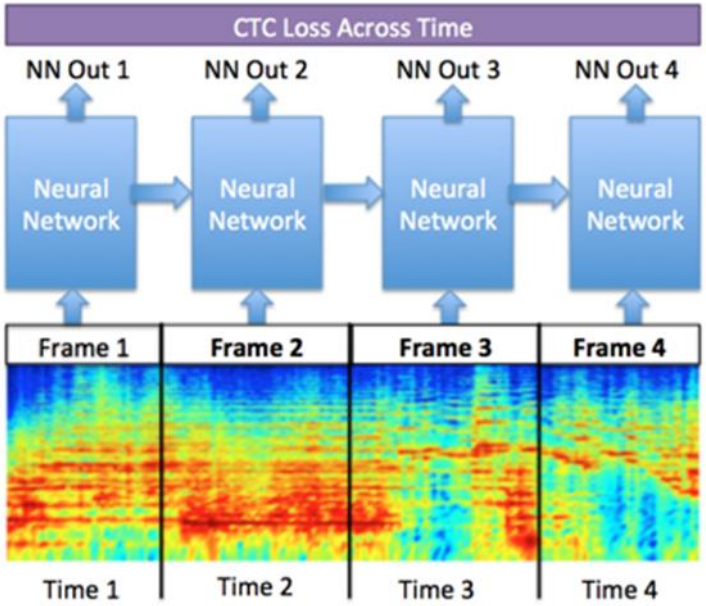
- Deep models are more powerful
  - GMM assumes data is generated from single component of mixture model
  - GMM with diagonal variance matrix ignores correlation between dimensions
- Deep models take data more efficiently
  - GMM consists with many components and each learns from a small fraction of data
- Deep models can be further improved by recent advances in deep learning

End-to-End Deep Models

- Connectionist Temporal Classification (CTC) based models
  - LSTM CTC models
  - Deep speech 2
- Attention based models
  - Transformer Models (focus of ASR research 2018-2020)

Connectionist Temporal Classification (CTC)

- Method for labeling unsegmented data sequences
  - Raw waveforms and text transcription
- Differentiable objective function
  - Gradient based training
- Used in various ASR and TTS architectures
  - DeepSpeech (ASR)
  - DeepVoice (TTS)



Deep Speech

Model	SWB	CH	Full
Vesely et al. (GMM-HMM BMMI) [44]	18.6	33.0	25.8
Vesely et al. (DNN-HMM sMBR) [44]	12.6	24.1	18.4
Maas et al. (DNN-HMM SWB) [28]	14.6	26.3	20.5
Maas et al. (DNN-HMM FSH) [28]	16.0	23.7	19.9
Seide et al. (CD-DNN) [39]	16.1	n/a	n/a
Kingsbury et al. (DNN-HMM sMBR HF) [22]	13.3	n/a	n/a
Sainath et al. (CNN-HMM) [36]	11.5	n/a	n/a
Soltau et al. (MLP/CNN+I-Vector) [40]	**10.4**	n/a	n/a
Deep Speech SWB	20.0	31.8	25.9
Deep Speech SWB + FSH	12.6	**19.3**	**16.0**

Table 3: Published error rates (%WER) on Switchboard dataset splits. The columns labeled "SWB" and "CH" are respectively the easy and hard subsets of Hub5'00.

Deep Speech Training

- Maps from acoustic frames *X* to subword sequences *S*, where *S* is a sequence of characters (in some CTC approaches, can be a sequence of phones)
- Uses the CTC loss function
- Makes good use of large training data
  - Synthetic additional training data by jittering the signal and adding noise
- Many computational optimizations
- N-gram language model used to impose word-level constraints
- Competitive results on standard tasks

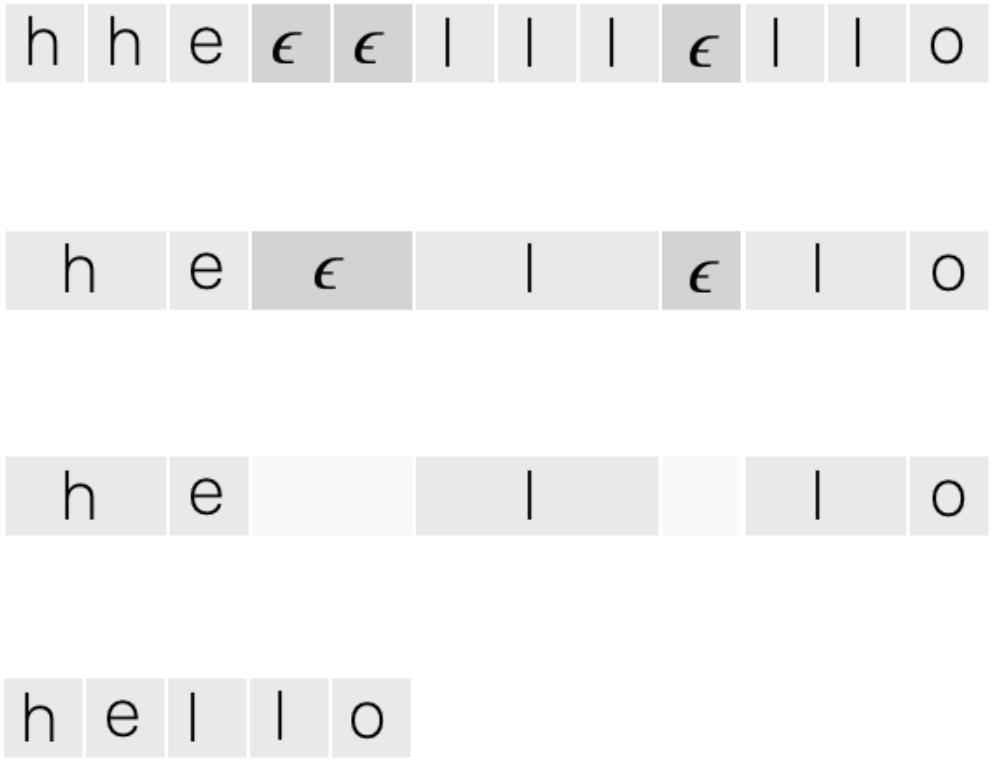
Connectionist Temporal Classification (CTC)

- Train a recurrent network to map from input sequences *X* to output sequences *S*
  - Sequences can be different lengths—for speech, input sequence *X* (acoustic frames) is much longer than output sequence *S* (characters or phonemes)
- CTC sums over all possible alignments (similar to forward-backward algorithm)
- Possible to back-propagate gradients through the CTC loss function

CTC: Alignment

- Imagine mapping *(x1, x2, x3, x4, x5, x6)* to *[a, b, c]*
  - Possible alignments: *aaabbc, aabbcc, abbbc, …*
- However
  - Don't always want to map every input frame to an output symbol (e.g., if there is silence between symbols)
  - Want to be able to have two identical symbols adjacent to each other
- Solve this using an additional blank symbol ε
- CTC output compression
  - Merge repeating characters
  - Remove blanks
- Some possible alignments for *[h,e,l,l,o]* and *[h,e,l,o]* given a 10-element input sequence
  - *[h,e,l,l,o]: hεεeεllεlo; heεllεlεoo*
  - *[h,e,l,o]: hεεeεllllo; hhεeεlεεoε*

Alignment Example



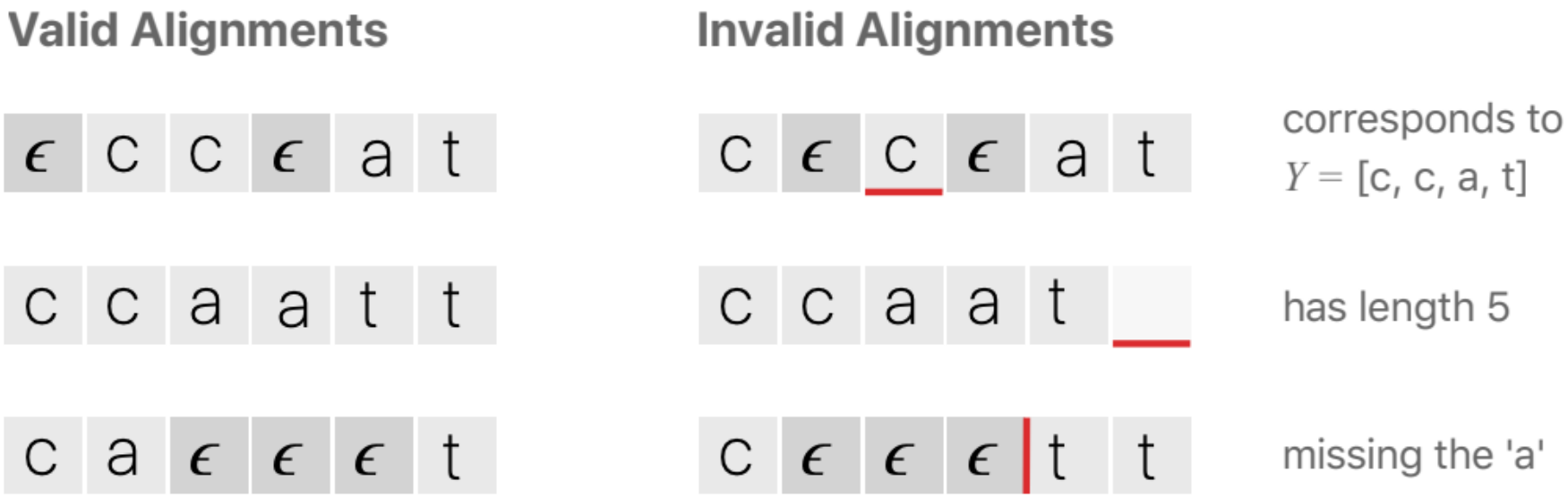
First, merge repeat characters.

Then, remove any ϵ tokens.

The remaining characters are the output.

Valid and Invalid Alignments

- Consider an output *[c,a,t]* with an input length of six



CTC Alignment Properties

- Monotonic—alignments are monotonic (left-to-right model); no re-ordering (unlike neural machine translation)
- Many-to-one—alignments are many-to-one; many inputs can map to the same output
- But a single input cannot map to many outputs
- CTC doesn’t find a single alignment: it sums over all possible alignments

CTC Loss Function

- Let *C* be an output label sequence, including blanks and repetitions—same length as input sequence *X*
- Posterior prob of output labels $C = (c_1, ..., c_t, ..., C_T)$ given input sequence $X = (x_1, ..., x_t, ..., x_T)$

$$p(C|X) = \prod_{t=1}^{T} p_t(c_t|X)$$

- This is the probability of a single alignment—we need to sum over all alignments consistent with *S*

CTC Loss Function

- Let *S* be the compressed target output sequence
- Compute the posterior probability of the target sequence

$$S = (s_1, ..., s_m, ..., s_M)(M \leq T)$$

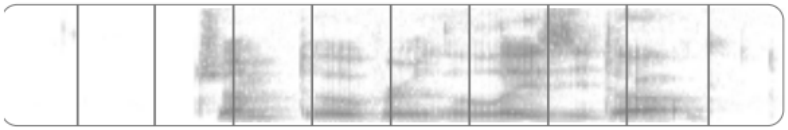
given *X* by summing over the possible CTC alignments

$$p(S|X) = \sum_{c \in A(S)} p(C|X)$$

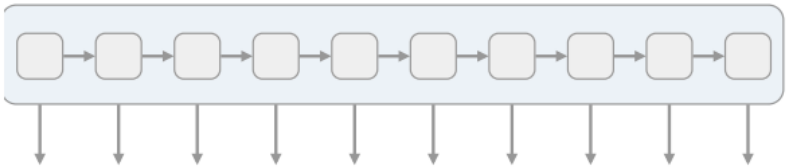
- *A* is the set of possible output labels sequences *c* that can be mapped to *S* using the CTC compression rules (merge repeated labels, then remove blanks)
- The CTC loss function is given by the negative log likelihood of the sum of CTC alignments

$$L_{CTC} = -\log p(S|X)$$

- Various NN architectures can be used for CTC—usually use a deep bidirectional LSTM RNN



We start with an input sequence, like a spectrogram of audio.



The input is fed into an RNN, for example.

h	h	h	h	h	h	h	h	h	h
e	e	e	e	e	e	e	e	e	e
l	l	l	l	l	l	l	l	l	l
o	o	o	o	o	o	o	o	o	o
ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ

The network gives $p_t(a \mid X)$, a distribution over the outputs {h, e, l, o, ϵ} for each input step.

h	e	ϵ	l	l	ϵ	l	l	o	o
h	h	e	l	l	ϵ	ϵ	l	ϵ	o
ϵ	e	ϵ	l	l	ϵ	ϵ	l	o	o

With the per time-step output distribution, we compute the probability of different sequences

h	e	l	l	o
e	l	l	o	
h	e	l	o	

By marginalizing over alignments, we get a distribution over outputs.

Dynamic Programming

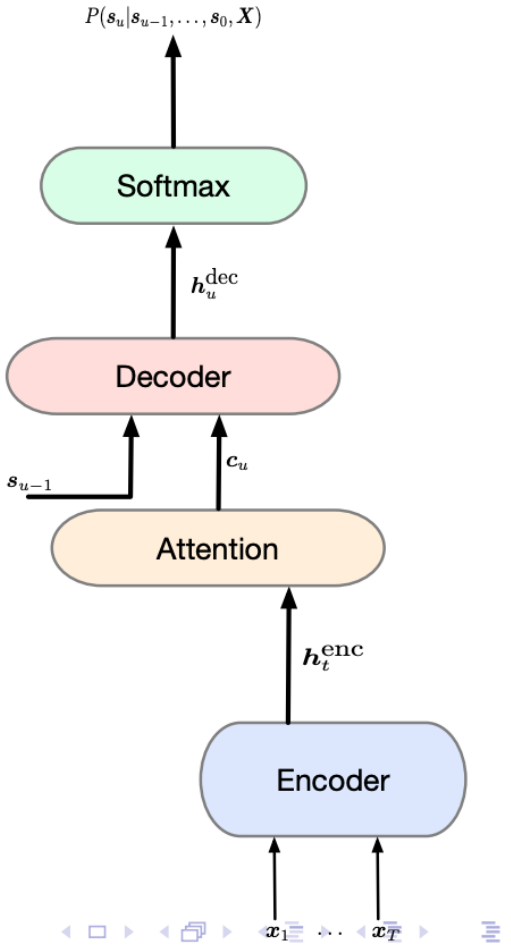
- Computing the loss involves performing a sum over alignments *A(S)*
  - Use dynamic programming via forward algorithm
- Also, performing decoding involves solving

$$S^* = \text{argmax}_S p(S|X)$$

- Uses beam search
- Won't cover these
- Additionally, need to incorporate the language model (also won't cover here)

Encoder-Decoder Models

- Encoder: acoustic model using a recurrent network to map acoustic features to hidden vectors
- Decoder: Computes distributions over labels conditioned on previously predicted labels and acoustics
- Attention: Constructs a context vector for the decoder network based on attention weights computed over all frames in the encoder output
- Google's "Listen, Attend, and Spell" model from ICASSP 2016



Recent ASR Research

- Much of the recent work (2018-2020) has focused on transformer-based models
- Still very much an active area of research!