(Deep) Neural Networks for Speech Processing

Kate Knill

September 2015

Cambridge University Engineering Department

DREAMS Summer School Tutorial 2015

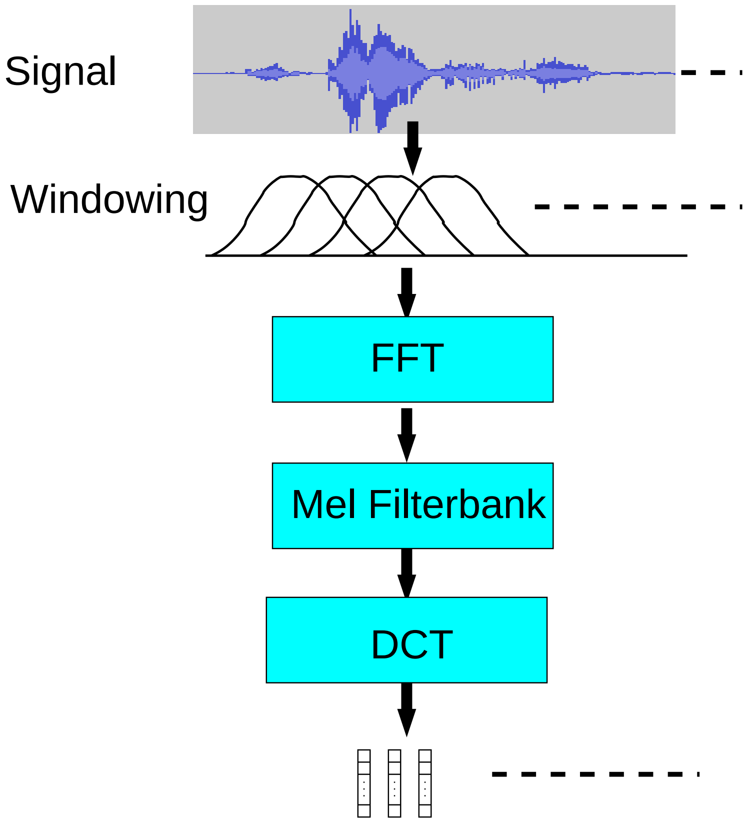
Overview

- Part 1:
  - Motivation
  - Basics of Neural Networks
  - Voice Activity Detection
  - Automatic Speech Recognition
- Part 2:
  - Neural Networks for ASR Features and Acoustic Models
  - Neural Networks for Language Modelling
  - Other Neural Network Architectures

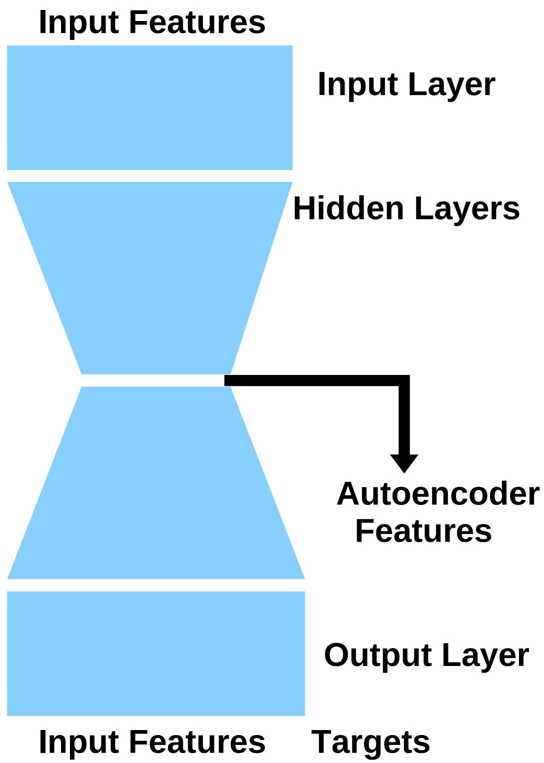
Neural Networks for ASR

Neural Network Feature Extraction

- Standard MFCC generation



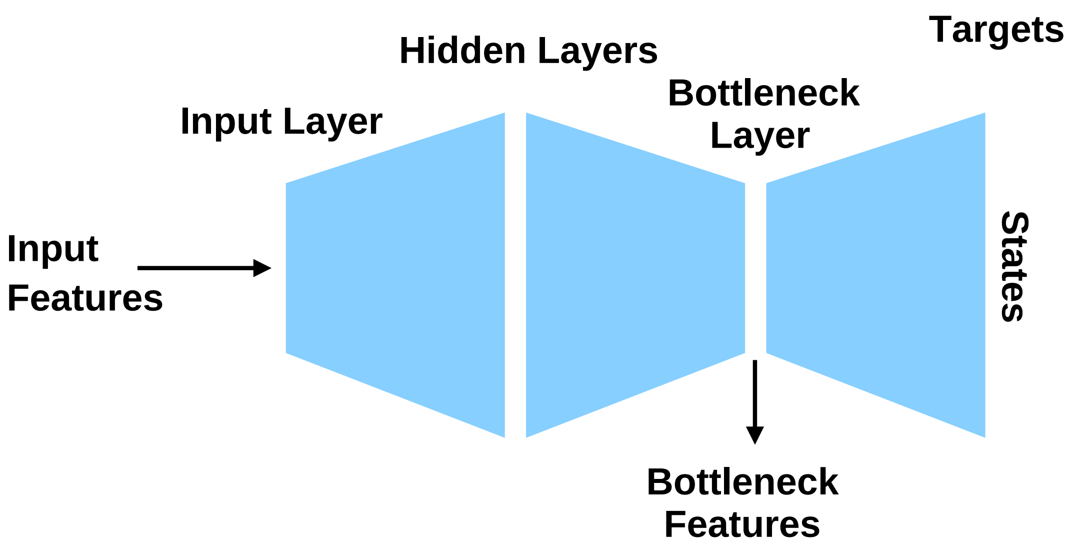
- Auto-encoder [1]



  – Low dimensional representation
  – Used e.g. noisy speech [2, 3]

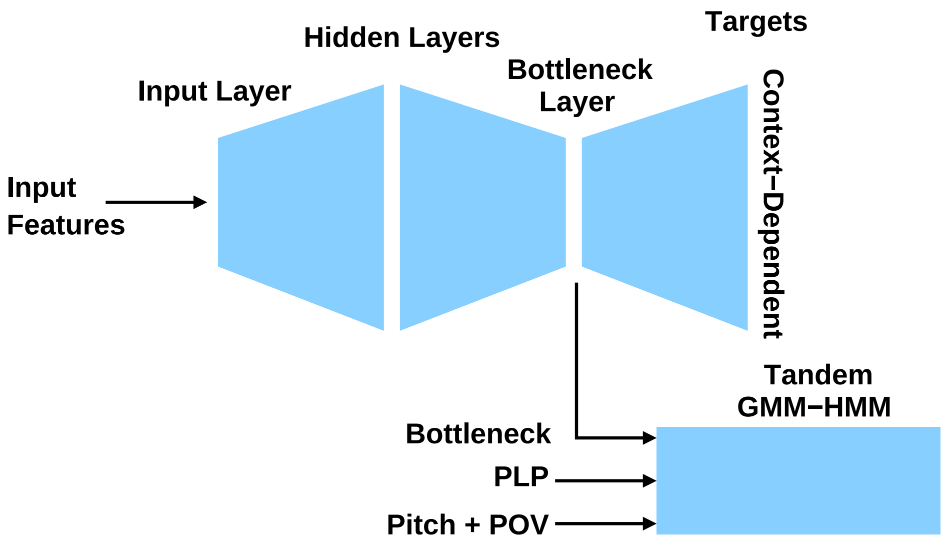
Neural Network Bottleneck Features

- Use discriminative projection capabilities of NN to extract features
- Hermansky etal in 2000 [4] proposed NN-based probabilistic features
  - NN posterior outputs converted to features
  - Log and PCA de-correlation with (possibly) dimensionality reduction
- Bottleneck features proposed by Grezl etal in 2007 [5]
  - Exploit NN's ability to non-linearly compress and classify input features
  - Need minimum 3 hidden layers



Tandem Systems

- Use NN features as input features for GMM-HMM system
  - "two models in tandem" [4]
  - bottleneck alone - doesn't match standard feature performance
  - in combination with standard input features - very competitive e.g. [6, 7]



- Number of investigations into optimal configuration e.g. [8, 9]
- Can learn speaker and environment transforms as per standard features

Hybrid Systems (1)

- Use neural networks to extract features still uses GMM-HMM
  - standard training approaches are used
  - BUT are GMMs the appropriate state output distribution

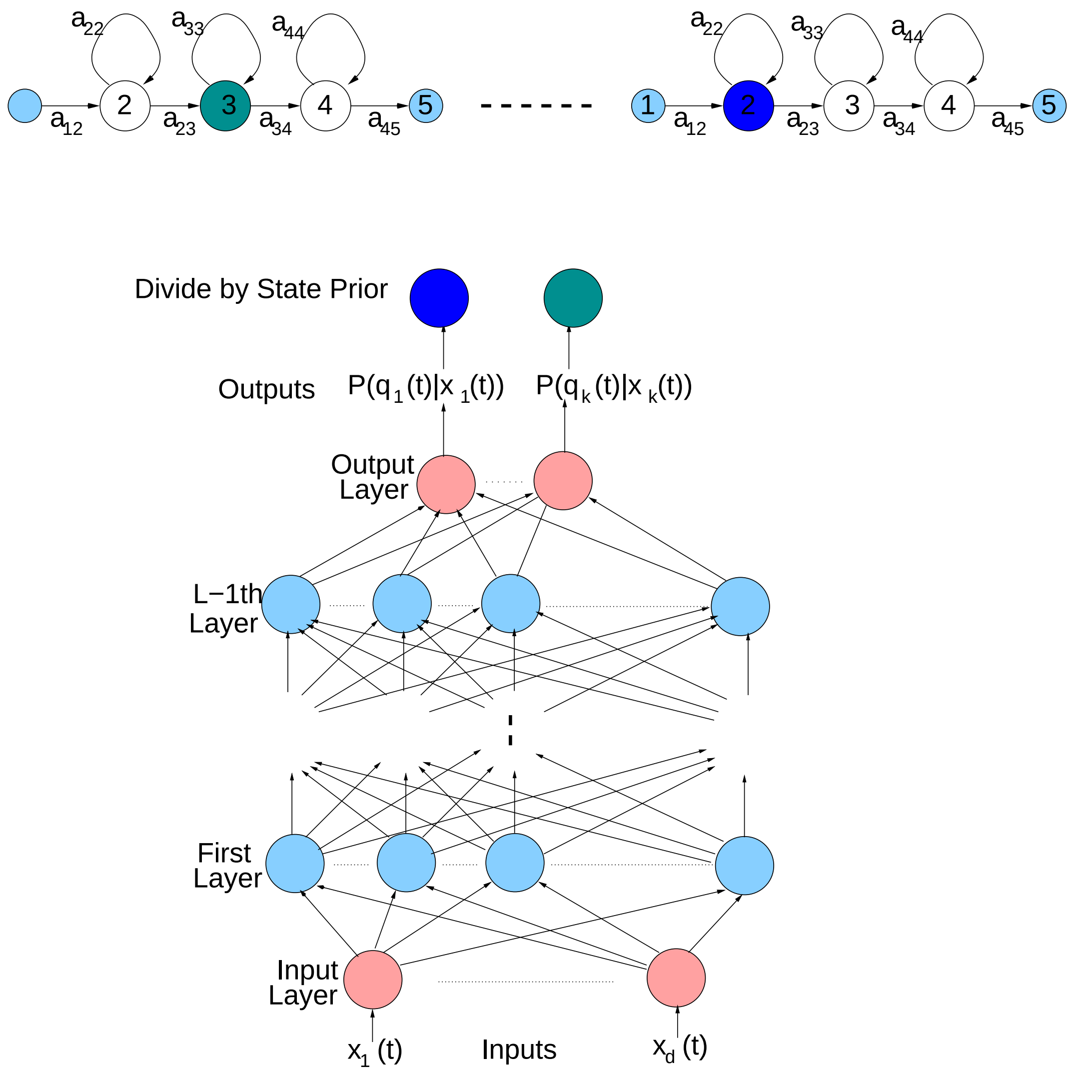
- Standard NN theory states that the outputs can be trained to yield posteriors
  - transform these posteriors to "likelihoods" as the state output distribution

$$p(\boldsymbol{x}_t|q_t) = \frac{P(q_t|\boldsymbol{x}_t)p(\boldsymbol{x}_t)}{P(q_t)} \propto \frac{P(q_t|\boldsymbol{x}_t)}{P(q_t)}$$

  - $P(q_t|\boldsymbol{x}_t)$ is the output from the network
  - $P(q_t)$ is the state prior

- The underlying model is still an HMM
  - replace output state distribution by a neural network - Hybrid system

Hybrid Systems (2)



From Neural Nets to DNNs (via MLPs!)

- Hybrid systems have been around since the late 80s

(Mainly) Ignored since the late 90s - What's Changed?

- Three important developments since the 80s

  1. vast quantities of data now available for ASR training
  2. dramatic shift in compute resources and GPU availability
  3. improvements in initialisation (and optimisation approaches)

  vast interest from researcher/industry driven by results

- These enabled important changes to the neural network topology (MLP)

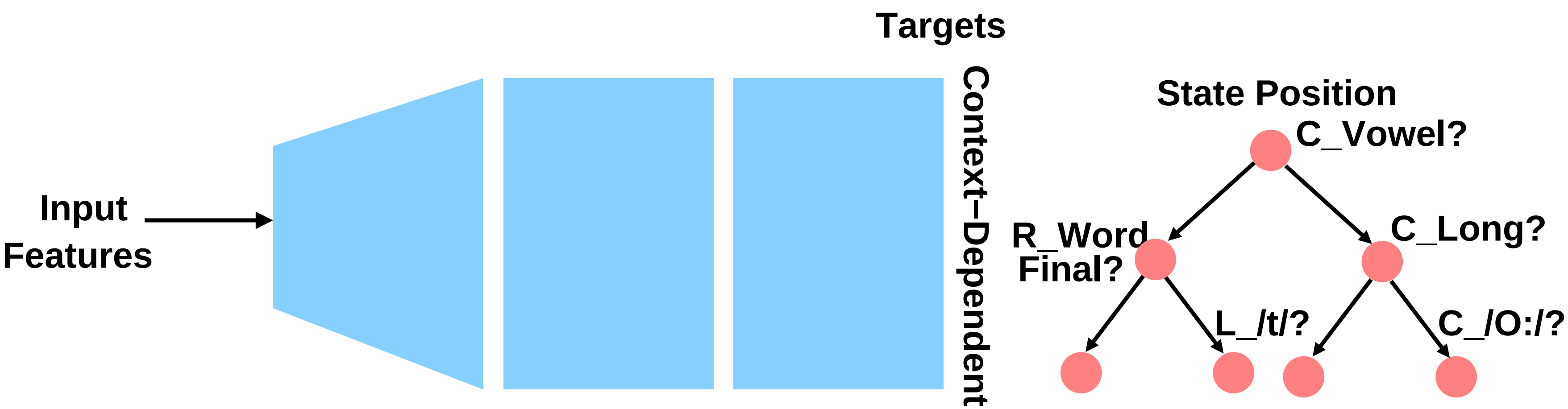
  - many hidden layers (data/initialisation/compute)
  - context-dependent state targets (data/compute)

Deep Neural Networks

- Some great advantages in this Hybrid architecture
  - very powerful general non-linear mapping (models anything)
  - able to handle wide input window of frames (frame-indpendence)
  - very flexible architecture for including input information
- Need to consider:
  - Selection of targets
  - Topology and unit type
  - Initialisation
  - Optimisation

Hybrid System Targets

- Target choices:
  - Context independent - small number, broad
  - Context dependent - large number, fine grained
- Increase in computational power makes CD possible
- Learn posteriors of the states tied by the GMM-HMM decision tree



- Limited work on learning state tying without GMMs [10, 11, 12]

DNN Topology

- In theory a single hidden layer allows any form of decision boundary to be modelled

- In practice: finite training data - bounds number of units per layer

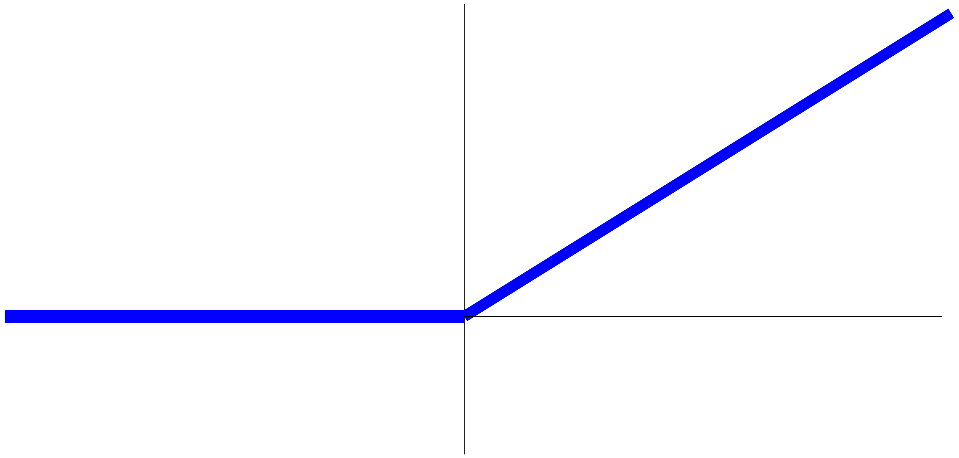
- Balance number and size of layers based on amount of data

- Example topologies from CUED systems

	MGB Challenge [7]	BABEL VLLP [6]
Amount training data	700 hours	3 hours
Network config	$720 \times 1000^5 \times 9500$	$963 \times 1000^4 \times 1000$
Hidden nodes	sigmoid	sigmoid
Output nodes	softmax	softmax

Hidden unit activation functions

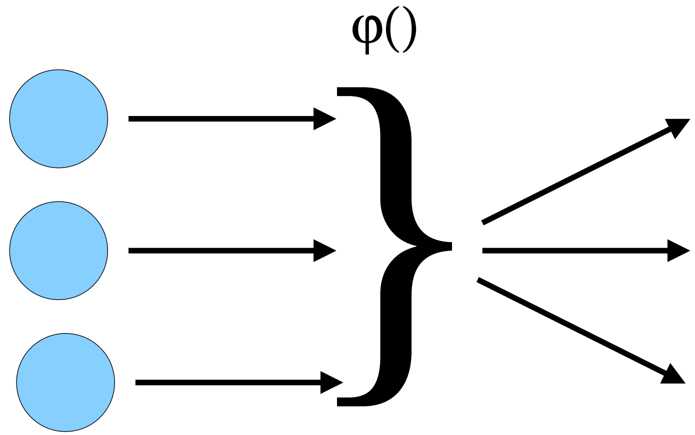
- Sigmoid: traditionally used
- Rectified Linear Units (ReLUs) [13]:



$$y_i(\boldsymbol{x}) = \max(0, z_i)$$

  - Faster convergence in training
  - Simple to optimise
  - Faster to compute at run-time - no exponentiation and division - 25% in [13]
  - Risk of over-fitting to the training data

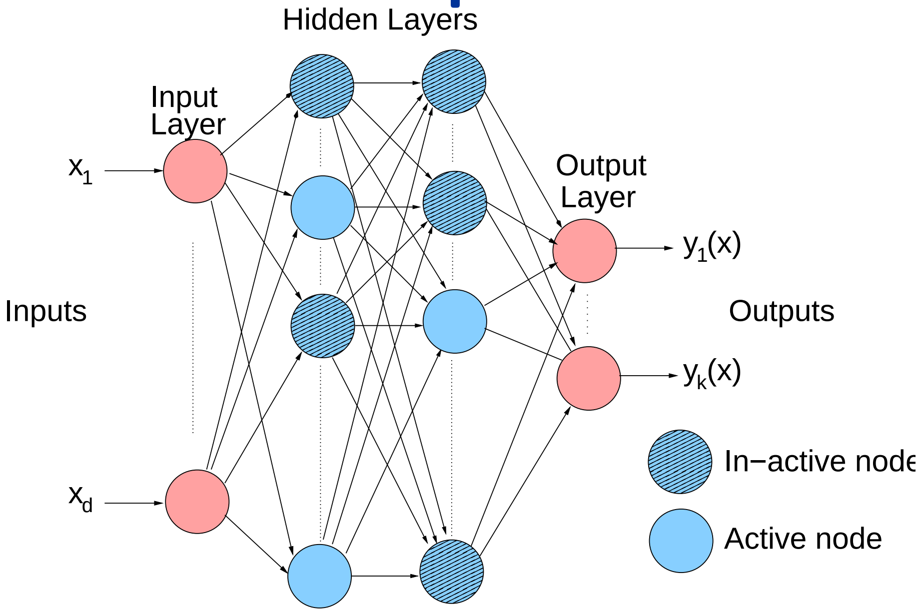
(Generalised) Max-Out Networks



- Reduce "dimensionality" of number of weights
  - apply operations $\phi()$ on a "pool" of output from hidden nodes
  - maxout $\phi(y_1, y_2, y_3) = \max(y_1, y_2, y_3)$
  - soft-maxout $\phi(y_1, y_2, y_3) = \log(\sum_{i=1}^{3} \exp(y_i))$
  - p-norm $\phi(y_1, y_2, y_3) = (\sum_{i=1}^{3} |y_i|)^{1/p}$

- Other approaches use "SVD" reduction (often on output layer)

Dropout



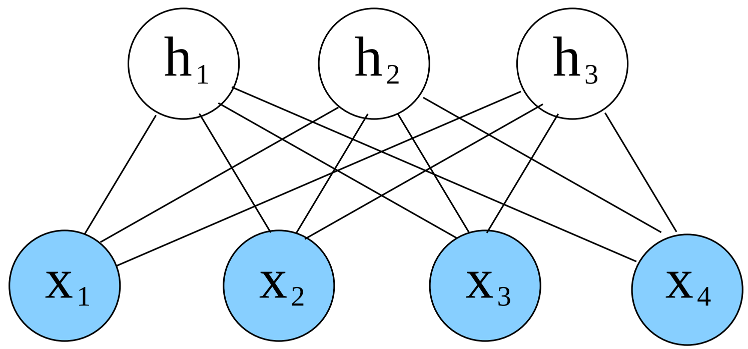
- Avoid over-fitting (beyond regularise weights to zero)
  1. randomly de-activate 50% of the nodes
  2. update model parameters, goto (1)
- Idea is to assume that nodes don't become too specific
  - Work well in conjunction with ReLUs

Network Initialisation

- Prior to optimisation, have to initialise the network parameters for each layer
- Three approaches used:
  1. random: randomly distribute parameters
     - usually samples drawn from a zero mean Gaussian distribution
     - standard approach for many years
  2. restricted Boltzmann machine: use generative RBM to set parameters
  3. layer-by-layer training: incrementally add hidden layers
- After the network has been initialised run standard error back-propagation
  - Called fine-tuning in the context of DNNs
- In 2009-11 it was thought that initialisation was key to performance
  ... now we realise we just need to have a sensible initialisation

Restricted Boltzmann machines

- (Restricted) undirected graphical model
  - Connections are only between the observations and the hidden layer



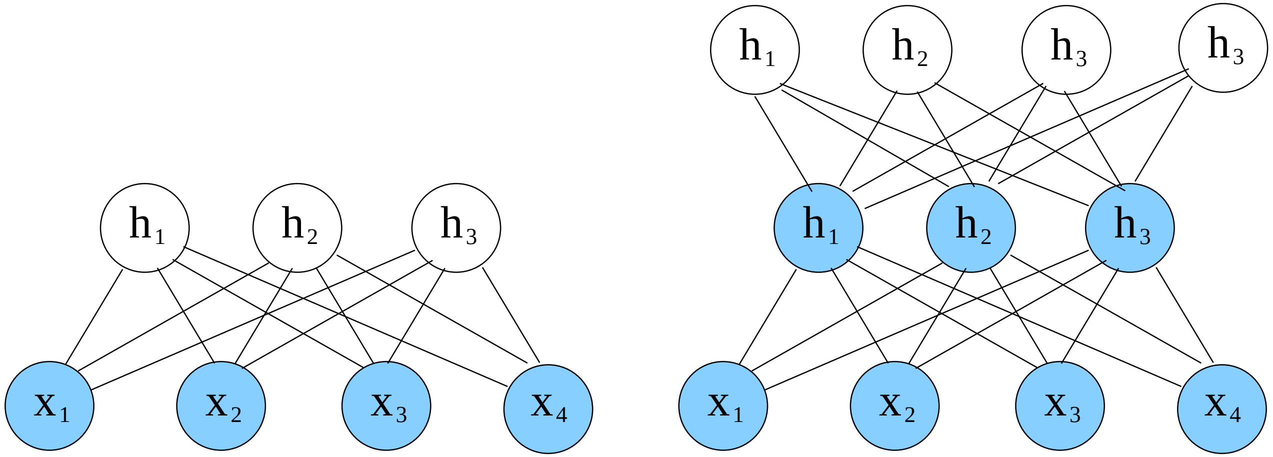
$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z}\sum_{\boldsymbol{h}} \exp\left(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\right)$$

  - $\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})$ is an energy function for the observation $\boldsymbol{x}$ and hiiden layer $\boldsymbol{h}$
  - Z is the normalisation term that ensures that there is a valid PDF

- Parameters can be estimated by contrastive divergence learning [14]

Initialisation with a RBM

- To initialise the NN, generate a RBM layer-by-layer
  - number of elements matches the number required in the final network

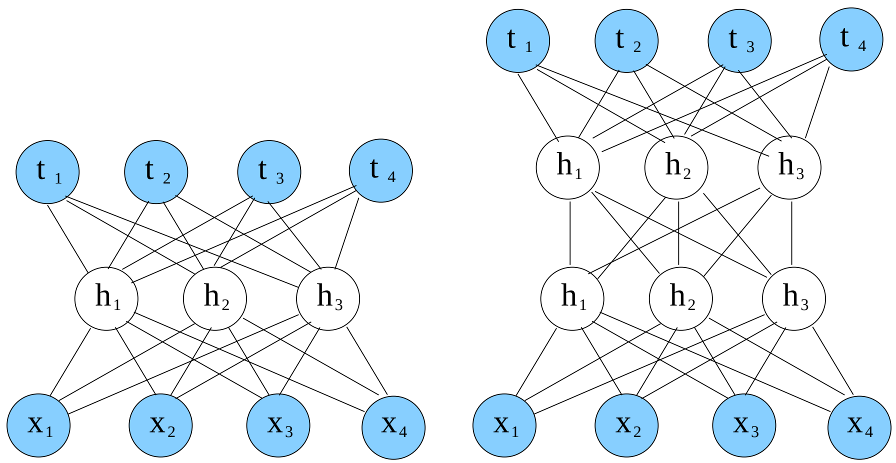


- Sometimes referred to as generative pre-training
- If the hidden layers are not further trained: deep belief network
  - In this case only the output layer is discriminatively trained

Layer-by-layer pre-training

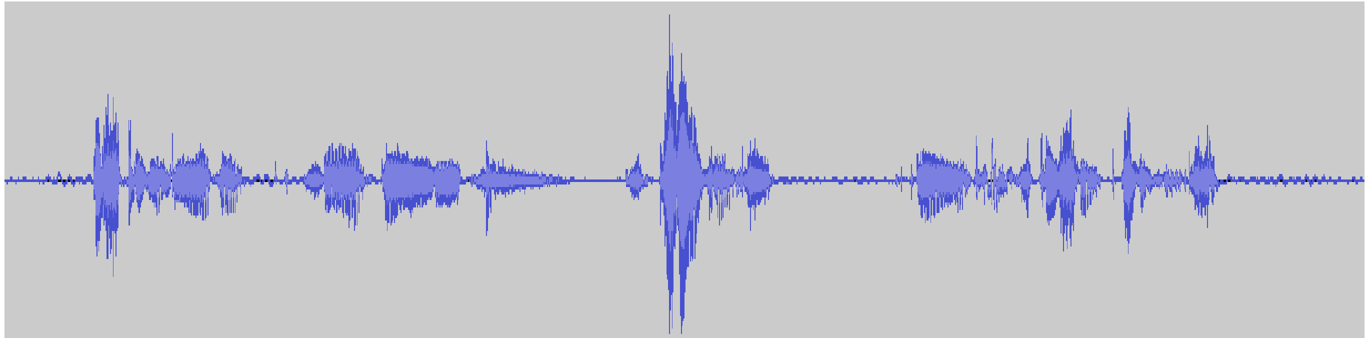
- Incrementally add hidden layers to the network

  1. Add new hidden layer with random initial values
  2. Train the new network - usually restrict number of iterations
  3. Repeat until desired network configuration reached



- Sometimes called discriminative pre-training

Training Criteria - Cross-Entropy



THE CAT SAT ON THE MAT WITH A RAT

- Standard criterion for training Neural Networks:

$$\mathcal{F}_{\texttt{CE}}(\boldsymbol{\lambda}) = -\sum_{r=1}^{R}\sum_{t}\sum_{i=1}^{K} t_{ti}^{(r)} \log(y_i(\boldsymbol{x}_t^{(r)}))$$

  – BUT now need to apply to sequence data with word-labels

- Need to obtain target $t_{ti}^{(r)}$ from the audio and word labels

  – use Viterbi with existing model for best state-sequence/labels

Training Criteria - Sequence-Discriminative training

- Cross-Entropy training converts sequence problem to standard training
  - BUT thrown away all aspects of speech as a sequence of observations
  - no interaction with nature of Hybrid model/Language Model

- Able to use discriminative training criteria used in GMM-HMM systems

- Minimum Bayes' Risk training very popular

$$\mathcal{F}_{\texttt{mbr}}(\boldsymbol{\lambda}) = \frac{1}{R}\sum_{r=1}^{R}\sum_{\mathbf{w}} P(\mathbf{w}|\mathbf{O}^{(r)};\boldsymbol{\lambda})\mathcal{L}(\mathbf{w},\mathbf{w}_{\texttt{ref}}^{(r)})$$

  - loss, $\mathcal{L}(\mathbf{w},\mathbf{w}_{\texttt{ref}}^{(r)})$, often computed at the state-level
  - possible to propagate gradients from this function into the network
  - considers speech as a sequential classification problem

Optimisation - Stochastic Gradient Descent (SGD)

- Training on large data-sets is problematic
  - gradient descent does not have a simple parallelisation (contrast EM)
  - vast number of model parameters to update

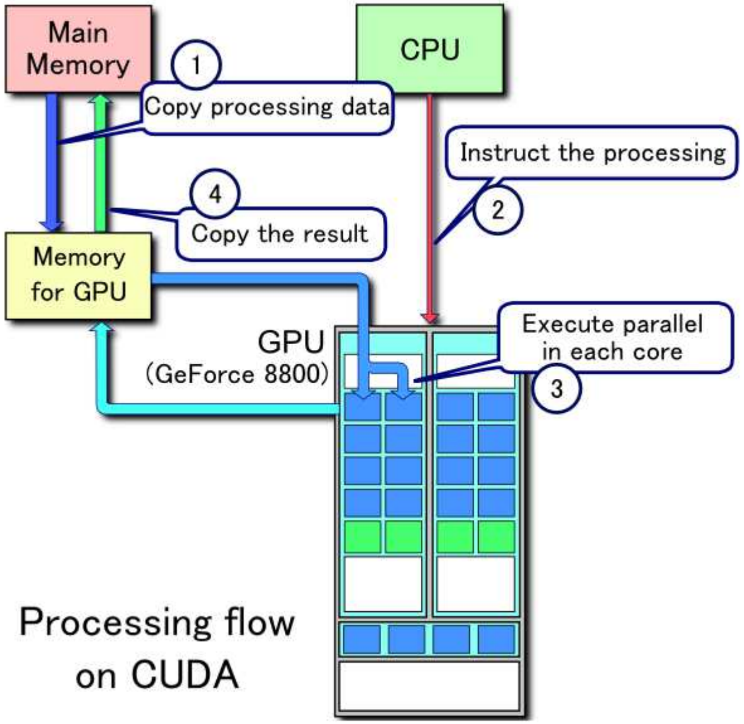
- SGD is combined with mini-batch updates to improve training time
  - randomly select from the complete training data set
  - for CE training randomly select frame samples
  - for sequence training randomly select utterances

BUT still far too slow

- Parallel versions (Google) have been developed, but ...

Optimisation - Graphical Processing Units (GPUs)

- Compute power (based on CPUs) keeps improving - Moore's Law



- Significant aid to NN training provided by introduction of GPU based training
  - originally designed to manipulate and create images

Example of ASR System

IARPA Babel Program

"The Babel Program will develop agile and robust speech recognition technology that can be rapidly applied to any human language in order to provide effective search capability for analysts to efficiently process massive amounts of real-world recorded speech." - Babel Program BAA

IARPA Babel Program Specifications - Option Period 2

- Language Packs
  - Conversational and scripted telephone data (plus other channels)
  - Full (FLP): 60-80 hours transcribed speech
  - Very Limited (VLLP): 3 hours transcribed speech (plus untranscribed speech)
  - 10 hour Development and Evaluation sets
  - No lexicon: Graphemic models
  - Collected by Appen

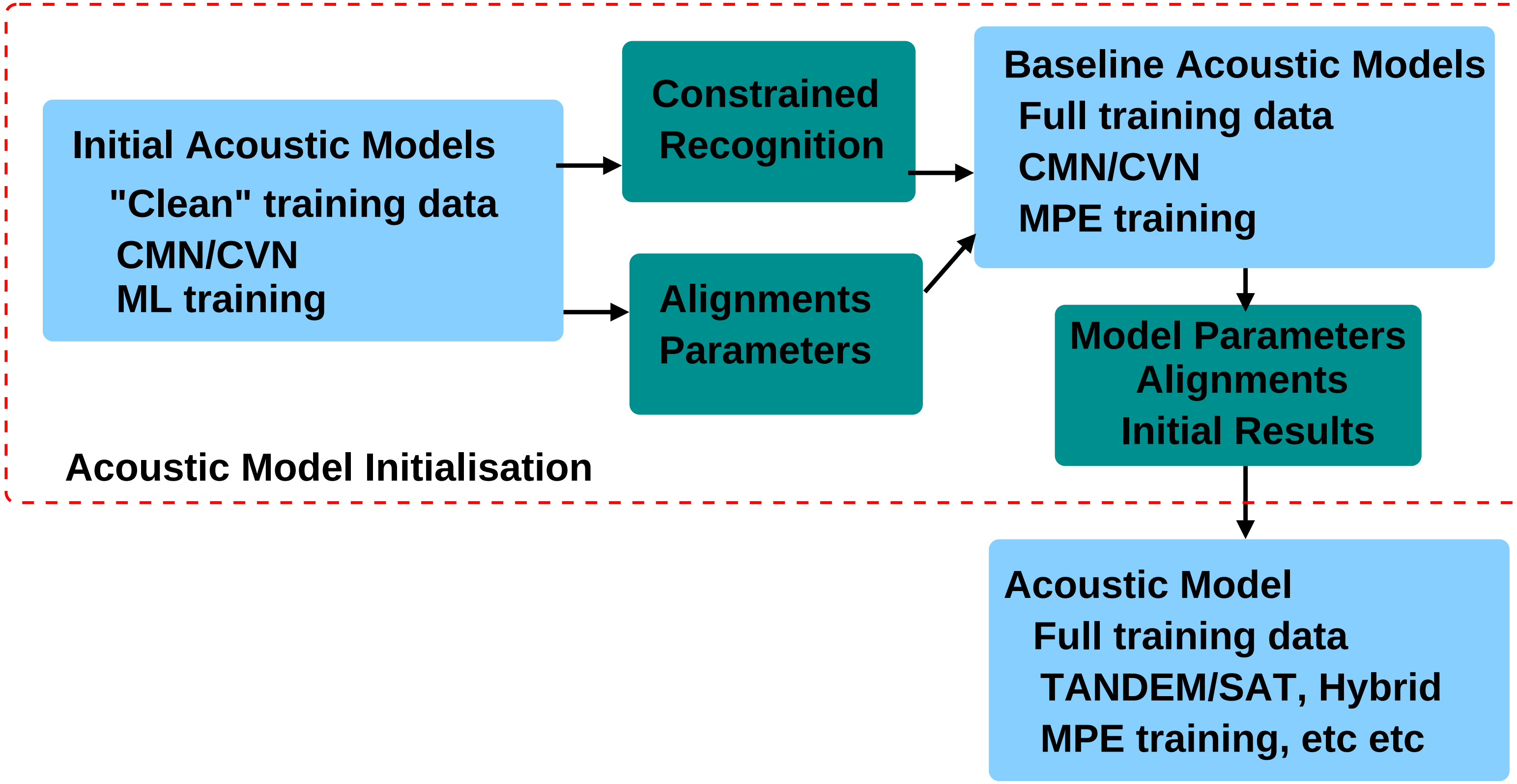
- Evaluation conditions
  - FLP - teams can only use data within a language pack
  - VLLP
    - can use multilingual features trained on BP and OP1 languages
    - can add text data from the web

OP2 Languages

- IARPA Babel Program OP2 language collection releases:

language	ID	Release
Swahili	202	IARPA-babel202b-v1.0d
Kurmanji Kurdish	205	IARPA-babel205b-v1.0a
Tok Pisin	207	IARPA-babel207b-v1.0a
Cebuano	301	IARPA-babel301b-v1.0b
Kazakh	302	IARPA-babel302b-v1.0a
Telugu	303	IARPA-babel303b-v1.0a
Lithuanian	304	IARPA-babel304b-v1.0b

General Training Procedure



- “Clean” training data - remove segments containing:
  - unintelligible, mispronounce, fragment words
- Convert PCM, 48KHz, 24-bit to A-law, 8KHz, 8-bit

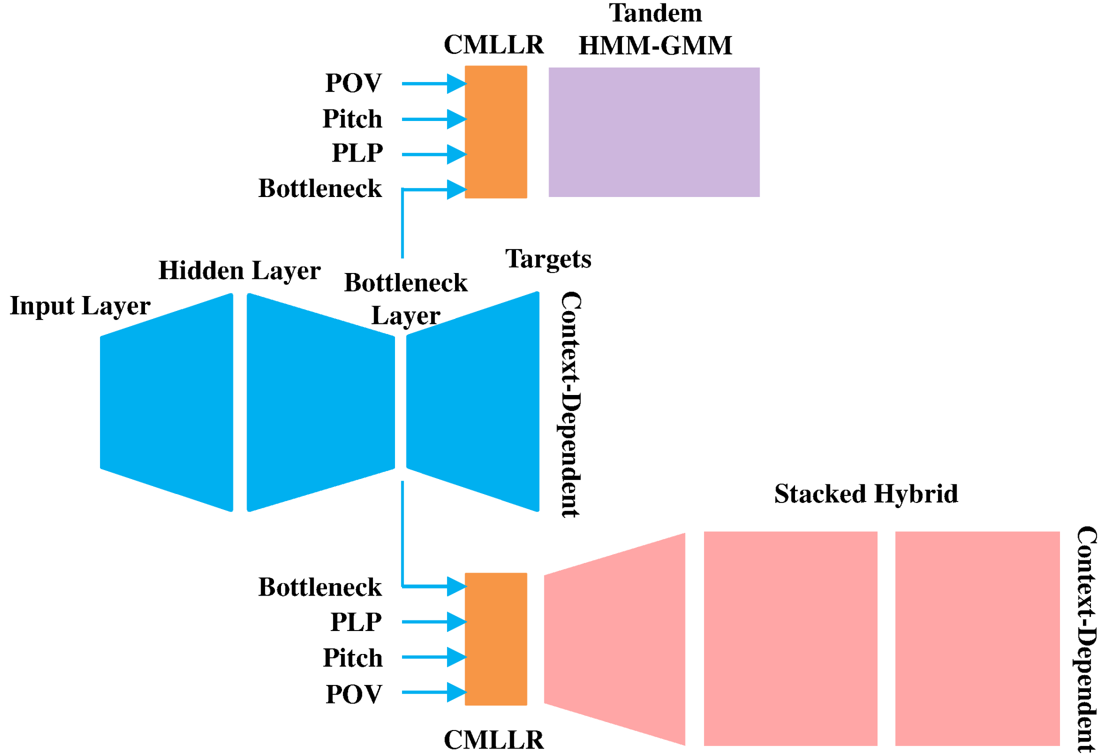
Common Development Language Configuration

- Trained with HTK V3.4.1 [15] with DNN extension [16]
- Graphemic dictionary - no language specific knowledge
- decision trees state-position roots of trees
- Tandem features
- MPE, speaker adaptive training at the conversation side level
- sequence-trained stacked Hybrid systems
- N-gram word and pseudo-syllable language models
- joint (Hybrid/Tandem) decoding and lattice generation
- unique arc-per-second pruning for diverse lattice generation
- arc-level score deweighting

FLP Configuration

- 6000 distinct decision-tree states
- CUED Bottleneck features (FBK+PoV+Pitch)
- Bottleneck NN structure: $936 \times 1000^4 \times 39 \times 6000$
- Hybrid NN structure: $486 \times 1000^5 \times 6000$

Tandem and Hybrid ASR



- Common front-end - speaker normalised stacked features
- Layer-wise pretraining all NNs
- Tandem ASR system discriminatively trained with MPE
- Hybrid ASR system: CE fine-tuning, MPE-based sequence training

Comparison Tandem and Hybrid

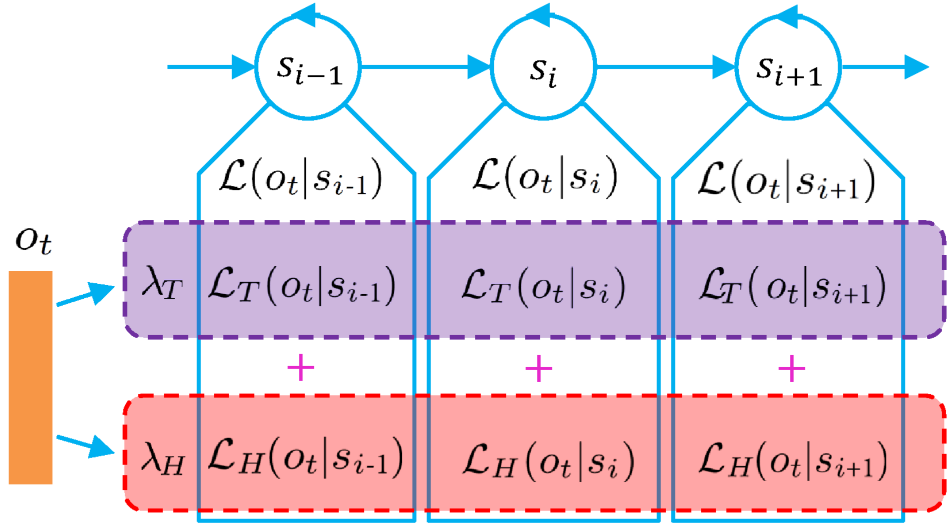
- Swahili speaker independent (SI) systems

System	%WER
GMM	57.8
Tandem	50.5
Hybrid-CE	51.1
Hybrid-MPE	49.4

- Clear reduction in WER over GMM-based system

- Sequence training is required to reduce Hybrid WER below Tandem

Joint Decoding Tandem/Hybrid [6]



- Frame-level combination: linear combination log likelihoods

$$\mathcal{L}(\boldsymbol{o}_t|\boldsymbol{s}_i) = \lambda_T\mathcal{L}_T(\boldsymbol{o}_t|\boldsymbol{s}_i) + \lambda_H\mathcal{L}_H(\boldsymbol{o}_t|\boldsymbol{s}_i)$$

  where λ_T, λ_H are weights for Tandem and Hybrid respectively
- single pass decoding and keyword search
- can rescore joint decoding lattices using a single Tandem or Hybrid model

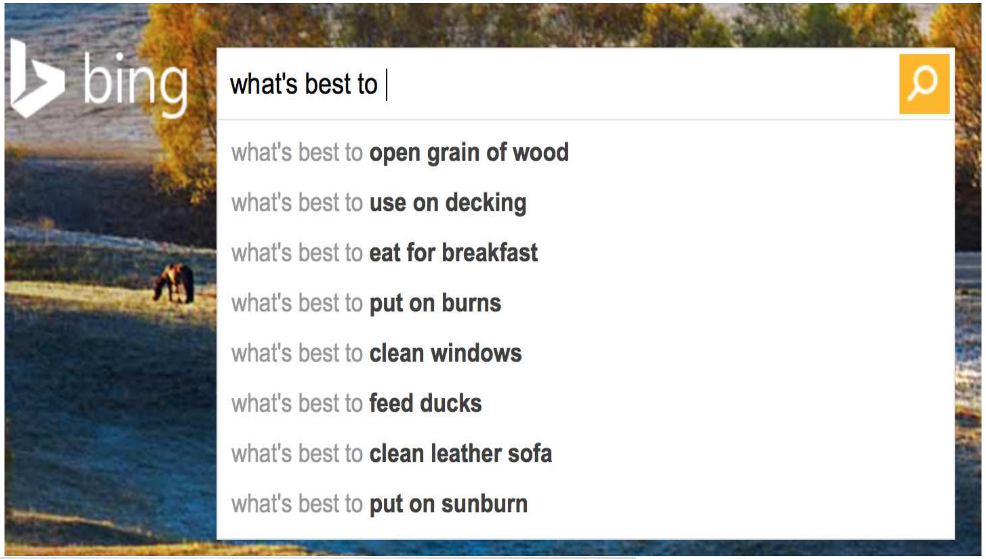
ASR Performance (%WER)

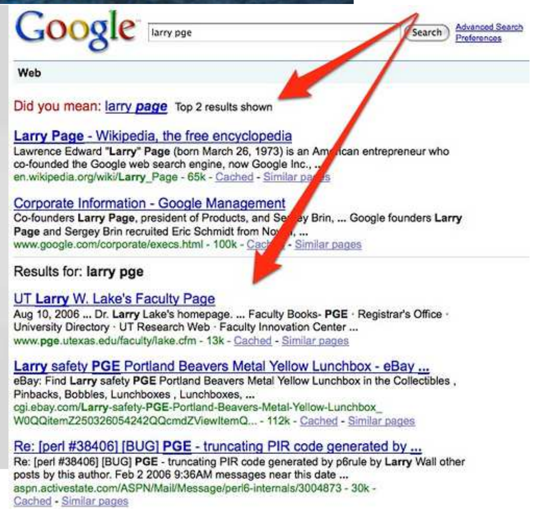
System	Tok Pisin	Cebuano
Tandem	40.7	54.2
Hybrid	39.2	52.8
Tandem $\oplus$ Hybrid	38.8	52.3
Joint	38.4	52.0

- Joint combination out-performs confusion network combination ($\oplus$)
- Carries over to keyword spotting

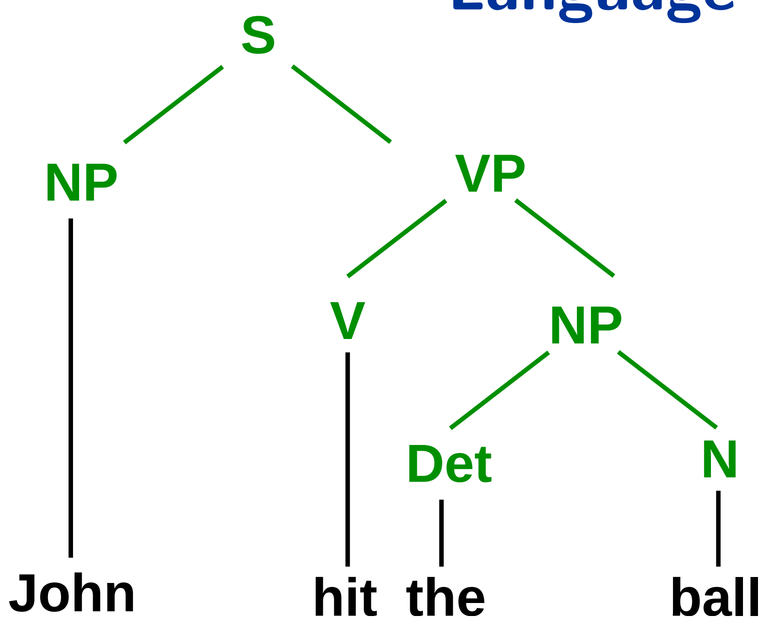
Neural Networks for Language Modelling

Language Modelling





Language Modelling



(a) Syntactic Parse Tree

P(John hit the ball) =
P(John) x
P(hit | John) x
P(the | John hit) x
P(ball| hit the)

(b) Trigram Model

- Syntactic/semantic information important
  - but hard to model robustly (especially for conversational style speech)
- Simple n-gram model-used: $P(w_1 w_2 ... w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-2} w_{i-1})$
  - don't care about structure - just the probability

N-Gram Language Models

- Consider a task with a vocabulary of V words (LVCSR $65K+$)
  - 10-word sentences yield (in theory) V^{10} probabilities to compute
  - not every sequence is valid but number still vast for LVCSR systems

Need to partition histories into appropriate equivalence classes

- Assume words *conditionally independent given previous* $N-1$ *words:* $N = 2$

$$P(\texttt{ball}|\texttt{John},\texttt{hit},\texttt{the}) \approx P(\texttt{ball}|\texttt{She},\texttt{dressed},\texttt{for},\texttt{the}) \approx P(\texttt{ball}|\texttt{the})$$

  - *simple form of equivalence mappings - a bigram language model*

$$P(\boldsymbol{w}) = \prod_{k=1}^{K+1} P(w_k|w_0,\ldots w_{k-2},w_{k-1}) \approx \prod_{k=1}^{K+1} P(w_k|w_{k-1})$$

N-Gram Language Models

- The simple bigram can be extended to general N-grams

$$P(\boldsymbol{w}) = \prod_{k=1}^{K+1} P(w_k|w_0, \ldots w_{k-2}, w_{k-1}) \approx \prod_{k=1}^{K+1} P(w_k|w_{k-N+1}, \ldots, w_{k-1})$$

- Number of model parameters scales with the size if N (consider $V = 65K$):
  - unigram (N=1): $65K^1 = 6.5 \times 10^4$
  - bigram (N=2): $65K^2 = 4.225 \times 10^9$
  - trigram (N=3): $65K^3 = 2.746 \times 10^{14}$
  - 4-gram (N=4): $65K^4 = 1.785 \times 10^{19}$

  Web comprises about 4.5 billion pages - not enough data!

- Long-span models should be more accurate, but large numbers of parameters

A central problem is how to get robust estimates and long-spans?

Modelling Shakespeare

- Jurafsky & Martin [17]: N-gram trained on the complete works of Shakespeare

Unigram

- Every enter now severally so, let
- Will rash been and by I the me loves gentle me not slavish page, the and hour; ill let

Bigram

- What means, sir. I confess she? then all sorts, he is trim, captain.
- The world shall- my lord!

Trigram

- Indeed the duke; and had a very good friend.
- Sweet prince, Fallstaff shall die. Harry of Monmouth's grave.

4-gram

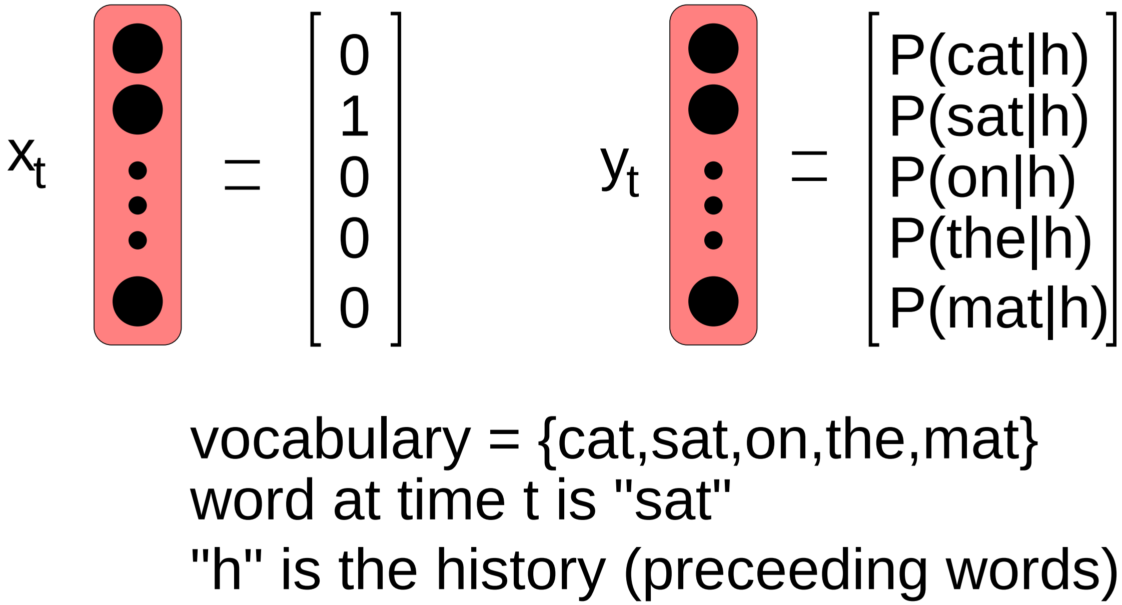
- It cannot be but so.
- Enter Leonato's brother Antonio, and the rest, but seek the weary beds of people sick.

Issues with N-grams

- N-grams are the dominant form of language model
  - simple estimation - just count
  - fast to train on large corpora (billions of words)
  - integrates well with Viterbi estimation
- N-grams are a discrete distribution over words
  - hard to share parameters
  - robustness of estimates handled by discounting/back-off

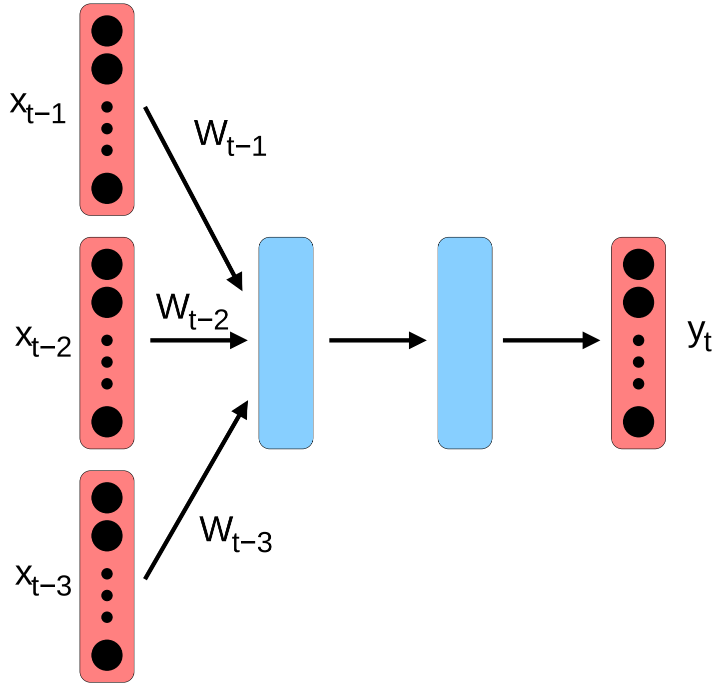
Is there a neural network language model option?

NN Input and Output Word Representation



- Simplest word representation is 1-of-K coding
  - input/output vectors size of vocabulary
  - at input only one element of vector non-zero

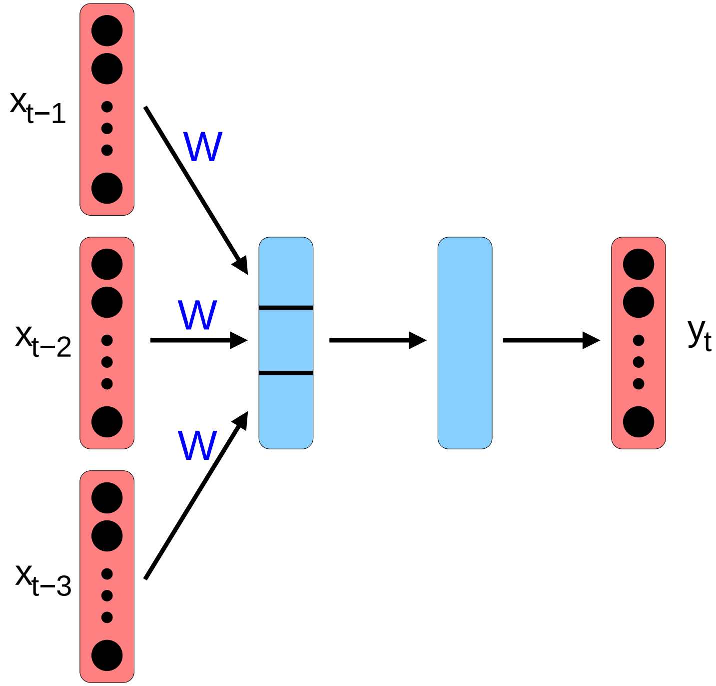
Naive Feed-forward neural networks



- Consider the equivalent of a 4-gram
  - output (y_t) a function of the three preceding words
  - number of parameters to project from input to hidden layer very large

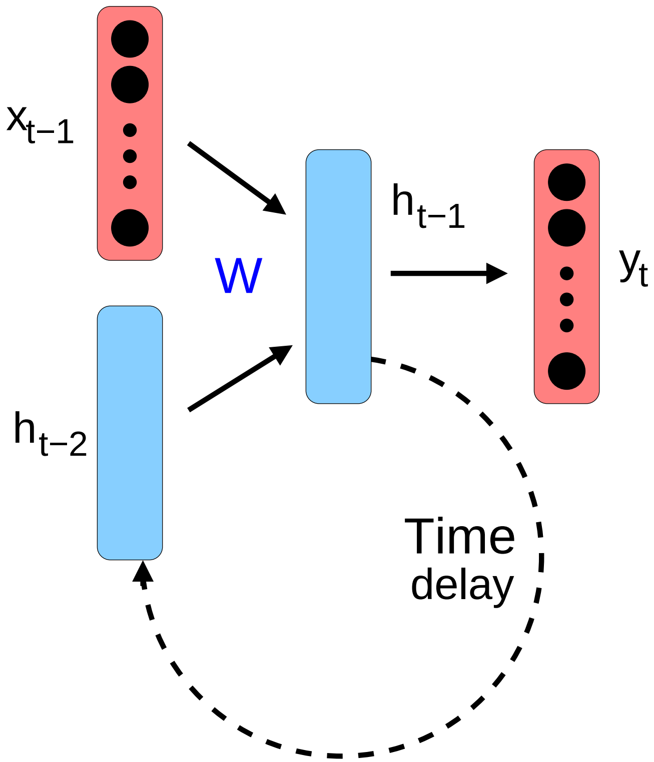
How to constrain the number of parameters?

Feed-forward neural networks



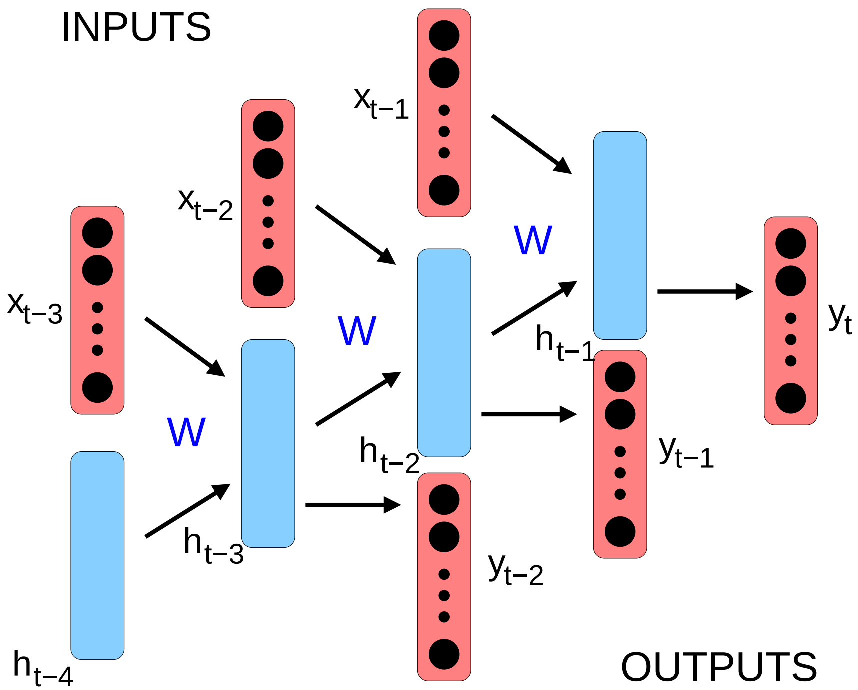
- Bengio et al [18] proposed a neural probabilistic language model
  - further developed by a number of sites [19, 20, 21, 22]
- Projects words into a continuous space word-space
  - improves generalisation - removed discrete aspects of N-gram
  - however still has finite history

Recurrent Neural Network Language Model (RNNLM)



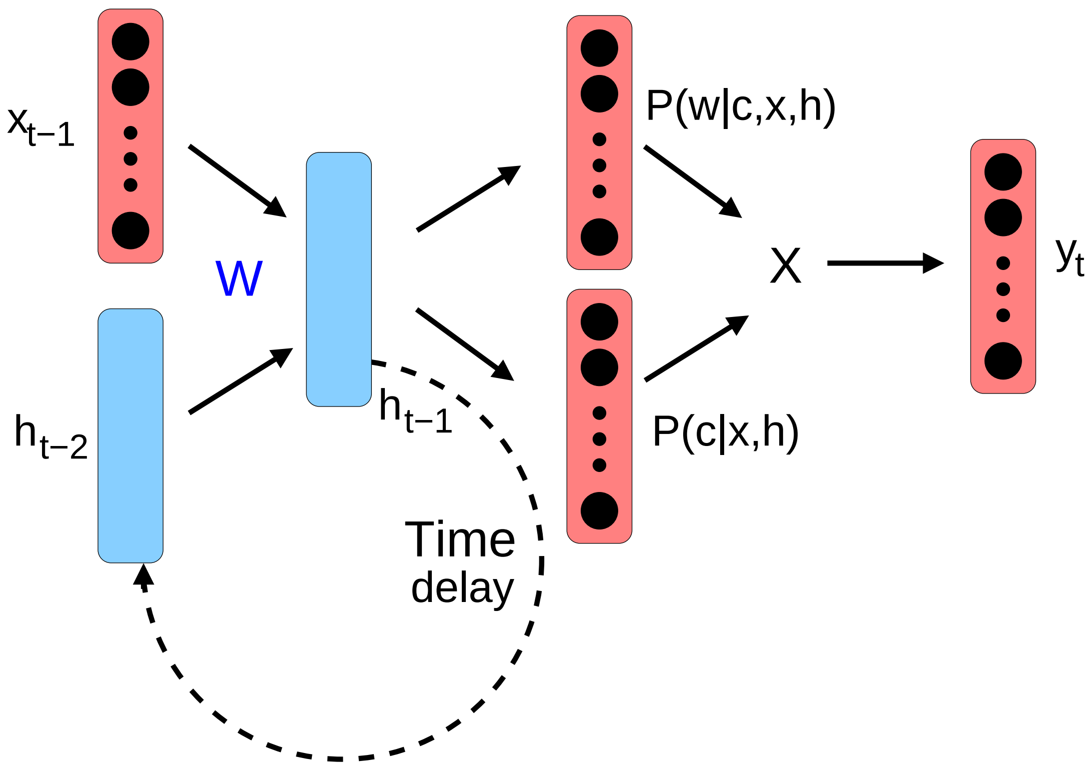
- Use the hidden state values as a compact history representation [23, 24]
  - allows network to model arbitrary length histories!

RNNLM Training



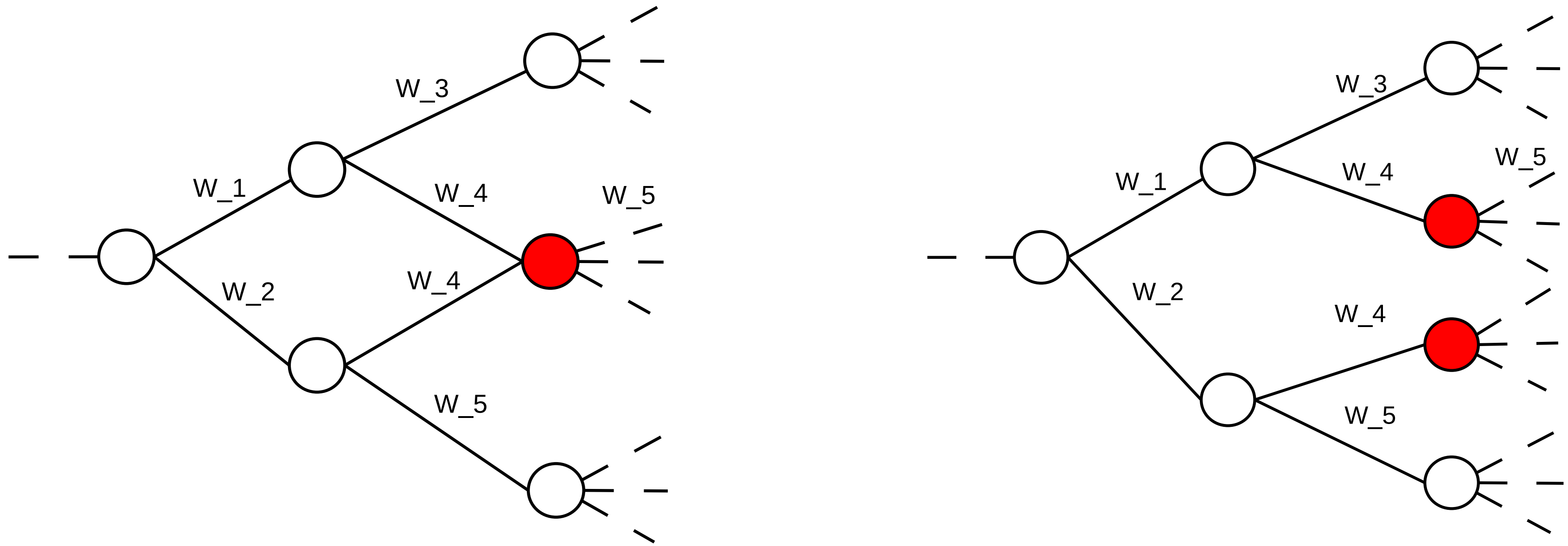
- RNN training uses Back-Propagation through time (BPTT)
  - implementation can be amusing - simplification truncate history [25, 26]
  - now looks like a standard DNN with weight sharing

Class-based RNNLM



- Softmax calculation at output layer dominates training and test time
  - CPU training impractical - use class-based output layers [27, 28, 29, 30]
  - recently GPU training allows full output layer to be used [31]

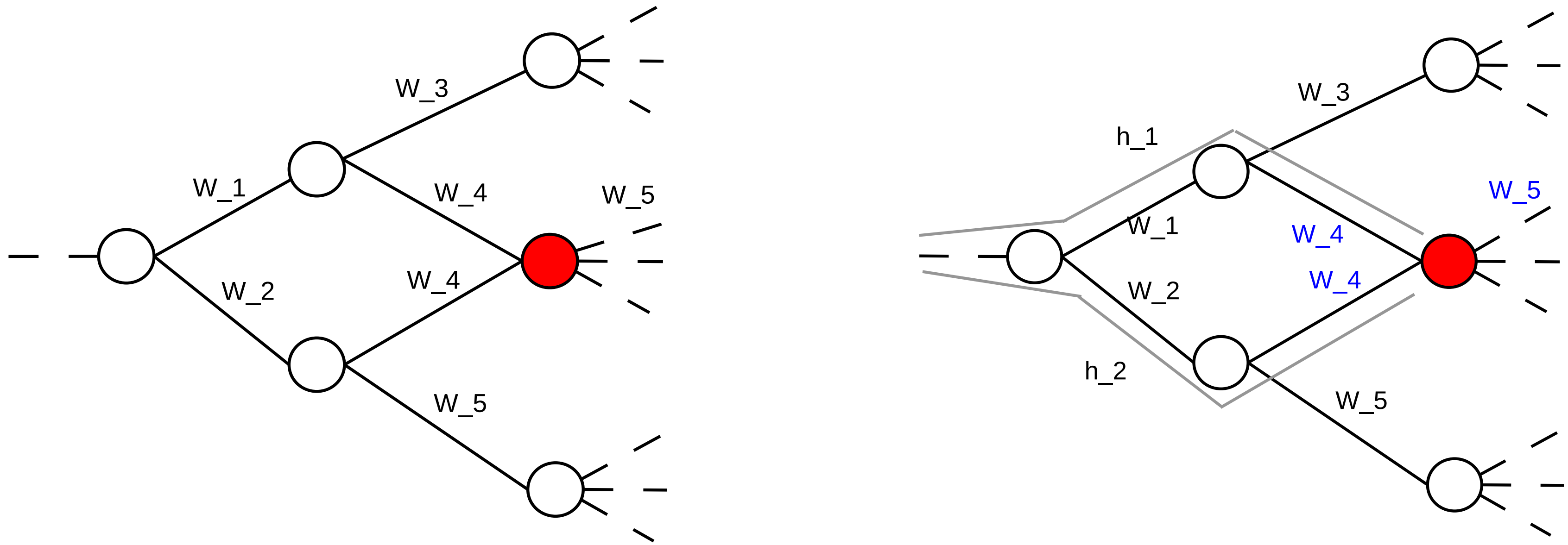
Decoding with RNNLM



- The history vector of RNN encodes complete path history
  - standard (bigram) path merging (left diagram) cannot be applied
  - exact decoding yields a prefix tree (right diagram)

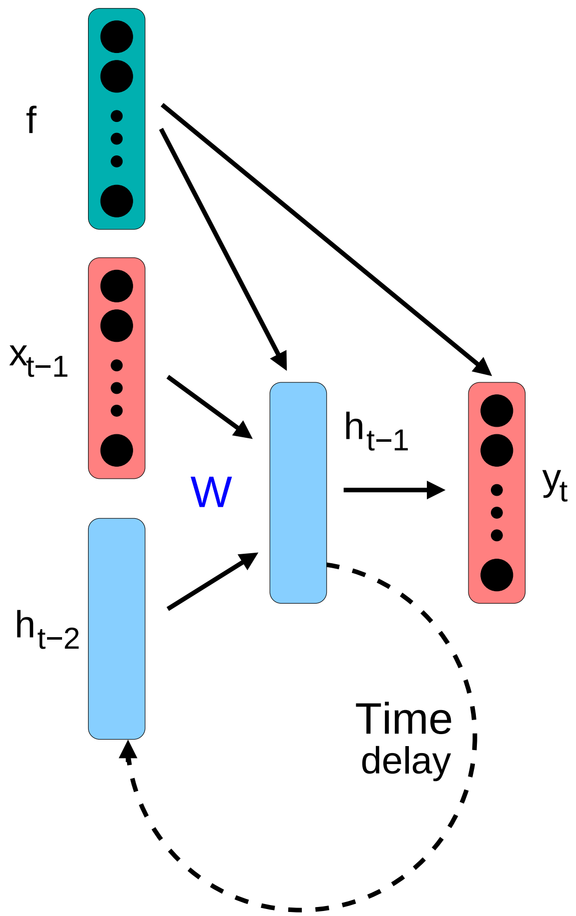
- For most tasks this rapidly becomes impractical

Approximations for RNNLM Decoding



- Simple approximation is to apply an N-gram approximation [25]
  - apply same path merging as standard N-gram (bigram in diagram)
  - however the language model score is computed from the RNN
  - allows lattices to be generated (and simply rescored)

RNNLM Adaptation



- Add auxiliary feature to input layer [32, 33, 34, 35, 36]

Example of Using RNNLMs

Multi-Genre Broadcast (MGB) Challenge

MGB CHALLENGE

Multi-Genre Broadcast (MGB) Challenge

- Training data
  - Approximately 1600 hours broadcast audio from 7 weeks BBC output
  - Captions as originally broadcast on TV
  - Several 100M words subtitles text from BBC TV output over 15 year period
  - Hand-compiled British English lexicon

- Evaluation - one week BBC output
  - Speech-to-text transcription of broadcast television
  - Alignment of broadcast audio to a subtitle file, ie. lightly supervised
  - Longitudinal speech-to-text transcription of a sequence of episodes from the same series of programmes
  - Longitudinal speaker diarization and linking, requiring the identification of common speakers across multiple recordings

Results

LM	PPlex	% WER
4-gram	103.1	25.6
+ RNN	93.0	25.0
+ RNN topic adaptation	81.0	24.4

- Interpolated 4-gram LM [7]
  - Trained on 650 million word text data and transcriptions
  - Vocaulary sizes: 160K; 65K
- RNNLM
  - 64K vocabulary

Other Neural Network Architectures

Convolutional Neural Networks (CNNs)

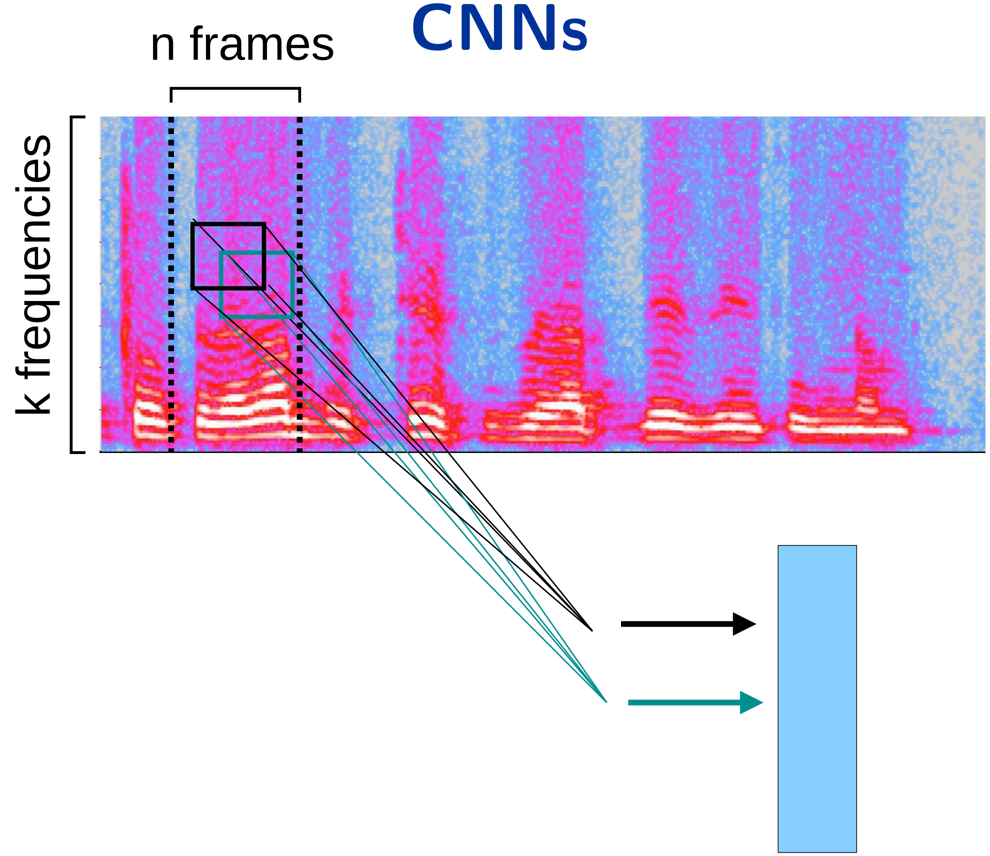
DNNs do not model well:

- Input correlations
- Translation variance

CNNs are potentially more powerful than DNNs as

- They reduce spectral variations and model correlations in the input signal
- Jointly while performing discrimination

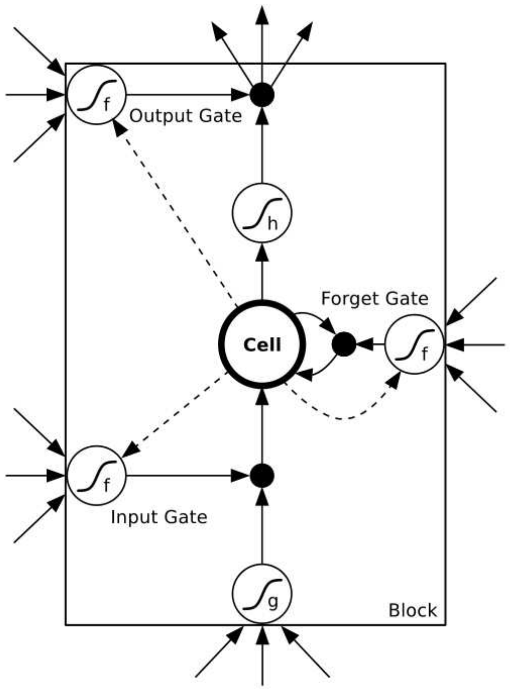
CNNs



- Parameters of each convolution kernel are trained by back propagation
- First layers obtain low-level features e.g. edges, lines, corners
- More abstract, higher-level features at higher layers

Long Short Term Memory (LSTM) Networks

- RNNs suffer from "vanishing gradients" problem
  - Derivatives of loss function relative to weights for past iterations becomes 0
  - Alternative: Long Short Term Memory (LSTM) [37] network



Source: A. Graves, "Supervised Sequence Labelling with Recurrent Neural Networks", Studies in Computational Intelligence, 2012.

Long Short Term Memory (LSTM) Networks

- LSTMs:
  - special kind of RNN, capable of learning long-term dependencies
  - sigmoid is replaced with a “perfect integrator memory cell”
  - allows LSTM to store and access information over long periods of time
  - as long as the input gate is closed the cell’s activation is not overwritten

End-to-End Speech Recognition

- Would it be better to learn the speech features at the same time as the model?
  - A number of sites are looking at inputting raw waveform directly to NN pipeline
  - Still early days - Log Mel filterbanks generally better
    Sainath et al Interspeech2015 small gain for noisy voice search trained on 2K hours [38]

- Can we eliminate the HMM?
  - Not yet - but people are working on it e.g. [39]

DNN Toolkits

- (Primarily) speech recognition toolkits
  - Kaldi http://kaldi.sourceforge.net
  - Aachen RWTH ASR (RASR) http://www-i6.informatik.rwth-aachen.de/rwth-asr/
  - HTK V3.5 DNN support [16] - end of 2015
- Microsoft's Computational Network Toolkit (CNTK) [40] http://cntk.codeplex.com
- Torch - ML algorithms in C and lua http://www.torch.ch
- Theano - Python library
- RNNLM Toolkit by Tomas Mikolov http://www.rnnlm.org

Summary

- Significant advances in speech processing performance achieved with NNs
- Recent advances enabled by
  - Increase in computing power
  - Increase in training data
- Not the complete solution
- More structured computational networks in the near future
- Worth looking at earlier work
  - Intern at Google ran over 10000 RNN architectures to find a better one [41]
    Best result: add bias of 1 to LSTM forget gate - published in 2000 [42]

Acknowledgements

- Thanks to Heiga Zen and Pierre Lanchantin for TTS and MGB samples

- Thanks to Mark Gales, Andrew Liu, Andrey Malinin, Anton Ragni and Haipeng Wang for help with slides

- Babel project experiments: This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U. S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0012. The U. S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U. S. Government.

References

[1] Y Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, pp. 1–127, 2009.

[2] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *Proc. ICASSP*, 2012.

[3] Xue Feng, Yaodong Zhang, and James Glass, "Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition," in *Proc. ICASSP*, 2014.

[4] H. Hermansky, D. Ellis, and S. Sharma, "Tandem Connectionist Feature Extraction for Conventional HMM Systems," in *Proc. ICASSP*, 2000.

[5] Frantisek Grezl et al., "Probabilistic and bottle-neck features for LVCSR of meetings," in *Proc. ICASSP*, 2007.

[6] H. Wang, A. Ragni, M. Gales, K. Knill, P. Woodland, and C. Zhang, "Joint Decoding of Tandem and Hybrid Systems for Improved Keyword Spotting on Low Resource Languages," in *Proc. Interspeech*, 2015.

[7] Philip Woodland, Xunying Liu, Yanmin Qian, Chao Zhang, Mark Gales, Penny Karanasou, Pierre Lanchantin, and Linlin Wang, "Cambridge University Transcription Systems for the Multi-Genre Broadcast Challenge," in *To appear Proc. ASRU*, 2015.

[8] Frantisek Grezl, Martin Karafiat, and Lukas Burget, "Investigation into bottle-neck features for meeting speech recognition," in *Proc. Interspeech*, 2009.

[9] Frantisek Grezl et al., "BUT Neural Network Features for Spontaneous Vietnamese in Babel," in *Proc. ICASSP*, 2014.

[10] M. Bacchiani and D. Rybach, "Context Dependent State Tying for Speech Recognition using Deep Neural Network Acoustic Models," in *Proc. ICASSP*, 2014.

[11] A. Senior, G. Heigold, M. Bacchiani, and H. Liao, "GMM-Free DNN Training," in *Proc. ICASSP*, 2014.

[12] C. Zhang and P. C. Woodland, "Standalone Training of Context-Dependent Deep Neural Network Acoustic Models," in *Proc. ICASSP*, 2014.

[13] M.D. Zeiler et al., "On rectified linear units for speech processing," in *Proc. ICASSP*, 2013.

[14] G.E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, , no. 8, pp. 1771–1800, 2002.

[15] S. J. Young et al., *The HTK Book (for HTK version 3.4.1)*, Cambridge University, 2009, http://htk.eng.cam.ac.uk.

[16] C. Zhang and P. Woodland, "A General Artificial Neural Network Extension for HTK," in *Proc. Interspeech*, 2015.

[17] Daniel Jurafsky and James H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*, Prentice-Hall, 2nd edition edition, 2009.

[18] Y. Bengio and R. Ducharme, "A neural probabilistic languagde model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[19] H. Schwenk and J.-L. Gauvain, "Training Neural Network Language Models on Very Large Corpora," in *Proc. HLT-EMNLP*, 2005.

[20] George Saon, Hagen Soltau, Upendra Chaudhari, Stephen Chu, Brian Kingsbury, Hong-Kwang Kuo, Lidia Mangu, and Daniel Povey, "The IBM 2008 GALE Arabic speech transcription system," in *Proc. ICASSP*, 2010.

[21] A. Emami and L. Mangu, "Empirical study of neural network language models for Arabic speech recognition," in *Proc. ASRU*, 2007.

[22] X. Liu, M. J. F. Gales, and P. C. Woodland, "Language Model Cross Adaptation For LVCSR System Combination," in *Proc. Interspeech*, 2010.

[23] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Interspeech*, 2010.

[24] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of Recurrent Neural Network Language Model," in *Proc. ICASSP*, 2011.

[25] X. Liu, Y. Wang, X. Chen, M.J.F. Gales, and P.C. Woodland, "Efficient Lattice Rescoring Using Recurrent Neural Network Language Models," in *Proc. Interspeech*, Sep 2014.

[26] X. Chen, X. Liu, M.J.F. Gales, and P.C. Woodland, "Improving the Training and Evaluation Efficiency of Recurrent Neural Network Language Models," in *Proc. ICASSP*, Apr 2015.

[27] Hong-Kwang Kuo, Ebru Arisoy, Ahmad Emami, and Paul Vozila, "Large scale hierarchical neural network language models," in *Proc. Interspeech*, 2012.

[28] Diamanino Caeiro and Andrej Ljolje, "Multiple parallel hidden layers and other improvements to recurrent neural network language modeling," in *Proc. ICASSP*, 2013.

[29] M. Sundermeyer et al., "Comparison of feedforward and recurrent neural network language models," in *Proc. ICASSP*, 2013.

[30] Geoffrey Zweig and Konstantin Makarychev, "Speed regularization and optimality in word classing," in *Proc. ICASSP*, 2013.

[31] X. Chen, Y. Wang, X. Liu, M.J.F. Gales, and P.C. Woodland, "Efficient Training of Recurrent Neural Network Language Models using Spliced Sentence Bunch," in *Proc. Interspeech*, 2014.

[32] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *Proc. SLT*, 2012.

[33] T.-H. Wen et al., "Recurrent neural network based personalized language modeling by social network crowdsourcing," in *Proc. Interspeech*, 2013.

[34] Y. Shi, *Language models with meta-information*, Ph.D. thesis, TU Delft, Delft University of Technology, 2014.

[35] O. Tilk and T. Alumae, "Multi-domain recurrent neural network language model for medical speech recognition," in *Proc. HLT*, 2014.

[36] X. Chen, T. Tan, X. Liu, P. Lanchantin, M. Wan, M.J.F. Gales, and P.C. Woodland, "Recurrent Neural Network Language Model Adaptation for Broadcast Speech Recognition," in *to appear Proc. ASRU*, 2015.

[37] Sepp Hochreiter and Jurgen Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the Speech Front-end with Raw Waveform CLDNNs," in *Proc. Interspeech*, 2015.

[39] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals, "Listen, Attend and Spell," *arXiv*, 2015, 1508.01211 [cs.CL].

[40] D. Yu et al., "An Introduction to Computational Networks and the Computational Network Toolkit," Tech. Rep. MSR-TR-2014-112, Microsoft, 2014.

[41] Rafal Jozefowicz et al., "An Empirical Exploration of Recurrent Network Architectures," in *arXiv*, 2015.

[42] Felix A. Gers et al., "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.