An Overview of HTK V3.5

Phil Woodland
& Cambridge HTK team
pcw@eng.cam.ac.uk

Cambridge University Engineering Department

UK Speech Meeting UEA, 3rd July 2015

Outline

- Background
  - What is HTK
  - Speech Recognition architecture
  - HTK v3.4.1 Main Features
  - Deep Neural Network acoustic models
  - Recurrent Neural Network language models
- HTK v3.5
  - Extensions for Deep Neural Network acoustic models
  - Lattice rescoring with recurrent Neural Network language models
  - Overview of key features
- Some recent ASR Systems built with HTK
  - BOLT Mandarin conversational telephone speech
  - MGB challenge (multi-genre broadcast data)
- Summary and Plans

HTK Contributors

- HTK V3.4.1 book has authors:
  Steve Young, Gunnar Evermann, Mark Gales,
  Thomas Hain, Dan Kershaw, Xunying (Andrew) Liu,
  Gareth Moore, Julian Odell, Dave Ollason,
  Dan Povey, Valtcho Valtchev, Phil Woodland

- Major additions in HTK 3.5 will be primarily due to
  - Chao Zhang (HTK-ANN extension) †
  - Xunying Liu (Language model interface / RNNLM decoding).

- Additional V3.5 input from Anton Ragni, Kate Knill, Mark Gales, Jeff Chen and many others at Cambridge.

† See also: C. Zhang & P.C. Woodland “A General Artificial Neural Network Extension for HTK”,To appear, Interspeech 2015

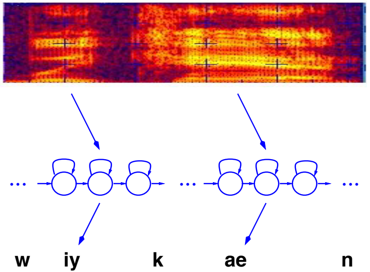
HTK Overview

- What is HTK?
  - Hidden Markov Model Toolkit
  - set of tools for training and evaluating HMMs:
    primarily speech recognition but also speech synthesis (HTS)
  - implementation in ANSI C
  - approx 400 page manual tutorial and system build examples
  - modular structure simplifies extensions
- History (1989-)
  - Initially developed at Cambridge University (up to V1.5)
  - ... then Entropic ... (up to V2.2)
  - Since 2000 back at Cambridge (V3 onwards)
  - Free to download from web, more than 100,000 registered users
  - Latest released version is V3.4.1 (in 2009 ...)
- Used extensively for research (& teaching) at CU
  - Built large vocabulary systems for NIST evaluations using HTK

http://htk.eng.cam.ac.uk/

Statistical ASR System

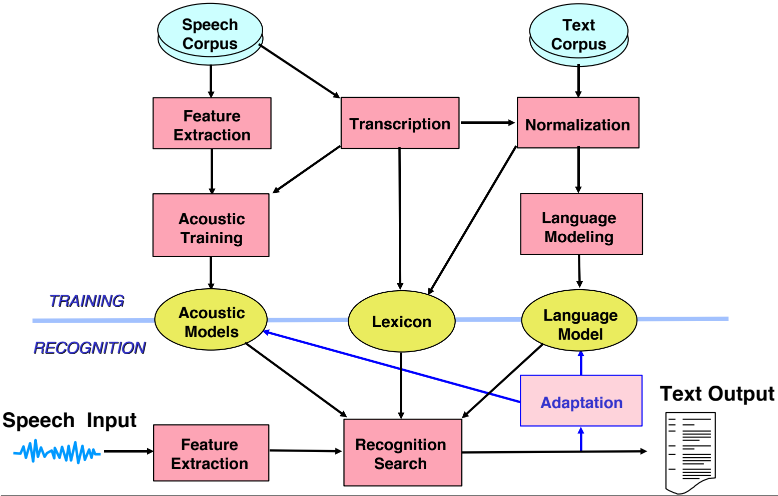
- Statistical speech models using context-dependent hidden Markov Models
  - Decision tree state tying
  - Gaussian mixture models (or Neural Networks)
- probabilities of word sequences (N-gram)
- Estimate the models from a large amount of data
- Find most probable word sequence using the models search (decoding) problem



$$
\begin{aligned}
\widehat{W} &= \operatorname{argmax}_W P(W|A) \\
&= \operatorname{argmax}_W \frac{P(A|W)\,P(W)}{P(A)} \\
&= \operatorname{argmax}_W \underbrace{P(A|W)}_{\substack{Acoustic \\ Model}} \; \underbrace{P(W)}_{\substack{Source \\ Language\,Model}}
\end{aligned}
$$

Training/Test Architecture

- HTK includes components for all stages of the speech recognition process

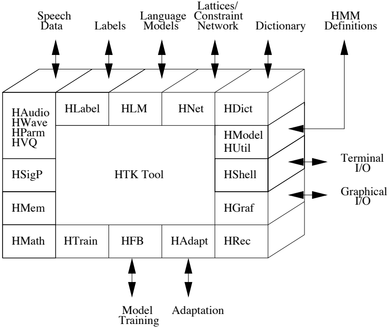


HTK Features

- LPC, mel filterbank, MFCC and PLP frontends
  - cepstral mean/variance normalisation + vocal tract length norm.
- supports discrete and (semi-)continuous HMMs
  - diagonal and full covariance models
  - cross-word triphones & decision tree state clustering
  - (embedded) Baum-Welch training
- Viterbi recognition and forced-alignment
  - support for N-grams and finite state grammars
  - Includes N-gram generation tools for large datasets
  - N-best and lattice generation/manipulation
- (C)MLLR speaker/channel adaptation & adaptive training (SAT)
- From V3.4
  - Large vocabulary decoder HDecode: separate license
  - Discriminative training tools, MMI and MPE HMMIRest

HTK Architecture

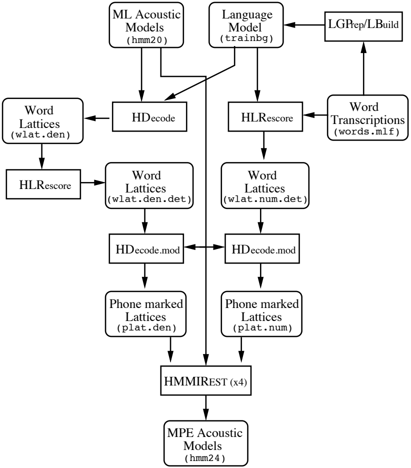
- HTK is structured as
  - a set of libraries
  - a set of tools
- Tools have uniform interface



- Text-based model formats are used where possible (with binary versions for efficiency)
- Built to scale to large data-sets
  - data-parallel operations for training (HERest/HMMIRest)
  - unsegmented data files (e.g. broadcasts)
  - multiple lattices/labels in one file

Typical HTK MPE HMM Build Process

- Start from maximum likelihood trained triphone HMMs
- Generate "numerator" (correct transcription) and "denominator" (recogniser with weak language model) lattices
- "phone mark" lattices
- Run MPE training with HMMIRest (extended Baum-Welch algorithm)

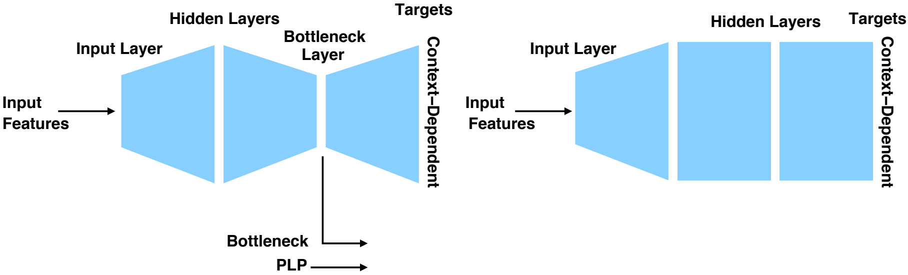


Deep Neural Network Acoustic Models

- Recently a resurgence in the use of Neural Network models for acoustic modelling
- Deep Neural Networks (DNNs) are Multi-Layer Perceptrons with many hidden layers (Sigmoid or ReLU units)
- Standard DNNs
  - Model posterior probability of standard HMM context-dependent phone states (1-of-k encoding, softmax)
  - Frame based criterion optimises the cross-entropy criterion
  - Stochastic gradient descent (SGD) via error back propagation
  - Initialised using generative model (RBM pre-training) or EBP (discriminative pre-training)
  - State-of-the-art DNNs also include sequence training via the MPE/MMI criteria computed over lattices
- HMM-DNN Hybrid models use the probabilities directly
- Tandem models use the DNN to produce features (possibly combined with e.g. PLP) and modelled by a GMM as usual.

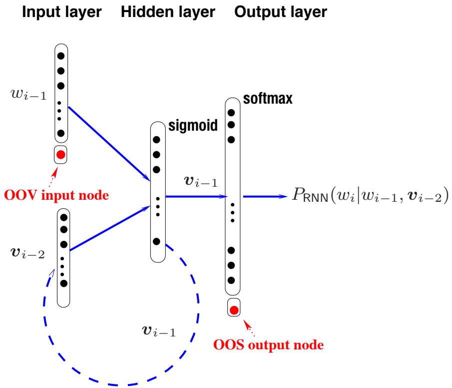
Tandem and Hybrid Approaches

- “Tandem (left): Generate features at bottleneck for HMM-GMMs
- “Hybrid (right) : replace GMMs with DNN scaled likelihoods
- Both give large reductions in WER (e.g. 25%) & are complementary
- Define state-of-the-art: used in all best research systems and some commercial systems



Recurrent Neural Network Language Models

- Predict probability of next word given current word & history (in recurrent units)
- SGD by back-propagation through time
- Continuous space vs discrete space for N-grams
- Significant reductions in WER
- Expensive to train (& expensive to decode due to multiple histories)



- Apply in combination with N-grams (via lattices preferred but computational issues)

Key HTK Attributes

Strong Points in HTK V3.4.1

- Widely used
- Flexible and modular (easy to modify/extend/use)
- Good documentation & examples
- Could build state of the art systems (in 2009 ...)

Issues

- lack of built-in Deep Neural Network support
  - for frame-based training use other tools
  - can't extend to "sequence training" (e.g. MMI/MPE)
- n-gram only lattice rescoring (no recurrent neural network LMs)
- only relatively small-scale recipes

HTK V3.5 aims to address issues while retaining strong points!

Overview of HTK-ANN Extensions

- Design Principles
- Implementation Details
  - Generic ANN Support
  - ANN Training
  - Data Cache
  - Other Features
- Example ANN definition
- New Modules and Tools
- Build Procedure
- A Summary of HTK-ANN

Design Principles

- The design should be as generic as possible.
  - Flexible input feature configurations.
  - Flexible ANN model architectures.
  - ... but don't sacrifice efficiency.
- Maintain compatibility with as many existing functions in HTK as possible.
- HTK-ANN should be compatible with existing functions.
  - To minimise the effort to reuse previous source code and tools.
  - To simplify the transfer of many technologies.
- HTK-ANN should be kept "research friendly".

Generic ANN Support

- In HTK-ANN, ANNs have layered structures.
  - An HMM set can have any number of ANNs.
  - Each ANN can have any number of layers.
- An ANN layer has
  - Parameters: weights, biases, activation function parameters
  - An input vector: defined by a **feature mixture** structure
- A feature mixture has any number of **feature element**s
- A feature element defines a fragment of the input vector by
  - Source: acoustic features, augmented features (e.g. ivectors), output of some layer.
  - A context shift set: integers indicated the time difference.

Generic ANN Support (cont'd)

- In HTK-ANN, ANN structures can be any directed cyclic graph.
- Since only standard EBP is included at present, HTK-ANN can train non-recurrent ANNs properly (directed acyclic graph).

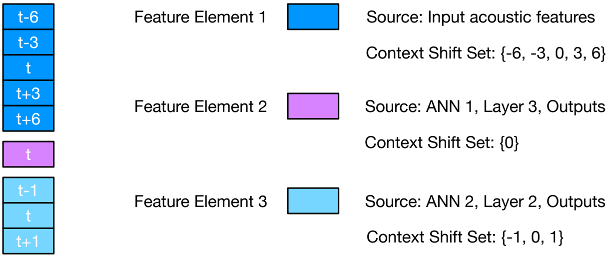


Figure: An example of a feature mixture.

ANN Training

- HTK-ANN supports different training criteria
  - Frame-level: Cross Entropy (CE), Minimum Mean Squared Error (MMSE)
  - Sequence-level: Maximum Mutual Information (MMI), Minimum Phone/Word Error (MPE/MWE)
- ANN model training labels can come from
  - Frame-to-label alignment: for CE and MMSE criteria
  - Feature files: for autoencoders
  - Lattice files: for MMI, MPE, and MWE criteria
- Gradients for SGD can be modified with momentum, gradient clipping, weight decay, and max norm.
- Supported learning rate schedulers include List, Exponential Decay, AdaGrad, and a modified NewBob.

Data Cache

- HTK-ANN has three types of data shuffling
  - Frame based shuffling: CE/MMSE for DNN, (unfolded) RNN
  - Utterance based shuffling: MMI, MPE, and MWE training
  - Batch of utterance level shuffling: RNN, ASGD

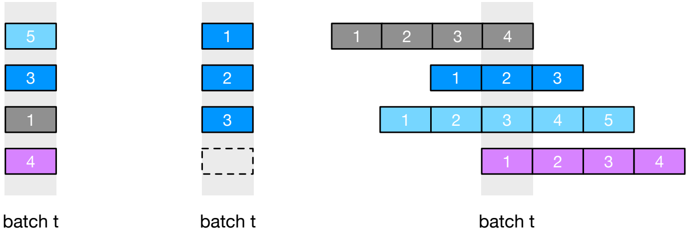


Figure: Examples of different types of data shuffling.

ANN Model Definition

```
~n "n_1"
<BEGINANN>
  <ANNKIND> "FNN"
  <NUMLAYERS> 3
    <LAYER> 2
      <OPERAND> "SUM"
      <ACTIVATION> "SIGMOID"
      <INPUTFEA>
        <NUMFEAS> 1 351
          <FEATURE> 1 39
            <SOURCE> <STREAM> 1
            <EXPAND> 9
              -4 -3 -2 -1 0 1 2 3 4
      <WEIGHT> 1000 351
        . . .
        . . .
      <BIAS> 1000
        . . .
    <LAYER> 3
      <OPERAND> "SUM"
      <ACTIVATION> "SOFTMAX"
      <WEIGHT> 6000 1000
        . . .
        . . .
      <BIAS> 6000
        . . .

<ENDANN>
```

- Example shows a 3-layer feed forward ANN with
  - a sigmoid hidden activation function
  - softmax output activation function.
- Structure is $351 \times 1000 \times 6000$.
- Input feature mixture of the second layer is omitted as it is just the output of the last layer.
- Also state definition to convert DNN-HMM posteriors to pseudo log-likelihoods

New Modules and Tools

- Extended modules:
  HFBLat, HMath, HModel, HParm, HRec, HLVRec
- New modules
  - HANNet: ANN structures & core algorithms
  - HCUDA: CUDA based math kernel functions
  - HNCache: Data cache for data random access
- Extended tools:
  HDecode, HDecode.mod, HHEd, HVite
- New tools
  - HNForward: ANN evaluation & output generation
  - HNTrainSGD: SGD based ANN training

Other Features

- Math Kernels: CPU, Intel MKL, and CUDA based new kernels for ANNs
- Input Transforms: compatible with HTK SI/SD input transforms (e.g. HLDA, CMLLR)
- Speaker Adaptation: an ANN parameter unit online replacement (e.g. parameterised activation function adaptation)
- Model Edit
  - Insert/Remove/Initialise an ANN layer
  - Add/Delete a feature element to a feature mixture
  - Associate an ANN model to HMMs
- Decoders
  - HVite: tandem/hybrid system decoding/alignment/model marking
  - HDecode: tandem/hybrid system LVCSR decoding
  - HDecode.mod: tandem/hybrid system model marking
  - A Joint decoder: log-linear combination of systems (same decision tree, not in initial release)

Building Hybrid SI Systems

- Building CE based SI CD-DNN-HMMs:
  - Produce desired tied state GMM-HMMs by decision tree tying (HHEd)
  - Generate ANN-HMMs by replacing GMMs with an ANN (HHEd)
  - Generate frame-to-state labels with a pre-trained system (HVite)
  - Train ANN-HMMs based on CE (HNTrainSGD)
- Building CD-DNN-HMMs with MPE sequence training
  - Generate numerator/denominator lattices (HLRescore & HDecode)
  - Phone mark numerator/denominator lattices (HVite or HDecode.mod)
  - Perform MPE training (HNTrainSGD)
- Note similarities to standard HMM build process for MPE training.

ANN Front-ends for GMM-HMMs

- ANNs can be used as GMM-HMM front-ends by using a feature mixture to define the composition of the GMM-HMM input vector.
- HTK can accommodate a tandem SAT (CMLLR) system as a single system
  - Mean and variance normalisations are treated as activation functions.
  - SD parameters are replaceable according to speaker ids.

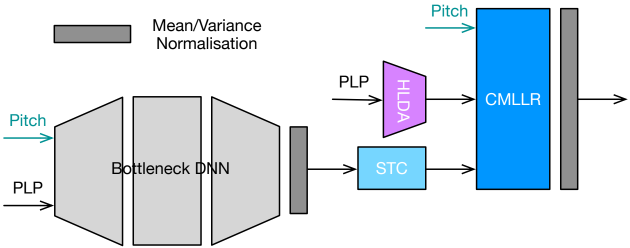


Figure: A composite ANN as a Tandem SAT system front-end.

BOLT Mandarin Chinese System Results

- 300h Mandarin conversational telephone transcription task, dev14 test set
- Hybrid DNN structure: $504 \times 2000^4 \times 1000 \times 12000$
- Tandem DNN structure: $504 \times 2000^4 \times 1000 \times 26 \times 12000$

System	Criterion	%CER
Hybrid SI	CE	34.5
Hybrid SI	MPE	31.6
Tandem SAT	MPE	33.2
Hybrid SI $\otimes$ Tandem SAT	MPE	31.0

- $\otimes$ is joint decoding of weighted combination hybrid and tandem models (combined at frame score level).
- hybrid with sequence training reduces error rate by 8% relative
- Joint decoding not available in initial release of HTK V3.5

HTK-ANN Summary

- HTK-ANN integrates native support of ANNs into HTK.
- HTK based GMM technologies can be directly applied to ANN-based systems.
- HTK-ANN can train DNNs with very flexible configurations
  - Topologies equivalent to DAG
  - Different activation functions
  - Various input features
  - Stochastic gradient descent optimisation
  - Frame-level and sequence-level training criteria
- Use in either tandem or hybrid configurations
- Efficient due to availability of CUDA GPU kernels (as well as CPU kernels)
- Experiments on 300h CTS task showed HTK can generate standard state-of-the-art tandem and hybrid systems.

HTK Language Model Interface

- Allows efficient lattice rescoring using various language models:
  - *n*-gram LMs, and recurrent neural network language models (RNNLMs);
  - linear interpolation between the two to draw strengths from both.
- Supports multiple forms of RNNLMs:
  - full output, and class based output RNNLMs for improved efficiency;
  - output layer short list and out-of-shortlist (OOS) node covering full vocab.
- Efficient RNNLM lattice rescoring approaches (ICASSP2014) provided:
  - using n-gram style history clustering;
  - or more flexible recurrent hidden vector distance based history clustering.
- Produces RNNLM rescored HTK format lattices:
  - fully integrated with other HTK lattice operations;
  - to be used for downstream applications.

HTK Language Model Interface (cont)

- General and extendable language model interface:
  - modularized design allows many more LM types to be supported in future
  - including class based *n*-gram LMs and feedforward NNLMs.
- Separate RNNLM training software also to be released in future:
  - to produce RNNLMs fully compatible in format with HTK V3.5;
  - also supports various modelling features to significantly improve RNNLM efficiency during both training and evaluation time.
  - bunch mode GPU training; full/class output RNN LMs;
  - NCE training and variance regularised training

Example of LM Interpolation

4-gram LM

```
\data\
ngram 1=58286
ngram 2=1322619
ngram 3=5768465
ngram 4=11151893

\1-grams:
-2.628496 !!UNK -0.7490927
-1.763285 </s>
-99 <s>-2.071745
-2.334805 A -0.9217603
... ...
```

RNNLM

```
!RNN
./RNNLM
./RNNLM.input.wlist.index
./RNNLM.output.wlist.index
31857
20001
```

Linear interpolation between 4-gram LM and RNNLM

```
!INTERPOLATE
2
!NGRAM 0.5 ./4g.txt
!NGRAM 0.5 ./rnnlm.txt
```

Key Features of HTK V3.5

- Support ANNs, maintaining compatibility with most existing functions.
  - Flexible input feature configurations
  - ANN structures can be any directed acyclic graph
  - Stochastic gradient descent supporting frame/sequence training
  - CPU/GPU math kernels for ANNs
  - Decoders extended to support tandem/hybrid systems, system combination
- Support for decoding RNN language models
  - Lattice rescoring using RNNLMs
  - Class / Full word outputs, interpolation with n-grams
- 64-bit compatible throughout
- Bug fixes
- Updated documentation and examples

Recent Experiments: MGB Challenge Systems

- Challenge for ASRU'15 (`http://www.mgb-challenge.org/`) to transcribe etc, general BBC programme output
- Some early development numbers (not our final systems ...)
- 700h training set from distributed data, manual segmentation, 64k vocab

AM	LM	%WER
GMM-HMM ML HLDA	4-gram	42.7
GMM-HMM MPE		40.7
Tandem SI MPE		27.0
Hybrid CE		28.4
Hybrid MPE		25.9
Hybrid MPE	RNNLM	25.0
Hybrid MPE	RNNLM + LDA	24.7

- Note included a line on RNNLM adaptation via LDA (see Interspeech 2015 paper)

Summary & Plans

- New version of HTK with significantly upgraded capabilities
- HTK V3.5 can produce state-of-the-art performance on large tasks (BOLT/MGB challenge)
- Expect to release a beta version for Interspeech 2015

Plan to continue to further extend HTK in future

- further NN models such as convolutional neural networks (CNNs)
- improved/alternative ANN estimation procedures
- other tools such as confusion networks (combination)
- complete recipe for large ASR task
- release tools for RNNLM training (can be used by HTK but not part of it)