Memory-Efficient Heterogeneous Speech Recognition Hybrid in GPU-Equipped Mobile Devices

Alexei V. Ivanov,
CTO, Verbumware Inc.

GPU Technology Conference,
San Jose, March 17, 2015

Autonomous Speech Recognition With Mobile Devices

Reduce the load on web-servers and the network;

Enable autonomous human-computer spoken interaction even in the absence of the network;

Increase privacy of the customer-device interaction;

Improve accuracy of the recognition by customization of the automated speech recognition system to a specific user.

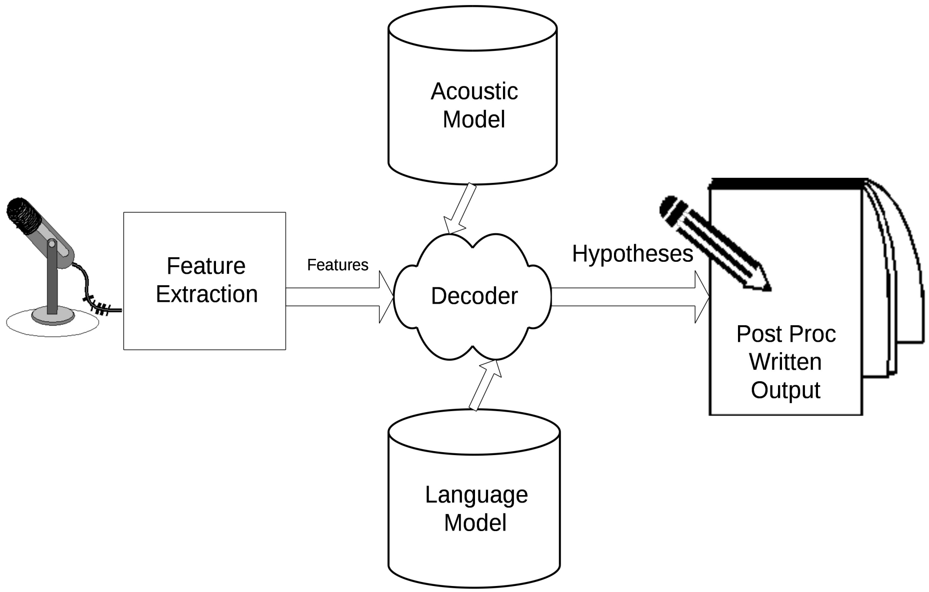
ASR application structure: FE, AM, LM, Decoding

Any ASR consists of:

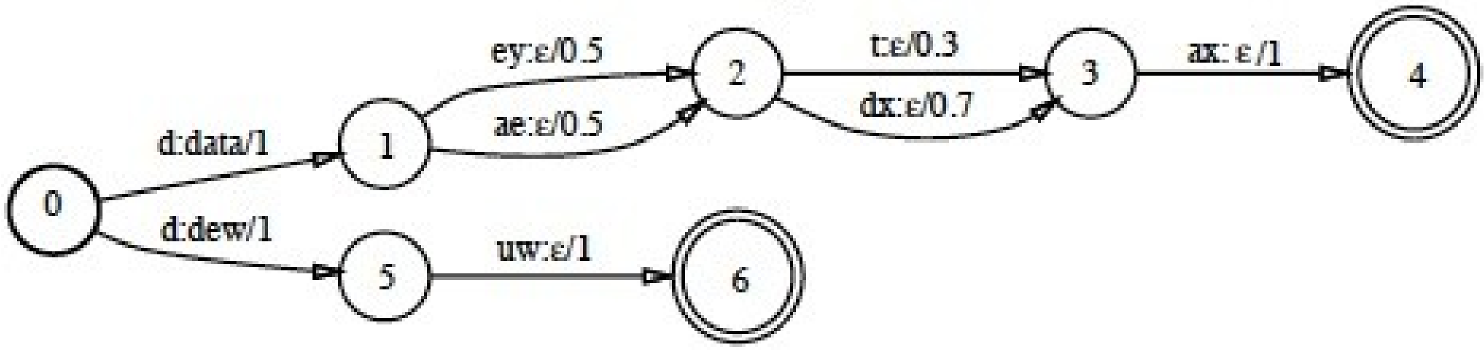
- **Feature Extraction** (FE) that provides of the **input phenomenon objective description**

- Several statistical models, that help to **subjectively interpret that phenomenon** (in relation to the **previous experience**), traditionally:
  - **Acoustic** Model (AM)
  - **Language** Model (LM)

- **Decoder** - A module that implements integration of objective measurements with knowledge stored in models to generate **hypotheses on interpretation** of the



Decoding with Weighted Finite State Transducers



A random walk through WFST converts strings (input into output) & accumulates a cost

Traditional way of doing WFST-based ASR:
Fuse all knowledge sources into a global network of alternatives

- AM is evaluated on the acoustic evidence (**input label costs** at a given time)
- LM is completely fused into the search graph (**costs of traversals** themselves)
- Search for **the single best** solution
- **PROBLEM:** The resulting network is too sparse to be handled efficiently by computing devices
- Even **more true for GPUs** than CPUs !

WFST Operations

Composition (∘) – **elimination of the intermediate alphabet** of two successively applied WFSTs

Determinization – each distinct sequence of tokens, resulting from traversing a graph, has a unique path associated with it;

Minimization – ensuring that graph does not contain equivalent states;

Epsilon removal – removing transitions, associated with empty observation symbol.

- **Why we need it?**
  - **Efficiency** (obviously, DFA traversal *has* ***the least computation cost***, ***minimal necessary set of stacks*** *for intermediate results*)
  - Surprisingly, **NFAs are less powerful**

GPU-based Baseline System Complexity

$$\min(det(\mathbf{H} \circ \min(det(\mathbf{C} \circ \min(det(\mathbf{L} \circ \mathbf{G}))))))$$

- EXAMPLE – WSJ 20K standard tri-gram LM — Arcs — Nodes

  G - “grammar” - N-gram Language Model

  L - “lexicon” - pronunciation rules;

  C – contextual phone loop;

  H – phone-internal topology;

	Arcs	Nodes
$\min(\det(L \circ G))$	16.0M	6.2M
$\min(\det(H \circ \min(\det(C \circ \min(\det(L \circ G))))))$	100M	35M
$\min(\det(H \circ C))$	150K	25K

GPU-based Baseline System Performance

TASKS\LMs	BCB05ONP	BCB05CNP	BCB05ONP	BCB05CNP	TCB20ONP	BCB05ONP	BCB05CNP	TCB20ONP
NOV'92 (5K) WER	**5.66%**	2.30%	**5.66%**	2.30%	1.85%	5.77%	2.19%	1.63%
NOV'92 (5K) xRT	0.4647	0.4683	**0.0327**	**0.0328**	**0.0364**	0.1967	0.1900	0.2203
NOV'93 WER	18.22%	**19.99%**	18.22%	**19.99%**	7.77%	18.13%	20.19%	7.63%
NOV'93 xRT	0.4658	0.4651	**0.0332**	**0.0331**	**0.0375**	0.2309	0.2382	0.2562
Power/RTchan.	**~3.6W**		**~9 W**			from 75**W** (1 ch) to **15W** (full load)		
Hardware	**Tegra K1 (32 bit)**		**GeForce GTX TITAN BLACK**			i7-4930K @3.40GHz		
	GPU-enabled					Nnet-latgen-faster		

- **Accuracy** of our GPU-enabled engine **is approximately equal** to that of the reference implementation. There is a small fluctuation of the actual WER (mainly) due to the differences in arithmetic implementation.

- For the single-channel recognition **the TITAN-enabled engine is significantly (~7 times) faster** than the reference. This is important in tasks like media-mining for specific a priori unknown events.

- Our implementation of the speech recognition in the **mobile** device (Tegra K1) enables **twice faster than real-time processing** without any degradation of accuracy.

- Our GPU-enabled engine allows **unprecedented energy efficiency** of speech recognition. The value of 15W per RT channel for i7-4930K was estimated while the CPU was fully loaded with 12 concurrent recognition jobs. This configuration is the most power efficient manner of CPU utilization.

GPU-based Baseline System Challenges

Completely composed non-trivial WFSTs

$$\min(det(\mathbf{H} \circ \min(det(\mathbf{C} \circ \min(det(\mathbf{L} \circ \mathbf{G})))$$

Consume **large amount of memory** ~ 6Gb

(100M arcs 35M states for WSJ 3-gram LM)

That is typically **far beyond what is available** in mobile devices

(~2-4Gb of RAM total Tegra K1)

GPU-based Phonetic Decoding

Phonetic decoding phase, where a sequence of acoustic observations is interpreted in terms of a sequence of phonetic symbols is

- Performed with a “dense” H∘C graph => Fast on GPU
- Equivalent to
  - HC composition with a fully-connected between time instances AM observation DAG (A) resulting in A∘H∘C graph (DAG)
  - Pruning into a “history tree”
  - Backtracking for the best hypothesis

GPU-based Phonetic Lattice Generation

Instead of backtracking for the best hypothesis lets merge all arcs in the history tree that do not generate meaningful output symbols

- Forward-path pruning (faster) ~ 20% computational overhead
- Backward-path pruning (more memory efficient) ~ 50% computational overhead

Result = Phonetic Lattice, a Compact Way to Store Alternatives (Report multiple good instead of the only best)

("Good" in oracle WER sense) lattice is ~7.5K arcs/sec (~ 500 kbit/s)

It is not entirely redundant compared to the original audio representation (256 kbit/s) as it contains some information AM about AM

Principle of Sequential Decoding

It is possible to make a **run-time dynamic composition** of sub-graphs

Lattice ~7.5K arcs/sec (pruned & epsilon-removed $A \circ H \circ C$)

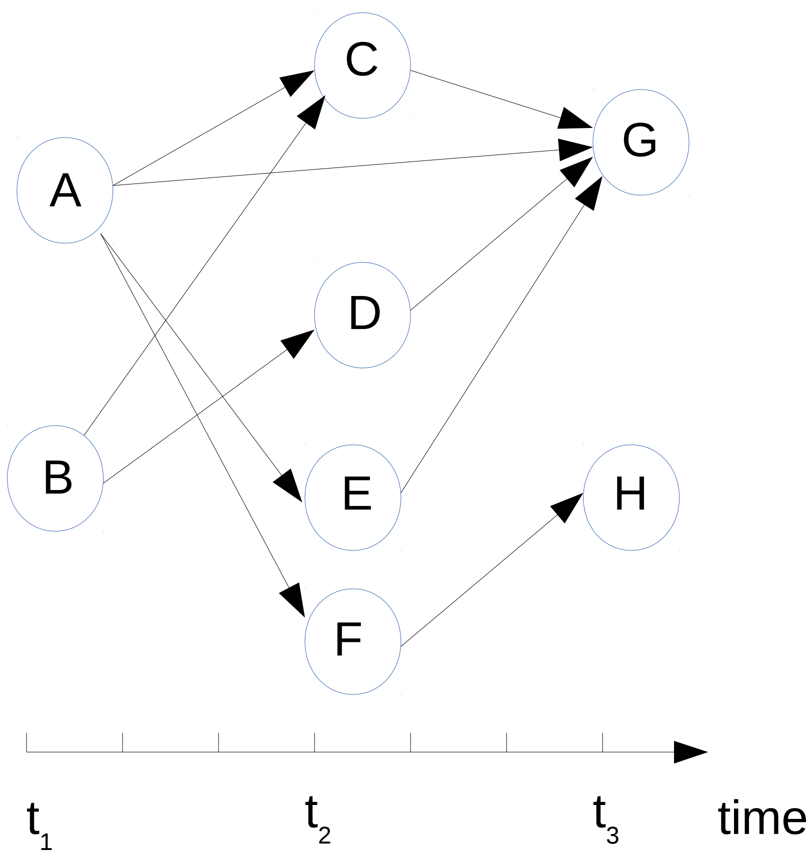
$L \circ G$ 16M arcs

This task is easier than propagating 100 times/sec through the HCLG graph with 100M arcs

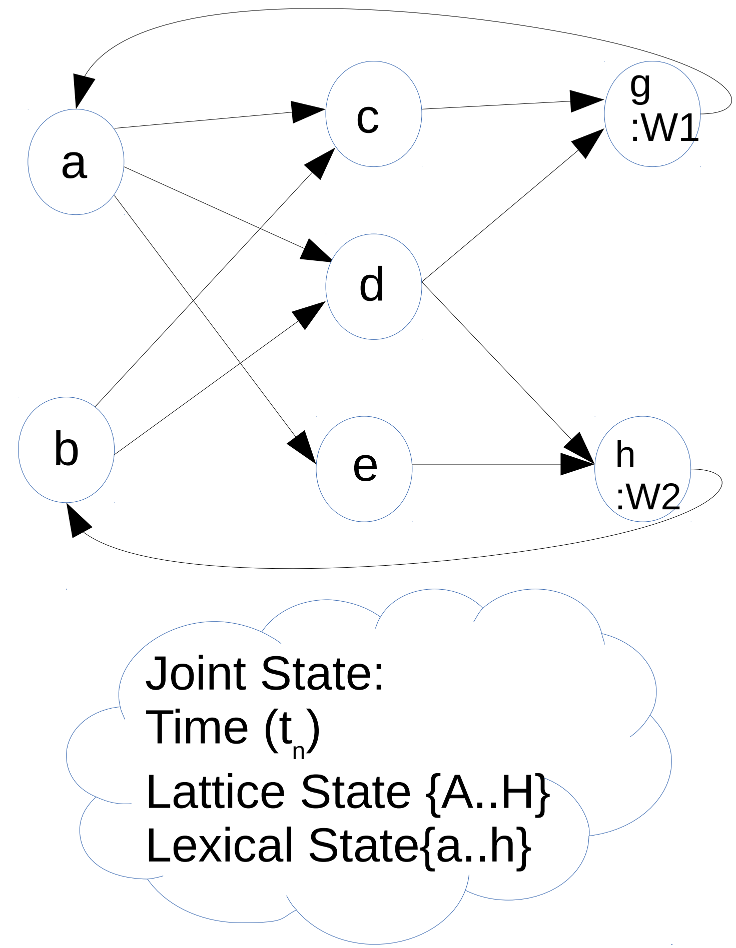
CPU-based Lexical Decoding

- Lexical decoding phase
  - A sequence of phonetic symbols is interpreted as a sequence of words
- Lattice traversal is no longer a strictly time-synchronous process
  - Hash & stack are required for the implementation
- LG graph is rather sparse

CPU-based Lexical Decoding



Lattice (DAG)



Lexical graph (L ∘ G)

GPU-CPU Hybrid Benchmarks

TCB20ONP on TK1		TK1 GPU	TK1 CPU
TASKS		NOV'92	NOV'93
PHONETIC LATTICE	**GPU xRT**	0.5128	0.5194
LEXICAL DECODING	**CPU xRT**	0.3820	0.3917
LEXICAL DECODING	**CPU WER**	1.85%	7.77%
COMPLETE RECOGNITION	**Total xRT**	**0.8948**	**0.9111**

Lexical Decoding step **follows** Phonetic Lattice Extraction

– back-track lattice generation

Total processing is still **faster than natural speech pace**

Conclusions

Our research confirms the possibility to implement complex recognition systems in devices with small footprint

Properties of decoding graphs dictate GPU-based phonetic decoding stage complemented with CPU-based lexical decoding

Multipath recognition is advantageous also from the multi criteria optimization point of view

Q & A

Do you have any questions?

www.verbumware.net info@verbumware.net

alexei_v_ivanov@ieee.org