Language, Dialect, and Speaker Recognition Using Gaussian Mixture Models on the Cell Processor.

Nicolas Malyska, Sanjeev Mohindra, Douglas Reynolds, and Jeremy Kepner.
MIT Lincoln Laboratory
244 Wood Street, Lexington, MA 02420, USA
{nmalyska,smohindra,dar,kepner}@ll.mit.edu

Introduction

Automatic recognition systems are commonly used in speech processing to classify observed utterances by the speaker's identity, dialect, and language. These problems often require high processing throughput, especially in applications involving multiple concurrent incoming speech streams, such as in datacenter-level processing.

Recent advances in processor technology allow multiple processors to reside within the same chip, allowing high performance per watt. Currently the Cell Broadband Engine has the leading performance-per-watt specifications in its class. Each Cell processor consists of a PowerPC Processing Element (PPE) working together with eight Synergistic Processing Elements (SPE). The SPEs have 256KB of memory (local store), which is used for storing both program and data.

This paper addresses the implementation of language, dialect, and speaker recognition on the Cell architecture. Classically, the problem of performing speech-domain recognition has been approached as embarrassingly parallel, with each utterance being processed in parallel to the others. As we will discuss, efficient processing on the Cell requires a different approach, whereby computation and data for each utterance are subdivided to be handled by separate processors. We present a computational model for automatic recognition on the Cell processor that takes advantage of its architecture, while mitigating its limitations. Using the proposed design, we predict a system able to concurrently score over 220 real-time speech streams on a single Cell.

Recognition for Speech Applications Using Gaussian Mixture Models

Recognition systems in speech technology allow us to identify the language, dialect, or speaker in a particular audio recording based on a set of known previous recordings. A recognition system consists of two primary stages, front-end processing and pattern-recognition. The goal of front-end processing is to transform incoming audio into a set of parameters, called *features* that capture the important characteristics of the signal. The pattern-recognition stage then takes these features and uses them to make a decision regarding the identity of the speaker, their language, or their dialect. In this paper, we shall focus on parallelizing the pattern-recognition stage of the process as this is where much of the time is spent in current systems.

This work is sponsored by the United States Air Force under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Gaussian mixture models (GMMs) are a state-of-the-art approach to pattern recognition that are popular in language, dialect, and speaker recognition systems, and that are also fundamental for speech recognition tasks. This technique models the probability-density function governing observed features and uses this distribution to classify incoming speech. Gaussian-mixture models represent the feature space as a set of Gaussian states, each with three parameters, *mean*, *covariance*, and *weight*. Multiple individual Gaussians combine to form the full density function model for a particular class.

Given the set of observed feature vectors, X, for an utterance, the recognition problem can be summarized as finding the ratio of the probability that X was generated by the *target* system with model λ_C to the probability that X was generated by the *background* model, $\lambda_{\bar{C}}$, which represents the distribution of features expected over all speakers. The logarithm of this ratio, the *log-likelihood ratio score*, becomes the criteria for deciding whether a given speaker, language, or dialect is in fact the target:

$$\Lambda(X) = \log[p(X \mid \lambda_C)] - \log[p(X \mid \lambda_{\bar{C}})]$$

$$\Lambda(X) \geq \text{threshold}, \quad \text{accept}$$
$$\Lambda(X) < \text{threshold}, \quad \text{reject}.$$

The probability, $p(X \mid \lambda)$, of observing a set of feature vectors given model λ is found using:

$$p(X \mid \lambda) = \tfrac{1}{K} \sum_{1}^{K} \left(\log \sum_{i=1}^{M} \exp\left(C_i - \tfrac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i) \right) \right),$$

where Σ_i is the (diagonal) covariance matrix, μ_i is the mean, and C_i is a constant derived from the weight and covariance matrix of the i^{th} Gaussian state. M denotes the number of states in a particular model and K is the number of observed features in a particular utterance. The calculation of $p(X \mid \lambda)$ reduces to a dot product, with a table lookup used to compute the function $f(u) = \log \sum_{i=1}^{M} \exp(u)$.

Parallel Implementation of the GMM

For programming the Cell architecture, it is important to choose an algorithm that minimizes the amount of data transfer between the PPE and the SPE. The background model is about 630K, far exceeding the size of the 256KB SPE local store. Thus, before GMM scoring begins, the background model must first be split across the SPEs (Step 1) and kept there throughout the scoring procedure.

Each feature vector is then broadcast to all SPEs (Step 2). The SPEs score the feature vector against their portion of the background model (Step 3), and the resulting scores are aggregated on the PPE (Step 4). The data flow is shown in Figure 1.

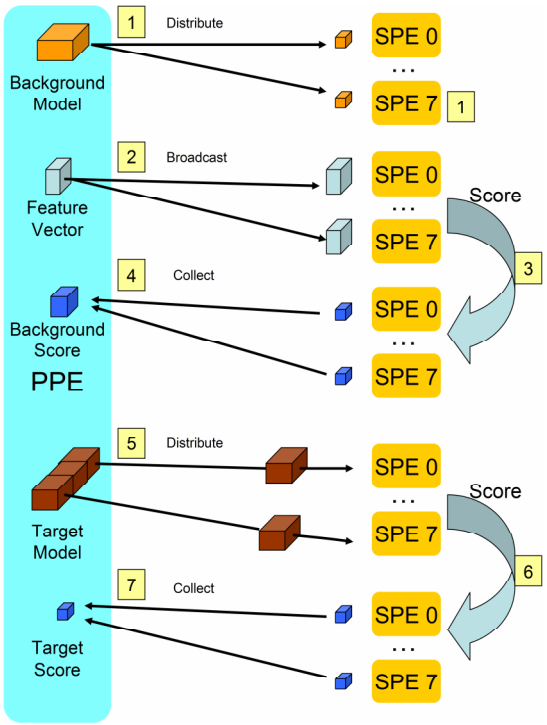

Figure 1: Parallel implementation of GMM. Numbers refer to corresponding step numbers in the text.

Modern methods do not require scoring against all states of each target model. Instead, the results of scoring a feature vector against the background model are used to select only a limited number (typically 5 out of 2048) of the target states [1]. Whereas the full target models would not fit in the SPE memory, several selected states can fit in the SPE memory and are sent to the SPE and scored at the same time (Step 5). The results of scoring each target on the SPEs (Step 6) are aggregated on the PPE, and the log-likelihood ratio scores are computed (Step 7).

Results

We have simulated the parallel-computing model in order to evaluate its performance characteristics. The metric that we use for performance is the number of concurrent real-time speech streams supported, when operating at the standard rate of 100 frames per second. In the model we define computational efficiency as the ratio of FLOPS to the maximum theoretical FLOPS for the Cell. The data transfer efficiency is defined to be the ratio of the observed data transfer rate to the peak theoretical bandwidth for the Cell Element Interconnect Bus. Based on empirical observations, a computational efficiency of 5 percent and a data-transfer efficiency of 12 percent have been used for the model [2]. We assume that communication and computation operations are overlapped using double buffering where possible and C extensions are used to take advantage of the SIMD capabilities of the SPEs. The default parameters for the system are 2048 38-dimensional Gaussians per model for each of 10 target models. These values are representative of the standard language-recognition systems in use. With these parameters, we are able to achieve a simulated throughput of 222 concurrent real-time speech streams on a single 8-SPE Cell.

As depicted in Figure 2, three parameters of the model—computational efficiency, data-transfer efficiency, and the number of targets—are swept. As shown, each of these factors has the ability to alter performance dramatically. The sweep of computational efficiency shows that this aspect dominates performance, yielding increased performance with increased efficiency and only minor saturation. Sweeping data-transfer efficiency shows that while it is an important parameter, with a similar effect to computational efficiency up to about 5 percent, its importance levels off after this point. Even high data-transfer efficiency cannot yield increases in performance comparable to high computational efficiency.

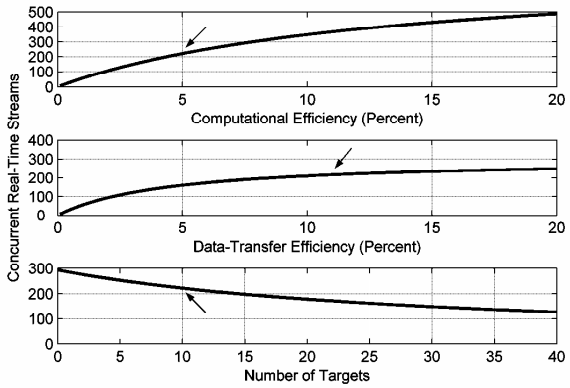

Figure 2: Sweep of three parallel-scoring-model parameters. Default system operating points are indicated with arrows.

The final sweep shows the effect of varying the number of target models that we score against. As can be seen, increasing the number of targets increases the computational load, reducing the number of possible concurrent streams. This is because each target model must be transferred to the SPEs, incurring additional data-transfer time, and each feature vector must be compared against each target model, incurring additional operations.

Conclusions and Future Work

We have presented a model for parallel language, dialect, and speaker recognition on the Cell processor. Our future work includes implementing the algorithms in order to validate our model and comparing our system against other state-of-the art serial and parallel approaches to this problem. The algorithm will become part of the PVTOL [3] library.

References

[1] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn. *Speaker Verification Using Adapted Gaussian Mixture Models*, Digital Signal Processing, vol. 10, pp. 19-41, 2004.

[2] J. Geraci and S. Raghunathan. *High Performance Simulations of Electrochemical Models on the Cell Broadband Engine*, HPEC Workshop, 2007, Lexington, MA.

[3] H. Kim, N. Bliss, R. Haney, J. Kepner, M. Marzilli, S. Mohindra, S. Sacco, G. Schrader and E. Rutledge. *PVTOL: A High-Level Signal Processing Library for Multicore Processors*, HPEC Workshop, 2007, Lexington, MA.