Chapter 5

Learning and Speech Recognition

5.1 Automatic Template Learning for Template-Based Correlation

Up to this point, I have not dwelled on the method of obtaining a valid template to use for the class example in transient classification. The acoustic transient processor is a purely feed-forward system which has no local hardware for on-chip (or on-board) learning. Speed is of utmost importance for classification once the templates are known, and so the hardware has been developed precisely for this purpose. Training can be performed off-line and not in real-time, as long as we can assume that a classification system does not need to reconfigure or retrain itself during operation (which may not be a completely valid assumption for speech, but should be true for simple transient applications). This does not mean that training is or should be entirely separate from the classifier hardware. In the preceding chapter, I have alluded to "chip in the loop" learning. The method of "chip in the loop" learning assumes that even the best feed-forward network has nonlinearities and mismatch, and that the best model of such a system is the system itself. To train the classifier, the training system sets all the weights (template values), presents the training data to the classifier, runs the classifier, and captures the output. In an iterative learning scheme, it adapts the weights according to some learning rule, then performs the learning cycle again.

The correlation algorithm works under the assumption that a template should be a "prototype" of its class: it should have all the characteristics of an example of its class. This could be interpreted in several ways. One interpretation is that the template is the average value of many examples of the class, aligned by a segmentation procedure. Another interpretation is that the template contains features which correlate positively with features of all examples of its own class, but may be missing features which would correlate positively with examples of a different class, thus

decreasing the likelihood of a false identification of the second class. Thus, the template may not look anything like the decomposed class examples against which it is correlated, but produces an output which is much larger than that for any other class input. Class averaging requires the least amount of training, and can be done with only the frontend filterbank system as part of the training system. Once the idea of simple class averages is abandoned and a more robust solution sought on the basis of standard pattern recognition, a number of possibilities arise and must be investigated for their applicability to the transient classification problem. This chapter provides an overview of what training methods are available and how they may be applied to the acoustic transient processor.

5.2 Average-Value Templates

The first interpretation of the form of the template, that of class average, is the one that I have been assuming throughout the text to this point. Average-value template learning computes the baseline template p_z for class z as

$$p_z(n,m) = \frac{1}{K}\sum_{k=1}^{K} y_{z,k}(t_0 - n, m) \qquad \forall n, m \tag{5.1}$$

where the training set contains K examples of class z. The examples are distinct and their boundaries known, and they are aligned to some time t_0 relative to the start of the event. The values $y_{z,k}$ are the response of the frontend filterbank to the k-th example of class z. This is at least a simple method to implement. The main difficulty in implementation arises from the need to accurately segment each example of the training set used to generate the template. The average of examples which are not properly aligned quickly becomes flat and featureless. Unfortunately, all acoustic signals have some temporal variation, and so the average value will always become increasingly flat and featureless away from the point of alignment.

To give a concrete example: The sound of a door closing (one of the transient classes in our recorded set) consists of two separate events: The sound of the door latch striking the metal plate, followed by the sound of the door itself bumping against the frame. The result of the two sounds occurring in sequence is the familiar "ker-chunk" of a closing door. Either of the two sounds, if separated from the other, fail to sound like a door. Clearly the sound that we recognize as that of a door closing is as much dependent on the juxtaposition of these two events as it is on the events themselves. On the other hand, the timing between the two events depends on the radial velocity of the door, which may vary considerably from one event to the next. Suppose the recorded transient examples are in every way exact except for the distance between the door latch and door

frame events (*i.e.*, the speed at which the door is closed). Assume that the recorded examples are perfectly aligned on the door latch event. When they are all averaged together and copied to the template, the door latch event will remain distinct while the door frame event will be spread out into a Gaussian shape describing the probability of timing of the door frame event relative to the door latch event rather than describing anything about the door frame event itself. This does not invalidate the average-template method. But it does show how template averaging can lose valuable information about the transient class.

In the case of the door it should be obvious that the direct template method is not sophisticated enough to capture all the nuances of a closing door. Another way to put this is that a closing door is not strictly an acoustic transient, but is a composite event which displays some of the properties of complicated acoustic events such as speech. The closing door, by the way, is not an isolated example. Many such composite events exist in nature. A piano key is another example of a composite event: The sound of one part of the wooden hammer mechanism striking another occurs so close to the sound emanating from the vibrating strings that it is impossible to separate the two sounds perceptually. But the sound of a piano key with the transient removed simply does not sound much like a piano.

The example just described shows that ensemble averaging can produce a probability distribution which, while masking features on a smaller time scale, can contribute important information about the relative timing of events on a larger time scale. This can be repeated on many hierarchical levels, possibly down to the level of zero-crossings (which in the acoustic transient processor we have eliminated by smoothing the signal out to a 2 ms time scale). By creating the acoustic transient processor as a one-level, non-hierarchical architecture, we are forced to find a single time scale which is optimal for transient processing rather than making use of information at all scales. In that sense, what the acoustic transient processor is missing is a wavelet-like interpretation of its input. The situation is illustrative of why a filterbank in and of itself is not a wavelet transformer: In the template correlator, the filterbank outputs are sampled evenly across the spectrum, which forces the system either to oversample the low frequency channels or undersample the high frequency channels. Encoding a wavelet-like representation in the template correlator is a problem without an obvious solution; at very best, it is inconvenient to format into hardware. But the wavelet representation doesn't tell the whole story. In the continuous wavelet representation, larger time scales are captured by filtering lower frequency bands. But if we wish to interpret the behavior of a complex acoustic event over a period of one second, we certainly do not mean that we wish to view the signal spectrum in the 1 Hz range. In this case, the interpretation is more

like a discrete wavelet transform where information is actively pushed from one dilation into the next, but where each filtering function is optimized for a particular transient event at each dilation (timescale). Those functions may be approximated by template correlators to give a high level of flexibility to function selection, but the templates no longer encode any function as simple as an ensemble average. What they encode are functions which extract features at each dilation which, collectively over all dilations, optimally isolate one acoustic class from all other classes.

Clearly, capturing the nuances of composite acoustic events requires that composite sounds be broken down into their fundamental, unchanging parts at each timescale. The template correlation should be affected by temporal shifts as little as possible, and probabilities between the timing of fundamental events probably should be dealt with separately, in a hierarchical manner, by using the outputs of the current timescale as the input to the next timescale. The largest timescale has Z templates (one for each class) and ideally separates the acoustic classes in a robust manner. The outputs of all timescales up to the last are effectively "hidden nodes" whose outputs do not necessarily have an obvious physical meaning. They are like hidden nodes in a neural network, and may be treated in a similar manner.

5.3 Deterministic Methods: Statistical Component Analysis

A major difficulty with finding the fundamental components of composite acoustic events lies in the fact that the fundamental transient parts are not necessarily separable. Training a network to learn to separate classes is made difficult when the training examples cannot be isolated from one another. A related problem is perhaps the most problematic of all for ensemble average training: the system is trained to recognize not only features which are unique to each transient class, but also features which are common to many or all of the transient classes. Unfortunately, these common features can dominate the structure of the sound input. Most transients are characterized by a sharp rise in total energy followed by a slower decay back to zero. Proper training should suppress the similarities and accentuate the differences between transient classes.

Fortunately, the techniques of *Principle Component Analysis* (PCA) and *Independent Component Analysis* (ICA) address some of these problems. They provide a mathematical framework for addressing the issues mentioned above. ICA operates under the assumption that every signal is formed of statistically independent parts which have been put together in various weighted distributions. The ICA algorithm determines what are the independent components of its input (after presentation of many training examples, of course), but its *inverse* is a *detector* of those fea-

tures. Under many situations, one of which includes the template correlator, the independent feature detector is easier to determine than the independent components themselves.

PCA and ICA tend to give qualitatively similar results, and formulating each in the framework of acoustic transient classification encounters the same problems, so I will focus here on only one of the two methods, ICA. Bell and Sejnowski [71, 72] have shown how ICA may be applied as a model of visual and auditory perception.

Independent Component Analysis is a purely linear construct. Is assumes that all inputs are a linear mixture of statistically independent parts, jumbled together in different amounts by some unknown natural process. Because the mixture is linear, the generating process is an unknown matrix of weights determining how much of each independent source is added to the output:

$$x = \mathbf{A}s \tag{5.2}$$

where x are the data presented to the input of the classifier, having been produced by a matrix of independent source vectors $\mathbf{A}$ and a mixing vector s (for simplicity, we assume that s is one-dimensional). Bell and Sejnowski present the problem as on of *blind source separation*; that is, the goal is to find the mixing matrix s and thus determine the original independent sources $\mathbf{A}$. This is useful, for example, to separate one speaker's voice from others in the background of a recording of a crowded room. ICA recovers the sources by applying the inverse of the problem,

$$u = \mathbf{W}x, \tag{5.3}$$

where u is a vector of weights which is a scaled and permuted version of s (the linear system cannot differentiate between different scales and orders of the input vectors, nor does it need to for the blind signal separation problem), and the inverse of $\mathbf{W}$ yields the similarly scaled and permuted version of matrix $\mathbf{A}$:

$$u = \mathbf{W}\mathbf{A}s. \tag{5.4}$$

A classification system poses the same problem in a different way. In an ideal situation, each class of input is independent from the rest. Each class is one independent source, and real examples presented to the system consist of the class independent source (the "prototype") mixed with lesser amounts of other sources (including noise). The classifier itself is $\mathbf{W}$, and given an input set x, it produces an output u, consisting of elements u_i representing the classification result for class i. In the ideal case, u is very nearly a binary vector which is (nearly) one for the element i of u which corresponds to the class to which input x belongs, and (nearly) zero elsewhere. Consequently, Equation (5.4) which forms the core of the ICA algorithm is itself a classifier.

In a more realistic case, transient events are themselves composed of other independent sources. As mentioned in the previous section, many transients have similar structures. Roughly speaking, the similar structures are independent components which are shared between transient classes, and there are other independent components which correspond to structures unique to certain classes. Each class produces a unique output u representing the unique mixture of independent sources which encode that acoustic transient class.

The acoustic transient processor baseline algorithm (4.1) is a linear matrix multiplication and so is directly applicable to ICA. The matrix **W** is the set of templates p, where each template is a one-dimensional vector of values, ignoring the relationships between the two dimensions of time and frequency. From the discussion above, it should be clear that in an ICA framework each template p encodes an independent component of the source, and does not encode a single class discriminator function unless the class happens to be representable by a single independent component (which would be extremely rare in practice). The output c of all the template correlators encodes the instantaneous mixtures of independent components present in the input.

It should be noted that standard ICA theory assumes a square matrix **W**; that is, the assumption is that there are as many independent sources as there are weights in the system (for each template). However, the final weight update rule derived does not require a square matrix **W**. Indeed, logically the size of **W** should correspond exactly to the true number of independent components of the source. When there are more weights than sources, the encoding is redundant but should have the positive effect of increasing system robustness in the presence of noise; the matrix **A** may be recovered if necessary using the matrix pseudo-inverse.

The weight update rule is [72]:

$$\Delta w_{ij} = \mu w_{ij} + 2 \tanh(u_i) \sum_k w_{kj} u_k \tag{5.5}$$

where μ is a learning constant. The weight update rule is similar to Hebbian learning.

Applied directly to the transient test data, this rule does an excellent job in breaking up the input into distinct output patterns. There are several major problems with the application of the ICA technique, however:

1. Equation (5.5) is an *unsupervised learning* rule. Because the true independent components of the input are unknown, knowing the class of the input is unhelpful to the algorithm. The problem of classification learning is really deferred, not solved. ICA can be viewed as a way to warp the input space into a form more amenable to simple classification. When there are

fewer outputs than weights per template, it does so by reducing the dimensionality of the input space and in that way is very similar to Linear Discriminant Analysis (LDA) techniques Initialization is a problem, but generally satisfactory performance results from initializing the templates with the original ensemble-average values.

2. The weight update rule applies only to continuously-valued weights. No ICA weight-update rule has been demonstrated for binary valued weights. This problem occurs for many kinds of learning systems, including the backpropagation algorithm for neural networks. All of these algorithms are related in one way or another to gradient descent, which requires an estimate of the slope of an error surface with respect to the derivatives of the weights. If the weights are binary, their derivatives are discontinuous functions, and an error surface cannot be evaluated. Solutions usually involve using continuous-valued weights but passing them through a sigmoidal function such as a hyperbolic tangent which has a well-defined derivative but can approximate a step function by scaling the x-axis. This tends to force weights to resolve to binary values but leaves open issues such as how to change the sigmoid scaling during training. This solution requires that training be performed on a model of the system rather than the system itself if the system (such as our acoustic transient processor hardware) contains only binary values. Trinary values are, naturally, even more complicated to deal with.

3. The ICA update rule is static in time and is best suited for static applications such as natural scene analysis. It does not adequately deal with statistical correlation in time that arises regularly in a system such as the acoustic transient classifier. It is not sufficiently powerful enough to know that independent components appear and disappear with transitions from one to another. A temporally-based ICA algorithm cannot be blind to its own history and must take such information into account. It may even require sufficient delay so as to observe both past and future inputs around the input which causes the weight update. This would significantly increase the complexity of the system.

5.4 Support Vector Machines

Another promising training method is the use of *Support Vector Machines* [73]. The theory of Support Vector Machines (SVM) provides a statistical framework for pattern classification for which most standard methods such as multilayer neural networks and radial basis functions are

a subset. The main difference is that SVM theory formulates a *deterministic* algorithm for learning based on Lagrangian multipliers. Solutions are iterative (because large nonlinear networks rarely have symbolic solutions) but not haphazard: quadratic programming methods can find the optimal separating hyperplanes for any classification task, given properly tagged training data. As usual, of course, it is necessary to choose the optimal complexity of the feedforward classifier such that it generalizes well and does not over-fit the data, but the ease of quadratic programming allows one to train and test networks of different complexity and determine exactly what system complexity yields the best generalization in a purely deterministic sense.

There are only two failures of standard SVM to meet the needs of the acoustic transient processor, and they are the same ones encountered with Independent Component Analysis: The temporal nature of the problem which gives rise to a high degree of correlation between successive values of the outputs in time, and the binary (or trinary) nature of the template values which plays havoc with learning algorithms due to the presence of discontinuous derivatives. Hopefully, extensions of SVM exist which can include both these cases. Also, SVM training uses a model of the feed-forward classification system and should be reformulated to allow the physical system to be its own model (as with "chip in the loop" learning) to account for properties of the real system which may not be captured in a simple model.

5.5 Heuristic Methods (Unnikrishnan/Hopfield)

Speech signals are quite complicated and contain transients as well as long-term (vowel) events of variable length. Successful recognition of transients can be viewed as a step towards recognition of continuous speech. Another important step is the detection of vowel formants. When separated from the more general task of speech recognition, it is a relatively easy problem, particularly when the vocabulary is small. The task of digit recognition provides a useful small-vocabulary system which has been used often as a benchmark for speech recognition systems. One system optimized to the task of continuous-speech digit recognition is the architecture by Hopfield, Tank, and Unnikrishnan [68, 69]. The architecture of [68] achieves a 99.3% recognition rate on continuous spoken digits when trained for a single speaker, and a similar architecture reported in [69] achieves 97.5% in a speaker-independent task (male speakers only). These figures are for the author's own recorded database, which is similar to the industry standard *TIDIGITS*, a database of both isolated and continuously-spoken digits, using male, female, and child speakers from around the country.

The authors claim that their architecture was designed with analog implementation in

mind, although the project never went further than the software simulation stage. The frontend to the system is quite similar to the ATP frontend (see Section 1.2.1): a filterbank of bandpass filters (in this case, 32) followed by rectification and smoothing, with the channel outputs modified by a center-surround computation (which in our tests of ATP architectures resulted in system behavior almost identical to that using pairwise channel differences).

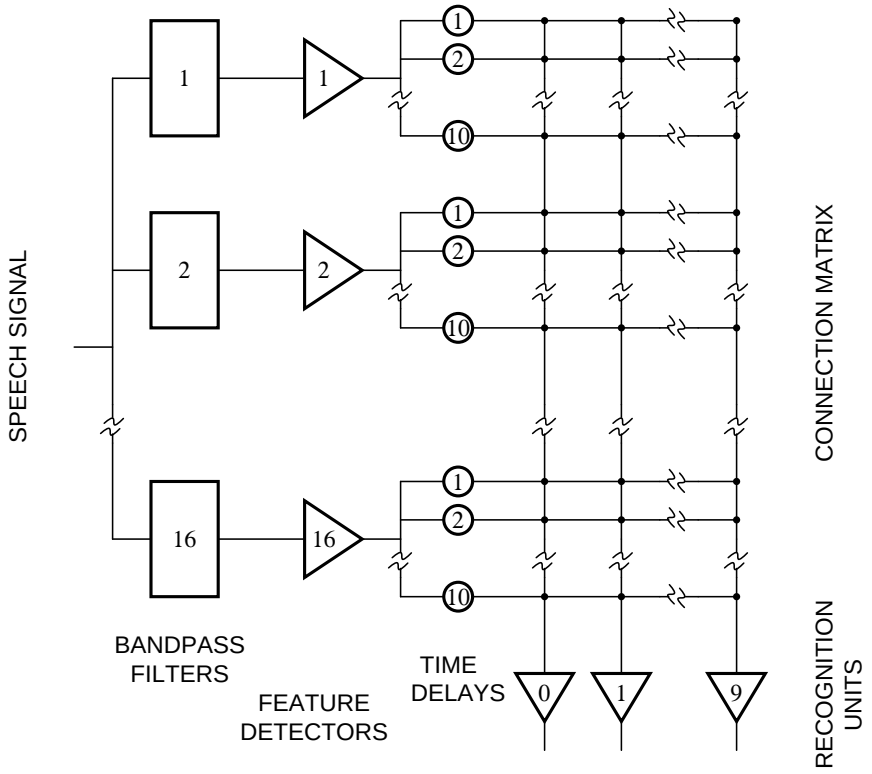

Figure 5.1: Unnikrishnan, Hopfield, and Tank correlator architecture, block diagram.

The correlation between input and template is also similar to that of the ATP, except the input signal is binarized, not the template. However, there are some key differences by which the Unnikrishnan et al. architecture copes with time variations in the input pattern: it treats event times as normally distributed variations around a fixed mean. Input signals from each (frequency) channel travel through a delay line where they diffuse as they travel, and the correlation is computed at ten "taps" spaced at regular time intervals (rather than at every time sample as in the ATP). The form of the correlation is (following the naming conventions of Equation (1.2)):

$$c_z[\,t\,] = \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{\tau=1}^{T} x[t-n,m]\, g[\tau,n]\, p_z[\tau,m] \tag{5.6}$$

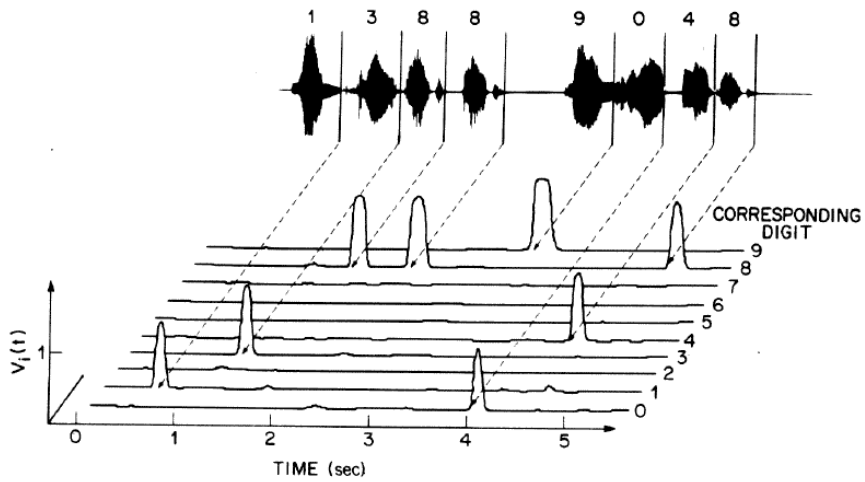

Figure 5.2: Unnikrishnan *et al.* correlator simulation results on isolated digit recognition.

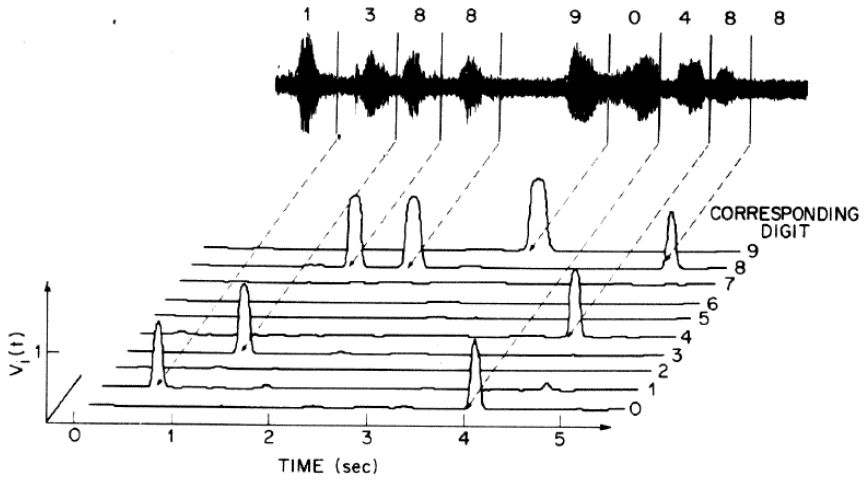

Figure 5.3: Response of architecture in simulation to the same input as Figure 5.2 but with white noise added to the input. The system remains robust in the presence of noise.

where T is the number of taps, and $g[\cdot]$ is the set of amplitudes defining the Gaussian diffusion profile at each tap. Here I have taken the liberty of replacing a continuous integral, as it is presented in the original paper ([68], Equation (3)), with a discrete sum, as it is implemented in their software simulation. The Gaussian diffusion kernels (Figure 5.4) follow the equation

$$g(n,\tau) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(\tau - n\tau_{\text{delay}})^2}{2\sigma^2}\right) \tag{5.7}$$

where $\sigma = \sigma_0^2(n - \lfloor N/2 \rfloor)^2$ defines the Gaussian width at each tap. Parameters for scaling the Gaussian width σ_0 and the spacing between taps τ_{delay} are the main free variables of the system. If

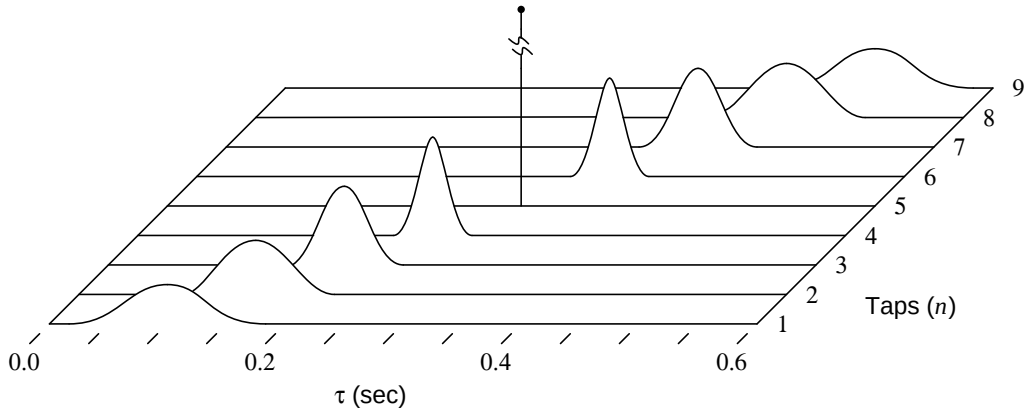

Figure 5.4: Gaussian kernels which diffuse the input in the Unnikrishnan *et al.* model.

we note that the weights and the Gaussian profiles are fixed, then we can define a function in which the weights are premultiplied by the Gaussian profiles to generate a new set of weights:

$$p'_z[n,m] = \sum_{\tau=1}^{T} g[\tau,n]\, p_z[\tau,m], \tag{5.8}$$

then the correlation takes a form equivalent to Equation (1.2):

$$c_z[\,t\,] = \sum_{m=1}^{M} \sum_{n=1}^{N} x[t-n,m]\, p'_z[n,m]. \tag{5.9}$$

This reveals the close connection with the ATP algorithm. In the previous section of this chapter, I explained how ensemble averaging over many inputs can produce templates which encode the probability distribution of the timing of events relative to the point of segmentation. This architecture does something very similar but assumes that the timing probability distributions are normal and encodes the normal distribution as a function (the Gaussian kernels) rather than part of the template. The arrangement of the equation in (5.9) moves the Gaussian functions back into the template. The

step may be regarded as an improvement or not depending on the capabilities of the hardware. The tapped-delay model of Unnikrishnan et al. requires much less storage per template, but on the other hand it depends heavily on the Gaussian multiplications. The authors envision the Gaussian kernel multiplications as diffusion profiles in an analog circuit; however, the model shows diffusion occurring both directions in time, an impossibility for a circuit implemented as a physically diffuse medium (such as the bucket brigade). We have emphasized throughout the thesis that analog multiplication is difficult and imprecise. The architecture chosen for the acoustic transient processor is one which avoids the problem of analog multiplication and it is not unreasonable to assume that the above equation, which casts the Unnikrishnan *et al*. model into the framework of the acoustic transient processor, is a reasonable compromise between system size and accuracy. It remains to be investigated whether the performance of this algorithm remains high if the same manipulations applied to the ATP algorithm are applied to it; namely, if the weights take binary values instead of, or in addition to, the input. If so, then an ATP-like architecture can be used to implement and test the algorithm.

The scheme used by the Unnikrishnan *et al.* architecture for training the system lies somewhere between the simplicity of ensemble averaging and the complexity of independent component analysis. Segmentation of each input example is performed by hand, but the system learns the correct position of the segmentation through a bootstrapping method. Segmentation is defined as the point near or at the end of the spoken word at which the detector should signal a correct classification. The set of weights for each word recognizer is updated with a gradient-descent equation (perceptron rule), where the word recognizer matching the training word presented is updated to increase its output at the segmentation point, while the word recognizers for all other words are updated to decrease their outputs at the segmentation point and also at other fixed intervals during the presentation of the input training example. When the training is near completion, the system can use its own output to determine the segmentation point, thus allowing it to overcome small errors in the estimated segmentation. The negative training on out-of-class weights helps the system isolate those parts of each class which are different from all the others, unlike ensemble averaging which will additionally encode those features which are common to all transient classes.

Because spoken digits are not transients, they have timespans on the order of half a second or longer, requiring a delay length (and template size) more than four times that of the ATP. This is because the Unnikrishnan *et al.* architecture is not hierarchical. It is not even phonetic, which is a simple form of hierarchy used by many systems. If a phoneme occurs twice in the same word, or if the same phoneme is shared among several words, the stored weights encoding the recognizer

of that phoneme are duplicated for every instance. A hierarchical structure could easily remove that redundancy. Using one correlator per *phoneme* and employing a state-based model to detect transitions between phonetic states is a much more efficient use of hardware, and should create a system more robust to differences in the timespan of different instances of the same spoken digit.

5.6 Biologically-Inspired Methods

A potential use of the acoustic transient processor is the investigation of acoustic processing in the brain. Recent research by Shamma [70] shows that certain areas of the mammalian auditory cortex encode time-frequency maps in neural "wetware." Shamma subjected neural recordings to analysis and was able to produce the time-frequency mapping encoded by single neuron outputs. These maps are qualitatively similar to spatial mappings of the independent components of natural images as produced by Bell and Sejnowski [72]. A large proportion of both the neural and natural scene ICA outputs have Gabor logon-like characteristics (that is, sine- and cosine-modulated Gaussians in 2-dimensional space). This is not entirely surprising given the relationship of all three systems (wavelets, independent component analysis, and biological neural processing) to information theory and the efficient representation of signals.

A typical example of a map in the auditory cortex is shown in Figure 5.5. The figure represents the level of neural activity in response to auditory signals over a range of frequency and change of frequency with time. The neural activity is measured in spikes per second and averaged over numerous trials. Most of the map is unencoded—that is, the neuron does not respond at all, either with excitation or inhibition, to inputs in that range of time and frequency. Other areas produce an excitatory response (red) and others, an inhibitory response (blue). In the figure, the mapping is tuned to a falling tone. If the mapping represents the input directly (that is, without differentiation in time or frequency), then the mapping is strongly tuned to a particular falling formant but at the same time is strongly inhibiting a tone slightly below it in frequency and/or slightly lagging in time. On the other hand, if the mapping represents a channel-differenced encoding of the input, then the mapping represents a simple response to a single falling tone. A simple model of the distributions which covers most cases observed in the auditory cortex can be formed by one or two Gaussians encoding regions of either odd symmetry, as shown in the figure, or even symmetry consisting of a large excitatory response surrounded by inhibitory side lobes. The angle of the Gaussians corresponds to the rate of the rise or drop. At the extremes, a Gaussian with no rotation (horizontal) encodes a constant tone, and a vertically-aligned Gaussian encodes a very fast transient or some

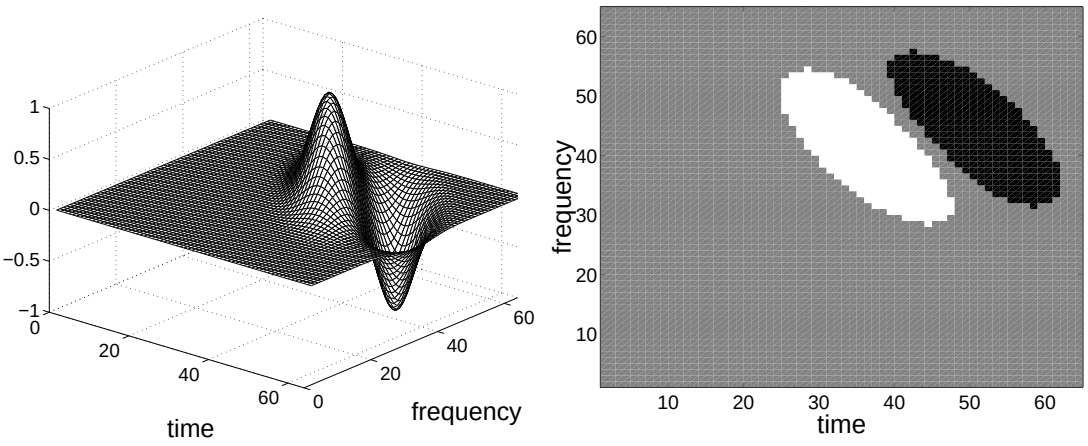

Figure 5.5: Example of a biologically-inspired template encoding a falling tone of specific frequency and rate. *left:* Continuous-valued encoding. *right:* Trinary encoding.

kind of instantaneous spectral information. The Gaussian is defined by the equation

$$\frac{1}{2\pi\sqrt{|U|}}\exp\left((\mathbf{x}-\mathbf{x}_0)U(\mathbf{x}-\mathbf{x}_0)^T\right) \tag{5.10}$$

where $\mathbf{x}$ is a 2-dimensional vector in time and frequency, $\mathbf{x}_0$ is the distribution center, and U is a covariance matrix

$$U = \begin{bmatrix} (\sigma_t)^2 & \rho \\ \rho & (\sigma_f)^2 \end{bmatrix} \tag{5.11}$$

having components for variance in time σ_t, in frequency σ_f, and the covariance coefficient ρ, which effectively represents a rotation of the distribution.

The acoustic transient architecture algorithm provides a way to actively test theories associated with the observations from the auditory cortex. In particular, the trinary encoding of the digital transient classifier hardware can encode regions of positive and negative response inside a region of zero response, as shown on the right-hand side of Figure 5.5. This is essentially an attempt to map the correlation algorithm onto a neuron, where each location of the template maps more or less directly to a synapse connection between the delayed output of the cochlear frequency mapping and the neuron under study in the auditory cortex. The neuron is more efficient than the correlator because it presumably uses no more synaptic connections than necessary for each response. It also has a higher resolution than the correlator because the synaptic weights encode continuous-valued weights: a neat little trick of nature that is, as noted in previous chapters, rather difficult to duplicate in silicon. Also, the neurons of the brain make optimally efficient use of the nonlinearities inherent

to the system, whereas the correlator is a linear system (certain methods of training, such as independent component analysis, depend on this fact). Despite the differences between a real neuron and its model implemented as a linear, trinary correlation, we can expect that the system is in most ways like the acoustic transient system and should therefore remain reasonably robust after the same algorithmic manipulations.

Conclusion

This thesis has sought to portray analog VLSI as a medium for computation and a viable alternative to digital signal processing. These aspects of analog design were taken for granted many years ago before the start of the so-called "computer revolution," but today require justification in an increasingly high-speed digital world.

Our use of the Analog VLSI medium concentrates on acoustic signal processing. We show how efficient use of mixed-mode processing, translinear circuit design, and other techniques can lead to compact, power-efficient systems which can rival DSPs in many aspects of performance. What we espouse is a view of system design which starts at the level of algorithms and looks at all ways, both analog, digital, or mixtures of both, which can best implement a specific system under the constraint of limited resources, namely, size and power. I believe that there is still a need for compact, power-efficient analog computing systems even in a primarily digital world. The conflicting demands of energy efficiency and raw processing power must be satisfied, and so system design must avoid many of the pitfalls of brute-force digital design and achieve ever greater levels of finesse from devices to circuits to algorithms and systems.

In this thesis, we have demonstrated the application of efficient analog and mixed-signal design to several acoustic signal processing systems, starting with ways of perceiving and mapping the time-frequency domain. Throughout the thesis, we have shown how each system deals with this mapping, from wavelet transforms to filterbanks and pattern classification systems. We have "zoomed in" from this view of signal processing to look at specific circuits for implementing time-frequency transformations of acoustic inputs and acoustic pattern recognition and classification. And we have "zoomed out" from this view to look at the larger picture of learning machines, speech recognition, and biological models of neural processing. We could continue on in either direction, from transistor physics to the human brain, because time-frequency mapping of acoustic data occurs ubiquitously in both nature and in engineering. At this important juncture of circuits, acoustics, biology, and machine and human learning, we hope we have provided some valuable insights and pointed the way to better engineering design.