Contribution Analysis: A Technique for Assigning Responsibilites to Hidden Units in Conectionist Networks

Dennis Sanger

CU-CS-435-89 May 1989

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Contribution Analysis: A Technique for Assigning Responsibilities to Hidden Units in Connectionist Networks

Dennis Sanger

AT&T Bell Laboratories and the University of Colorado at Boulder

Abstract: Contributions, the products of hidden unit activations and weights, are presented as a valuable tool for investigating the inner workings of neural nets. Using a scaled-down version of NETtalk, a fully automated method for summarizing in a compact form both local and distributed hidden-unit responsibilities is demonstrated. Contributions are shown to be more useful for ascertaining hidden-unit responsibilities than either weights or hidden-unit activations. Among the results yielded by contribution analysis: for the example net, redundant output units are handled by identical patterns of hidden units, and the amount of responsibility a hidden unit takes on is inversely proportional to the number of hidden units.

1. The Problem

Three-layer feed-forward neural nets have become very popular of late, with much research exploring how to apply these nets to a wide variety of problems and how to improve their performance, especially in terms of learning rate. Yet in spite of their popularity, relatively little research has investigated how neural nets solve problems, i.e. how the hidden units implement the mapping from input to output. This has resulted in a situation where researchers are using a tool that works well, while not understanding in any great detail why the tool works, analogous to an instructor not requiring pupils to show how they arrive at solutions to problems.

Understanding this mapping would not be a difficult problem if the activation function used by the net were linear: given inputs and outputs, the net would essentially be solving a large system of linear equations, eliminating much of the mystique surrounding the problem of understanding what goes on inside neural nets.

Neural nets do not, however, use linear activation functions. Worse yet, most nets use at least one additional, hidden layer of non-linearly activated units. Now, the problem of mapping inputs to outputs clearly does not have a simple, closed-form solution. In this paper, I propose a method, contribution analysis, for deriving the responsibilities of individual hidden units in implementing the input-output mapping.

2. Contribution Analysis

While most attempts at understanding neural net internals focus on weights or hidden-units, I define a new quantity, a combination of weights and hidden-unit activations: the contribution. For a specific input presentation, a specific hidden unit, and a specific output unit, the contribution is defined as the product of the hidden unit's activation when the net is presented with the specified input and the weight from the hidden unit to the output unit. There is a distinct contribution for each combination of input presentation, hidden unit, and output unit.

Unless stated otherwise, for the analysis described in this paper, the contribution is slightly modified. It is possible for a contribution to be very large, yet push the output unit in the wrong direction. Further, the sign of a contribution indicates only whether the contribution stimulates or inhibits the output unit. To transform the contribution into a measure of correctness, it is necessary only to adjust the sign of the contribution so that contributions in the wrong direction are negative, while contributions in the right direction are positive.

3. An Example Problem Domain: Micro-NETtalk

Contribution analysis is best explained by example. The neural net used here is a small version of NETtalk (Sejnowski & Rosenberg, 1986; 1987; 1987), a net that learns to convert written English text to the corresponding spoken English phonemes.

Like NETtalk, this net uses the backward-propagation rule (Rumelhart, Hinton, & Williams, 1986) to learn to "pronounce" text. In this case, as shown in Figure 1, the input to the net is a pair of letters and the output is a phoneme corresponding to the pronunciation of the first letter in the pair.

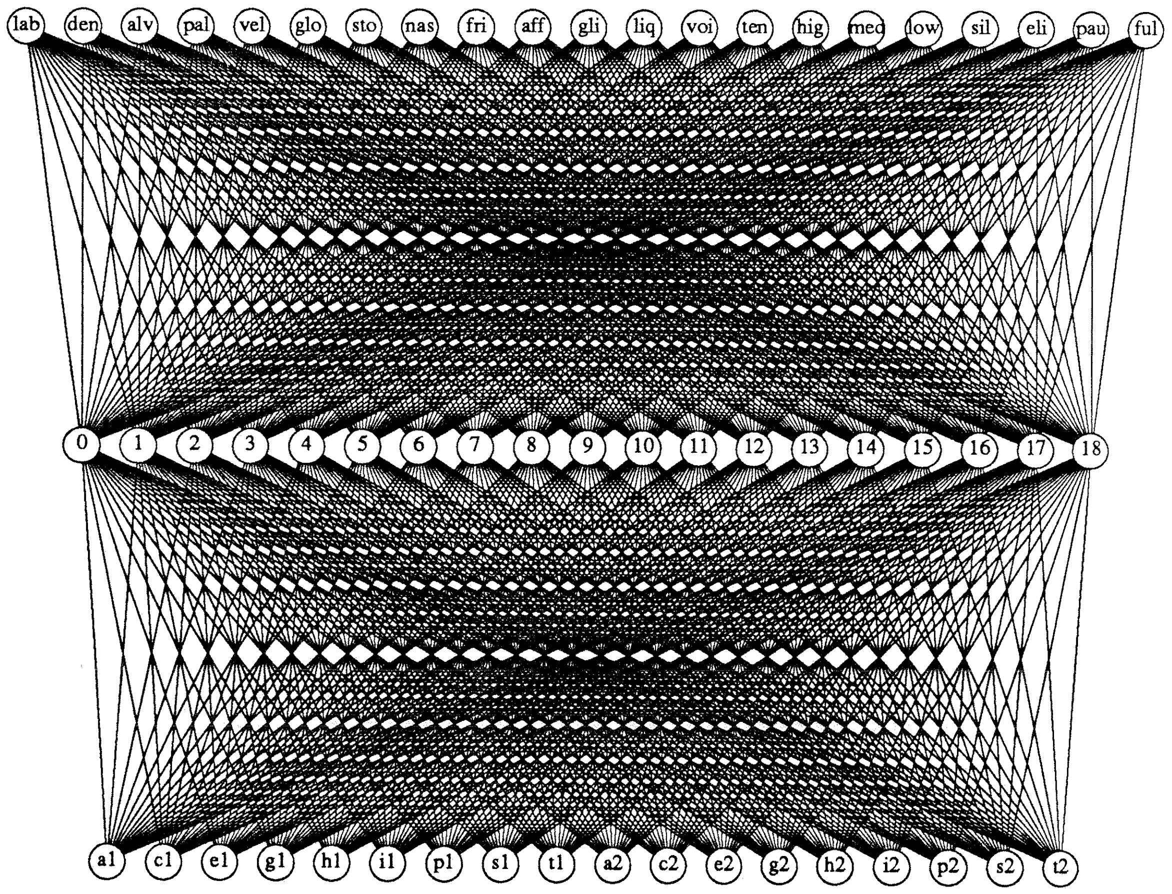

Figure 1 - Micro-NETtalk Architecture

There are eighteen input units, logically divided into two groups of nine units, with one group representing the first letter in the pair (the letter to be "pronounced") and the other group representing the second. Within the groups, each of the nine units represents one of the letters from the set {a, c, e, g, h, i, p, s, t}. For valid input, only one unit in each group will be active. The input units are referred to by names consisting of a letter followed by a number, such that the letter is the letter the unit represents and the number is the position of the letter in the input, i.e. "1" or "2". For example, for input "ag", only the units **a1** and **g2** will be active.

The number of hidden units may be varied. As few as five and as many as twenty-five were used in this study — unless stated otherwise, the data presented here comes from a net with nineteen hidden units. The hidden units are referred to by number, with the lowest-numbered one being called **0**, and the highest-numbered one, in this case, being **18**.

The twenty-one output units correspond to phonetic features, e.g. position of the tongue in the mouth or vowel pitch. The output units are referred to by three-letter abbreviations of the features they represent. For example, the unit **lab** represents the labial feature. The various features and their abbreviations are defined in Table 1, based on systems used by Sejnowski and Rosenberg (1986) and Ladefoged (1975).

Unit	Feature	Feature Type
lab	labial	position in mouth
den	dental	
alv	alveolar	
pal	palatal	
vel	velar	
glo	glottal	
sto	stop	phoneme type
nas	nasal	
fri	fricative	
aff	affricative	
gli	glide	
liq	liquid	
voi	voiced	
ten	tensed	
hig	high	vowel frequency
med	medium	
low	low	
sil	silent	punctuation
eli	elide	
pau	pause	
ful	full stop	

Table 1 - Phonetic Features

A phoneme is defined by a particular pattern of activation in the output units; see Table 2, again based on Sejnowski and Rosenberg (1986) and Ladefoged (1975). Note that the features **liq**, **pau**, and **ful** are never present.

Symbol	Phoneme	lab	den	alv	pal	vel	glo	sto	nas	fri	aff	gli	liq	voi	ten	hig	med	low	sil	eli	pau	ful
		Phonetic Features																				
-	*silence*																		√	√		
@	bat		√															√				
A	bite		√	√											√		√					
C	*ch*in				√						√											
E	bet	√	√														√					
I	bit	√														√						
J	*g*in					√			√					√								
S	*sh*in				√					√												
a	father				√										√			√				
e	bake		√												√		√					
f	*f*in	√								√												
g	*g*uess					√		√						√								
h	*h*ead						√					√										
i	Pete	√													√	√						
k	*K*en					√		√														
p	*p*et	√						√														
s	*s*it			√						√												

Table 2 - Phonemes and Associated Phonetic Features
(√ = feature present in phoneme)

The training set consisted of fifty-five two-letter pairs and a phoneme to represent the pronunciation of the first letter, as shown in Table 3. Only one pronunciation is associated with any pair; the pronunciation for a given pair was chosen by extracting occurrences of the pair from a dictionary and using the most frequent pronunciation. Note that none of the pairs started with the letter "t".

	a	c	e	g	h	i	p	s	t
a		@	e	@	a	e	@	@	@
c	k	k	s		C	s		k	k
e	i	E	i	E	E	i	E	E	E
g	g		J	g	-	g	g	g	-
h	h					h			
i	A	I	A	I			I	I	I
p	p		p		f	p	p	s	
s	s	s	s		S	s	s	s	s
t									

Table 3 - Letter Pairs and Their Pronunciations
(first letter on left, second letter on top)

The net was trained over two-thousand epochs, with weights updated after each of the fifty-five input presentations and used the activation function

$$a_j = \frac{1}{1 + e^{-(\sum_i w_{ji} a_i + \beta_j)}}$$

where a_j is the activation of the jth unit in this layer, e is the constant 2.719..., β_j is the bias, a_i is the activation of unit i in the layer below, and w_{ji} is the weight on the connection from that unit. The learning rate was .8; momentum was .2.

4. Applying Contribution Analysis to Micro-NETtalk

Since a contribution is defined in terms of a specific input presentation, a specific hidden unit, and a specific output unit, the set of all contributions resides in a three-dimensional array. Extracting useful generalizations from a large three-dimensional array of data — in the Micro-NETtalk example, there are 21,945 contributions — is not an easy task. Since the distribution of the contributions is more interesting in the pursuit of hidden-unit responsibilities than the magnitude of the contributions, principal component analysis on two-dimensional cross-sections of the contribution array suggests itself as the method of attack.

Principal component analysis (Fukunaga, 1972) is a procedure for taking a set of n-vectors and rotating n-space such that the new axes are aligned with the direction of greatest variation in the set of vectors. For contribution analysis, there are two types of cross sections that principal components analysis can operate on. First, the set of vectors can be defined such that each vector consists of the contributions from all hidden units to a specific output unit for a specific input presentation. There is one such vector for each input presentation, and a separate analysis must be performed for each output unit. In this case, the principal components represent patterns of hidden units that are responsible for activating the specific output unit: distributed hidden-unit responsibilities can be found this way.

Second, the set of vectors can be defined such that each vector consists of the contributions from a specific hidden unit to all output units for a specific input presentation. Again, there is one such vector for each input presentation, and a separate analysis must be performed for each hidden unit. In this case, the principal components represent the patterns of output units that the specific hidden unit is responsible for: local hidden-unit responsibilities can be found this way.

4.1 Distributed Hidden-Unit Responsibilities

As described above, a separate principal component analysis is performed for each output unit. The results of such an analysis for the output unit **lab** are shown in Figures 2 and 3. Figure 2 plots the value of the first rotated variable for each input presentation. The input presentations are clearly divided into those for which **lab** is active and those for which it is not. Note that it is possible for the axes of contribution space to be rotated in a way that a negative value for a rotated variable actually denotes correctness; this is not the case here, where the input presentations that are handled correctly most strongly for **lab** are those with large positive values of the first rotated variable.

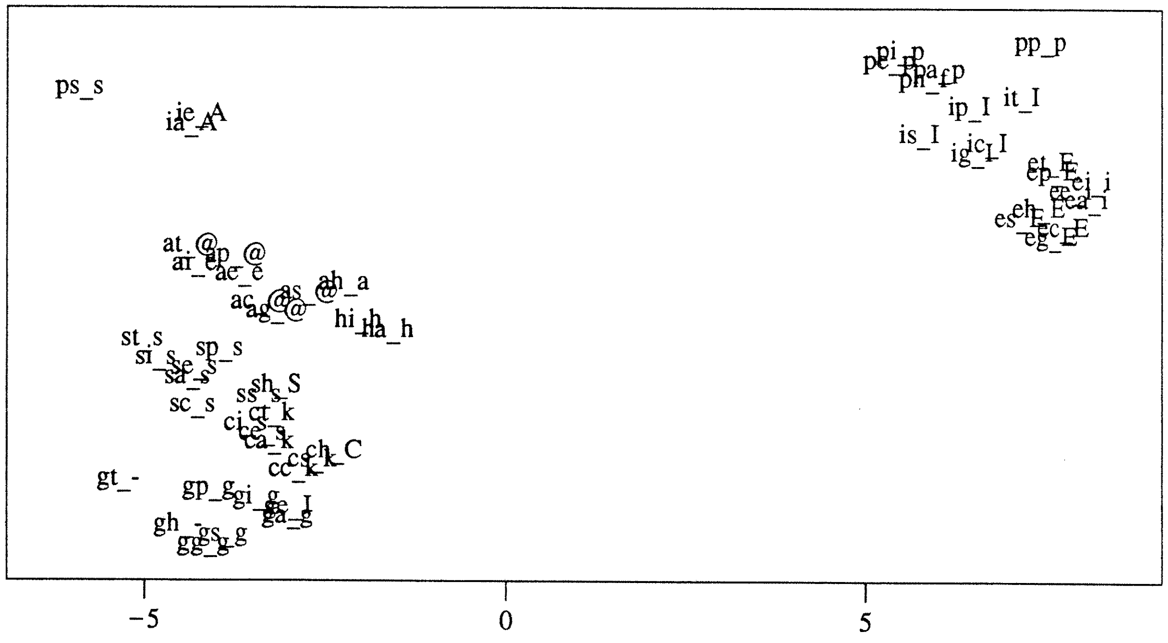

Figure 2 - Results of Contribution Analysis for the Output Feature "lab"
(Values of the first rotated variable for each input presentation -
"eg_E" means input presentation "eg", output phoneme "E")

Figure 3 plots the first column of the rotation matrix; the values represent the weight each hidden unit was given in forming the first principal component. The hidden units with the most positive weights are those that contribute most strongly in the right direction to **lab**. Thus Figure 3 yields the pattern of hidden units that are responsible for **lab**.

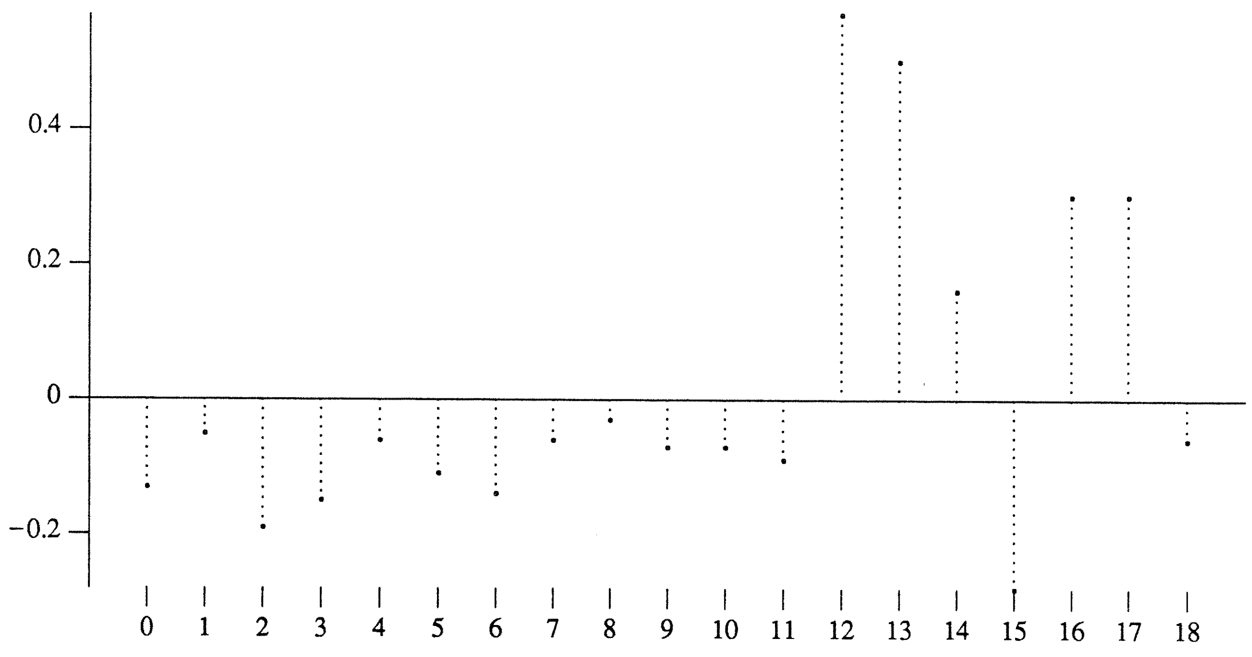

Figure 3 - First Principal Component Weight for Each Hidden Unit

Similar analyses can, of course, be performed for all the hidden units. Most of the graphs corresponding to Figure 2 show a similar quality of separation, i.e. the pattern of hidden units has a very clearly delineated domain of responsibility for this feature. A few features exhibit a lesser degree of separation — in general, the degree of separation increases as the proportion of inputs for which the output feature is active approaches one half. For example, **liq** is active for none of the input presentation, and the corresponding graph shows little separation.

The charts corresponding to Figure 3, on the other hand, showed with some consistency a clear separation between hidden units in the pattern and hidden units not in the pattern. This indicates that membership in hidden-unit patterns is clearly defined.

The results of analyses for all output units are summarized in Table 4. The second column shows the patterns of hidden units responsible for the features in the first column; when more than one principal component was significant, each was displayed as a separate row, in descending order of significance. The third column indicates for which of the input presentations the pattern properly activates the output unit, again, in descending order of significance.

An intriguing conclusion can be drawn from Table 4: the units **glo** and **gli** are handled by the same pattern of hidden units. What these output units have in common is that they are activated only for the two inputs "ha_h" and "hi_h". Likewise, the units **sil** and **eli** are activated only for "gt_-" and "gh_-", and are handled by identical patterns of hidden units. Finally, the same effect can be seen for the units **liq**, **pau**, and **ful**, which are never activated, and are all handled by the same pattern of hidden units. Apparently the net has learned which output units can be handled identically and thus consolidates the responsibility for identical units.

Feature	Hidden Units	Input Presentations
lab	12 13	ei_i ea_i ee_i ec_E ep_E et_E eg_E pp_p eh_E it_I es_E ic_I ig_I ip_I pa_p ph_f is_I pi_p pe_p
den	9	et_E ec_E eh_E eg_E ep_E es_E at_@ ap_@ ac_@ ag_@ as_@
alv	6	st_s si_s sa_s se_s sc_s sp_s ci_s ss_s ce_s ie_A ia_A ps_s
pal	0	cc_k ct_k cs_k ga_g ca_k ge_J gi_g gs_g gh_- gp_g gg_g
	7	ch_C sh_S ah_a
	6 11 15 4 1	se_s si_s
vel	0	gs_g gg_g gp_g ga_g ct_k cc_k gi_g ge_J ca_k cs_k
glo	8	hi_h ha_h
sto	16	pp_p gs_g ca_k pa_p gg_g cc_k gi_g gp_g cs_k ga_g pi_p pe_p ct_k
nas	2 0 13	ge_J
fri	18	si_s se_s sa_s st_s sp_s sc_s ss_s ps_s ci_s ce_s sh_S ph_f
aff	6 18 5 15	se_s si_s ci_s ce_s sp_s sc_s sh_S ss_s sa_s st_s
	0 7 2	gt_- gg_g gp_g gs_g gh_-
gli	8	hi_h ha_h
liq	0 3 2	cc_k gs_g gg_g cs_k gp_g ct_k ga_g gt_- gh_- gi_g ca_k
voi	2 0	gs_g gg_g gp_g ge_J ga_g gi_g
ten	15 10	ai_e ae_e ie_A ee_i ei_i ia_A ea_i ah_a
hig	12	it_I ic_I ig_I ip_I is_I ea_i ei_i ee_i
med	9 13 11	et_E ec_E eh_E ep_E eg_E es_E ai_e ae_e ia_A ie_A
low	10 11 9	at_@ ap_@ ac_@ ag_@ as_@ ah_a
sil	10	ae_e
	12	gh_- gt_-
eli	10	ae_e
	12	gh_- gt_-
pau	0 3 2	cc_k gs_g gg_g cs_k gp_g ct_k ga_g gt_- gh_- gi_g ca_k
ful	0 3 2	cc_k gs_g gg_g cs_k gp_g ct_k ga_g gt_- gh_- gi_g ca_k

Table 4 - Hidden Unit Patterns (19 Hidden Units)

4.2 Local Hidden-Unit Responsibilities

As described earlier, a separate principal component analysis is performed for each hidden unit. The results of such an analysis for hidden unit **0** are shown in Figures 4 and 5. Figure 4 plots the value of the first rotated variable for each input presentation. The input presentations are clearly divided into those that **0** handles correctly and those it does not especially care about. Hidden unit **0** appears to be responsible for all inputs resulting in the phonemes "k", "g", and "J".

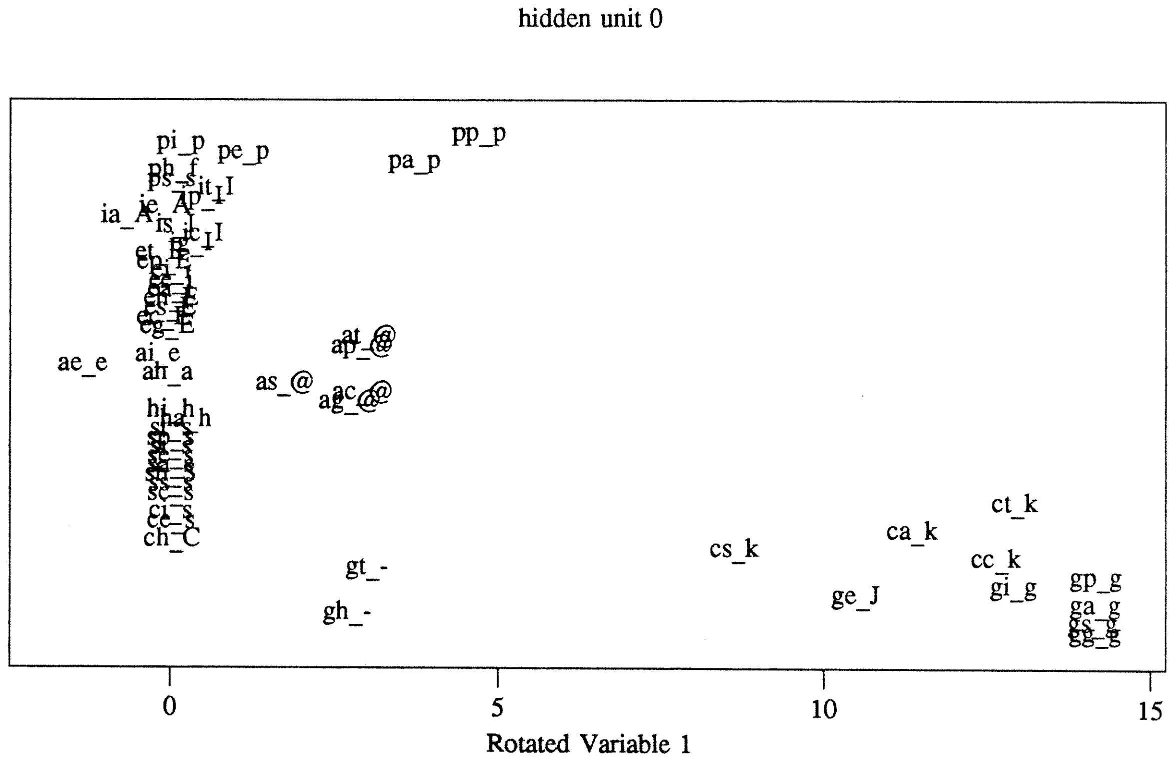

Figure 4 - Results of Contribution Analysis for Hidden Unit 0
(Values of the first rotated variable for each input presentation -
"eg_E" means input presentation "eg", output phoneme "E")

Figure 5 plots the first column of the rotation matrix; the values represent the weight each output unit was given in forming the first principal component. The output units with the largest weights are those that are contributed to most strongly by **0**. While Figure 4 shows which input presentations **0** is responsible for, Figure 5 shows which output units **0** handles. **0** turns out to be responsible for the output unit **vel**, which is active for the input presentations that **0** is responsible for.

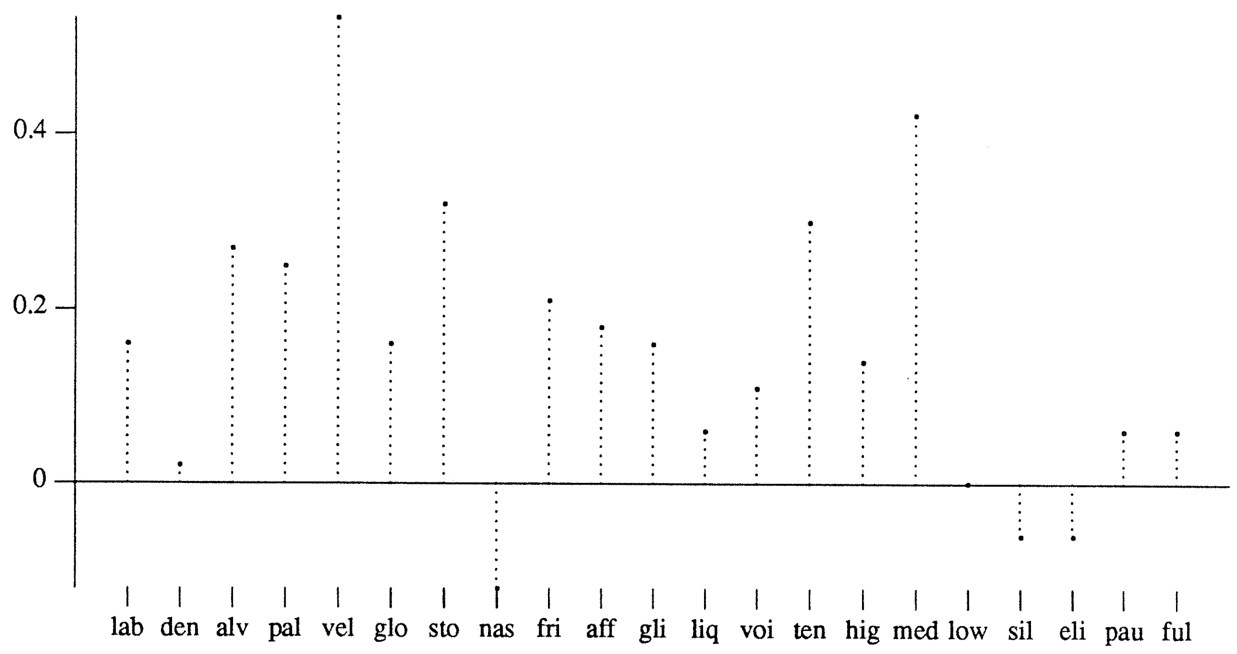

Figure 5 - First Principal Component Weight for Each Output Unit

The graphs corresponding to Figure 4 for the other hidden units exhibit strong separation, in most cases with one cluster of input presentations on one edge of the graph, another on the opposite edge, and no inputs in between. From this, one can conclude that each hidden unit has a very clearly defined domain of input presentations for which it is responsible. The separation is also clear, albeit not as strongly, in the charts corresponding to Figure 5, suggesting that hidden units tend to be responsible for correctly activating a small set of output units for a small set of input presentations.

The results of analyses for all hidden units are shown in Table 5. The output features for which the hidden units are responsible are shown in the second column. Some hidden units have more than one row of responsibilities — this occurs when more than just the first principal component has a significant standard deviation associated with it. A separate row of the table is devoted to each component, in descending order of significance. The input presentations for which the hidden units are responsible are listed in the third column, again, in descending order of significance.

Two results are readily apparent from Table 5. First, there is a strong dichotomy between units that are responsible for vowels and those that are responsible for consonants. Second, "exceptions", such as "gt_-" and "gh_-", appear frequently in the table, suggesting that exceptions require more resources to learn.

Hidden Unit	Features	Input Presentations
0	vel	gp_g gg_g ga_g gs_g ct_k gi_g cc_k ca_k ge_J cs_k
	med	gt_- at_@ ac_@ ap_@ gh_- ag_@
1	med	ah_a ch_C sh_S gh_- gt_- at_@
	sil eli	gh_- gt_-
2	lab	gs_g gg_g gp_g gi_g ga_g ge_J
3	nas med ten	cc_k gp_g gg_g ca_k cs_k ga_g gs_g
4	voi	ct_k
5	fri alv	ss_s st_s sc_s si_s se_s sp_s sa_s ps_s ci_s ce_s
6	alv	si_s se_s sa_s st_s sp_s sc_s ci_s ce_s ss_s ie_A ia_A
7	sto vel den	si_s se_s sp_s ci_s sh_S ce_s sc_s ss_s
8	gli glo	hi_h ha_h
9	den	et_E ep_E ec_E es_E eg_E eh_E
10	sto	es_E as_@ ap_@ ag_@ ac_@ is_I at_@ ep_E eg_E ai_e ae_e ip_I ig_I ec_E ic_I ah_a ea_i eh_E ei_i ee_i ia_A ie_A et_E it_I ps_s
11	vel sto alv	at_@ ac_@ as_@ ag_@ ap_@ eh_E et_E ai_e ae_e ah_a ec_E gh_- es_E ph_f eg_E ep_E sh_S gt_-
12	lab vel	et_E ec_E eh_E eg_E ep_E es_E it_I ea_i ei_i ic_I pp_p ee_i ph_f ig_I ip_I is_I
13	fri low lab	ei_i ea_i ee_i ep_E et_E ec_E ip_I eg_E it_I ic_I pi_p ig_I pa_p pe_p eh_E es_E pp_p ia_A ie_A is_I ai_e ae_e
14	hig	ip_I it_I ic_I ig_I is_I
15	sto	se_s si_s sa_s ce_s
	ten	ee_i ei_i ae_e ai_e ea_i ie_A ia_A ah_a
16	sto	cs_k cc_k gs_g ca_k pp_p pi_p pa_p gg_g gp_g gi_g pe_p ga_g ct_k
17	lab	eh_E et_E ec_E ep_E eg_E
18	fri	se_s ps_s si_s sc_s ss_s st_s sp_s sa_s ce_s ph_f ci_s sh_S

Table 5 - Hidden Unit Responsibilities (19 Hidden Units)

Table 5 may seem unwieldy for those seeking a short summary of why Micro-NETtalk works, but, upon inspection, it becomes obvious that Table 5 can be summarized in a much more compact form. For the reader familiar with regular-expression syntax, Table 5 is summarized in Table 5a. An asterisk represents a wild-card, while brackets represent a set of alternatives. For example, "g*_g" represents all input presentations with "g" in the first position and with output phoneme "g", while "p[hs] *" represents the input presentations "ph_f" and "ps_s".

Hidden Unit	Features	Input Presentations
0	vel	g*_[gJ] c*_k
	med	g*_- a*_@
1	med	[acs]h_* g*_- at_@
	sil eli	g*_-
2	lab	g*_[gJ]
3	nas med ten	c*_k g*_g
4	voi	ct_k
5	fri alv	[csp]*_s
6	alv	[cs]*_s i*_A
7	sto vel den	[cs]*_s sh_S
8	gli glo	h*_h
9	den	e*_E
10	sto	[aei]*_* ps_s
11	vel sto alv	a*_* e*_E [ps]h_* g*_-
12	lab vel	e*_* i*_I p[ph]_*
13	fri low lab	[ei]*_* p*_p
14	hig	i*_I
15	sto	s*_s
	ten	e*_i a*_[eh] i*_A
16	sto	c*_k g*_g p*_p
17	lab	e*_E
18	fri	[cps]*_s [ps]h_*

Table 5a - Hidden Unit Responsibility Summary (19 Hidden Units)

4.3 The Effect of Varying the Number of Hidden Units

To illustrate the usefulness of contribution analysis in studying net internals, consider the effect on hidden-unit responsibilities of varying the number of hidden units. Table 6 shows hidden-unit patterns found in a net with five hidden units, while Table 7 shows hidden-unit responsibilities for the same net. Tables 8 and 9 display similar data for a net with twenty-five hidden units.

Feature	Hidden Units	Input Presentations
lab	0	si_s sa_s se_s ce_s st_s ci_s sp_s sc_s ss_s ie_A sh_S ia_A ch_C ai_e ae_e ah_a ge_J
den	0	ie_A ia_A ai_e ae_e at_@ as_@ ac_@ ag_@ ap_@ es_E eh_E ep_E et_E eg_E ec_E ig_I ea_i ic_I ip_I is_I ei_i ee_i it_I
alv	4	si_s se_s sa_s ie_A ia_A ce_s sp_s st_s sc_s ci_s ss_s ps_s
pal	0	sh_S ah_a ch_C
vel	2	ge_J ca_k gs_g gi_g ga_g gg_g gp_g cs_k ct_k cc_k
glo	3	hi_h ha_h
sto	0	st_s sh_S sa_s si_s se_s ce_s ss_s sc_s ci_s ge_J sp_s gh_- ch_C gt_- ps_s ah_a ph_f ha_h hi_h ai_e ae_e ie_A ia_A
nas	0	ge_J
fri	4	st_s ps_s si_s sa_s se_s ss_s sc_s sp_s ce_s ci_s ph_f sh_S
aff	0	ch_C
gli	3	hi_h ha_h
liq	4	si_s sa_s se_s st_s ce_s sc_s sp_s ss_s ci_s ps_s
	4	ei_i ee_i ea_i ia_A ie_A ae_e ai_e pe_p pa_p pi_p ip_I it_I ic_I ig_I ep_E pp_p ec_E et_E eg_E es_E eh_E is_I ph_f
	2	pa_p pi_p gt_- pe_p gi_g ga_g gg_g gp_g gh_- gs_g ge_J
voi	2	gs_g gi_g ge_J gg_g ga_g gp_g
ten	0 4	ie_A ia_A ah_a ai_e ae_e
hig	4	ee_i ei_i ea_i ip_I it_I ic_I is_I ig_I
med	0	ie_A ia_A ai_e ae_e
low	3	as_@ ag_@ ac_@ at_@ ap_@ ah_a eh_E ep_E et_E ec_E eg_E es_E ae_e ai_e ea_i ee_i ei_i ia_A ie_A ip_I it_I ic_I ig_I is_I
sil	1	gh_- gt_-
eli	1	gh_- gt_-
pau	4	si_s sa_s se_s st_s ce_s sc_s sp_s ss_s ci_s ps_s
	4	ei_i ee_i ea_i ia_A ie_A ae_e ai_e pe_p ip_I pa_p pi_p it_I ic_I ig_I ep_E pp_p ec_E et_E eg_E es_E eh_E is_I ph_f
	2	pa_p pi_p pe_p gt_- gi_g ga_g gg_g gp_g gh_- gs_g ge_J
ful	4	si_s sa_s se_s st_s ce_s sp_s sc_s ss_s ci_s ps_s
	4	ei_i ee_i ea_i ia_A ie_A ae_e ai_e pe_p pa_p ip_I pi_p it_I ic_I ep_E ig_I ec_E et_E eg_E pp_p es_E eh_E is_I ph_f
	2	pa_p pi_p gt_- pe_p gi_g ga_g gg_g gp_g gh_- gs_g ge_J

Table 6 - Hidden Unit Patterns (5 Hidden Units)

Hidden Unit	Features	Input Presentations
0	lab	se_s si_s sa_s ce_s st_s ci_s sp_s sc_s ss_s sh_S ie_A ia_A ae_e ai_e ch_C ah_a ge_J
1	den	ph_f sh_S st_s ps_s ah_a ha_h hi_h gh_- gt_-
2	low den med hig ten	ss_s gt_- sa_s si_s ps_s ci_s se_s gh_- st_s sc_s ce_s sp_s cs_k ca_k ct_k cc_k pa_p pi_p gs_g ga_g gi_g gg_g gp_g pe_p sh_S ge_J
3	med ten voi sil eli nas lab	cs_k cc_k ct_k ss_s sc_s st_s sp_s ci_s sh_S si_s ps_s ce_s ca_k ch_C se_s sa_s is_I hi_h ha_h as_@
4	alv fri	se_s si_s sa_s ps_s sp_s sc_s st_s ce_s ci_s ss_s ie_A ia_A
	low pal den aff vel med voi sil eli nas glo gli	pe_p pi_p pa_p pp_p ee_i ei_i ip_I ph_f se_s si_s ic_I sa_s ps_s it_I ea_i ig_I is_I sp_s sc_s st_s ce_s ci_s ss_s

Table 7 - Hidden Unit Responsibilities (5 Hidden Units)

Feature	Hidden Units	Input Presentations
lab	14	pp_p ph_f pi_p eh_E et_E ec_E eg_E ep_E pa_p pe_p es_E it_I is_I ic_I ig_I ei_i ip_I ee_i ea_i
den	20	eh_E es_E eg_E ec_E ep_E et_E at_@ as_@ ac_@ ag_@ ap_@
alv	3 19	st_s sc_s sp_s ss_s si_s ps_s sa_s se_s ie_A ia_A ci_s ce_s
pal	6 23 12	ah_a ch_C sh_S
vel	13	cs_k gs_g gg_g gp_g cc_k ca_k ct_k gi_g ga_g ge_J
glo	19	si_s se_s sa_s ci_s ce_s sp_s sc_s sh_S ss_s st_s ie_A ia_A ca_k ai_e ae_e
	24	ha_h hi_h
sto	4	pi_p cs_k pp_p pa_p cc_k ca_k gs_g pe_p ct_k gg_g gp_g gi_g ga_g
nas	9	ge_J
fri	18	sh_S st_s ph_f ps_s ss_s sc_s si_s sp_s se_s ce_s ci_s sa_s
aff	15	ch_C eh_E ec_E et_E eg_E ep_E es_E ee_i ei_i ea_i
	13	ch_C
	19	ch_C se_s si_s ie_A sa_s ce_s ci_s ia_A sp_s ae_e sc_s ai_e ss_s st_s ee_i ei_i ea_i hi_h sh_S ha_h
gli	19	si_s se_s sa_s ci_s ce_s sp_s sc_s sh_S ss_s st_s ie_A ia_A ca_k ai_e ae_e
	24	ha_h hi_h
liq	15	eh_E ec_E et_E eg_E ep_E es_E ei_i ee_i ea_i ah_a ip_I at_@ it_I ic_I ig_I is_I ae_e ac_@ ie_A ag_@ ap_@ ia_A ai_e as_@
voi	10 13	gs_g gg_g gp_g ga_g gi_g ge_J
ten	15	ai_e ia_A ah_a ie_A ei_i ea_i ae_e ee_i
hig	14	ei_i ea_i ee_i it_I ic_I ig_I is_I ip_I
med	20	es_E eg_E eh_E ec_E ep_E et_E
low	20	ah_a at_@ ac_@ ag_@ ap_@ as_@
sil	18	gh_- gt_-
	15 18 6 13	gh_- gt_-
eli	18	gh_- gt_-
	15 18 6 13	gh_- gt_-
pau	15	eh_E ec_E et_E eg_E ep_E es_E ei_i ee_i ea_i ah_a at_@ ip_I it_I ic_I ig_I ae_e is_I ac_@ ie_A ag_@ ap_@ ia_A ai_e as_@
ful	15	eh_E ec_E et_E eg_E ep_E es_E ei_i ee_i ea_i ah_a ip_I at_@ it_I ic_I ig_I ae_e is_I ie_A ac_@ ap_@ ag_@ ia_A ai_e as_@

Table 8 - Hidden Unit Patterns (25 Hidden Units)

Hidden Unit	Features	Input Presentations
0	lab ten	ss_s st_s sc_s sh_S cs_k ct_k ci_s sa_s cc_k ps_s ch_C sp_s si_s gs_g ca_k gt_-
1	fri alv	si_s se_s ss_s st_s sc_s sp_s ci_s sa_s ce_s
2	ten	pp_p ph_f pa_p pe_p es_E eh_E pi_p et_E is_I ec_E eg_E it_I ep_E ic_I ig_I ch_C ip_I cs_k ct_k cc_k
3	alv	ia_A st_s sc_s sp_s ss_s ps_s ie_A si_s sa_s se_s
4	sto	cs_k cc_k gs_g gp_g gg_g ca_k gi_g pp_p pi_p pa_p ct_k pe_p ga_g
	ten	ps_s ci_s ss_s sp_s sc_s si_s
5	ten	it_I et_E st_s
6	den	sh_S ph_f ah_a ch_C
7	ten voi hig	pa_p ca_k pp_p ct_k cc_k pi_p
8	fri	ss_s si_s se_s sc_s ph_f sp_s sa_s sh_S st_s ps_s ci_s ce_s
9	sto fri alv	se_s ce_s si_s sp_s st_s sc_s ci_s
10	ten sil eli	ps_s ss_s sp_s sc_s sa_s cs_k si_s pp_p gs_g gp_g gg_g pa_p cc_k ga_g ca_k pi_p gi_g ci_s se_s pe_p as_@ ap_@ ag_@ ge_J
11	lab	pe_p ee_i pi_p ei_i pp_p ep_E pa_p et_E ea_i ph_f ec_E eg_E eh_E
12	sto vel	eh_E ph_f si_s es_E ss_s
13	med	gg_g gs_g gp_g ga_g cc_k ct_k cs_k gt_- gh_- ch_C ge_J gi_g ah_a ag_@ ac_@ at_@ ca_k as_@ ap_@
14	hig den med	ei_i ee_i ea_i it_I ic_I is_I ig_I ip_I
	med den	ps_s gs_g gg_g gp_g gh_- gt_- ah_a
	lab	pp_p pi_p ph_f pe_p pa_p
15	ten gli glo	cs_k ca_k cc_k ct_k gi_g gs_g ga_g gg_g gp_g gt_- ge_J pp_p gh_- pa_p pe_p pi_p st_s sp_s sc_s ce_s ps_s
16	voi	sa_s st_s ps_s ca_k ce_s ct_k pa_p sc_s ss_s ci_s sp_s se_s cc_k cs_k si_s pp_p pe_p ph_f pi_p ch_C sh_S ia_A at_@
17	ten voi nas den hig med lab	ch_C ct_k ca_k cc_k cs_k ci_s sh_S st_s sa_s sc_s ph_f ss_s si_s ha_h ps_s hi_h pa_p gh_- sp_s gt_- pi_p ce_s pp_p
18	sto	at_@ as_@ ac_@ eh_E et_E ag_@ ah_a ap_@ es_E ec_E gh_- ph_f eg_E gt_- sh_S st_s ps_s ep_E ae_e
19	alv	si_s sa_s se_s ci_s sc_s sp_s ce_s st_s ss_s sh_S ia_A ie_A

Table 9 - Hidden Unit Responsibilities (25 Hidden Units) - cont.

Hidden Unit	Features	Input Presentations
20	den	es_E eg_E ep_E eh_E ec_E et_E ai_e as_@ ag_@ ap_@ ac_@ at_@ ae_e
	hig	as_@ ag_@ ap_@ ac_@ at_@ ah_a cs_k ch_C cc_k ps_s ct_k pp_p ph_f
	hig med	cs_k cc_k ch_C pp_p es_E eg_E ps_s ep_E eh_E ec_E ct_k et_E ph_f
21	fri	ss_s se_s si_s sc_s st_s ps_s sp_s sa_s ce_s sh_S ci_s ph_f
22	alv	pp_p eh_E ph_f pi_p pe_p pa_p ep_E gh_- ch_C ah_a et_E ec_E eg_E sh_S hi_h ap_@ ha_h
	hig den	ch_C sh_S ah_a gh_-
23	med alv voi	ca_k cc_k ct_k cs_k ch_C ap_@ ac_@ ag_@ at_@ sh_S ah_a as_@ ph_f
	pal	ch_C ah_a sh_S
	low	ah_a ap_@ ac_@ ag_@ at_@ as_@
24	den ten low med hig lab eli sil	ci_s ss_s si_s cs_k ps_s ch_C sh_S sa_s ca_k ce_s sc_s cc_k sp_s se_s gs_g gi_g pi_p

Table 9 - Hidden Unit Responsibilities (25 Hidden Units)

The preceding tables suggest some interesting hypotheses. In Table 6, there is only one pattern consisting of more than one hidden unit, while Table 8 shows five such patterns, with up to four hidden units in a pattern. It appears that in nets with small numbers of hidden units, there is little distribution of responsibility.

A related effect can be observed in the hidden-unit-responsibility tables. In Table 7, three of the five hidden units are responsible for seven or more output features, while in Table 9, only two of the twenty-five units take on such a heavy workload. Similarly, while all but one of the hidden units in Table 7 are responsible for seventeen or more input presentations, less than half of the hidden units in Table 9 bear a comparable burden. Put simply, hidden units in a smaller net work harder.

5. The Effectiveness of Contribution Analysis

5.1 Contributions Versus Weights and Hidden-Unit Activations

Having developed a new method for analyzing what goes on inside neural nets, it is necessary to gauge its effectiveness relative to other methods. A simple comparison between weights, hidden-unit activations, and contributions involves thresholding activations passed to the output units. Suppose that all contributions below a given threshold are set to zero. If contribution analysis is a valid method for characterizing what is going on inside the net, then the error rate should remain low even if a large number of contributions to the output units are zeroed, if those contributions are all small, since the size of a contribution is directly proportional to its importance in implementing the input-output mapping. An analogous thresholding can be applied to hidden-unit activations and weights. If contributions are more useful in assigning meaning to hidden units than weights or hidden-unit activations, then the percentage of contributions that can be removed while still maintaining a given error level should be higher than the percentage of hidden-unit activations or weights that can be removed.

This is indeed the case, as is shown in Figure 6. The curve labelled "C" represents the number of erroneous outputs, i.e. output units with activations on the wrong side of .5, as the threshold value is raised and the percentage of contributions thresholded increases. The curve labelled "H" is the analogous curve for hidden-unit activations, while the curve for weights is labelled "W". Note that all three curves start out at one error with no information thresholded out, since there is a combination of input presentation and output unit, "ps_s"-**alv**, that is not mapped correctly by the net after two thousand epochs. If contributions were thresholded out in order of correctness, least correct first, rather than in order of magnitude, this error would disappear with only 3.9% of the contributions removed.

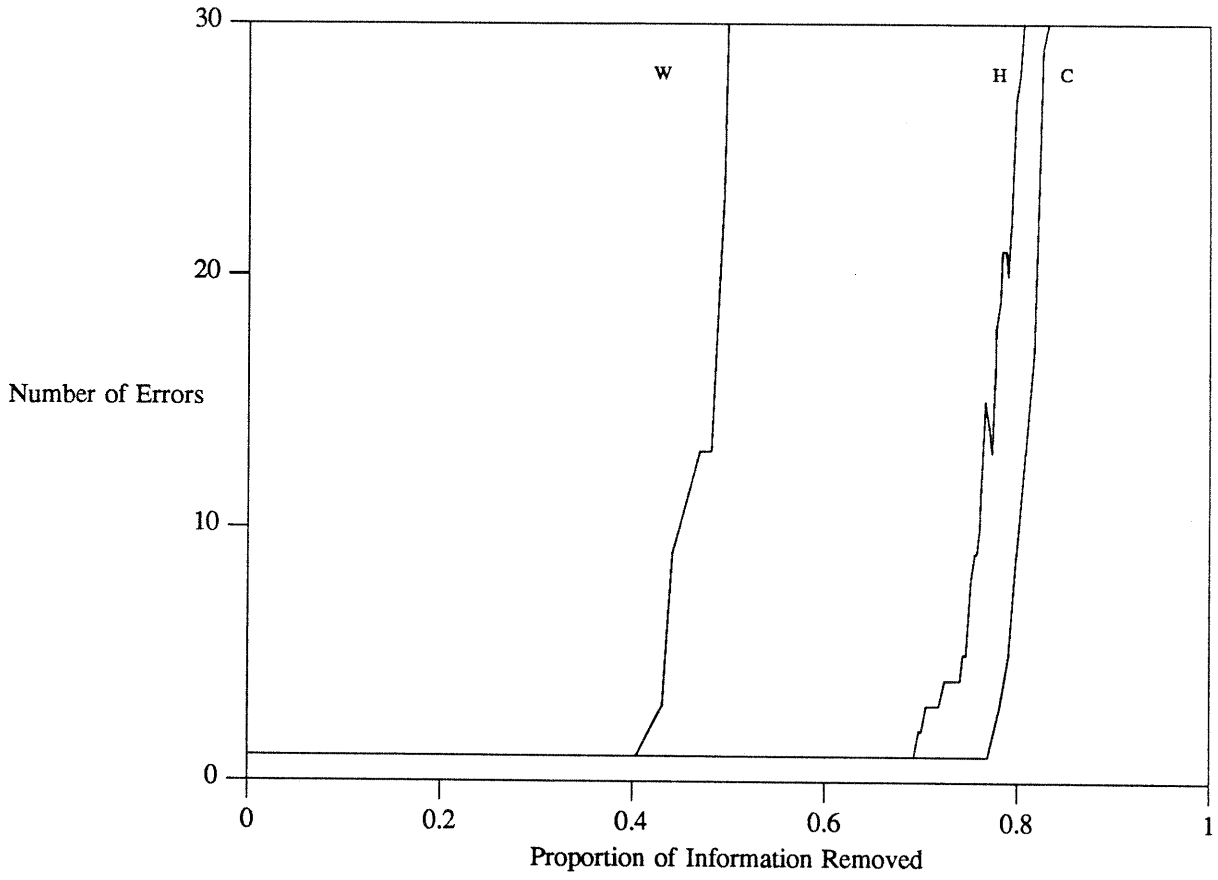

Figure 6 - Errors As a Result of Thresholding

Having determined that the contribution is a highly relevant quantity in terms of understanding connectionist net internals, the next question to ask is whether the hidden-unit responsibilities extracted from contribution array via principal components analysis are accurate. Table 10 shows the errors that occur when hidden units are removed from the net one at a time, i.e. it represents an empirical attempt at determining hidden unit responsibilities. Seven hidden units create no errors when they are removed, indicating that they are perhaps superfluous units, and this task could be learned with fewer hidden units. Note that only local hidden-unit responsibilities can be derived in this way without incurring combinatorial computational expense.

This experiment cannot yield all hidden-unit responsibilities, but only the ones that result in errors when the hidden unit is removed. Whether deleting a hidden unit produces an error is highly dependent on what other hidden units are doing: the sum of the contributions from all of the other hidden units may be larger than the contribution from the deleted hidden unit, although this contribution may be far larger than any other *single* contribution, and no error will result. Conversely, an output unit may be hovering on the brink of error, and deleting a small contribution may be enough to produce an error. As a result, Tables 5 and 10 are not identical, although they are similar enough to verify that the hidden-unit responsibilities revealed by contribution analysis are indeed accurate.

Hidden Unit	Erroneous Input/Output Pairs
0	gg_g_vel gs_g_vel ge_J_vel ca_k_vel
1	
2	
3	
4	
5	
6	ci_s_alv st_s_alv ia_A_alv ia_A_hig
7	ch_C_pal ch_C_sto
8	ha_h_glo ha_h_gli hi_h_glo hi_h_gli
9	at_@_den at_@_ten eg_E_den eg_E_hig eg_E_med ec_E_den ec_E_hig ec_E_med eh_E_den
10	ah_a_ten
11	sh_S_alv ag_@_vel ae_e_alv
12	gt_-_vel es_E_lab ea_i_hig ee_i_hig ei_i_hig
13	ae_e_low ie_A_fri pe_p_fri
14	
15	ge_J_sto
16	gg_g_sto gs_g_sto
17	
18	ps_s_fri

Table 10 - Errors Created by Removing Hidden Units
("ps_s_fri" means input "ps", output phoneme "s", output feature "fri")

5.2 An Extension to Contribution Analysis: Activation Change

As an extension to the contribution concept, consider the change in activation caused by deleting a single contribution. There is, then, an activation change associated with each contribution. It could be argued that this new quantity is a better measure of a hidden unit's responsibility, since a large contribution may do absolutely nothing to change an output unit's activation, if the sum of the other contributions may already have driven the unit perfectly to 0 or 1.

This argument may be countered by the assertion that contributions measure a hidden unit's responsibility ***in the absence of all other hidden units.*** Further, if several hidden units cooperate with large contributions to an output unit, the activation changes would all be small, leading one to believe that none of the large contributions are useful in activating the output unit. Deleting all the large contributions quickly erases that impression.

An example of this effect is readily apparent in Micro-NETtalk. Table 11 shows the relationship between contributions and activation changes. Points on a graph of contribution versus activation change have been grouped into one hundred clusters by dividing each axis into ten segments, and the number of points in each cluster is displayed in the table.

	-1 - -.8	-.8 - -.6	-.6 - -.4	-.4 - -.2	-.2 - 0	0 - .2	.2 - .4	.4 - .6	.6 - .8	.8 - 1
8 - 10	0	0	0	0	0	0	0	0	0	6
6 - 8	0	0	0	0	0	11	2	4	8	24
4 - 6	0	0	0	0	0	218	49	61	24	1
2 - 4	0	0	0	0	0	1386	71	1	0	0
0 - 2	0	0	0	0	0	14375	0	0	0	0
-2 - 0	0	0	0	0	5220	0	0	0	0	0
-4 - -2	0	0	0	0	369	0	0	0	0	0
-6 - -4	0	0	0	0	98	0	0	0	0	0
-8 - -6	0	0	0	0	16	0	0	0	0	0
-10 - -8	0	0	0	0	1	0	0	0	0	0

Table 11 - Contributions versus Activation Change
(Contributions on vertical axis, activation changes on horizontal axis)

Even though the net never learns the proper state of **alv** for the "ps_s" input, there is no large activation change in the wrong direction. This is because, as shown in Table 12, five hidden units cooperatively contribute in the wrong direction. I maintain that the contribution yields more valuable information in this case than the activation change.

Hidden Unit	Contribution	Activation Change
0	-1.482	0.000
1	-0.011	0.000
2	-0.664	0.000
3	-0.913	0.000
4	-0.072	0.000
5	3.056	0.000
6	0.002	0.000
7	0.006	0.000
8	-0.970	0.000
9	-0.215	0.000
10	-1.327	0.000
11	-4.742	-0.001
12	-2.018	0.000
13	0.073	0.000
14	-0.080	0.000
15	0.001	0.000
16	-2.660	0.000
17	-0.439	0.000
18	2.752	0.000

Table 12 - Contributions to "alv" for input "ps_s" with corresponding activation changes

The same effectiveness analysis that was used to compare contributions to weights and hidden-unit activations can be used to compare contributions to activation changes - the result, as shown in Figure 7, where the curve labelled "A" represents activation change, is that contributions perform better at first with activation change slowly catching up. Clearly, the extra effort involved in computing activation changes is not worth the effort.

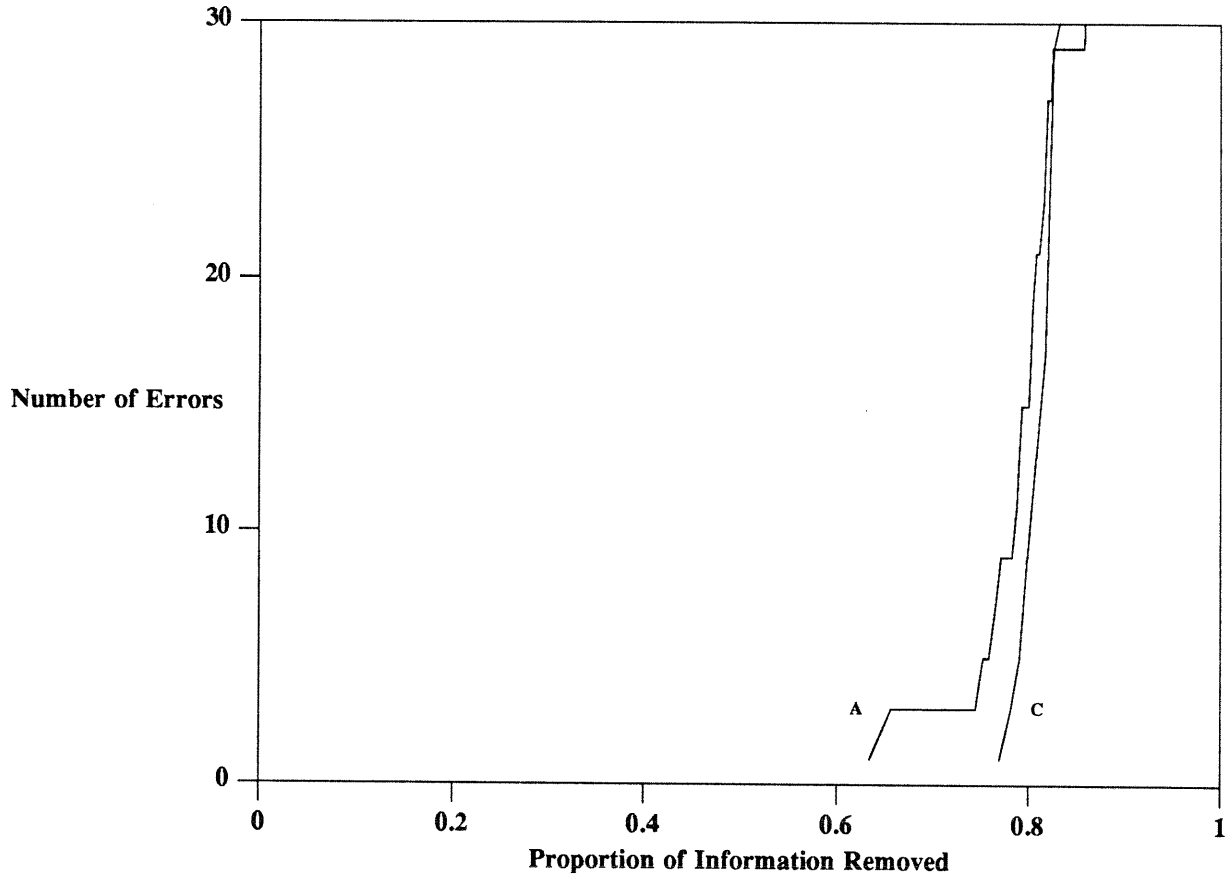

Figure 7 - Errors As a Result of Thresholding

6. Conclusion

In this paper, I argue for contribution analysis as a method for assigning responsibilities to hidden units. As demonstrated above, contributions are a more accurate indicator of how the net maps inputs to outputs than weights or hidden unit activations, although the cost of the analysis is greater.

The conclusion that contributions are more useful in assigning hidden-unit responsibilities than weights or hidden-unit activations is an intuitive one. It is common for highly active hidden units to be connected to some output units via minuscule weights. The activity of a hidden unit says little about how that hidden unit will affect output units. Similarly, for some input presentations, connections with large weights will be carrying a small amount of activation from a hidden unit to an output unit. The size of the weight on a connection correlates poorly with the amount of activation flowing across the connection. The contribution combines the weight and the hidden-unit activation in a quantity that is proportional to the effect on the activation of the output unit.

The example analysis demonstrates the ease with which contribution analysis brings to light interesting conclusions about neural net internals. Among the obvious results of this analysis: at least for the Micro-NETtalk application, redundant output units are handled by identical patterns of hidden units, and the amount of responsibility taken on by a hidden unit is inversely proportional to the number of hidden units.

7. *Future Work*

The most obvious piece of future work is to apply contribution analysis to a variety of nets and pick apart the results in greater detail than has been done in this paper, in the hope that general conclusions can be drawn can be drawn about the inner workings of neural nets. Since the size of the net that can be analyzed is limited only by the amount of available computing resources, I plan to look at larger nets than described here, in particular, the full-sized NETtalk. The bottleneck of the analysis is the principal components computation; the number of hidden units or output units that can be used is equivalent to the order of the largest matrix for which eigenvectors can be computed. As the number of input presentations increases, on the other hand, the computational expense increases only linearly.

The analysis is, as described in the appendix, fully automatable — C programs took as input the net's weights and training set and generated *pic* and *tbl* source for all the figures and tables (except for Table 5a) displaying contribution analysis results. A possibility for the future is the release of a software package for performing contribution analysis.

Finally, there are other, more practical applications for contribution analysis than just answering the question of how neural nets work. Mike Mozer and Paul Smolensky (1989) of the University of Colorado at Boulder have used a quantity called relevance, similar in spirit to contributions, to implement a sort of dynamic allocation of hidden units. Contributions could be used for the same sort of experimentation, as well as for investigations into how the net learns over epochs, potentially leading to a method for increasing learning speeds.

8. *Acknowledgments*

Thanks go to Paul Smolensky of the University of Colorado for his many useful comments on both the concepts and the format of this paper, to Mike Mozer and the Boulder Connectionist Research Group for their constructive criticism, and to Henry Dittmer and Dave Ruby of AT&T Bell Laboratories for their support.

9. *References*

Fukunaga, K. (1972) *Introduction to Statistical Pattern Recognition.* New York: Academic.

Ladefoged, P. (1975) *A Course in Phonetics.* New York: Harcourt Brace Jovanovich.

Mozer, M.C. & Smolensky, P. (1989) Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment. Technical Report CU-CS-421-89, University of Colorado, Department of Computer Science.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986) Learning Internal Representations By Error Propagation. In D.E. Rumelhart & J.L. McClelland (Eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations.* Cambridge: MIT Press.

Sejnowski, T.J. & Rosenberg, C.R. (1986) NETtalk: A Parallel Network That Learns To Read Aloud. Technical Report 86/01, Johns Hopkins University, Dept of Electrical Engineering and Computer Science.

Sejnowski, T.J. & Rosenberg, C.R. (1987) Parallel Networks That Learn to Pronounce English Text. *Complex Systems*, 1, 145-168.

Sejnowski, T.J. & Rosenberg, C.R. (1987) Connectionist Models of Learning. In M.S. Gazzaniga (Ed.) *Perspectives in Memory Research and Training.* Cambridge: MIT Press.

APPENDIX 1 - The Mechanics of Contribution Analysis

This appendix is intended as a sort of cookbook for generating contribution analysis results such as those presented in Tables 4 through 9 and Figures 2 through 5. There are three steps in the process: generating the contribution array, performing principal component analyses on a two-dimensional slice of the array, and extracting the significant information out of the principal component analysis output.

Since the contribution is the product of a hidden-unit activation evaluated for a specific input presentation and a weight from that hidden unit to an output unit, the contribution array has three dimensions: input presentation, hidden unit, and output unit. As the net is shown each input presentation, every weight between a hidden unit and an output unit is multiplied by the appropriate hidden-unit activation and the sign of the product is adjusted such that it is positive if it is pushing the output unit in the correct direction. The resulting contributions form a two-dimensional slice of the contribution array. When the net has seen all input presentations, the contribution array has been fully populated. Symbolically,

$$c_{ijk} = w_{kj}a_{ji}(2t_{ik}-1)$$

where w_{kj} is the weight from hidden unit j to output unit k, a_{ji} is the activation of hidden unit j when the net is shown the ith input presentation, and t_{ik} is the target output for output unit k and input presentation i. It is assumed that target outputs will always be 0 or 1. The last factor in the equation serves to adjust the sign of the contribution so that it is a measure of correctness, i.e. positive contributions are in the right direction.

As described in the body of the paper, a two-dimensional slice of the contribution array, taken for either a specific hidden unit or a specific output unit, is used as input to a principal components analysis. In one sentence, principal components analysis finds the eigenvalues and eigenvectors of the covariance matrix of the contribution slice. This amounts to rotating contribution space so that the new axes are in the directions of greatest variance. Each eigenvector represents the weightings given to each original factor in forming a new, rotated variable. The product of the eigenvector and the contribution slice yields a vector with values of that component's rotated variable. The eigenvalues correspond to the significance of the associated principal components. The rotated variable associated with the largest eigenvalue will show the clearest possible separation among the input presentations. It is this vector whose elements are graphed in Figures 2 and 4. The elements of the corresponding eigenvectors are charted in Figures 3 and 5.

To summarize the large amount of data present in these figures into tables such as Tables 4 through 9, a simple algorithm is used. To determine which principal components are significant, the associated eigenvalues are sorted. The largest gap in the sequence is found and principal components with eigenvalues above the gap are deemed significant.

For each significant component, the significant factor weightings (i.e. eigenvector elements) and well-represented input presentations (i.e. elements of the rotated variable) must be found. As for the eigenvalues, these vectors have their elements sorted and the largest gap in the sequence is sought. However, the search for the largest gap does not encompass the entire sorted set in these cases. Since positive values of the rotated variable do not necessarily denote well-represented inputs — an eigenvector may be multiplied by an arbitrary constant, even a negative one, and it remains an eigenvector — a decision must be made regarding which side of the origin is the "well-represented" side. The two extremes of the sorted set are compared to the set's average, and the side whose extreme is farther from the average is declared the "well-represented" side. This decision is motivated by the realization that all rotated variables that I have looked at have consisted of a cluster around the origin and a cluster of well-represented inputs far from the origin. Only gaps on the "well-represented" side of the origin, which is the same for both factor weightings and values of the rotated variable are considered in finding a cutoff point for significant factor weightings. In the search for a cutoff point for well-represented inputs, only the better-represented half of the rotated variable values are considered. Once these cut-off points have been found, values above the cutoffs are displayed in the tables.

Although this sounds like an enormous amount of effort, it is easy to automate. The C program that generated *pic* and *tbl* source for Tables 4 through 9 and Figures 2 through 5 ran in less than thirty CPU seconds on an Amdahl 580.