Neural Network Based Text to Phoneme Alignment and Mapping for Speech Technology

John A. Bullinaria

School of Computer Science, University of Birmingham
Birmingham, B15 2TT, UK

j.a.bullinaria@cs.bham.ac.uk

Abstract: Many speech technology applications rely upon having a good mapping between representations of text and phonemes that generalizes well to novel inputs, and that in turn relies upon having a good underlying alignment of the text and phonemes. For large scale systems, any approaches other than machine learning will be impractical, so that is the approach normally adopted. The current state-of-the-art appears to be a symbolic rule-based learning approach, which seems surprising given the range of neural network systems for text to phoneme mapping that have been developed over the years. This paper reviews the neural network based approaches to this problem and identifies the key problems involved. It then goes on to solve those problems and demonstrate how it is possible for neural networks to simultaneously perform text to phoneme alignment and mapping with performance levels at least as good as the best existing systems.

Keywords: Text to Phoneme, Learning, Mapping, Alignment, Neural Networks, Speech Technology.

1 Introduction

There are many speech technology applications (such as speech recognition and production) that require a mapping between text and phonemes that not only performs well on known words, but can also generalize appropriately to new words, such as previously unseen proper nouns [1]. Such mappings are generally most effectively produced by automated systems that learn from an enormous number of representative exemplars. However, the first stage of that process typically requires the alignment of the text and phonemes in the training data, so that an appropriate mapping can be learned, and that in itself remains a difficult research problem [2, 3, 4]. For most successful symbolic rule-based systems [1, 5] and neural network systems [6, 7, 8], that alignment is usually performed as a separate data pre-processing stage. The current state-of-the-art for this alignment process is the rule-based Expectation-Maximization (EM) algorithm of Damper et al. [9], and data aligned in that way has been used in the rule-based Pronunciation by Analogy (PbA) system of Damper et al. [1, 9] to provide state-of-the-art large-scale text to phoneme mappings for English [5].

It might seem surprising, given the large number of neural network based text-to-phoneme conversion systems found in the literature, that neural network approaches are not more competitive in this area. The most likely reason is that the vast majority of the older neural network approaches [7, 8, 10, 11] were primarily aimed at modeling psychological data and understanding human language abilities, rather than providing high performance applications for speech technology. They consequently tended to concentrate on modeling human-like performance on relatively small-scale empirically testable datasets, rather than creating large-scale systems of the kind required for real world applications. Another important issue has been the computational resources required for training neural networks, which has hampered progress in scaling them up to cope with larger words and larger training datasets. However, computers continue to grow more powerful, and a simple scalable neural network based approach for dealing with both the alignment and mapping problems has actually existed in the psychological modeling literature for some time [10, 11, 12, 13], so it seems timely to explore what can now be achieved with a neural network approach to this problem.

The remainder of this paper begins by outlining the key issues involved in the alignment and mapping processes, and reviews the existing neural network approaches to those tasks. That leads to a consideration of the problems faced when attempting to scale up simple psychological models to practical large-scale systems. Results from a series of computational experiments are then presented which optimize the scaling-up processes and allow direct comparisons with the results obtained by the state-of-

the-art symbolic approach of Damper et al. [9]. It is shown how the neural networks can not only learn a better mapping than the symbolic approach, but are also able to perform high quality alignment as part of the standard learning phase. The paper ends with some conclusions and discussion.

2 The Neural Network Architecture

Aligning the text (i.e. letters/graphemes) and phonemes in the training data is a crucial first step for text-to-phoneme conversion systems, irrespective of whether they are small scale psychological models of reading, or larger scale practical speech technology applications. The key issue is that some individual letters (e.g., "X") can sometimes correspond to more than one phoneme, and other strings of letters (e.g., "TH") can correspond to only one phoneme, and consequently, given corresponding strings of letters and phonemes, it is not always obvious which letters should map to which phonemes.

Various different approaches have previously been employed to deal with this alignment problem in neural network models. The first neural network text-to-phoneme system was the NETtalk model of Sejnowski & Rosenberg [6], but that chose to simply bypass the whole problem by using pre-aligned training data. For psychological models of reading, pre-alignment was considered unacceptable, and that led Seidenberg & McClelland [7] to develop models based on letter and phoneme trigrams known as Wickelfeatures [14]. Those models were groundbreaking in that they allowed direct comparisons with empirical human reading performance (such as stages of learning and reaction times), but ultimately they had relatively poor performance levels, even on small numbers of mono-syllabic words. The problem was that Wickelfeature style approaches are unable to represent reliably the information required to generate appropriate mappings for realistic languages [15]. The shortcomings of that type of model were addressed by Plaut, McClelland, Seidenberg & Patterson [8] who went on to develop more accurate models of human performance, but they had to return to pre-aligned training data to get good performance. That deficiency led Bullinaria to develop variations of the original NETtalk model that were able to avoid the pre-alignment by learning appropriate alignments at the same time as learning the mapping itself [10, 11, 13].

Those improved NETtalk models were based on a standard fully connected feed-forward neural network architecture [16], with two layers of connection weights similar to the original model [6]. The inputs consist of a sufficient number of blocks of $N_{letters}$ binary input units to represent the longest words, with each block containing one unit for each letter of the alphabet. The outputs consist of one or more blocks of $N_{phonemes}$ output units, with each block containing one unit for each phoneme of the language. In between, there is one layer of N_{hidden} hidden units, which is sufficient to accommodate appropriate learned

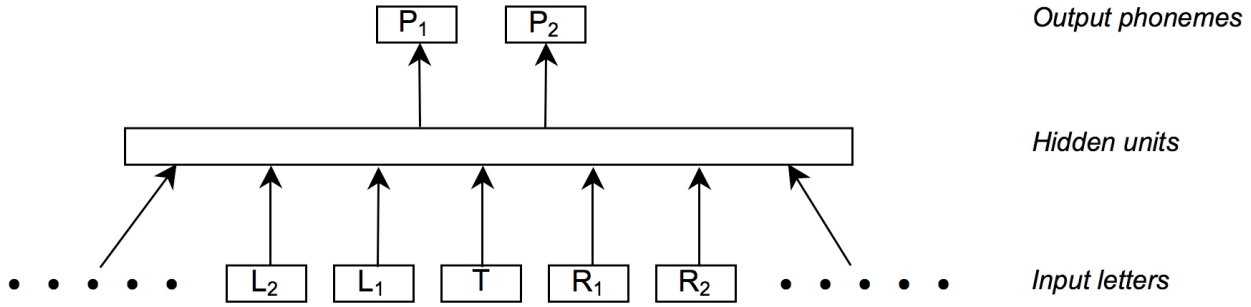

Figure 1. The neural network architecture with one letter represented in each input block and one phoneme represented in each output block. The phonemes represented in the two output blocks P_1 and P_2 correspond to the target letter represented in the centre block T of the input window, in the context of the letters represented in the blocks L_1, L_2, … to its left and R_1, R_2, … to its right.

hidden representations to facilitate the required input-to-output mappings. Each input word is initially positioned with its first letter in the middle input block, and then slides one letter at a time to the left, until its last letter is in that block. The network is trained so that, for each word in each position, the activated output phonemes correspond to the letter represented by the central input block, in the context of the letters surrounding it. To deal with the possibility that some single letters need to map to more than a single phoneme, the network usually requires more than one output block. Finally, to deal with strings of letters that map to shorter strings of phonemes, the output blocks allow the representation of a "null phoneme" (or "blank output") with equal status to the real phonemes. It is clear that, with sufficiently large networks of this type, words of any length can be accommodated, along with any number of one-to-one or many-to-one mappings between letter strings and phoneme strings. Obviously, one-to-many mappings (e.g., homographs that are not homophones) cannot be successfully accommodated without further context information, but supplementing the network with appropriate additional context inputs is straightforward [11]. For standard English words, two blocks of output phonemes are sufficient, but this will result in a few non-standard words and abbreviations (such as "XMAS") having less than optimal alignment. Thus, the basic representational requirements have led to the neural network architecture shown in Figure 1.

To train the neural network on pre-aligned training data, it is straightforward to apply any standard learning algorithm (such as traditional back-propagation, or any other gradient descent based connection weight updating [16]) so that the correct output phonemes are produced for each input position of each word. Clearly, to learn appropriate alignments too, a more sophisticated learning algorithm will be required. The first issue is that, as explained in detail by Damper et al. [9], there is actually no unique

correct alignment to be learned (e.g., the single phoneme corresponding to a "TH" could equally well be aligned with the letter "T" as with the "H"). However, some alignments will still be more appropriate than others. For example, consider the word "THIN", which maps to the three phonemes "dh ih n" (using the standard BEEP notation [17]). If "_" is used to represent a blank (null phoneme), then "dh _ ih n" and "_ dh ih n" would be considered equally acceptable alignments, but "dh ih n _" would not, because mapping the letter "I" to the phoneme "n" would be highly irregular compared with mapping the letter "N" to the phoneme "n". The crucial concept here is the idea of *regularity*: good alignments are highly regular, and poor alignments lead to many irregularities. It is the ability of neural networks to identify and exploit regularities in data that underlies their ability to generalize well, so there is good reason to expect them to be able to learn good alignments and have that lead to good mappings too.

It was this confidence in the ability of neural networks to identify regular alignments that led to the adoption of *multi-target training* [10, 13]. The general idea underlying this is that each neural network input pattern has not one, but a whole set of possible target output patterns, and the network is only trained on the target that already exhibits the lowest output error (e.g., the lowest sum-squared difference between the actual network outputs and the target outputs). For the case of mapping text to phonemes, the multiple targets are the complete set of possible text-to-phoneme alignments. Then, even if the network starts with random initial weights, the learning process tends to settle down to using the set of targets that correspond to a good regular set of alignments. The reason that happens is because, even if the starting alignments are completely random, any coherent regular weight updates will tend to build up, while the irregular incoherent weight updates will tend to cancel out, with the overall result that the regular alignments will have their output errors reduced by more that the irregular alignments. Then, with each round (or epoch) of weight updates, even more of the potential targets with regular alignments will have the lowest errors and be chosen as targets for learning (instead of those corresponding to irregular alignments), and the good alignments will come to dominate the weight updates even more. This process continues until all the chosen targets end up corresponding to good regular alignments, the alignment process is complete, and the same targets are chosen every time until all the output activation errors are reduced to whatever termination criterion is specified.

This neural network architecture and learning algorithm formed the basis of a series of models of reading aloud, including human-like generalization ability for reading non-words, accounts of frequency and regularity effects in reaction times, and models of developmental and acquired surface dyslexia [10, 11]. It also proved robust enough that straightforward extensions of it could be used to model the even harder reverse task of spelling, i.e. phoneme-to-text conversion [11, 12]. This naturally leads to the

question of why this approach has not been applied to task of text-to-phoneme mapping more generally. The most likely reason is that those earlier models were all rather small scale, and there are various practical technical problems with scaling it up to cope with much larger datasets, such as the British English Example Pronunciation (BEEP) dictionary [17] used in the Damper et al. [9] study.

3 Solving the Problems of Scaling-up

The main difficulties encountered when attempting to scale up the multi-target learning approach for the text to phoneme mapping are: the enormous numbers of words in the larger training datasets, the increased word lengths involved, and the combinatorial explosion that arises in the number of possible target alignments for longer words.

The first problem is very common with large training datasets: During each epoch of learning, the neural network weight updates associated with the irregular mappings (e.g., the letters "LB" being pronounced "p aw n d") are very small compared with the combined updates associated with more regular mappings, and that can render the irregular mappings extremely hard to learn. This issue tends to have been less of a problem for the earlier psychological models, because they have usually taken word usage frequencies into account in the training process, either by using the training words, or scaling the weight updates, in proportion to the word frequencies. The reason this works is that language evolution has resulted in irregular words generally being of higher frequency than regular words [18], which usually provides sufficient compensation for all the words to be learned well [7, 8, 11]. A more general alternative solution, that works even when word frequency information is not available (e.g., as with the BEEP dictionary data), is to simply stop updating the weights for words that have already reached some threshold output error level (e.g., that have each output unit activation within 0.2 of its binary target value). Since the latter approach is fairly standard practice, and proved to work very well for this application, it was adopted as standard for all the networks discussed in this paper.

Another problem with the larger training datasets is that they tend to contain longer compound words, and many of those will require the accommodation of very long range pronunciation dependencies. For example, two closely related words in the BEEP dictionary are

RECONSIDERATIONS ⇒ r iy k ax n s ih d ax r ey sh ax n z

PRECONSIDERATIONS ⇒ p r iy k ax n s ih d ax r ey sh n z

in which the presence or absence of the initial "P" affects the pronunciation of the final "IONS". It is debatable whether this is a real idiosyncrasy of English, or simply a problem with the BEEP dictionary

which has its content drawn from many different sources and is known to contain errors [19]. However, any system learning the mapping from this dataset will still have to deal with it, and to do that it will need to have a large enough input context window to accommodate the "P" while the "IONS" is being processed. To cope with long range dependencies like this in the architecture of Figure 1, 20 context letters each side of the crucial central position were required. That results in an input window of 41 letters, and hence 1066 input units in total, which is a rather large neural network to train. Computational resource problems could easily arise with naïve implementations of the network that simply multiply all the input activations by the corresponding weights. In practice, it is possible at each stage to simply ignore any input units that are not activated, because they do not contribute to any activations or weight updates, and just add the weights for those that are. This means, on average, adding only nine weights, rather than adding 1066 weighted inputs, for each hidden unit.

An unfortunate consequence of having to make large amounts of context information available for some words, is that it could easily be misused for other words. The crucial concept of regularity here is underpinned by the idea that the text-to-phoneme mapping consists of a whole hierarchy of rules embodied in the neural network connection weights, and that the simpler and more general that rule set is, the better it is likely to generalize. Thus, good generalization depends on making the *minimum* use of the available context information that is consistent with performing the mapping accurately on the training data, which means only using long range context information when more local context proves insufficient. That could be problematic given that the standard gradient descent based neural network learning automatically updates any weights connected to any useable input information. One obvious solution would be to have different learning rates for the weights connected to different blocks of inputs: large for the central block, and increasingly smaller for more distant blocks. Although that does work, it proves extremely difficult in practice to have the rates fall off sufficiently fast with distance without a considerable slow-down in the training time of the networks. A simple alternative solution corresponds to the standard practice of teaching children to read shorter words before they are exposed to longer words. This proves to work extremely well for training the neural networks. At each stage, the network waits till it has learned a certain fraction θ of the current word set before words of one letter longer are added to the used part of the training data. This incremental approach starts with learning words of length one and proceeds till the network is trained to completion on the whole training dataset. The final generalization performance depends on the chosen criterion θ, so that is an aspect of the learning process that needs to be optimized empirically.

A further problem resulting from having many long words in the training data, is that the

combinatorics of the text to phoneme alignment results in an explosion in the number of targets for the multi-target approach, and that becomes particularly troublesome for words of more than eight or nine letters long. The BEEP dictionary [17] contains around 200,000 words, with over 25% of them eleven or more letters long, and some as long as 28 letters. In practice, this means that the number of possible alignments that can be considered during training will eventually have to be restricted, but, of course, that needs to be done in such a way that a large proportion of the best alignments are still found. Fortunately, the incremental learning approach just discussed not only helps avoid the misuse of context information, but also provides a workable solution for dealing with the combinatorial explosion of the target alignments. In particular, it allows the training data subsets of different maximum word lengths to be easily dealt with in different ways.

The BEEP training data contains only 829 words of three or fewer letters, and these can easily be aligned by hand. The earlier small scale studies [10, 11, 13] showed that this was not actually necessary for achieving good final alignments, but it will be carried out here because it serves to prevent some of the arbitrariness in the resulting alignments, which simplifies the subsequent analysis and evaluation. In particular, the small hand-aligned words only have a non-blank in the right-hand phoneme block when there is already one in the left-hand block, and when two letters map to a single phoneme, that phoneme is always aligned with the left most letter. When the networks progress to learning the later words sets that also include longer words, all the words are unaligned, and the networks are free to change the hand-crafted alignments they were forced to learn previously, but that will only happen when there is some computational advantage to it. Generally, the alignment convention built into the hand-alignments of the initial short words leads to similar consistency in the alignments of the longer words too.

Once a word is pronounced correctly by the neural network, there is no need to look for a better target alignment for it, because any other alignment will clearly result in a larger error. This is actually quite common because each time the network moves on to learning a new larger word-set, most of the shorter words are already pronounced correctly, and many of the new words are also already correct without the need for further training, because they are regular and the network automatically generalizes well to them. That massively reduces the number of target alignments that need processing, and means it is computationally feasible to search through all the possible alignments for each incorrect word at each stage of training for all words of fewer than 16 letters.

A full search through all the potential alignments does, however, become impractical for longer words. Fortunately, in practice for the BEEP training data, by the time 16 letter words are introduced into the training, over 80% of the whole training set (that is about 160,000 words) is typically aligned and

learned correctly, and the principal alignment rules are firmly established in the network weights. Moreover, many of the remaining words are aligned correctly, and are only pronounced incorrectly due to subtle variations in the vowel sounds. Also, there are no new words introduced beyond the 15 letter word stage that have no letters at all aligned and pronounced correctly, and those letters that are correct act to strongly restrict how the rest are aligned. It turns out that a very good indication of the remaining unconstrained target phoneme positions can be obtained by simply identifying the existing non-blank outputs for each word, and "patching-up" the largest errors to result in the right number of output phoneme positions. Following that with a simple check of how swapping each blank with each non-blank affects the total error, usually gives a confirmation that the alignment is good, but occasionally finds an improved alignment. This faster "patch-up" process might still appear costly, but since all the network output activations are already computed, it usually has negligible computational cost compared with passing the activations through the network and computing the outputs in the first place. Even if this approach fails to identify the best possible alignment for a particular word, it will rarely get it totally wrong, and the combination of weight updates for all words will still improve the alignments ready for the next round. In the same way that random initial alignments eventually lead to good alignments, so do any less than perfect choices of output targets here. The effect of this switch to a faster approximate multi-target approach will be tested empirically by introducing the switch at an earlier stage and comparing the resulting performance against the standard switch point.

4 Optimizing the Learning Process

Having specified the neural network architecture, and identified the key problems and solutions for scaling up to large scale datasets, there remain several details that need optimizing for the best performance levels to be obtained.

The earlier small-scale implementations [10, 11, 13] all used the standard sum-squared-error measure for the gradient descent learning, but it has since been established that the cross-entropy error measure works better for classification type networks with sigmoidal outputs [16, 20], so that is used here. A gradient descent learning rate of 0.1 proved to result in learning that progressed reasonably quickly, but not so fast that the networks settled prematurely into inappropriate alignments, or kept switching unnecessarily between different target alignments. Varying the output error training tolerance between 0.1 and 0.3 led to no significant differences in the performances achieved, so a value of 0.2 was used. The number of hidden units had to be large enough that the associated connection weights could easily accommodate all the necessary pronunciation rules, but not so large that the network took too long to

train. There is no need to worry about restricting the numbers to avoid over-fitting here – what might generally be called "unwanted noise" in the training data are the irregular pronunciations, and in this case the network does need to learn them. A suitable compromise for the BEEP training data was found to be 2000 hidden units.

The neural network learning traditionally begins with all the weights and biases drawn from the same uniform distribution of random numbers [16]. However, more complicated set-ups are consistently found to emerge in studies in which different initial weight distributions are optimized for each part of the network by processes such as simulated evolution by natural selection [20, 21], so it makes sense to consider more carefully what is most appropriate here. As discussed above, when thinking in terms of having a good hierarchy of rules determining the correct phoneme outputs for each letter, it is natural to expect that it would be best to only use the long range context information when the more local context is not sufficient. For the neural network architecture of Figure 1, that means the average magnitudes of the weights connecting the more central input blocks to the hidden units should be higher than those connecting the less central input blocks. However, the learning algorithm determines those weight values to minimize the training data errors, and that will not necessarily result in a set of weights that implements a rule set that generalizes well. The network designer can guide the learning process, though, by making sure that the random initial values for the input to hidden unit weights are not set too high, so they are more likely to stay low until larger values are needed. This is also important for the incremental learning approach adopted here, since it minimizes the disruption caused when extra context blocks suddenly introduce additional random activations into the network each time the maximum word length is increased. The approach adopted here takes this idea to the extreme and has all the initial input to hidden layer weights start from zero, to minimize any unnecessary contributions from the outer context blocks. For similar reasons, it also starts all the hidden unit and output unit biases from zero. That only leaves the hidden unit to output weights starting from a traditional uniform distribution of random values in the range [-1,+1]. To establish the advantage of this variation, it will later be compared empirically with the more conventional approach of starting all the network weights and biases with a uniform distribution of random values in the range [-1,+1].

The final detail that needs optimization is the fraction θ of words that must be learned to within the output error tolerance at each stage of training before the maximum training data word length is incremented again. The initial expectation was that fairly high values of θ would work best, so the preliminary version of this study [22] used 0.95, and found that varying it up to 0.99, or down to 0.6, had no significant effect on performance. However, a more complete investigation for this paper revealed that

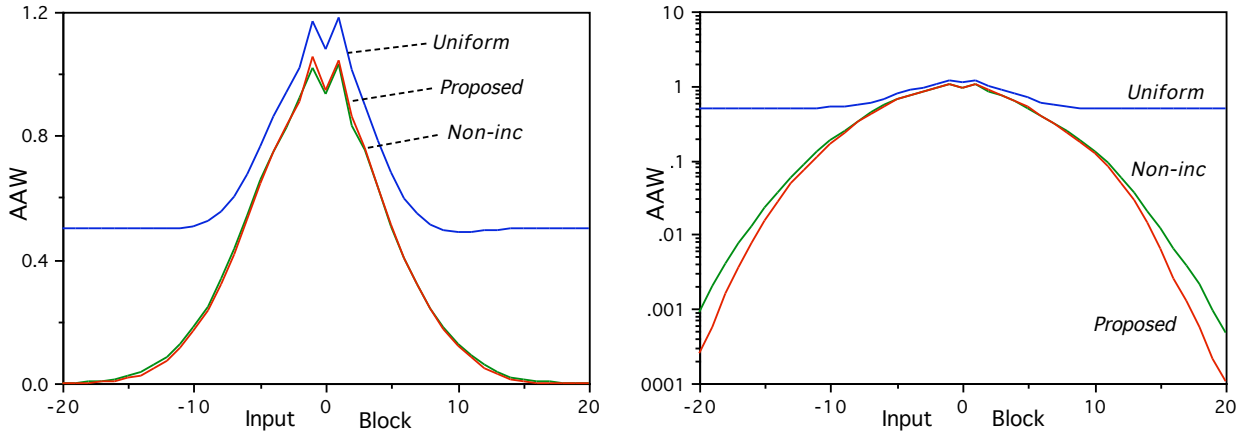

Figure 2. Average absolute weights (AAW) for each of the 41 input unit blocks, shown on a linear scale (left) and a log scale (right). The weights for the long range context blocks are lowest for the proposed network, more with non-incremental learning, and highest with a uniform initial weight distribution for the whole network.

using much lower values actually leads to small, though significant, improvements in performance, so $\theta = 0.2$ will be used as standard for the remainder of this paper.

5 Quantifying the Learning Performance

To test the large scale neural network approach described above, 198618 words from the BEEP dictionary [17] were used for training. These consisted of the same 198632 words used in the Damper et al. study [9], except for 14 very non-standard "words" that had many more phonemes than letters (e.g., REV" pronounced "r eh v ax r ax n d" and "UNIV" pronounced "y uw n ih v er s ih t iy"), which were excluded on the grounds that they were abbreviations that clearly deviated considerably from standard English pronunciation. Twenty independent runs of the standard set-up, using different random number seeds, and numerous further sets of runs designed to explore variations in the details and parameters specified above, all achieved 100% correct phoneme outputs on the training data.

When introducing non-standard features into neural network training, it is always important to check that they really do lead to the expected advantages. For that reason, the effect of the incremental training approach and the non-traditional initial weight distributions on the resulting learned weights were investigated. Figure 2 shows the average absolute weight distributions after training for the proposed networks, equivalent networks that learn from the complete set training words throughout (rather than using the incremental scheme with the shorter words largely learned before moving on to longer words),

and equivalent networks started with a uniform distribution for all the initial weights and thresholds (rather than only having non-zero initial values for the hidden to output weights). This confirms that the proposed approach, particularly the initial weight distribution, really does restrict the unnecessary use of long range context information when more local information is sufficient.

Naturally, it is really the quality of the resulting network outputs that is of primary importance here, not the weight distributions that lead to them. The crucial questions for this paper are: how good are the resulting alignments, and how well do they compare with the symbolic approach of Damper et al. [9]? There are two commonly used approaches for establishing alignment quality: intrinsic evaluation involving direct alignment comparison against a manually-constructed gold standard, and extrinsic evaluation involving comparisons of the resulting performances on the text-to-phoneme mapping task [2]. Generally, gold standard alignments are impractical for the very large datasets of interest here, so Damper et al. [9] took the extrinsic route, comparing their alignments by measuring the associated generalization performance on the Pronunciation by Analogy (PbA) system [1]. It is not totally obvious, though, that the associated pronunciation performance really is the best measure of alignment. All the neural networks represented in Figure 2, and all the others tested with less than optimal parameter values, achieved 100% on the training data, despite spanning a range of different alignment qualities, so good performance on the training data will clearly not be a reliable indicator. However, one would expect a more consistent or regular alignment to allow a better hierarchy of rules to emerge (particularly in a rule based system like PbA, but also in a neural network) and that will inevitably result in better generalization. That measure will therefore be used in the next section, but, first, a more direct measure of alignment quality will be used to give an alternative performance indicator that might be deemed more reliable.

A suitable grapheme to phoneme consistency measure has actually already been formulated by Wolff, Eichner & Hoffmann [23]. Given aligned strings of graphemes $g \in G$ and phonemes $f \in F$, one can easily compute the associated probabilities $p(g)$, $p(f)$ and $p(g,f)$, and thus determine the entropy

$$H = -\sum_{g \in G, f \in F} p(g,f) \log p(g,f)$$

and mutual information

$$I = \sum_{g \in G} \sum_{f \in F} p(g,f) \log \left(\frac{p(g,f)}{p(g)p(f)} \right).$$

Then the required measure of mapping consistency is simply their ratio $C = I/H$. A totally random G-F mapping will have $p(g,f) = p(g)p(f)$, leading to $I = 0$, and hence $C = 0$ (i.e., no consistency). Conversely, a perfect 1-1 mapping, with each grapheme associated with a single distinct phoneme, will have $I = H$,

Approach	C Average	C Std. Dev.	C Maximum	Ensemble C
Proposed NN	0.5631	0.0010	0.5645	0.5637
Inc. Step $\theta = 0.95$ NN	0.5606 **	0.0012	0.5619	0.5630
Non-incremental NN	0.5553 **	0.0055	0.5625	0.5560
Uniform initial weights NN	0.5612 **	0.0020	0.5637	0.5628
Approx MT at length 8 NN	0.5629	0.0010	0.5643	0.5636
Approx MT at length 4 NN	0.5624	0.0012	0.5637	0.5628
Approx MT throughout NN	0.4868 **	0.0076	0.5012	0.4934
Damper et al. [9]	0.5630	–	0.5630	–
Naïve Alignment	0.2276 **	–	0.2276	–

Table 1. Alignment consistencies C (average, standard deviation, maximum and ensemble average over 20 runs) for the proposed neural network, six representative variations of it, the alignment generated by Damper et al. [9], and the naïve alignment. Averages differing significantly (unpaired two-tailed t-test) from the corresponding proposed network are indicated by a * for $p < 0.05$, and ** for $p < 0.01$.

and hence $C = 1$ (i.e., perfect consistency).

Natural languages are never totally consistent, but never totally random either. Wolff et al. [23] have found consistencies around 0.75 for German and Dutch, and somewhat lower consistencies, around 0.65, for English. Large dictionaries like BEEP [17] also tend to include mixtures of regional variations and significant numbers of errors, so they will result in lower consistencies, even if the alignment is carried out as perfectly as possible. Moreover, even basic features like the sizes of the grapheme and phoneme sets can vary across different dictionaries or representations of the same language [9]. This makes it very difficult to perform absolute consistency evaluations, but the measure C can still be used to provide reliable comparisons of consistency for different alignments based on the same dictionary.

The neural network architecture, as shown in Figure 1, will map each individual letter to either zero, one or two phonemes. Thus. the natural "graphemes" G here are the letters, and the "phonemes" F should correspond to the set of possible outputs. This leads to the key neural networks studied achieving the alignment consistencies C shown in Table 1, with the averages, standard deviations, and maxima obtained from 20 runs. The proposed neural network here achieves a consistency of 0.5631, which is significantly better than the non-incremental learning approach (0.5553, unpaired two-tailed t-test, $p = 0.00004$), the uniform initial weight distribution approach (0.5612, $p = 0.0008$), and even the case that simply has the incremental threshold θ increased from 0.2 to 0.95 (0.5606, $p < 10^{-5}$). This provides further justification

of the non-standard features and parameters of the proposed approach.

A natural concern with the multi-target approach is the computational costs involved in choosing which of the many targets to train on, so faster variations that switch from the full multi-target learning approach to the simpler approximate "patch-up mechanism" at word lengths less than the standard 16 were tested. Switching at word length 8 does not give significantly different results to the standard case (0.5629, $p = 0.75$), and even switching as early as word length 4 does not quite make a significant difference (0.5624, $p = 0.06$). However, the full multi-target approach really is needed for the initial stages: attempting to use the "patch-up mechanism" throughout leads to a large reduction in consistency (0.4868, $p < 10^{-5}$). This provides justification that the simplified "patch-up" approach is a good enough approximation to the full multi-target approach, as long as it is not introduced too soon.

For comparison purposes, the consistency of the Damper et al. alignment [9] is also shown in Table 1. However, computing the consistency in this case is complicated by the fact that the alignments are specified by inserting both "null letters" and "null phonemes". The null phonemes act in the same way as they do in the neural network approach, but rather than allowing more than one output phoneme per letter as in the neural network approach, any extra phonemes need to be aligned with null letters. For example, the word BOX leads to the 3-letter mapping "B O X ⇒ b oh k+s" in the neural network approach, but the 4-letter mapping "B O X _ ⇒ b oh k s" in the Damper et al. [9] approach. In the latter case, it is most natural to take the grapheme set G to be the letters plus null, and the phonemes set F to be the actual phonemes plus null. That leads to a consistency C of 0.5560, which is rather poor compared to the neural network results. However, to have a fair comparison with the neural networks, one really needs to modify their representation to match that used by the neural networks. In particular, equivalent grapheme and phoneme sets (G and F) should be used. This can be done most easily by simply removing all the null letters, and assigning the phonemes associated with them to adjacent real letters, but even this is not straightforward. For null letters at the beginning or end of a word, there is no doubt which real letter the corresponding phoneme should be associated with. However, when they appear mid-word, it is not so obvious which real letter is appropriate. The strategy that matches the neural representation most closely has the extra phoneme associated with the real letter that comes first. One can then compute the consistency in exactly the same way as for the neural networks, and that results in the increased value of 0.5630, which is not significantly different to the average result for the proposed neural network.

The final result shown in Table 1 is the consistency of the naïve alignment that simply assigns one phoneme to each letter from left to right till there are no more phonemes left. This results in very low consistency (0.2276), and confirms the need to do something more sophisticated.

Given that the very different neural network and Damper et al. [9] approaches lead to remarkably similar consistencies, one might wonder if a ceiling level of performance has been reached by both approaches, or whether there remains scope for further improvements of the neural network approach. It is certainly clear from the details of the outputs for individual words that the two approaches are not resulting in exactly the same alignments. For example, the neural network does not allow a letter to map to more than two phonemes, and that is a restriction the Damper et al. approach does not suffer. For standard English, this limitation will have little effect, because letters virtually never need to map to more than two phonemes. However, the BEEP dictionary contains a number of entries that are clearly incorrect pronunciations, for example

DISGUSTEDNESS ⇒ d ih s ih n t r ax s t ih d n ax s

IMPROBABLENESS ⇒ ih m p r ae k t ih k ax b l n ax s

RECONGELATION ⇒ r eh k ax n g r ae ch uh l ey sh ax n

and errors like this will be difficult to avoid completely in any very large pronunciation dictionary [19]. For many such anomalies, it will be impossible with only two phonemes per letter to accommodate the extraneous phonemes at their appropriate positions, which can disrupt the alignment of the letters that are actually pronounced correctly, and consequently result in reduced consistency. Fortunately, the restriction of the networks to two output blocks is not a fundamental limitation of the multi-target learning approach. For example, exactly the same type of multi-target learning network has been demonstrated to successfully map individual phonemes to as many as four letters in a model of English spelling [11, 12]. Of course, the number of potential alignments, and hence target outputs, will rise rapidly as the number of output blocks is increased, so there are computational resource implications to consider. That, together with the relatively small number of serious errors in large dictionaries such as BEEP, means that allowing more than two output phonemes per letter is probably not the most fruitful route for seeking potential improvements.

A more promising route involves simpler optimizations of the neural network. As with all neural networks, the various details interact, and the key learning parameters (in particular, the learning rate, the training tolerance, and the incremental threshold θ) all need to be optimized together. A more exhaustive testing of the various combinations has already resulted in the performance levels shown above being a significant improvement over those reported in the preliminary version of this study [22]. Using an evolutionary algorithm to automate that kind of simultaneous optimization process is known to be a good strategy [20, 21], but the computational cost of doing that for the large-scale networks of interest here is

prohibitive. It remains possible that the parameter values and results presented in this paper are still not the best that can be achieved, but extensive explorations of variations in the learning details and parameter values have so far failed to identify any further significant improvements.

Another factor of relevance here is the inherent variance across runs of the neural network outputs arising from the various random factors involved, such as the different random initial weights and the different random orders of presentation of the training words. It has been demonstrated in numerous previous studies that ensembles of neural networks arising from independent training runs can be combined together as a voting committee machine to achieve improved performance over the average individual results, and sometimes even over the best individual results [21, 24, 25]. Such an approach can be expected to work well whenever the errors made by individual networks are relatively rare and uncorrelated, so they can be out-voted by the rest of the ensemble. The final column of Table 1 shows the consistencies C of the ensemble alignments for the main neural network variations. It can be seen that the ensembles do have a slightly better consistency than the average for each network type, but only in the case of the sub-optimal θ value are they better than the most consistent individual network. It seems that any remaining alignment errors are either too few or too correlated for the ensemble approach to be of significant advantage here.

6 Quantifying the Generalization Performance

The ultimate test of performance for text to phoneme mapping systems is their generalization ability, i.e. how well they perform on inputs they were not trained on. Damper et al. [9] used a leave-one-out cross-validation scheme to measure that, but training a new neural network for every word in the training set is clearly not a practical option. Instead, the generalization of the proposed neural network was measured using a standard 10-fold cross-validation approach. That involves randomly splitting the full set of training data into 10 subsets, training a new network on each of the ten combinations of nine subsets, and testing on the remaining subset.

Unfortunately, a simple application of the standard cross-validation approach can potentially provide misleading results here, because the whole concept of correct generalization is not straightforward. The first problem is that the full BEEP dataset actually contains multiple pronunciations for many words, and only the first for each word was selected to give a workable set for training without additional context information [9]. If part of that reduced word set is used for testing, it will obviously not include many acceptable pronunciations, and the generalization performance will appear poorer than it really is. The obvious remedy might therefore be to use the full set of pronunciations for testing purposes, but there is

still no guarantee that the BEEP dictionary contains the complete set of possibilities. In fact, given the range of acceptable regional variations in English pronunciation, and checking a small subset of the dictionary by hand, it seems likely that it is actually very far from complete. Moreover, as noted above, the BEEP dictionary contains a number of items of dubious accuracy [19], and there is little chance of any system generalizing to get those correct.

Most of the earlier smaller-scale psychological models of reading were trained on a standard set of only a few thousand mono-syllabic words, and it was feasible to test them on standard representative sets of a few hundred hand-crafted non-words, many of which did have a range of acceptable pronunciations [7, 8, 11]. It is clear that such a hand-crafted approach is never going to be feasible for testing the large-scale systems of interest here, and producing a reliable automatically generated set is effectively what the proposed neural network model is being expected to do.

One computationally feasible way to proceed would be to pick a particularly consistent set of alignment data and then use that to identify acceptable generalizations which may differ from the potentially under-representative pronunciation found in the test set. Clearly, it will not be sufficient to simply take the most frequently occurring phoneme for each letter. The first problem is that for many letters, particularly the vowels, there is no clear winning phoneme. For example, for the letter "A", the top three pronunciations found in the most consistently aligned dataset produced in this study are the phonemes "ae" (30%), "ax" (25%) and "ey" (17%), leaving 28% for the others. Moreover, simply allowing the most common correspondences will not encompass many of the widely accepted rules of pronunciation, such as the length of the preceding vowel sound depending on the presence or absence of a final letter "E", e.g., "MAT" being pronounced "m ae t", while "MATE" is pronounced "m ey t".

A compromise strategy involves working with a whole series of generalization measures, that begins with only allowing the specific pronunciation listed in the test set, and then becomes increasingly lenient about what variations are allowed. This can easily be implemented by truncating the frequency ordered list of phoneme alignments for each letter at the point at which the total coverage (as a proportion of all mappings) is K, and accepting any of the phonemes in that list. Only counting the test set pronunciation as correct will correspond to $K = 0$. Then using $K = 0.2$ also allows the most common phoneme for each letter, that is a total of up to 26 extra matches. Increasing as far as $K = 0.8$ allows multiple possibilities for all the vowels and some consonants ("C" and "S"), totaling 51 matches in all. By $K = 0.99$, the only letter still matching a single phoneme is "V", and there are 121 allowable extra matches in total, none of which are obvious errors. For the extreme $K = 1.00$, there are 838 matches in total, many of which would be considered totally unacceptable by most English speakers. Obviously, the number of generalizations

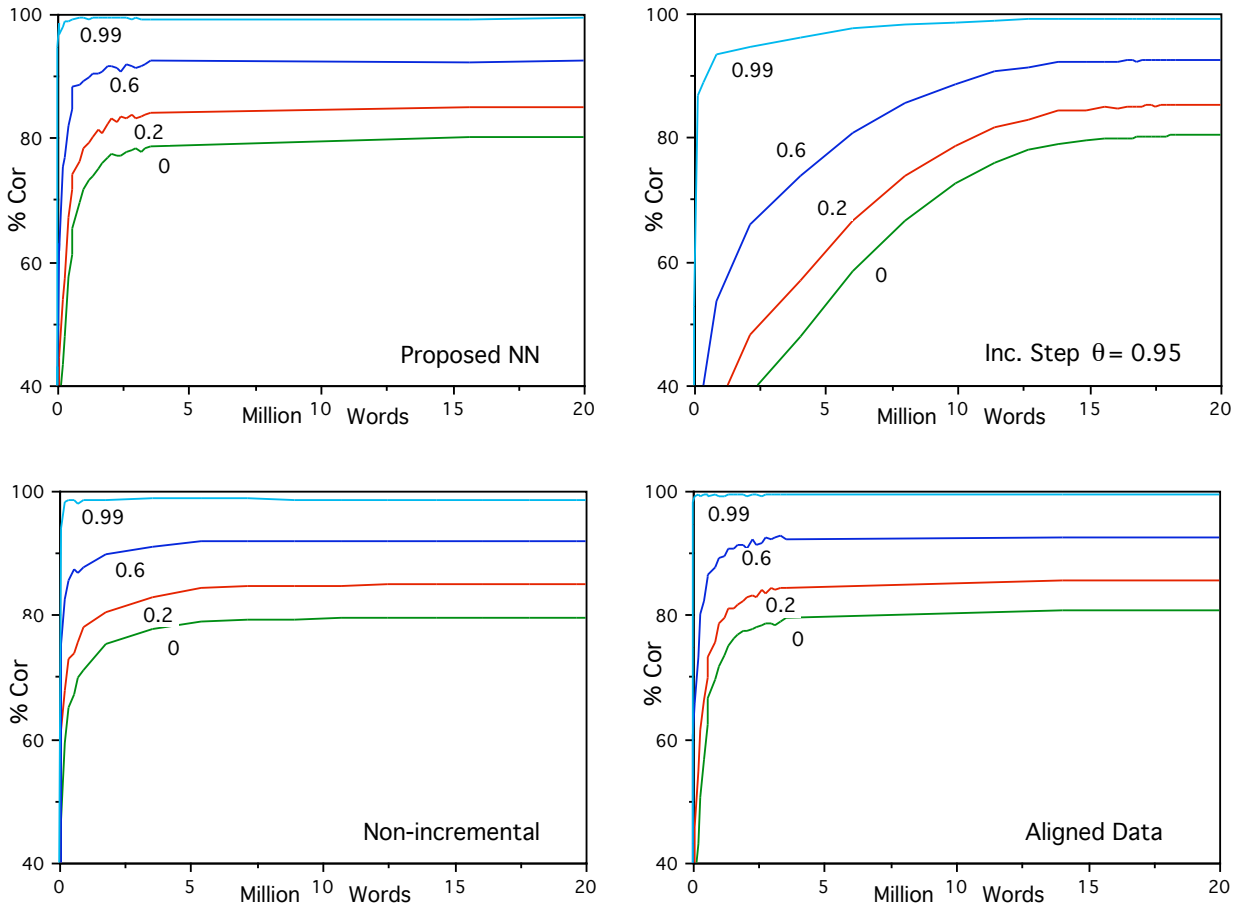

Figure 3. Typical progression of the generalization performance (percentage correct) during training of the proposed network and three key variations, for leniency parameter K = 0, 0.2, 0.6, 0.99, as a function of the number of training word presentations.

being counted as correct will increase with K, and there is no guarantee that the allowed variations follow the normally accepted pronunciation rules, but it does enable individual users to choose which cut-off point they consider most appropriate for their particular purposes.

The improving generalization performances during typical training runs are shown in Figure 3 for the proposed network with standard incremental step threshold θ = 0.2, the same network with the much higher θ = 0.95, the non-incremental variation of the proposed network, and the proposed network with the training data pre-aligned. The incremental step criterion θ clearly has a large effect on the speed of the learning process, but the final performances are remarkably similar for all four cases. Having access to the whole training dataset at once in the non-incremental case does lead to a faster initial improvement in performance for all values of the leniency parameter K, but by the end of training, the incremental

Approach	$K = 0$	$K = 0.2$	$K = 0.4$	$K = 0.6$	$K = 0.8$	$K = 0.9$	$K = 0.99$
Proposed NN	80.35	85.52	88.17	92.61	95.10	96.96	99.26
Inc. Step $\theta = 0.95$	80.67 *	85.40	87.98	92.35	94.85	96.65 *	98.95 *
Non-incremental	79.49 **	84.61 **	87.27 **	91.06 **	93.66 **	95.39 **	97.64 **
Uniform initial weights	80.08	85.12 *	87.85 *	92.19 *	94.75 *	96.63	99.03
Approx MT at length 4	80.26	85.42	88.15	92.58	95.09	96.97	99.26
Aligned data	80.36	85.42	88.18	92.53	95.12	96.92	99.29
Aligned data, Non-inc.	79.77 *	84.96 **	87.74 *	92.26 **	94.92	96.80	99.29
Aligned data, Uniform	80.06 *	85.18 *	87.91 *	92.40 *	95.05	96.87	99.29
Damper et al. data	79.31 **	84.85 **	87.68 **	92.40 *	95.15	97.18 *	99.28
Naïve aligned data	52.97 **	58.90 **	60.71 **	63.63 **	65.73 **	69.53 **	72.98 **

Table 2. Generalization performances (percentages correct) of the proposed neural network and the key variations for a range of leniency parameters K. Values differing significantly (unpaired two-tailed t-test) from the corresponding proposed network results are indicated by a * for $p < 0.05$, and ** for $p < 0.01$.

approach is performing slightly better.

A more rigorous comparison of the generalization results is provided by Table 2, which presents the final performances of the key neural network variations for a range of values of the leniency parameter K. Overall, nothing does better than the standard proposed network. However, only training the standard network on the naïve aligned data leads to performance that is much worse, with very large (>25%) and highly significant ($p < 10^{-5}$) reductions compared with the proposed approach. Increasing the incremental step threshold to $\theta = 0.95$ has little effect, with marginally better results for the $K = 0$ case, and marginally worse for $K \geq 0.9$. Switching to the non-incremental version leads to small (~1.0%), but significant ($p < 0.01$), reductions in performance compared to the proposed approach, for all values of K. Using a uniform initial weight distribution results in a surprisingly small performance reduction (~0.5%), with the individual differences only reaching significance ($p < 0.05$) for intermediate values of K. Approximating the later stages of the full multi-target learning approach by the faster "patch-up mechanism" does not make a significant difference, even when it is introduced as early as word length four. This pattern of generalization performances is in line with the alignment consistencies seen in Table 1, providing further confidence that the consistency measure C really is a useful and reliable indicator of the alignment quality and of the level of generalization performance that can be expected.

An important remaining question is whether the performance of the proposed multi-target learning neural network is as good as that of an equivalent neural network that learns from the best pre-aligned

training data throughout. Table 2 shows three sets of results obtained by training on data with the best alignment previously achieved by the multi-target approach. For the standard proposed network, there is no significant difference in the generalization for any value of K, confirming that performing the alignment and mapping together gives the same results as performing them separately. Using the incremental learning approach is still important when using pre-aligned training data, but the reduction in performance due to not using it is much smaller than for the non-aligned training data case. For networks with a uniform distribution of initial weights, there is no significant difference between the pre-aligned and non-aligned training data cases, and the pre-aligned case still suffers a marginal performance reduction compared with the standard proposed neural network using non-aligned training data.

The final comparison of interest is with the alignments achieved by the symbolic approach of Damper et al. [9]. Unfortunately, testing their alignments by training the proposed neural network on them is not straightforward, because of the significant number of words that are aligned by introducing null letters. As discussed above, the same words presented related problems in performing fair comparisons using the alignment consistency measure C. Accommodating those words in the neural network representation requires the null letters to be removed and the associated phonemes treated as additional phoneme outputs for the adjacent real letters. However, that results in a some letters needing to be mapped to more than two phonemes, which cannot be accommodated by the two output blocks of the neural network. That is relevant for 3625 words, and if all those words are simply removed from the dataset, the Damper et al. [9] alignment consistency C increases from 0.5630 to 0.5685, which suggests that doing so would give their alignment an unfair advantage over the neural network alignment. Nevertheless, the resulting generalization comparisons as shown in Table 2 are mixed, with the Damper et al. performance significantly worse (by up to ~1.0%) for the $K \leq 0.4$ cases, marginally worse (~0.2%) for $K = 0.6$, and marginally better (~0.2%) for $K = 0.9$. Being unfair in the other direction, and simply counting all the removed words as being incorrect, reduces all the Damper et al. results in Table 2 by a factor of 0.9818, rendering them significantly worse than the proposed neural network results for all values of K.

The standard ($K = 0$) generalization performances achieved by the neural networks are typically in the region of 80%, which seems rather low compared to the earlier small scale psychological models [11]. However, that is not surprising given what was noted above about the test sets not containing the full set of acceptable outputs, nor even the most regular acceptable output, for each word. Damper et al. [9] managed to achieve around 86% with their leave-one-out cross-validation scheme, but that involved using a reduced dataset because, like the proposed neural network, the PbA pronunciation system they used

could not accommodate "null letters". They proceeded by omitting all the words that had alignments involving "null letters", which accounted for about 10% of the full word set, and those missing words are the very ones that any alignment and mapping system would be most likely to find difficult to accommodate. In fact, with those problematic words removed, the Damper et al. [9] alignment consistency *C* increases from 0.5630 to 0.5711, providing confirmation that the reduced word set is likely to over-estimate the generalization performance on the full word set. Damper et al. [9] noted that if all the omitted words were counted as incorrect, they would achieve a generalization performance of 76.1%, which they considered "a very respectable result on such a sizeable dictionary". For the proposed neural network approach of this paper, no such word removals are needed, so it appears that its generalization results are actually better than those of the existing symbolic approaches.

7 Conclusion and Discussion

This paper has demonstrated how earlier neural network based psychological models of reading [10, 11, 13] can be scaled up to cope with the much larger dictionaries and word lengths required for modern speech technology. These neural networks automatically learn the text-to-phoneme alignment and mapping simultaneously, with no performance reduction compared to learning the alignment and mapping separately. Key details of the proposed approach (such as non-standard initial weight distributions, incremental training regimes, computational speed-ups, and different parameter values) have been explored, with the best specifications identified. The resulting text-to-phoneme alignment consistencies and pronunciation generalization performances have been shown to be at least comparable to the existing state-of-the-art symbolic rule-based systems [5, 9].

There remain numerous potential variations of the standard approach presented in this paper. For example, one might expect to obtain a more consistent mapping and better generalization by initially training the networks on a subset of the training data that has the most consistent letter-phoneme mappings, and only training on the remaining words after those have been learned. The difficultly is in deciding what to use as the initial consistent training data subset. If only the most common phoneme were allowed for each letter, there would clearly be 26 allowed alignments, and the consistency *C* of the allowed training words would be high, but there would be very low coverage of the whole training data set. As Figure 4 shows, the more alignments that are allowed (e.g., by increasing the parameter *K* discussed previously), the lower the consistency, and the greater the coverage of the whole set of training data. A reasonable compromise uses $K = 0.8$, which corresponds to 51 alignments (i.e. an average of about two phonemes per letter), $C = 0.7332$, and 29.6% coverage. The slightly counterintuitive result of

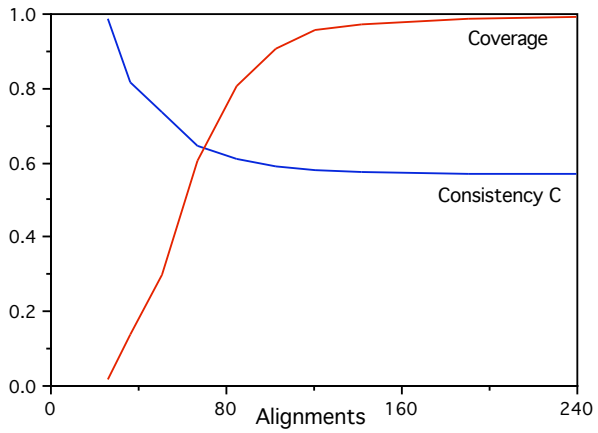

Figure 4. Increasing the number of allowed letter-phoneme alignments, from 1-1 to as many as needed, results in increasing coverage of the training data set but decreasing alignment consistency *C*.

pre-training on this high-consistency data subset is that the alignment consistencies *C* of the final trained networks are actually reduced rather than increased. The average final *C* over the ten cross-validation sets is 0.5591 for this two stage approach, compared with 0.5620 for the standard approach. The difference is small, but it is highly significant ($p = 0.0002$). Investigating the individual mappings learned by the networks reveals that forcing them into a fixed consistent framework in the initial stage acts to restrict the flexibility for accommodating the less regular mappings required later, and that reduces the final overall mapping consistency. This reduction in alignment consistency is also reflected in the generalization results, with the two-stage approach slightly, but significantly, worse for all $K \geq 0.2$, and not significantly different for $K = 0$.

In some ways it is reassuring that more complex training regimes like this, which one might think would be required to achieve the best performances, are not actually necessary, and that the whole neural network approach is so robust with respect to its details. Even when there are significant performance reductions due to not incorporating a certain feature, such as the proposed incremental learning scheme, the resulting differences in generalization performance are rarely more than about one percent. It is also encouraging to find that employing relatively fast approximations to the full multi-target training approach does not significantly degrade the resulting performance, rendering the whole approach not much more computationally intensive than a standard back-propagation network.

So far, no further variations of the standard neural network approach proposed in this paper have been found to lead to improved results. Obviously, there remain other possible variations that have yet to be explored, but the similarity of the alignment consistencies and generalization results from the key

variations studied suggests that both the neural network and the symbolic approaches are now operating at close to ceiling performance levels for this particular task. Even though further refinements of both approaches might be possible, there is currently no indication that either will end up with substantial performance advantages over the other. The fairly straightforward neural network approach of this paper, with its ability to learn the alignment and mapping simultaneously, will be considered by many to be a simpler and superior approach. However, given the long history of the symbolic versus neural network debate [15], it is seems unlikely that this paper will have the last word on this matter. Nevertheless, it will certainly require very little extra effort now to apply the ideas and neural networks presented in this paper to related tasks, such as alternative English dictionaries, other languages, and even the reverse mapping (i.e. phonemes to text as required for spelling).

Acknowledgements

A shorter preliminary version of this study was published in the proceedings of the IJCNN 2011 conference [22].

References

1. Damper RI, Marchand Y, Adamson MJ & Gustafson K (1999) Evaluating the pronunciation component of text-to-speech systems for English: A performance comparison of different approaches. *Computer Speech and Language*, **13**, 155-176
2. Jiampojamarn S & Kondrak G (2010) Letter-phoneme alignment: An exploration. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 780-788, ACL
3. Reichel U (2012) Perma and Balloon: Tools for string alignment and text processing. In: *Proceedings of the 13th Annual Conference of the International Speech Communication Association*, Paper 346, ISCA
4. Stan A, Bell P & King S (2012) A grapheme-based method for automatic alignment of speech and text data. In: *Proceedings IEEE Workshop on Spoken Language Technology*, 286-290, IEEE
5. Davel M & Barnard E (2008) Pronunciation prediction with Default&Refine. *Computer Speech and Language*, **22**, 374–393
6. Sejnowski TJ & Rosenberg CR (1987) Parallel networks that learn to pronounce English text. *Complex Systems*, **1**, 145-168
7. Seidenberg MS & McClelland JL (1989) A distributed, developmental model of word recognition and naming. *Psychological Review*, **96**, 523-568
8. Plaut DC, McClelland JL, Seidenberg MS & Patterson KE (1996) Understanding normal and impaired word reading: Computational principles in quasi-regular domains. *Psychological Review*, **103,** 56-115
9. Damper RI, Marchand Y, Marsters JDS & Bazin AI (2005) Aligning text and phonemes for speech technology applications using an EM-like algorithm. *International Journal of Speech Technology*, **8**, 149-162
10. Bullinaria JA (1993) Neural network models of reading multi-syllabic words. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN 1993),* 283-286. Piscataway, NJ: IEEE
11. Bullinaria JA (1997) Modelling reading, spelling and past tense learning with artificial neural networks. *Brain and Language,* **59**, 236-266
12. Bullinaria JA (1994) Connectionist modelling of spelling. In: *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society,* 78-83. Hillsdale, NJ: Erlbaum
13. Bullinaria JA (1995) Neural network learning from ambiguous training data. *Connection Science*, **7**, 99-122

14. Wickelgren WA (1977) Speed-accuracy tradeoff and information processing dynamics. *Acta Psychologica*, **41**, 67-85
15. Pinker S & Prince A (1988) On language and connectionism: Analysis of a parallel distributed model of language acquisition. *Cognition*, **28**, 73-193
16. Bishop CM (1995) *Neural Networks for Pattern Recognition*. Oxford, UK: Oxford University Press
17. Robinson A (1997) *British English Example Pronunciation Dictionary (BEEP),* Version 1.0, Cambridge University
18. Hare M & Elman JL (1995) Learning and morphological change. *Cognition*, **56**, 61-98
19. Martirosian OM & Davel M (2007) Error analysis of a public domain pronunciation dictionary. In *Proceedings of the 18th Annual Symposium of the Pattern Recognition Association of South Africa (PRASA*), 13-16
20. Bullinaria JA (2003) Evolving efficient learning algorithms for binary mappings. *Neural Networks*, **16**, 793-800
21. Bullinaria JA (2007) Using evolution to improve neural network learning: Pitfalls and solutions. *Neural Computing & Applications*, **16**, 209-226
22. Bullinaria JA (2011) Text to phoneme alignment and mapping for speech technology: A neural networks approach. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2011)*, 625-632. Piscataway, NJ: IEEE
23. Wolff M, Eichner M & Hoffmann R (2002) Measuring the quality of pronunciation dictionaries. In *Proceedings of the ISCA Tutorial and Research Workshop on Pronunciation Modeling and Lexicon Adaptation (PMLA),* 117-122
24. Battiti R & Colla AM (1994) Democracy in neural networks: Voting schemes for classification. *Neural Networks,* **7**, 691-709
25. Hansen LK & Salamon P (1990) Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, 993-1000