Combining Frame and Segment Level Processing via Temporal Pooling for Phonetic Classification

Sumit Chopra, Patrick Haffner, Dimitrios Dimitriadis
AT&T Labs-Research, 180 Park Ave, Florham Park, NJ
{schopra,haffner,ddim}@research.att.com

Abstract

We propose a simple, yet novel, multi-layer model for the problem of phonetic classification. Our model combines a frame level transformation of the acoustic signal with a segment level phone classification. Our key contribution is the study of new temporal pooling strategies that interface these two levels, determining how frame scores are converted into segment scores. On the TIMIT benchmark, we match the best performance obtained using a single classifier. Diversity in pooling strategies is further used to generate candidate classifiers with complementary performance characteristics, which perform even better as an ensemble. Without the use of any phonetic knowledge, our ensemble model achieves a 16.96% phone classification error. While our data-driven approach is exhaustive, the combinatorial inflation is limited to the smaller segmental half of the system.

Index Terms: deep network, multi-layer perceptron, ensemble method, phonetic classification, TIMIT.

1. Introduction

Over the last few years, there has been renewed interest in improving the performance of segmented phoneme classification, using the TIMIT benchmark [1, 2]. We can identify two motivations behind this. First, to study, in isolation, improvements in acoustic modeling with the goal to later combine them with Markov modeling. This leads to framed based approaches, where the focus is on defining a frame-level transformation of the acoustic parameters that outperforms standard Gaussian modeling. The idea is to compute one feature vector per time frame and perform temporal integration over the phone state scores, using algorithms like the Viterbi or the Baum-Welsh. While at train time, supervision may happen at the phone segment level. Whereas at test time, the frame scores can be used by an HMM. Several of the latest paradigms in machine learning have been tried, including Hidden Conditional Random Fields [3], and Compressed Sensing [5]. The classification performance of all of these methods hovers above 20% error. Some of these studies were successfully extended to continuous phone recognition [6, 7].

The second motivation is to show that a segmental approach could significantly outperform Markov modeling. Such segment based approaches [8, 9, 10] assume that the phoneme boundaries are known at test time. Acoustic features are directly optimized at the segment level, usually by temporal integration, and processed through a static architecture that has no concept of time. More recently, with the help of both phoneme hierarchies and ensemble methods, a classification error of 16.7% was achieved [11]. Despite their reliance on phonetic expertise, one major drawback of such segmental approaches is the lack of a clear path to continuous speech recognition tasks. In this context, another algorithm should be employed that produces candidate segments and searches for the optimal segmentation. While such an approach has been successfully deployed in Optical Character Recognition [12], a combinatorial explosion of segment candidates is observed, requiring segmental classifiers that share as much computation as possible.

This paper proposes a multi-layer model that combines both frame-based and segment-based processing, interfaced via a temporal pooling layer. In this work, we focus on the pooling strategies and their impact on performance. Without relying on any expert phonetic knowledge the performance of our model is significantly better than any frame based models proposed to date, and is comparable to the best segmental approaches. The proposed model is more practical in the sense that it is quite modular, using spectral acoustic features as input, a Time Delay Neural Network (TDNN) at frame level, a fully connected neural network at the segment level, with a temporal pooling architecture in between. These modules could be further optimized for the task at hand. Furthermore, our model offers a clearer path for extension to phone recognition. Majority of runtime computation happening at the frame level enables a connected speech recognition implementation where this computation would be shared. This is also, to our knowledge, the first study of learnable temporal evidence integration models that go beyond the simple summation done through dynamic programming (Viterbi or Baum-Welsh) or the direct extraction of segmental features.

2. Multi-Layer Architecture

Our architecture combines both the frame based and segment based approaches into two levels, with a temporal integration in-between (called the pooling interface) and is represented as a multi-layer Perceptron (MLP). More formally, we have combined a frame processing unit and a segmental classification unit with a pooling interface unit, in between. Amongst the novelty of this work is the pooling interface unit that offers much more flexibility than the temporal integration procedures used in either segmental or frame-based approaches.

2.1. The Frame Processing Unit (FPU)

Our frame processing unit consists of a temporal filter bank followed by a point-wise non-linearity. Its input consists of a feature sequence associated with every frame extracted from pre-segmented phone segments. The details of the features used are discussed in Section 3. Let us denote by \( \mathbf{x}_j \) the \( d \) dimensional feature vector associated with the \( j \)-th frame of the phone segment. If \( T \) is the number of frames associated with this segment, then the input feature sequence can be represented as a matrix \( \mathbf{X} \in \mathbb{R}^{d \times T} \), such that, \( \mathbf{X} = [x_1 x_2 \ldots x_T] \), where the
j-th column of $X$ consists of the feature vector $x_j$. The functionality of this layer resembles that of a standard Time Delay Neural Network (TDNN) that takes as input a two dimensional feature map (the matrix $X$) and produces, as output, another two dimensional feature map $Z$. Herein, time refers to the notion of order associated with the frame sequence. In particular, if $i$ is the index of the filter, then the value associated with the output feature map at time $t$ is a convolution using a filter of width $k$: $z_i \leftarrow \sum_{j=1}^{k} w_{ij} \cdot x_{i+j}$, $1 \leq t \leq T - k + 1$, where $w_{ij} \in \mathbb{R}^d$ ($1 \leq j \leq k$) are the parameters associated with the $i$-th filter. If $h$ is the number of filters in the filter bank layer, then the above operation can be written as $z_i = \sum_{j=1}^{h} W_j \cdot x_{i+j}$, where $W_j \in \mathbb{R}^{d \times d}$, $1 \leq j \leq k$ are the parameter matrices, and $z_i \in \mathbb{R}^e$ is the hidden representation of the input at time $t$. In summary, the output of this layer is another two dimensional feature map $Z = [z_1 \ z_2 \ldots \ z(T-k+1)]$, with $Z \in \mathbb{R}^{e \times (T-k+1)}$. Finally, each element of this matrix is passed through a point-wise non-linearity: a hyperbolic tangent function $y_{ij} = \tanh(z_{ij})$. This architecture element is denoted $P_T$.

2.2. The Pooling Interface Unit (PIU)

The pooling interface unit accumulates frame-level scores over specific segment of times. Typically the objective of pooling, as in case of TDNN, is to extract time invariant features. Here we refine the objective to detecting phonetic events during a given period of time, with the choice that this event is detected continuously (e.g. vowels) or over a short burst of time (e.g. plosives).

Recall that $T-k+1$ is the number of temporal features generated by the temporal filter bank layer, i.e., the number of columns of the matrix $Z$. We divide $Z$ along its columns into $S$ non-overlapping contiguous sub-matrices with the number of columns in the ratio of $\alpha_1 : \ldots : \alpha_S$ with $\sum \alpha_i = 1$. After selecting the partitioning, we have the choice over several pooling strategies: (i) **Average Pooling ($P_A$):** This operation applied to $k$-th such sub-matrix ($k \in \{1, \ldots, S\}$) results in a feature vector $y^k \in \mathbb{R}^{d_i}$, given by $y^k = \frac{1}{\alpha_k} \sum z^k_i$ where $d_i$ is the frame size of segment $S$ (essentially a row-wise average of sub-matrix $k$). When this operation is applied to all the $S$ sub-matrices, it results in a feature vector $y \in \mathbb{R}^{S \times d}$ obtained by concatenating the outputs from the $S$ sub-matrices. (ii) **Rectified Average Pooling ($P_R$):** Before averaging to every component of its input, this module applies the absolute value function and returns the result as output: $y_{ij} = |z_{ij}|$. (iii) **Max Pooling ($P_M$):** The max pooling module is similar to the average pooling module, except that the weighted average operation over the $S$ sub-matrices is replaced by a max operation. The function of this layer can be interpreted as the extraction of the most relevant features over time according to a given feature map.

2.3. The Segmental Classification Unit (SCU)

The classification unit takes as input the feature vectors generated by the underlying pooling interface and produces as output a vector of class probabilities, associated with each phone class. In our experiments this layer was composed of multiple (zero, one, or two) standard perceptron layers, one stacked on top of the other (top part of Figure 1). Each such perceptron layer consists of a linear module followed by a tanh non-linearity applied component-wise to the output of the linear module. The final perceptron layer is followed by a fully connected linear layer with size equal to the number of classes. Finally the output of this layer can be interpreted as the extraction of the most relevant features over time according to a given feature map.

2.4. Training the Model

Let us denote by $S = \{X_i, t' : i = 1, \ldots, N\}$, the training set consisting of the input-output pairs. Each input $X_i$ is a sequence of features vectors corresponding to the segmented phone. The label associated with the corresponding phone segment is denoted by $t'$. The multi-layer architecture discussed in Section 2, and shown in Figure 1 can be represented as a non-linear parametric function $G_M$, with parameters $\mathcal{M}$, that takes as input the features $X_i$ and produces a vector of class probabilities. The likelihood associated with the $i$-th training sample under the model is given by $p_i = [G_M(X_i)]_i = e^{y_i}/\sum_j e^{y_j}$. The likelihood of the entire training set is given by $L = \prod_{i=1}^{N} p_i[G_M(z^i)]_i = \prod_{i=1}^{N} \frac{e^{y_i}}{\sum_j e^{y_j}}$. The system training involves adjusting the parameters $\mathcal{M}$ to maximize this likelihood. This is achieved by minimizing the negative log likelihood loss with respect to the parameters $\mathcal{M}$, using a standard stochastic gradient descent algorithm.

3. Experiments and Results

The performance of our models was evaluated on the TIMIT phone classification task. In this task, the boundaries of the phone segments are known and the goal is to identify the phone name associated with each segment. The standard grouping scheme proposed in [1] was used to map the 61 phone classes onto 39 classes. For training we used the standard NIST training set consisting of 3,096 utterances from 462 speakers. For testing we used the *core test set* consisting of 192 utterances from 24 speakers. For preliminary choices and experiments with selective ensembles, a validation set was used. It combines the NIST development set consisting of 400 utterances.
from 50 speakers and the portion of the full test set remaining after removing the speakers belonging to core test set from it.

Given the utterance along with phone boundaries and their labels, initial acoustic features were extracted for every phone segment using log-filterbank Teager-Kaiser energy coefficients [13]. The energy coefficients, the log-Mean Teager Energy coefficients, i.e., \( \log (\langle \Psi [s(t) * g_j(t)] \rangle) \), are extracted using a mel-spaced Gabor filterbank, with 25 filters and the 3db-bandwidth overlap of 50%. Preliminary experiments show that, on clean speech data such as TIMIT, the performance improvement over standard MFCC coefficients is not significant.

This feature extraction phase resulted in a total of 142910 tokens for the training set and 7333 tokens for the core test set. Independent of the phones, frames from all phone segments in the training set were collected together to compute the mean and standard deviation of each of the \( d \) frame dimensions. Then, the data set was standardized so as to have zero mean and unit standard deviation along each dimension. These standardized features for every pre-segmented phone were supplied as input to the models. Following previous work [9], an initial temporal partitioning \((\alpha_1 : \alpha_2 : \alpha_3) = (3 : 4 : 3)\) was considered.

Table 1: Error rate on the core test set for the TIMIT phone classification task of a number of models. The performance of Hierarchical LM-GMM is reported using their best single classifier not selected using the test error, with glottal stops (/h/) removed resulting in a subset consisting of 7215 frames.

<table>
<thead>
<tr>
<th>ARCHITECTURE</th>
<th>TEST ERR(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM [9]</td>
<td>22.4</td>
</tr>
<tr>
<td>HIDDEN CRF [3]</td>
<td>20.8</td>
</tr>
<tr>
<td>LARGE MARGIN GMM [2]</td>
<td>21.1</td>
</tr>
<tr>
<td>COMRESSED SENSING [5]</td>
<td>20.0</td>
</tr>
<tr>
<td>HETEROGENEOUS GMM [10]</td>
<td>21.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>H0</th>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F_T - P_A)</td>
<td>21.06</td>
<td>19.58</td>
<td>19.68</td>
</tr>
<tr>
<td>(F_T - P_M)</td>
<td>22.77</td>
<td>20.29</td>
<td>20.17</td>
</tr>
<tr>
<td>(F_T - P_R)</td>
<td>21.48</td>
<td>21.34</td>
<td>21.41</td>
</tr>
</tbody>
</table>

While the various hyper-parameters associated with the model, such as the number and size of the filters in the frame processing unit, the number of hidden units in various layers in the classifier, their learning rates, the number of epochs, etc., should be chosen using the process of cross validation, we have found that the model performance on the validation set was fairly robust with respect to these parameters, hence they are kept fixed across all the experiments. In all reported experiments except first 3 lines of Table 3, no validation set was used.

The three factors which have a significant influence on the classification performance are the number of layers used in segmental classifier, the type of pooling interface used, and the point of initialization of the parametric function. While we report this impact, we do not add this choice to diversify our ensemble method. Table 1 summarizes the classification performance on the core test set of a number of architectures reported in the literature and the various combinations of our model. Our best performing PIU - SCU combination achieves the state-of-the-art performance of 19.58%. Another interesting result that comes out of the table is that even with a simple linear classifier the performance of our model is quite reasonable. This points towards the quality of the frame based features extracted by our frame processing unit.

3.1. Ensemble Architectures

Since both the FPU and SCU are highly non-linear, the performance of the model is sensitive to the point of initialization in the parameter space. Depending on the initialization the system could potentially get stuck in a “sub-optimal” local minimum. To circumvent this problem we have trained five independent models for each PIU - SCU combination. The training of each instance is started from a different initial point in the parameter space. For a particular feature PIU - SCU combination, the final model is obtained by taking an ensemble of the five instances trained for that combination. Our ensemble scheme involves taking a simple majority vote across the instances. However, it is easy to use any other complex scheme to combine. The results for each of the model combinations are reported in Table 2. Clearly we gain a significant performance boost using such a simple ensemble, reducing the classification error to 18.49%.

Table 2: Error rate using the ensemble of five different instances of each for each pooling units classifier pairs.

<table>
<thead>
<tr>
<th></th>
<th>H0</th>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F_T - P_A)</td>
<td>19.85</td>
<td>19.24</td>
<td>18.95</td>
</tr>
<tr>
<td>(F_T - P_M)</td>
<td>21.60</td>
<td>19.29</td>
<td>18.84</td>
</tr>
<tr>
<td>(F_T - P_R)</td>
<td>19.71</td>
<td>18.77</td>
<td>18.49</td>
</tr>
</tbody>
</table>

Looking at both Tables 1, and 2, a number of other interesting observations can be noted. The overall best performing ensemble model uses the \(F_T - P_R\), where as any single instance of \(F_T - P_R\) performs worse than other feature extractors. This indicates that each instance of \(F_T - P_R\) learns a different subset of the training space, and their combination results in a system which spans the largest subset of the training space.

3.1.1. Ensemble Across Pooling Interface Units

There is no reason to restrict oneself to a single type of pooling unit while creating an ensemble. Based on our hypothesis that different pooling strategies work for different types of phonemes, by potentially learning a subset of the training space with different properties, we create ensembles using instances from different types of pooling units. To study the effect of aggregating instances across different pooling architectures, we have performed the following experiments: For every pair of pooling units, say \((PIU_1, PIU_2)\), we made an ensemble consisting of \(n_1\) instances from \(PIU_1\) and \(n_2\) instances from \(PIU_2\), such that \(n_1 + n_2 = 5\). We have created a number of different such ensembles by varying the values of \(n_1\) and \(n_2\), and computing the performance of each. Among the five instances associated with each of the two pooling unit pairs, the combination which performed best on the validation set is picked as the final ensemble.

Table 3: Error rate using ensembles obtained by taking different number of instances from different pairs of pooling units. The last row gives the error rate of ensemble of all pooling units.

<table>
<thead>
<tr>
<th></th>
<th>H0</th>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F_T - P_A, F_T - P_M)</td>
<td>19.77</td>
<td>18.68</td>
<td>18.10</td>
</tr>
<tr>
<td>(F_T - P_A, F_T - P_R)</td>
<td>19.07</td>
<td>18.12</td>
<td>18.00</td>
</tr>
<tr>
<td>(F_T - P_M, F_T - P_R)</td>
<td>19.41</td>
<td>18.45</td>
<td>17.83</td>
</tr>
<tr>
<td>(F_T - (P_A, P_M, P_R))</td>
<td>19.14</td>
<td>18.06</td>
<td>17.56</td>
</tr>
</tbody>
</table>

For illustration purpose, in Figure 2 we plot the performance on the core test set of the pair \((F_T - P_A, F_T - P_R)\), for
different values of $n_1$ and $n_2$. From the figure, there is a “sweet spot” which involves optimal number of instances from both the pooling units (in this case 2 and 3 respectively) which yields the best performance. First three rows of Table 3 gives the performance on the core test set using the ensemble consisting of the combinations which performed best on the validation set for all the pooling pairs. By comparing Tables 2 and 3, one can conclude that creating an ensemble using instances from different types of pooling units is better than just using instances from a single pooling unit.

A reasonable next step is to create an ensemble consisting of all the five instances from all three types of pooling units. Last row of Table 3 reports the performance using such an ensemble for the three types of classification units. Our best classification error with a single phone partition is 17.56%.

3.1.2. Ensemble Across Different Phone Partitions

The next step is to try the previously discussed architectures using different partitioning of the phone segments. In particular, in addition to the (3 : 4 : 3) partitioning, we partitioned the phones in three additional ways: (5 : .25 : 25), (25 : .5 : 25), and (25 : .25 : .5). For each such partition we trained 5 different models starting with different set of initial parameters. For each PUI-SCU pair, this resulted in an ensemble consisting of 20 (= 4 × 5) models. First three rows of Table 4 gives the performance of such an ensemble for each PUI-SCU pair. From the table one can see that composing an ensemble with different partitions of phones results in better accuracy. This points towards the fact that each partition is good at classifying different types of phones. The questions of which partition is good for which type of phone and whether these partition choices are optimal are yet to be studied. Finally, the last row of Table 4 gives the performance of the ensemble consisting of models from all the three types of pooling architectures (a total of 60 models per classification unit). An error rate of 16.96% obtained by the best performing architecture is significantly better than most models proposed so far in the literature, and is close to the performance reported in [11]. This result is particularly remarkable taking into account the fact that unlike the model in [11], our model does not use any specific expert phonetic knowledge.

4. Conclusion and Future Work

We have proposed a simple multi-layer model for phoneme classification which combines frame level and segment level processing of phonemes via a novel pooling interface. Without the use of any expert knowledge, our model is able to achieve close to state of the art performance for the task.

To extend our model to phone recognition, the simplest approach would use the frame processing unit only to extract more relevant features for an HMM: because of pooling, these features are different from those produced by frame-based MLP approaches [14], and could be added using for instance a merger bottle-neck MLP. A more complex approach would involve the use of graph transformer networks [12] to produce multiple segmentations, followed by a search for the best-scoring sequence.

In our model, since the frame processing unit (most computationally expensive unit) can be shared across segments and classifiers used in the ensemble, a combinatorial explosion of computation would be limited to the segmental classification unit. Finally, our model can be applied to tasks where high confidence phoneme segmentation is necessary, but the processing can be done offline.

5. References


While the idea to use different pooling and phone partition schemes was motivated by phone variability, we just picked up the simplest schemes that came to our mind, letting data decide if they are suitable...