A Deep Generative Acoustic Model for Compositional Automatic Speech Recognition

Erik McDermott
Google Inc., USA
erikmcd@google.com

Abstract

Inspired by the recent successes of deep generative models for Text-To-Speech (TTS) such as WaveNet and Tacotron [20, 23], this article proposes the use of a deep generative model tailored for Automatic Speech Recognition (ASR) as the primary acoustic model (AM) for an overall recognition system with a separate language model (LM). Two dimensions of depth are considered: (1) the use of mixture density networks, both autoregressive and non-autoregressive, to generate density functions capable of modeling acoustic input sequences with much more powerful conditioning than the first-generation generative models for ASR, Gaussian Mixture Models / Hidden Markov Models (GMM/HMMs), and (2) the use of standard LSTMs, in the spirit of the original tandem approach, to produce discriminative feature vectors for generative modeling. Combining mixture density networks and deep discriminative features leads to a novel dual-stack LSTM architecture directly related to the RNN Transducer [7], but with the explicit functional form of a density, and combining naturally with a separate language model, using Bayes rule. The proposed generative models are compared experimentally in terms of log-likelihoods and frame accuracies.

1 Introduction

1.1 The channel model & compositional ASR

For most of its history, the field of ASR has used a collection of separate modules to represent different stages of an overall processing chain. Fred Jelinek formalized this approach within his concept of the “noisy channel model”, in which components chain together to form a consistent overall joint probability \( P(X, S) \), decomposed using Bayes’ theorem [9]. This model was particularly convenient given the strong independence assumptions in the model structures used. Gaussian Mixture Models (GMMs) used jointly with 1st-order Hidden Markov Models (HMMs) were well-suited to this modular approach, as they directly provide an acoustic likelihood \( p(X|S) \) (decomposed using strong independence assumptions) for a sequence of acoustic feature vectors \( X = x_{1:t} \) conditioned on a given sequence of modeling units, e.g. phonemes, graphemes or words, \( S = s_{1:m} \). This then combines naturally with a separate language model probability, \( P(S) \) to form the joint probability \( P(X, S) \) using Bayes rule. Other probabilistic modules such as a pronunciation dictionary can be introduced into the overall chain, again combining with the other components according to Bayes’ rule. In that sense, this approach to ASR is “compositional”.

1.2 Shoehorning discriminative models into the modular channel model

The adoption of discriminative models such as Deep Neural Networks (DNNs) and Long Short Term Memory models (LSTMs) did not at first alter this model. The popular “hybrid” approach (still considered state-of-the-art by most the community today) simply converts the posteriors \( P(s|x) \)
Deep Fusion [17] and Cold Fusion [17]. Though giving some practical gains, these are heuristic Shallow Fusion, is simple interpolation of the joint end-to-end model score with the LM score, e.g. What has also been reported is the combination of end-to-end models with other models, e.g. where the input \( S \) represents e.g. text and speaker features, and where observed samples \( x_t \) are targets of e.g. an \( N \)-ways quantized softmax trained with CE, using e.g. a DNN with dilated convolutions.

1.3 Goodbye, modularity?

In contrast, end-to-end models such as Listen, Attend & Spell (LAS) [4] or RNN Transducer (RNN-T) [2] dispense with the modular channel model altogether, and directly optimize a discriminative sequence-level model with a discriminative criterion. By construction, these models are all-in-one models that represent \( P(S|X) \) directly, with no chaining of sub-module probabilities. If one has a large enough training set of supervised data pairs, this can be thought of as the perfect model. State-of-the-art, or near state-of-the-art results have been reported for these models on challenging tasks [5,1].

1.4 Combination of end-to-end models with other modules: Fusion

What has also been reported is the combination of end-to-end models with other models, e.g. an LM \( P^*(S) \) trained on vastly more text data than is found in the transcripts of the audio data used to train \( P(S|X) \). Here the story is much less rosy. One approach, sometimes referred to as “Shallow Fusion”, is simple interpolation of the joint end-to-end model score with the LM score, e.g. \( P(S|X) \alpha \ast P^*(S)^{(1-\alpha)} \) [10]. The external LM has also been brought in through other methods, e.g. “Deep Fusion” [27] and “Cold Fusion” [17]. Though giving some practical gains, these are heuristic approaches with no clear mathematical justification such as Bayes’ theorem.

1.5 Hybridization & Joint training

An additional approach to model combination, paralleling the hybrid model for ASR described earlier, is to form a separate estimate \( P(S) \) based only on the limited audio + text transcript training data available to train \( P(S|X) \), and use it to form a scaled likelihood of the acoustics given \( S \). This can then combine with \( P^*(S) \) as \( P(S|X)/P(S) \ast P^*(S) \), following the concept of Bayes rule, but with some uncertainty whether the scaled likelihood is a meaningful representation of \( p(X|S) \), and requiring separate estimation of both \( P(S|X) \) and \( P(S) \). It remains to be seen whether this approach is more effective empirically than the fusion techniques. As for the conventional ASR approach with discriminative acoustic models, the scaled likelihood can be plugged into an overall sequence training objective function such as MMI or sMBR [12,14].

1.6 A better way: a deep generative acoustic model

As described earlier, generative acoustic models offer a mathematically principled and interpretable approach to module combination, following Bayes rule. While the first-generation generative models used for ASR were an excellent fit to the modular channel model, the strong independence assumptions of those early models severely hobbled their effectiveness, especially compared to the deep discriminative models, DNNs and LSTMs, that eventually replaced them [16,14]. In contrast, generative models have made large advances in the area of speech synthesis. State-of-the-art TTS approaches such as WaveNet and Tacotron [20,23] specifically model \( p(X|S) \) as a fully-conditioned autoregressive model, using the entire unidirectional sequence of past observations \( x_1 \) (where \( x_t \) is either an acoustic feature vector or a single scalar waveform sample), and typically, the entire sequence of symbols \( s_t \) constituting the sequence \( S \). The “conditional WaveNet” model [20] exemplifies this well, defining

\[
p(X|S) = \prod_t p(x_t|x_1, ..., x_{t-1}, S),
\]

where the input \( S \) represents e.g. text and speaker features, and where observed samples \( x_t \) are targets of e.g. an \( N \)-ways quantized softmax trained with CE, using e.g. a DNN with dilated convolutions.
Mixture density networks \[2\], based on either DNNs or RNNs, have also been effective as deep generative TTS models \[25, 26\]. Their use for ASR was proposed 20 years ago but not fully investigated \[15\]. Mixture density networks can use the same conditioning over observations and labels as the sample-by-sample WaveNet model, but specifically adopt a Gaussian mixture density function:

\[
p(x_t | x_1, ..., x_{t-1}, S) = \sum_i c_i(x_1:t-1, S)N(x_t | \mu_i(x_1:t-1, S), \sigma_i(x_1:t-1, S)).
\] (2)

This is a reasonable choice when \(x_t\) is a feature vector, as opposed to a scalar sample as in WaveNet. As ASR models typically operate on feature vectors, and not at the sample level, mixture density networks may be a good first step to investigate deep generative models for ASR.

1.7 ASR with a deep generative acoustic model

The central insight of this study is that there is nothing preventing us from doing ASR with deep generative acoustic models, along the lines of the models used for TTS just discussed. Fundamentally, whenever a likelihood \(p(X | S)\) is available, ASR is possible - as long as a separate LM \(P(S)\) is available to form the joint probability \(P(X, S)\), and also assuming a decoder is available that can generate, score and prune different symbol sequence hypotheses \(S\). The strong conditioning on the entire symbol sequence history characterizing the models discussed here means that ASR decoding for deep generative acoustic models faces practical challenges, but those challenges are not significantly different than those faced by the end-to-end ASR models such as LAS and RNN-T. The easiest implementation may be \(N\)-best rescoring of a list of hypotheses from an existing conventional ASR system.

1.8 Deepening the original tandem approach

One generative approach to ASR that has shown lasting effectiveness is the tandem approach \[8, 6\], still yielding near state-of-the-art results \[19, 13\]. In contrast to the use of scaled likelihoods formed directly from the output of a discriminative model, described earlier, in the tandem approach, features from a penultimate layer are extracted from the discriminative model, and used in a separate generative model such as GMMs. Typically these deep features are concatenated with acoustic features, and transformed using e.g. Principal Components Analysis (PCA) before being used as input to the generative model\[19\]. The result is a consistent generative model, but now operating on features that are far more discriminative than raw acoustic features by themselves. Related work has investigated the embedding of a gaussian layer directly into the same DNN architecture, enabling joint discriminative training of feature extractor and gaussian density parameters \[21, 18\]. These studies limited themselves to DNNs for the discriminative feature extraction, and to standard GMMs for the generative model. The approach proposed here extends this past work in both dimensions.

1.9 Overview

This study explores two dimensions of depth: the depth of the features being modeled, and the depth of the density function itself. More specifically, the features considered were:

- **Shallow features**: raw acoustic features, such as logmel energies or cepstral features;
- **Deep features**: features obtained from the last layer of an LSTM stack trained discriminatively with a standard criterion such as CE, in the spirit of the original Tandem approach.

The density models considered were:

- **Shallow density layers**: vanilla gaussian mixture models, though implemented in TensorFlow and trained with SGD (using either ML or CE);
- **Deep density networks**: mixture density density networks, both autoregressive and non-autoregressive, used to generate density functions capable of modeling acoustic feature vector sequences strongly conditioned on past features and labels, trained with ML.
Figure 1: Shallow density layer modeling shallow acoustic features. The density parameters are represented explicitly as tensor variables indexed by label class membership, and used in a TensorFlow definition of log-likelihood.

\[ p(x_t|w_m) = \sum_i c_{m,i} N(x_t, \mu_{m,i}, \sigma_{m,i}) \]

where the log of the standard multi-variate gaussian pdf has been decomposed to emphasize the similarity to the typical DNN/LSTM final layer, \( \log(\text{softmax}(wx+b)) \); the main differences being the use of multiple mixture components, and more importantly, the self-normalized nature of the

Figure 2: Simple 1D mixture of 4 gaussians estimated via ML/SGD on synthetic data drawn from a gaussian mixture of known parameters. Starting from a random initialization (in red), the estimated distribution (in green) closely matches the true distribution (in blue).

2 Shallow mixture density models estimated using ML/SGD, modeling shallow features

Figure 1 illustrates the simplest of the models described here, a vanilla gaussian mixture,

\[ p(x_t|w_m) = \sum_i c_{m,i} N(x_t, \mu_{m,i}, \sigma_{m,i}) \]

defined for any label \( w_m \) and any \( D \)-dimensional feature vector \( x_t \). This likelihood makes strong independence assumptions; it is not conditioned on previous observations, nor on previous label symbols. Nonetheless, it can plug into the complete likelihood of the utterance, Equation (1).

Implemented in TensorFlow, the parameters for this model are represented as TensorFlow Variables, and TensorFlow operations (exponential transform and softmax) are used to enforce the constraints that the standard deviations and the mixing weights are positive, and that the mixing weights sum to 1.

A simple radial covariance (using a single scalar standard deviation per mixture component) model was found to be convenient and effective:

\[ \log N(x, \mu_i, \sigma_i) = -D \left( \frac{1}{2} \log 2\pi - \log \sigma_i \right) - \frac{1}{2\sigma_i} \left( \sum_d \mu_{i,d}^2 - 2\mu_i^T x + \sum_d x_d^2 \right), \]
density function. The model can be strengthened through a transformation $A$ of the input feature $x$, where $A$ is either a diagonal or lower-triangular matrix, with exponentially-transformed diagonal values to ensure the positivity of the matrix determinant. The transformed feature vector $Ax$ can then be used in Equation 4, corresponding to a gaussian mixture model with shared diagonal covariances or shared full covariances, overlaying the gaussian-specific radial model.

SGD-based optimization of the density function can be done with a number criteria; the natural fit for this simple generative model is Maximum Likelihood (ML), but discriminative criteria such as Cross Entropy (CE) or sequence-level discriminative criteria such as MMI or sMBR can also be used.

2.1 Verifying the effectiveness of ML/SGD training

ML-based estimation of GMMs was traditionally performed with the highly effective Expectation-Maximization (EM) algorithm [11]. In contrast to the EM algorithm, SGD is not specifically suited to ML/GMM optimization, but its generality is highly appealing in the context of joint optimization of density functions with deep features or density parameters that are themselves generated by arbitrary, possibly recurrent neural network stacks. The use of SGD for ML-based estimation of GMMs has not been widely reported. A first step in this study was to verify that ML/SGD can effectively learn the correct estimates given synthetic data drawn from known distributions, such as illustrated in Figure 2. For low-dimensional scenarios easily inspected visually, it was found that as long as the features were mean/standard-dev normalized, standard rules of thumb of GMM parameter initialization [24] led to surprisingly effective estimation. For real-world data, two rough sanity checks were used: (1) TensorBoard histograms of mixing weights can diagnose unhealthy situations where a single gaussian component overwhelms all others in the mixture; (2) log-likelihoods on the training set should improve significantly with increasing numbers of mixture components. Careful comparison of ML/SGD-estimated GMMs with EM-estimated GMMs, and schemes for mixture splitting in TensorFlow, could yield insights and better performance.
3 Autoregressive and non-autoregressive deep mixture density networks
modeling shallow features

Figure 3 illustrates a mixture density network generated from an LSTM stack, predicting the next acoustic feature frame using all label symbols \( w_{1:t} \) up to that point in time (input to the LSTM stack via a class embedding), and in the autoregressive version of the model illustrated here, all previous acoustic features as well:

\[
p(x_t|x_{1:t-1}, w_{1:t}) = \sum_i c_i \left(x_{1:t-1}, w_{1:t}\right) N(x_t, \mu_i(x_{1:t-1}, w_{1:t}), \sigma_i(x_{1:t-1}, w_{1:t})) ,
\]

(5)
derived for a specific alignment of labels \( w_t \) to observations \( x_t \).

In the non-autoregressive version, the mixture density network uses only the label symbols (and no previous acoustic features) to predict the next acoustic feature frame:

\[
p(x_t|w_{1:t}) = \sum_i c_i (w_{1:t}) N(x_t, \mu_i (w_{1:t}), \sigma_i (w_{1:t})) .
\]

(6)

The power of either of these mixture density network models is that the density function changes dynamically as a function of the input it is provided. In principle, via the input of the unidirectional embedded symbol sequence, the model can leverage long-span symbol context, a key feature in the current climate of heightened interest in non-standard acoustic modeling units such as graphemes. The autoregressive version has the further feature that the ground truth of the observed past acoustic feature vector sequence enables a kind of adaptation of the predictive model to the actual speech signal input to the model. In contrast, the non-autoregressive model can only leverage the given unidirectional symbol context in making its predictions; it has to absorb all acoustic variability of the speech signal observed in the training set, across all speakers, speaking styles and acoustic environments.

3.1 Variants between autoregressive and non-autoregressive

Some variants on the full autoregressive model were considered:

- \( SH=n \): size of the frame shift between observations input into the LSTM stack, and target of the prediction, if different from 1. E.g., \( SH=3 \) refers to predicting \( x_t \) from \( x_{1:t-3} \).
- \( BN=n \): size of linear bottleneck on features before being fed into the LSTM stack, if any.
- \( ST=n \): size of the stride over the frames input into the LSTM stack, if different from 1. E.g. \( ST=10 \) refers to only using every 10-th frame, feeding in 0s in between.
Figure 5: Deep mixture density network modeling deep features.

4 Shallow mixture density layer modeling deep discriminative features

Figure 4 illustrates a shallow density model of deep features. The deep features are obtained from an LSTM stack, separately trained with CE as a discriminative acoustic encoder. The deep features in question are simply the output of the last LSTM layer. In contrast to previous tandem work using DNNs [21], no bottleneck layer was found to be necessary, presumably due to the more compact LSTM layer size (e.g. 512 for LSTMs in this study, vs 2048 for the DNNs in [21]). The model here is:

\[
p(d(x_t)|d(x_{1:t-1}), w_{1:t}) = \sum_i c_i(w_{1:t}) \mathcal{N}(d(x_t), \mu_i(w_{1:t}), \sigma_i(w_{1:t}))
\]

(7)

If the LSTM stack implementing the discriminative acoustic encoder \(d()\) is frozen, this is just a vanilla GMM layer that can be trained with ML, whose features happen to be highly discriminative. Joint training of the density function and the feature extractor \(d()\) using ML is feasible, but requires proper handling of the Jacobian of the inverse of \(d()\). Joint training using CE, sMBR or MMI, however, has no such issue [21, 18, 13].

5 Deep mixture density networks modeling deep discriminative features

Finally, one can apply a deep density network to the modeling of deep features, as illustrated in Figure 5. The result is an architecture closely mirroring the well-known RNN Transducer [7], but explicitly formulated as a density function generated from an LSTM stack encoding a label sequence, applied to deep features encoding the acoustic sequence. The resulting likelihood is:

\[
p(d(x_t)|d(x_{1:t-1}), w_{1:t}) = \sum_i c_i(w_{1:t}) \mathcal{N}(d(x_t), \mu_i(w_{1:t}), \sigma_i(w_{1:t}))
\]

(8)

If the LSTM stack encoding the deep features is frozen, this is a straightforward mixture density network that can be trained with ML, but with highly discriminative features. As with the shallow density model of deep features, it can be trained jointly using discriminative criteria; and joint training with ML is again feasible but requires proper handling of the Jacobian of the inverse of \(d()\).

6 Experiments

The goal of the experiments was to verify that the added conditioning of the mixture density networks examined indeed improves log-likelihoods as expected, and to get some insights about likely WER from frame accuracies for these models.
Figure 6: Label class input-batching for computing frame accuracy.

Table 1: Cost and frame accuracy for density models predicting (shallow) mel-cepstral features.

<table>
<thead>
<tr>
<th>Model</th>
<th>Phones</th>
<th>Autoreg</th>
<th>Topology</th>
<th>Mixture size</th>
<th>Params</th>
<th>Criterion</th>
<th>Cost</th>
<th>FAcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>CI</td>
<td>-</td>
<td>5x512</td>
<td>-</td>
<td>1.1M</td>
<td>CE</td>
<td>-</td>
<td>55%</td>
</tr>
<tr>
<td>Shallow density</td>
<td>CI</td>
<td>-</td>
<td>256 / phone</td>
<td>1.1M</td>
<td>ML</td>
<td>127</td>
<td>35%</td>
<td></td>
</tr>
<tr>
<td>Shallow density</td>
<td>CI</td>
<td>-</td>
<td>256 / phone</td>
<td>1.1M</td>
<td>CE</td>
<td>-</td>
<td>48%</td>
<td></td>
</tr>
<tr>
<td>Deep density</td>
<td>Unsup</td>
<td>AR</td>
<td>5x512</td>
<td>10 total</td>
<td>4.5M</td>
<td>ML</td>
<td>43.0</td>
<td>-</td>
</tr>
<tr>
<td>Deep density</td>
<td>CI</td>
<td>AR</td>
<td>5x512</td>
<td>10 / phone</td>
<td>4.5M</td>
<td>ML</td>
<td>43.5</td>
<td>25%</td>
</tr>
<tr>
<td>Deep density</td>
<td>CD</td>
<td>AR</td>
<td>5x512</td>
<td>10 / phone</td>
<td>4.5M</td>
<td>ML</td>
<td>42.3</td>
<td>-</td>
</tr>
<tr>
<td>Deep density SH=3</td>
<td>Unsup</td>
<td>AR</td>
<td>5x512</td>
<td>10 total</td>
<td>4.5M</td>
<td>ML</td>
<td>118</td>
<td>-</td>
</tr>
<tr>
<td>Deep density SH=3</td>
<td>CI</td>
<td>AR</td>
<td>5x512</td>
<td>10 / phone</td>
<td>4.5M</td>
<td>ML</td>
<td>116</td>
<td>30%</td>
</tr>
<tr>
<td>Deep density</td>
<td>CI</td>
<td>AR</td>
<td>5x512</td>
<td>256 / phone</td>
<td>9.8M</td>
<td>ML</td>
<td>38.0</td>
<td>27%</td>
</tr>
<tr>
<td>Deep density BN=5</td>
<td>CI</td>
<td>AR</td>
<td>5x512</td>
<td>256 / phone</td>
<td>9.8M</td>
<td>ML</td>
<td>114</td>
<td>32%</td>
</tr>
<tr>
<td>Deep density ST=10</td>
<td>CI</td>
<td>AR</td>
<td>5x512</td>
<td>256 / phone</td>
<td>9.8M</td>
<td>ML</td>
<td>115</td>
<td>36%</td>
</tr>
<tr>
<td>Deep density</td>
<td>CI</td>
<td>-</td>
<td>5x512</td>
<td>10 / phone</td>
<td>4.5M</td>
<td>ML</td>
<td>129</td>
<td>38%</td>
</tr>
<tr>
<td>Deep density</td>
<td>CI</td>
<td>-</td>
<td>5x512</td>
<td>256 / phone</td>
<td>9.8M</td>
<td>ML</td>
<td>126</td>
<td>44%</td>
</tr>
<tr>
<td>Deep density</td>
<td>CD</td>
<td>-</td>
<td>5x512</td>
<td>256 / phone</td>
<td>9.8M</td>
<td>ML</td>
<td>122</td>
<td>-</td>
</tr>
</tbody>
</table>

6.1 Computation of frame accuracy

Frame accuracy for the shallow density layers (whether they use shallow or deep features) is straightforward to compute in TensorFlow, as \( N \) label class outputs for any given feature vector \( x_t \) can easily be made available by batching Equation 3 or Equation 7, and compared with the output for the ground-truth label at time \( t \).

For the deep density networks, in principle only one label class output is generated at a time, from the input of a specific embedded class label. Frame classification for density networks can be computed by producing \( N \) density outputs for \( N \) class label hypotheses input to the LSTM stack. In all experiments here, a simple label class prior \( P(s) \) is estimated in-network [21] and used jointly with the likelihood produced by the density model to make the frame classification.

An issue for deep density networks is the label context to use when measuring frame accuracy. In a real ASR decoding scenario, multiple partial label sequences would be hypothesized by the decoder. The approach adopted here was to measure frame accuracy for the density networks via conditioning on the ground-truth label sequence up to time \( t - 1 \), with only the label at time \( t \) being hypothesized. This can be computed efficiently in TensorFlow by batching the input of the \( N \) label hypotheses at all times \( t \), but splicing into the inference pass a set of separately generated LSTM states up to \( t - 1 \) obtained from ground-truth only input. Figure 6 illustrates the use of class input-batching to do this efficiently in TensorFlow.
Table 2: Cost and frame accuracy for density models predicting deep discriminative features.

<table>
<thead>
<tr>
<th>Model</th>
<th>Phones</th>
<th>Autoreg</th>
<th>Topology</th>
<th>Mixture size</th>
<th>Params</th>
<th>Criterion</th>
<th>Cost</th>
<th>FAcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>CI</td>
<td>-</td>
<td>5x512</td>
<td>-</td>
<td>4.3M</td>
<td>CE</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Shallow density</td>
<td>CI</td>
<td>-</td>
<td>5x512</td>
<td>1 / phone</td>
<td>4.3M</td>
<td>CE</td>
<td>-</td>
<td>87%</td>
</tr>
<tr>
<td>Shallow density</td>
<td>CI</td>
<td>-</td>
<td>5x512</td>
<td>10 / phone</td>
<td>4.5M</td>
<td>CE+ML</td>
<td>-305</td>
<td>77%</td>
</tr>
<tr>
<td>Deep density</td>
<td>CI</td>
<td>-</td>
<td>2*(5x512)</td>
<td>1 / phone</td>
<td>8.6M</td>
<td>CE+ML</td>
<td>-280</td>
<td>81%</td>
</tr>
<tr>
<td>Deep density</td>
<td>CI</td>
<td>-</td>
<td>2*(5x512)</td>
<td>10 / phone</td>
<td>8.8M</td>
<td>CE+ML</td>
<td>-400</td>
<td>85%</td>
</tr>
<tr>
<td>Deep density</td>
<td>CI</td>
<td>-</td>
<td>2*(5x512)</td>
<td>32 / phone</td>
<td>9.3M</td>
<td>CE+ML</td>
<td>-425</td>
<td>84%</td>
</tr>
<tr>
<td>LSTM</td>
<td>CD</td>
<td>-</td>
<td>5x512</td>
<td>-</td>
<td>8.5M</td>
<td>CE</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Shallow density</td>
<td>CD</td>
<td>-</td>
<td>5x512</td>
<td>1 / phone</td>
<td>8.5M</td>
<td>CE</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Shallow density</td>
<td>CD</td>
<td>-</td>
<td>5x512</td>
<td>46 / phone</td>
<td>46M</td>
<td>CE+ML</td>
<td>-</td>
<td>63%</td>
</tr>
</tbody>
</table>

Though one could of course use in-network estimation of much more powerful (e.g. LSTM-based) LMs to tremendously boost frame-accuracy, the simple class prior was deemed fit to provide more insight into the discriminative power of the AM itself.

6.2 Datasets

The dataset set used here is a set of 20000 hours of spontaneous speech from anonymized, human-transcribed Voice Search data for American English (en-us). The training examples are sequences of unstacked (single-frame) feature vectors extracted from a 64 ms window shifted over the audio samples in 30 ms increments. The acoustic features vectors used in the “deep feature” models, based on discriminative acoustic encoder architectures, are 256 dimensional logmel energies; those used in the “shallow feature” models are 96 dimensional cepstral coefficients, consisting of a stack of 32+32+32 mel filterbank cepstral coefficients (MFCCs) and their first and second derivatives, extracted from the same 64 ms windowing scheme.

The overall training architecture closely follows the TensorFlow based implementation described in [22]. Each training step has a batch size of 64 example utterances, with a fixed static LSTM unrolling of 20 frames. All LSTM stacks used are 5 layer models with 512 units each, for a total of roughly 4.3M parameters, not counting the final layer, whose size will depend on the number of output classes and, for the density models, on the number of mixture components. The class outputs are 42 context-independent (CI) phonemes or 8192 decision-tree clustered context-dependent (CD) phonemes. A label class embedding of size 128 was used for all experiments. CE training of the density models combines the acoustic likelihood output by the density model with an in-network estimate of the state prior $P(s)$ (as in used in the computation of frame accuracy) [21].

The statistics reported were somewhat unscientifically obtained from smoothed TensorBoard plots when the models were deemed to have converged. Frame accuracies are based on a small held out dataset and are hence quite noisy; only 2 significant digits are reported for them. The costs (negative log-likelihoods) are smoothed costs for the training criterion at convergence. As roughly 27% of the data is silence, a frame accuracy of anything not surpassing that can officially be considered abysmal.

6.3 Results

Table 1 describes the costs (negative log-likelihoods) and frame accuracies for all models using shallow mel-cepstral features.

Shallow density with shallow features: The frame classification of an ML-trained radial gaussian mixture barely surpasses abysmal levels, at 35%. In contrast, a reference 5 layer DNN with a matching number of parameters is at 55%. However, CE-training the gaussian mixture model brings the frame accuracy up to 48%. This may be a reasonable result, suggesting that the flat structure of the GMM isn’t as effective a classifier as a deeply structured DNN, but isn’t completely off either, if trained discriminatively. The cost of 127 provides a baseline for the deep density models in the rest of Table 1. Those models are expected to improve significantly over that.

Deep autoregressive densities with shallow features: As expected, the deep density models do much better in terms of cost than the shallow density model just discussed. Also as expected,
the autoregressive models have better log-likelihoods than the non-autoregressive models, and furthermore, the full autoregressive models do better than the autoregressive models whose input was limited via the schemes for prediction shift (SH), feature bottleneck (BN) and frame stride (ST). However, the trend for frame accuracy is the opposite. The more past input is provided to the LSTM stack generating the mixture density, the worse the frame accuracy. One perspective is that providing too much information about past observations makes the prediction problem too easy, the label class information is not necessary, and hence the prediction is not discriminative. Results are also shown for the unsupervised versions (Unsup), where all class input is merged into a single symbol. One expects that the label class information would help improve the prediction cost over the unsupervised prediction, but that is not the case for the CI phone scenario, corresponding to the “CI AR” result, with a cost of 43.5, compared to the “Unsup AR” cost of 43.0. The “CD AR” model does a bit better with a cost of 42.3. Shifting the prediction target 3 frames into the future (SH=3), however, produces a gain for the CI model over the unsupervised version, 116 vs 118. Going from 10 mixture components to 256 significantly improves prediction cost, but doesn’t significantly affect the frame accuracies for the autoregressive models here.

Deep non-autoregressive densities with shallow features: Removing all past observation input makes the prediction cost much worse than for the autoregressive models, but the frame accuracies are much better. They are however still not competitive with the simple DNN, and in fact a bit worse than the CE-trained shallow density model.

Table 2 ups the ante on frame accuracy, via the use of deep features. A standard CE-trained LSTM is listed at the top of the table for reference.

Shallow densities with deep features, joint training with CE: Given the form of the radial density defined in Equation 4 it should not surprise us that the classification capacity of an LSTM stack using the radial density as the last layer, trained discriminatively, would closely match the classification capacity of an LSTM stack with a standard logits layer. This is what we see for both CI and CD versions, with a single radial component per mixture, achieving frame accuracies of 87% and 79% respectively, matching the standard CI and CD LSTMs.

Shallow densities with deep features, ML training: All the ML results here require pre-training of the standard LSTM (either CI or CD) with CE first. There is a clear drop in performance compared to the pure CE result, especially for the CD case, with 8192 output classes to discriminate compared to 42. However, the deep features here are not mean/standard-dev normalized (nor PCAed, etc.). The simple radial covariance model clearly could be improved upon for this particular configuration, if desired.

Deep densities with deep features: The dual LSTM stack architecture does quite well as a predictor of the deep features, using just the label class context. The prediction cost (e.g. -400 for the 10 component mixture version) is much better than the cost (-305) for the shallow density model of the same deep features, and in fact has significantly better frame accuracy too, 85% vs 77%. It may be argued that the deep features here have essentially extracted phoneme information from the CE training of the standard LSTM stack, and that now the prediction problem is really a label prediction problem, given the previously observed label class input. However, there is nothing wrong with that from an ASR point of view, as the deep features are a function of the acoustics. Furthermore, this may be exactly what is interesting about this last configuration: using an encoding of a label sequence, we are trying to predict an encoding of an acoustic sequence.

7 Summary

Four generative models were described and evaluated, the cartesian product of {shallow, deep} features and {shallow, deep} densities. This recapitulates but also extends the original tandem approach to ASR. As no WER evaluations were run, these results are rather tentative, but do give some insights into the models proposed. The log-likelihood results follow our intuitions about the strength of the models. The weak result for supervised vs unsupervised may suggest an issue with the experimental setup, e.g. the fixed alignments used are poor, or it could reflect the weakness of the radial covariance model. The somewhat better result for the CD phone model suggests either that the CI label context is not being completely represented by the LSTM state, or that “future blindness” (not knowing the identity of the immediately following phoneme) is a significant issue.
The frame accuracies for the models using shallow acoustic features seem too low to be useful for ASR. One could argue, however, that the autoregressive models, properly used in an ASR decoding framework, may perform much better than the frame accuracies suggest.

The use of deep features solves many of the problems generative models have in modeling environmental and speaker variability, and immediately provides strong discriminative power, but it may be seen as cheating and no longer purely generative. Given their state-of-the-art frame accuracy, presumably the corresponding generative models described here would be viable for ASR. The question then is, how does the generative nature of these models help us? E.g. Does it allow for better compositionality with separate LMs, as claimed in the Introduction? Does it provide the advantages attributed to generative models regarding unsupervised training or adaptation? Full ASR decoding experiments with WERs are needed to address those questions.

References


