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 [29, 33], 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 [8], 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-likelihood and frame accuracy.

1 Introduction

1.1 The noisy 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” (abbreviated to "channel model" in the following), in which components chain together to form a consistent overall joint probability $P(X, W)$ for e.g. a sequence of acoustic observations X and a sequence of words W. In turn, $P(X, W)$ enables the Bayes' decision procedure based on the posterior $P(W|X)$, and Bayes' theorem [14]:

$$P\left(W|X\right) = \frac{p\left(X|W\right)P\left(W\right)}{p\left(X\right)}, \tag{1}$$

where $p(X)$ can be dropped from the recognition procedure, as it doesn't affect the relative posteriors of hypotheses W. 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 $p(X|W)$ 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, $W = w_{1:M}$. Denoting a possible state alignment of W to X as $S_W = \{w_1, ..., w_T\}$, the overall acoustic log-likelihood is

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

defined with strong independence assumptions, using

$$p(X|S_W) = \prod_{t=1}^{T} p(x_t|w_t), \tag{2}$$

and a sum over alignments (computed efficiently using the forward-backward algorithm),

$$p(X|W) = \sum_{S_W} p(X|S_W), \tag{3}$$

or the best state alignment (the Viterbi approximation):

$$p(X|W) = \max_{S_W} p(X|S_W). \tag{4}$$

This then combines naturally with a separate language model probability, $P(W)$ to form the joint probability $P(X, W)$.

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. One can keep adding modules, or substituting modules, and the overall model still holds. In this sense, this approach to ASR is "compositional".

1.2 Shoehorning discriminative models into the modular channel model: the hybrid model

The adoption of more powerful, 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 of the community today) simply converts the posteriors $P(s|x)$ (defined at the level of a single observation x for a model state s) obtained from DNNs or LSTMs to a scaled likelihood by normalizing the posterior by a state prior $P(s)$ [3]. Fleshing out this picture, the hybrid model for DNNs then is

$$p(X|S_W) = \prod_{t=1}^{T} p(x_t|w_t) \doteq \prod_{t=1}^{T} P(w_t|x_t)/P(w_t), \tag{5}$$

and for a bidirectional LSTM is

$$p(X|S_W) \doteq \prod_{t=1}^{T} P(w_t|X)/P(w_t). \tag{6}$$

These (scaled) likelihoods can then plug back into the channel model using Bayes rule, as before, and can also plug into an overall sequence training objective function such as utterance-level MMI or sMBR [19, 21]. Note that for the typical DNNs and LSTMs used by the community, sequence training has the potential to estimate implicitly the scaling term $p(s)$ as an output unit bias term, so that it optimizes the sequence-level criterion.

The overall construction of a sequence-level training objective, converting local frame-level scores from a discriminative model into "generative" scaled likelihoods, only to then plug those into a discriminative sequence training criterion, may seem like a strange hybrid indeed. Nonetheless, this has been a remarkably effective approach, that still constitutes the state-of-the-art [32].

1.3 Goodbye, modularity?

In contrast, end-to-end models such as Listen, Attend & Spell (LAS) [4] or RNN Transducer (RNN-T) [8] 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 $P(W|X)$ directly, with no chaining of sub-module probabilities. The model directly defines the posterior needed for recognition. 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 [1, 5].

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 $P^*(W)$ trained on vastly more text data than is found in the transcripts of the audio data used to train $P(W|X)$. 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. [15]:

$$P\left(X, W\right) \approx P\left(W|X\right)^{\alpha} P^{*}\left(W\right)^{(1-\alpha)}. \tag{7}$$

The external LM has also been brought in through other methods, e.g. "Deep Fusion" [37] and "Cold Fusion" [24]. 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, the "sequence version" of the hybrid model for ASR described above [3], is to form a separate estimate $P(W)$ based only on the transcript-side of the $\{$audio + transcript$\}$ data used to train $P(W|X)$, and use it to form a scaled likelihood of the acoustics given W:

$$q\left(X|W\right) = \frac{P\left(W|X\right)}{P\left(W\right)}, \tag{8}$$

related to $p(X|W)$ by Bayes rule,

$$p\left(X|W\right) = p\left(X\right) q\left(X|W\right). \tag{9}$$

As the $p(X)$ term doesn't affect recognition, the scaled likelihood can then combine with $P^*(W)$ for recognition using

$$P\left(X, W\right) = q\left(X|W\right) P^{*}\left(W\right), \tag{10}$$

following Bayes rule, but with some uncertainty whether the scaled likelihood is a meaningful representation of $p(X|W)$. 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 utterance-level MMI or sMBR with a separate language model, if desired. Depending on the model used for $P(W|X)$, note that sequence training has the potential to estimate implicitly the scaling term in Equation 8, so that it optimizes the sequence-level criterion.

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 [21, 23]. 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 [29, 33] specifically model $p(X|W)$ as a fully-conditioned autoregressive model, using the entire unidirectional sequence of past observations x_t (where x_t is either an acoustic feature vector or a single scalar waveform sample), and typically, the entire sequence of symbols w_i constituting the sequence W. The "conditional WaveNet" model [29] exemplifies this well, defining

$$p(X|S_W) = \prod_{t=1}^{T} p(x_t|x_1, ..., x_{t-1}, S_W), \tag{11}$$

where the input S_W 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 [35, 36]. Their use for ASR was proposed 20 years ago but not fully investigated [22]. Mixture density networks can use the same conditioning over observations and labels as the sample-by-sample WaveNet model, but use a Gaussian mixture density function:

$$p(x_t|x_1, ..., x_{t-1}, S_W) = \sum_i c_i(x_{1:t-1}, S_W) N(x_t|\mu_i(x_{1:t-1}, S_W), \sigma_i(x_{1:t-1}, S_W)). \quad (12)$$

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 deep channel model

The original channel model provides a principled foundation for ASR with deep generative acoustic models such as the TTS models just discussed [13, 22]. Whenever a likelihood $p(X|W)$ is available, ASR is possible - as long as a separate LM $P(W)$ is available to form the joint probability $P(X, W)$, and given a decoder that can generate, score and prune different symbol sequence hypotheses W. Time-synchronous beam-search can be used, among several possible decoding strategies. The strong conditioning on the entire unidirectional symbol sequence history characterizing several of the models discussed here means that ASR decoding in principle cannot merge partial hypotheses, but that is also true for decoding with end-to-end ASR models such as LAS and RNN-T. Another approach to ASR decoding with the models described here is N-best rescoring of a list of hypotheses from an existing conventional ASR system. Note that the LM $P(W)$ can of course be "deep" too, e.g. an RNN-LM. The "deep channel model" is just the channel model, with deep modules.

1.8 Extending the original Tandem approach

One generative approach to ASR that has shown lasting effectiveness is the tandem approach [7, 11], still yielding near state-of-the-art results [20, 28]. 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 obtained from a bottleneck layer [9], concatenated with acoustic features, and transformed using e.g. Principal Components Analysis (PCA) before being used as input to the generative model [28]. 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 [27, 31]. 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 Towards integrated ASR & TTS, and semi-supervised training

Though not explored in this study, the goal of integrating or chaining ASR and TTS together has a long history [6, 13, 25, 26]. Recently, this has been used for semi-supervised ASR and TTS training [12, 26]. The models discussed here are clearly related to this body of work, as well as to the ideas for prediction-based unsupervised training discussed in [17].

1.10 Overview

This study explores two dimensions of depth for ASR-oriented generative models: the depth of the features being modeled, and the depth of the density function itself. 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);

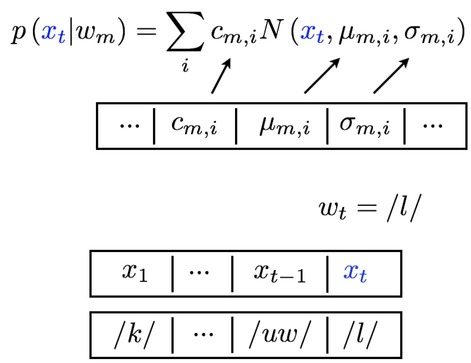

Figure 1: Shallow density layer modeling shallow acoustic features. The density parameters are represented explicitly as tensor variables indexed by label class membership, and used in a TensorFlow definition of log-likelihood.

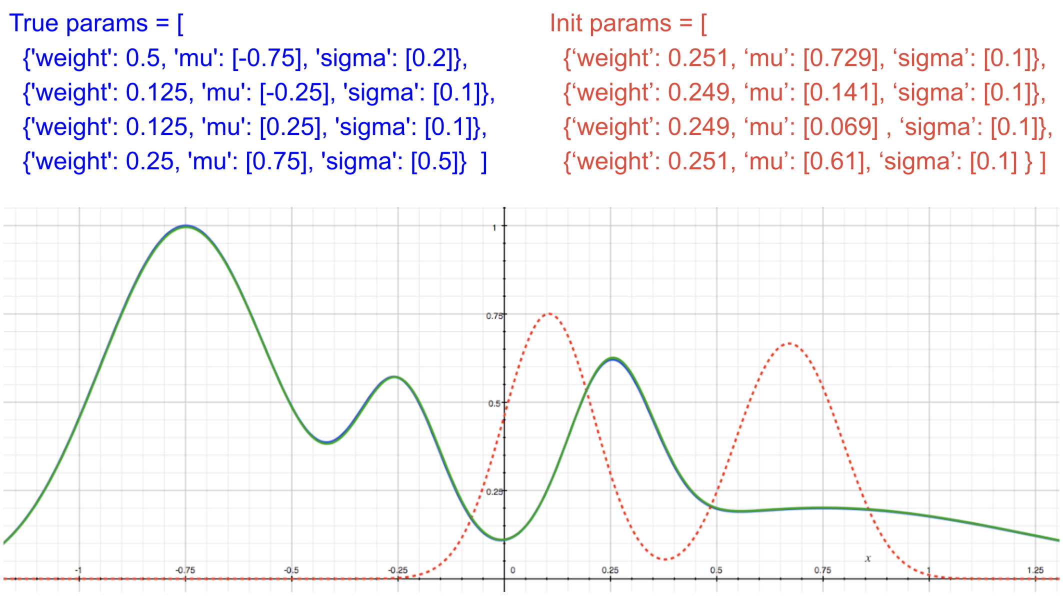

Figure 2: Simple mixture of 4 gaussians estimated via ML/SGD on synthetic 1D data drawn from a gaussian mixture of known parameters. Starting from a random initialization (in red), the estimated distribution (in green) closely matches the true distribution (in blue).

- *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.

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,

$$p(x_t|w_m) = \sum_i c_{m,i} N(x_t, \mu_{m,i}, \sigma_{m,i}), \tag{13}$$

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, based on Equation 2.

Implemented in TensorFlow, the parameters for this model are represented as TensorFlow Variables. Standard deviations and mixing weights are represented in the log domain, with *log_softmax()* used to enforce the constraint that the mixing weights sum to one [31].

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

$$\log N\left(x,\mu_i,\sigma_i\right) = -D\left(\frac{1}{2}\log 2\pi - \log\sigma_i\right) - \frac{1}{2\sigma_i}\left(\sum_d \mu_{i,d}^2 - 2\mu_i^T x + \sum_d x_d^2\right), \tag{14}$$

where the log of the standard multi-variate gaussian pdf has been decomposed to emphasize the similarity to the typical DNN/LSTM final layer, $log(softmax(wx+b))$; the main differences being the use of multiple mixture components, and more importantly, the self-normalized nature of the density function. (See [10] for discussion of the equivalence of gaussian and log-linear models under certain conditions). 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 14, corresponding to a gaussian mixture model with shared diagonal covariances or shared full covariances, overlaying the gaussian-specific radial model. The results described in the main body of this article are all for densities using the simple radial model; results with stronger covariance models are described in Appendices A and B.

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-discriminative criteria such as utterance-level 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 [18]. 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. 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 [34] led to surprisingly effective ML/SGD estimation, even for mixtures with highly overlapping components, such as illustrated in Figure 2. 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:

$$p\left(x_t|x_{1:t-1},w_{1:t}\right) = \sum_i c_i\left(x_{1:t-1},w_{1:t}\right) N\left(x_t,\mu_i\left(x_{1:t-1},w_{1:t}\right),\sigma_i\left(x_{1:t-1},w_{1:t}\right)\right), \tag{15}$$

defined for a specific alignment of labels w_t to observations x_t. This likelihood can plug into Equation 11, a much more powerful model than Equation 2.

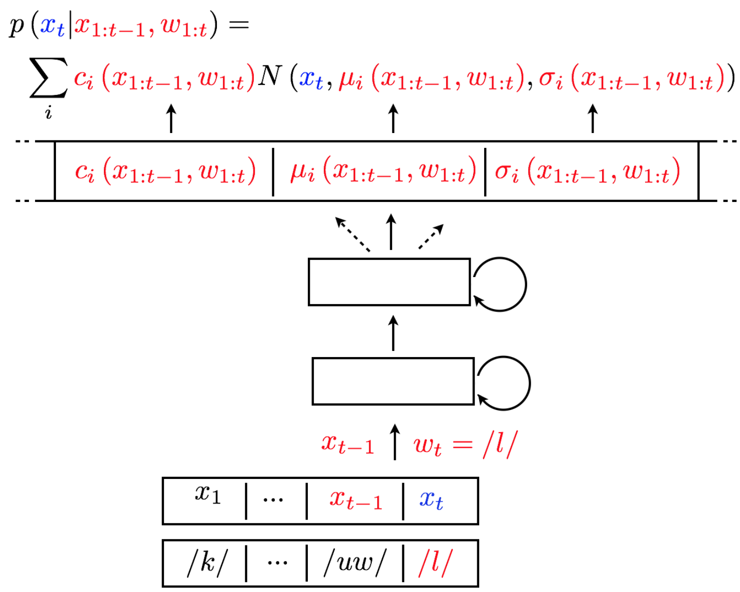

Figure 3: Deep mixture density network modeling shallow acoustic features. The density parameters are generated dynamically as TensorFlow activations from label class embeddings, and (in the autoregressive version) previous observations as well, input into an LSTM stack.

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:

$$p(x_t|w_{1:t}) = \sum_i c_i(w_{1:t}) N(x_t, \mu_i(w_{1:t}), \sigma_i(w_{1:t})) . \tag{16}$$

The power of these density networks is that the density function changes dynamically as a function of the input it is provided. Via the input of the unidirectional embedded symbol sequence, the model can leverage long-span symbol context, a key feature when using non-standard acoustic modeling units such as graphemes. The autoregressive version has the further feature that the ground truth of past observed acoustic feature vectors 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 model all acoustic variability 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.

These are ways to provide less than the full past input to the autoregressive model. One can expect that there would be a worsening of prediction log-likelihood, but perhaps an improvement in frame accuracy. Note that "Professor Forcing" [16] was proposed to address the related problem of "Teacher Forcing" in autoregressive models.

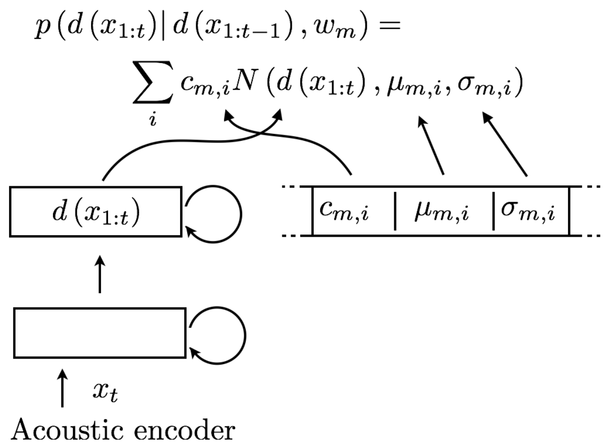

Figure 4: Shallow density layer modeling deep features.

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, before the logits and softmax layer [30]. In contrast to previous tandem work using DNNs [9, 31], 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 [31]). The model here is:

$$p\left(d\left(x_t\right)|d\left(x_{1:t-1}\right), w_m\right) = \sum_i c_{m,i} N\left(d\left(x_t\right), \mu_{m,i}, \sigma_{m,i}\right). \tag{17}$$

If the LSTM stack implementing the discriminative acoustic encoder $d\left(\right)$ 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 $d\left(\right)$ using ML is feasible, but requires proper handling of the Jacobian of the inverse of $d\left(\right)$. Joint training using CE, or utterance-level sMBR/MMI, however, has no such issue [20, 27, 31].

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 [8], 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:

$$p\left(d\left(x_t\right)|d\left(x_{1:t-1}\right), w_{1:t}\right) = \sum_i c_i\left(w_{1:t}\right) N\left(d\left(x_t\right), \mu_i\left(w_{1:t}\right), \sigma_i\left(w_{1:t}\right)\right). \tag{18}$$

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 $d\left(\right)$. The labels w can be input time-synchronously (repeating input labels for the length of their frame alignment) as well as label-synchronously, only inputting label transitions; the latter approach produced better frame accuracies, and was used in the experiments for this model. To our knowledge, this is a novel application of mixture density networks to the modeling of deep features.

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.

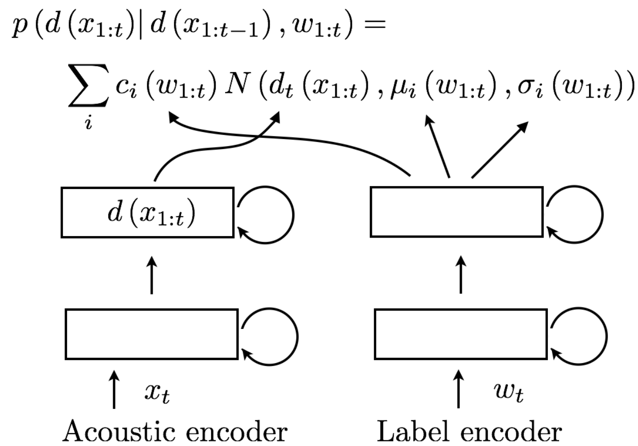

Figure 5: Deep mixture density network modeling deep features.

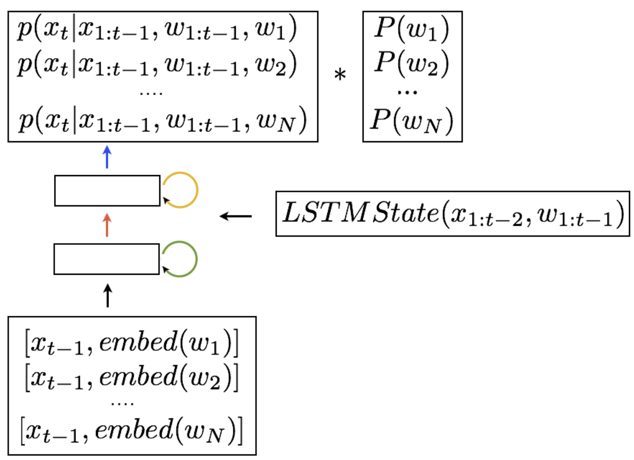

Figure 6: Label class input-batching for computing frame accuracy.

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 13 or Equation 17, 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 [31] and used jointly with the likelihood produced by the density model to make the frame classification. (Though one could 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 insight into the discriminative power of the AM itself).

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 $t-1$

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

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

Table 2: Cost and frame accuracy for density models predicting deep discriminative features.

Model	Phones	Autoreg	Topology	Mixture size	Params	Criterion	Cost	FAcc
LSTM	CI	-	5x512	-	4.3M	CE	-	87%
Shallow density	CI	-	5x512	1 / phone	4.3M	CE	-	87%
Shallow density	CI	-	5x512	10 / phone	4.5M	CE+ML	-305	77%
Deep density	CI	-	2*(5x512)	1 / phone	8.6M	CE+ML	-280	81%
Deep density	CI	-	2*(5x512)	10 / phone	8.8M	CE+ML	-400	85%
Deep density	CI	-	2*(5x512)	32 / phone	9.3M	CE+ML	-425	84%
LSTM	CD	-	5x512	-	8.5M	CE	-	79%
Shallow density	CD	-	5x512	1 / phone	8.5M	CE	-	79%
Shallow density	CD	-	5x512	10 / phone	46M	CE+ML	-	63%

obtained from ground-truth only input. Figure 6 illustrates the use of class input-batching to do this efficiently in TensorFlow.

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 feature 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. Compared to logmel features, MFCCs are a reasonable choice when using a radial or diagonal covariance model, and the time derivatives provide a representation of temporal context [22]. Results for shallow feature density networks using logmels are reported in Appendices A and B; the frame accuracies detailed there are significantly lower than those for MFCCs with derivatives described in the following.

The overall training architecture closely follows the TensorFlow based implementation described in [32]. Each training step has a batch size of 64 example utterances, with a fixed static LSTM unrolling of 20 frames. Given the total batch size of 1280 frames, 1 epoch of training over the 20000 hour training set corresponds to 1.875M training steps. For every configuration, at least 10 epochs of SGD training were run. 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) [31].

The statistics reported were somewhat unscientifically obtained from smoothed TensorBoard plots when the models were deemed to have converged. The smoothing factor for the TensorBoard moving average was 0.95. Frame accuracies are based on a small held out dataset of 1000 utterances (with an average of 120 feature frames per utterance) 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 the costs are running averages over the entire training set, they are less noisy than the frame accuracy statistics. These metrics are admittedly not very rigorous, but nonetheless offer insights into the conditions explored. 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, confirming that the flat structure of the GMM is not 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. (Professor Forcing [16] may provide a solution to this problem). 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. (See Appendix B for a deeper exploration of the effect of shifting the prediction target). 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. Note that CI or CD phones were used consistently for both the deep features and the deep density.

Shallow densities with deep features, joint training with CE: Given the form of the radial density defined in Equation 14 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 PCA'd, 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.

Visualization of predictive quality for deep densities with shallow logmel features: See Appendix C for plots of generated logmel spectra, specifically focusing on the difference between supervised and unsupervised prediction quality as a function of the prediction shift ("SH").

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, though other strategies such as re-alignment or decision delaying could address that too. Appendix A and Appendix B explore the strength of the covariance model used, and the difference between supervised and unsupervised models, in greater depth.

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 difference in prediction cost between supervised and unsupervised scenarios discussed earlier (and in Appendices A and B) may be a better metric than frame accuracy for getting insight into ASR performance.

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 as measured by frame accuracy, 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.

Acknowledgments

The author thanks Tom Bagby, Matt Shannon, Ehsan Variani, Michiel Bacchiani, Khe Chai Sim, Arun Narayanan, David Rybach, John Hershey, Heiga Zen, Mike Schuster, Samy Bengio, Tara Sainath, Bo Li, Rohit Prabhavalkar, Ananya Misra, Patrick Nguyen, Navdeep Jaitly, Shankar Kumar, Mitch Weintraub, Rif A. Sauros, Hasim Sak and Trevor Strohman for practical and conceptual help with this study.

References

[1] Kartik Audhkhasi, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Michael Picheny. Building competitive direct acoustics-to-word models for english conversational speech recognition. In *Proc. IEEE ICASSP*, 2018.

[2] Christopher M. Bishop. Mixture density networks. Technical report, 1994.

[3] Hervé Bourlard and Nelson Morgan. *Connectionist speech recognition - A Hybrid Approach*. Kluwer Academic Publishers, 1994.

[4] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[5] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani. State-of-the-art speech recognition with sequence-to-sequence models. In *Proc. IEEE ICASSP*, 2018.

[6] Peter Denes and Elliot Pinson. *The speech chain*. Macmillan, 1993.

[7] D. P. W. Ellis, R. Singh, and S. Sivadas. Tandem acoustic modeling in large-vocabulary recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2001.

[8] Alex Graves. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711, 2012. URL `http://arxiv.org/abs/1211.3711`.

[9] Frantisek Grézl, Martin Karafiát, Stanislav Kontar, and Jan Cernocký. Probabilistic and bottle-neck features for lvcsr of meetings. *Proc. IEEE ICASSP*, 2007.

[10] G. Heigold, P. Lehnen, R. Schlueter, and H. Ney. On the Equivalence of Gaussian and log-linear HMMs. In *Proc. INTERSPEECH*. ISCA, 2008.

[11] Hynek Hermansky, Dan Ellis, and Sangita Sharma. Tandem connectionist feature extraction for conventional hmm systems. In *Proc. IEEE ICASSP*, 2000.

[12] Takaaki Hori, Ramon Astudillo, Tomoki Hayashi, Yu Zhang, Shinji Watanabe, and Jonathan Le Roux. Cycle-consistency training for end-to-end speech recognition. *CoRR*, 2018. URL `https://arxiv.org/abs/1811.01690`.

[13] Toshio Irino, Yasuhiro Minami, Tomohiro Nakatani, Minoru Tsuzaki, and H. Tagawa. Evaluation of a speech recognition / generation method based on hmm and straight. In *Proc. INTERSPEECH*. ISCA, 2002.

[14] Frederick Jelinek. Continuous speech recognition by statistical methods. In *Proceedings of the IEEE 64*, 1976.

[15] Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N. Sainath, Zhijeng Chen, and Rohit Prabhavalkar. An analysis of incorporating an external language model into a sequence-to-sequence model. In *Proc. IEEE ICASSP*, 2018.

[16] A. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. *CoRR*, 2016. URL `https://arxiv.org/abs/1610.09038`.

[17] Yann LeCun. Power and limits of deep learning for signal understanding. Plenary talk given at Proc. IEEE ICASSP, 2018.

[18] C M Bishop and SpringerLink Service ligne. *Pattern Recognition and Machine Learning*, volume 1. 01 2006.

[19] Erik McDermott, Georg Heigold, Pedro J. Moreno, Andrew W. Senior, and Michiel Bacchiani. Asynchronous stochastic optimization for sequence training of deep neural networks: towards big data. In *Proc. Interspeech*. ISCA, 2014.

[20] Matthias Paulik. Lattice-based training of bottleneck feature extraction neural networks. In *Proc. Interspeech*. ISCA, 2013.

[21] Hasim Sak, Oriol Vinyals, Georg Heigold, Andrew W. Senior, Erik McDermott, Rajat Monga, and Mark Z. Mao. Sequence discriminative distributed training of long short-term memory recurrent neural networks. In *Proc. Interspeech*. ISCA, 2014.

[22] Michael Schuster. *On Supervised Learning From Sequential Data With Applications For Speech Recognition*. PhD thesis, 1999.

[23] F. Seide, G. Li, X. Chen, and D. Yu. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Proc. ASRU*, 2011.

[24] Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. Cold fusion: Training seq2seq models together with language models. In *Proc. Interspeech*. ISCA, 2018.

[25] Kenneth N. Stevens. Toward a model for speech recognition. *Journal of the Acoustical Society of America*, 32:47–55, 01 1960.

[26] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. Listening while speaking: Speech chain by deep learning. *CoRR*, abs/1707.04879, 2017. URL `http://arxiv.org/abs/1707.04879`.

[27] Z. Tuske, M. A. Tahir, R. Schlüter, and H. Ney. Integrating gaussian mixtures into deep neural networks: Softmax layer with hidden variables. In *Proc. IEEE ICASSP*, 2015.

[28] Zoltan Tuske, Wilfried Michel, Ralf Schlüter, and Hermann Ney. Parallel neural network features for improved tandem acoustic modeling. In *Proc. Interspeech*. ISCA, 2017.

[29] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *Arxiv*, 2016. URL `https://arxiv.org/abs/1609.03499`.

[30] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In *Proc. IEEE ICASSP*, 2014.

[31] Ehsan Variani, Erik McDermott, and Georg Heigold. A gaussian mixture model layer jointly optimized with discriminative features within a deep neural network architecture. In *Proc. IEEE ICASSP*, 2015.

[32] Ehsan Variani, Tom Bagby, Erik McDermott, and Michiel Bacchiani. End-to-end training of acoustic models for large vocabulary continuous speech recognition with tensorflow. In *Proc. Interspeech*. ISCA, 2018.

[33] Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc V. Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis. In *Proc. INTERSPEECH*. ISCA, 2017.

[34] Steve J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book Version 3.4*. Cambridge University Press, 2006.

[35] H. Zen and A. Senior. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In *Proc. IEEE ICASSP*, 2014.

[36] Heiga Zen and Hasim Sak. Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *Proc. IEEE ICASSP*, 2015.

[37] Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *CoRR*, abs/1503.03535, 2015. URL `https://arxiv.org/abs/1503.03535`.

Appendix A: Evaluation of different covariance models for autoregressive density networks modeling shallow logmel features

This Appendix describes the effect of covariance models stronger than the radial model used in the main body of the article, evaluated on logmel features modeled by autoregressive density networks. Only a single gaussian per mixture is used for all radial ("Rad"), diagonal ("Diag") and full ("Full") covariance models. The same prediction shift ("SH") of 2 frames is used. (A shift of 2 frames ensures there is nearly no overlap between input frames and prediction target frame, given the 64 ms audio analysis window size and 30 ms window step size). Costs and frame accuracies for these models are shown in Table 3. One can see that there is no big difference in cost between the radial and diagonal covariance models, but a big improvement is observed for the use of full covariance models. The difference between supervised and unsupervised models, also reported here, is not clearly correlated with the strength of the covariance model. The improvements in prediction cost do not improve the low frame accuracy observed for the logmel density models, but as discussed in the main body of the article, the frame accuracy metric used may be misleading.

Table 3: Cost and frame accuracy for radial, diagonal and full covariance models used in autoregressive density networks predicting (shallow) logmel features, trained with ML.

Model	Phones	Autoreg	Topology	Mixture size	Params	Cost	FAcc
Deep density Rad SH=2	Unsup	AR	5x512	1 total	4.5M	215	-
Deep density Rad SH=2	CI	AR	5x512	1 / phone	4.5M	190	31%
Deep density Diag SH=2	Unsup	AR	5x512	1 total	4.6M	205	-
Deep density Diag SH=2	CI	AR	5x512	1 / phone	4.6M	185	31%
Deep density Full SH=2	Unsup	AR	5x512	1 total	21M	-2.5	-
Deep density Full SH=2	CI	AR	5x512	1 / phone	21M	-6.5	30%

Appendix B: Evaluation of target prediction shift for autoregressive density networks modeling shallow logmel features

This Appendix describes the effect of different target prediction shifts ("SH") for autoregressive density networks modeling logmel features, focusing on the difference between supervised and unsupervised models. The results are shown in Table 4. One can see that (1) the prediction cost is substantially worse for the larger prediction shifts, (2) the difference between supervised and unsupervised is substantially larger, and (3) no real impact is observed on frame accuracy.

Table 4: Cost and frame accuracy for different prediction shifts ("SH") used in autoregressive density networks predicting (shallow) logmel features, trained with ML.

Model	Phones	Autoreg	Topology	Mixture size	Params	Cost	FAcc
Deep density Rad SH=1	Unsup	AR	5x512	1 total	4.5M	142	-
Deep density Rad SH=1	CI	AR	5x512	1 / phone	4.5M	135	29%
Deep density Diag SH=1	Unsup	AR	5x512	1 total	4.6M	140	-
Deep density Diag SH=1	CI	AR	5x512	1 / phone	4.6M	130	29%
Deep density Rad SH=2	Unsup	AR	5x512	1 total	4.5M	215	-
Deep density Rad SH=2	CI	AR	5x512	1 / phone	4.5M	190	31%
Deep density Rad SH=3	Unsup	AR	5x512	1 total	4.5M	247	-
Deep density Rad SH=3	CI	AR	5x512	1 / phone	4.5M	210	31%
Deep density Rad SH=4	Unsup	AR	5x512	1 total	4.5M	300	-
Deep density Rad SH=4	CI	AR	5x512	1 / phone	4.5M	240	29%

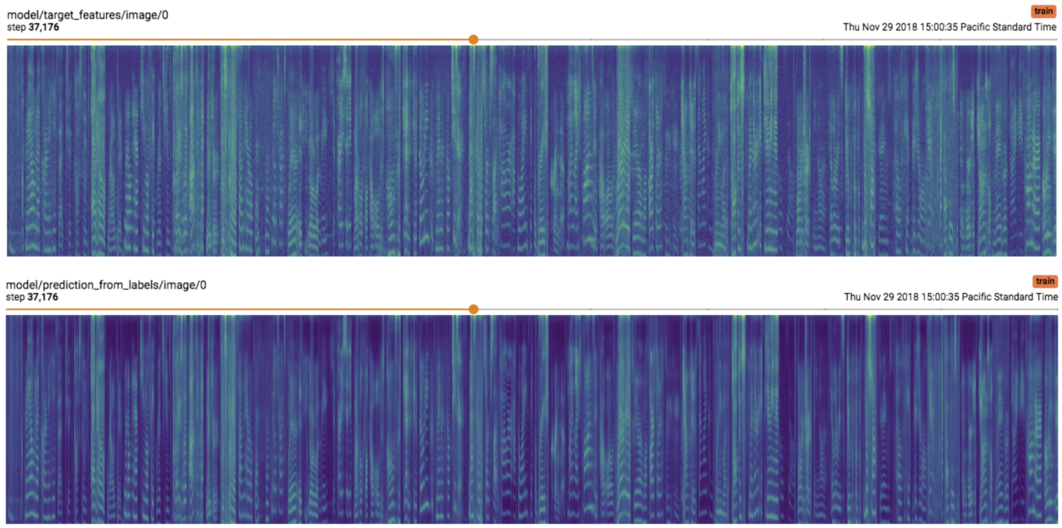

Figure 7: Unsupervised autoregressive density net, SH=1, target (top) vs. predicted (bottom) features

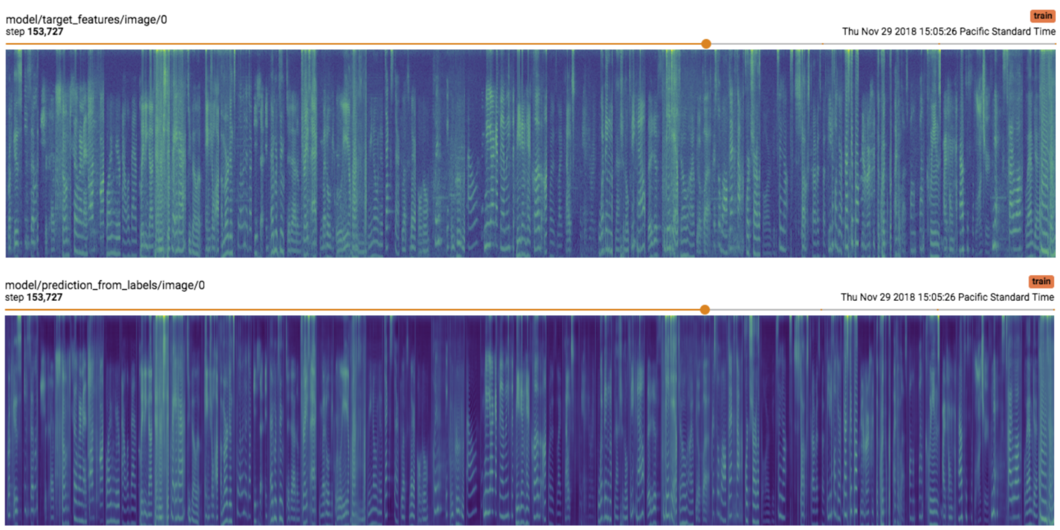

Figure 8: Supervised autoregressive density net, SH=1, target (top) vs. predicted (bottom) features

Appendix C: Visualization of predicted vs. target logmel feature vectors using autoregressive density networks with different prediction target shifts

This Appendix uses logmel spectrograms to illustrate the behavior of both unsupervised and supervised autoregressive deep density networks trained to predict logmel features with different prediction target shifts. Radial deep density networks were trained with a single gaussian per mixture, and the sequence of generated mean vectors plotted over time for unseen data from a separate dataset of read speech, alongside the actual target logmel feature vectors for the same data. The figures here illustrate target features and generated means for individual data mini-batches, i.e. 64 concatenated segments of 20 frames each, used in the truncated LSTM unrolling scheme described in Section 6.2. For a prediction shift of 1 frame (corresponding to the "SH=1" results for the radial model in Table 4), there is no visually discernible difference between unsupervised and supervised prediction, illustrated in Figure 7 and Figure 8, respectively; the label encoding is not necessary to make a good prediction. (A shift of 1 frame corresponds to a nearly 50% overlap in analysis window, given the 64 ms window and 30 ms window step). This behavior matches the rather small advantage in prediction cost observed for the supervised model over the unsupervised model in the corresponding "SH=1" results in Table 4.

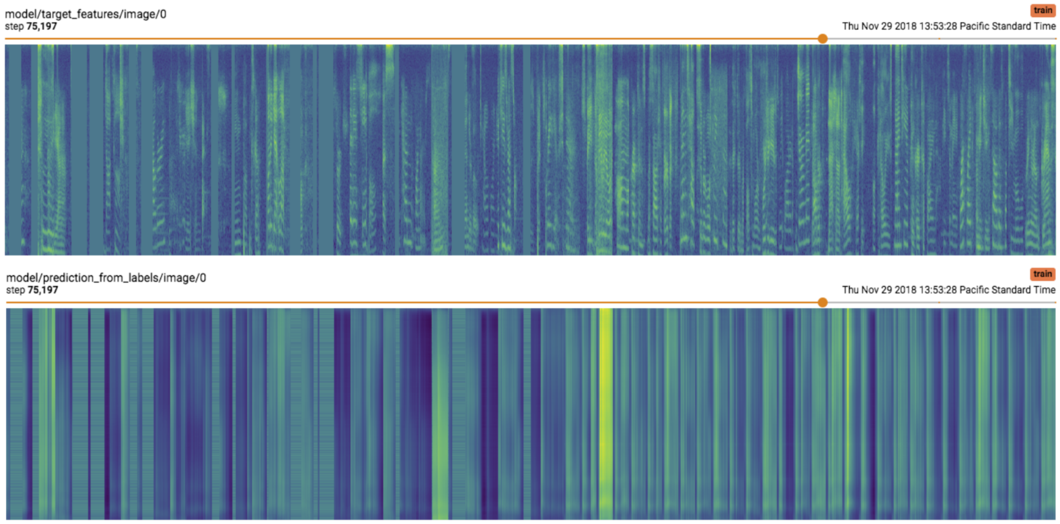

Figure 9: Unsupervised autoregressive density net, SH=4, target (top) vs. predicted (bottom) features

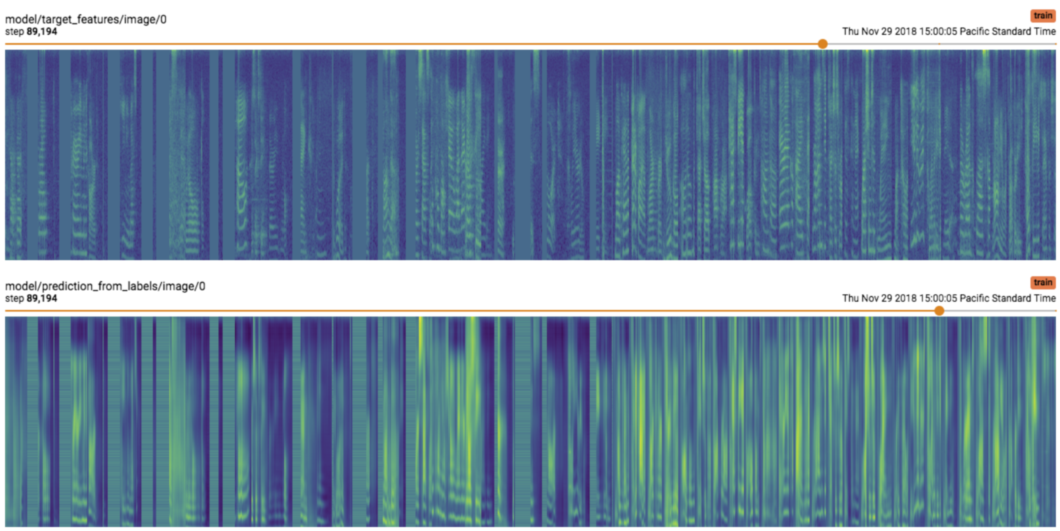

Figure 10: Supervised autoregressive density net, SH=4, target (top) vs. predicted (bottom) features

In contrast, a shift of 4 frames makes the unsupervised prediction quite poor (Figure 9), and there is a noticeable advantage for the supervised prediction, which can leverage the label encoding (Figure 10). These plots intuitively match the much better prediction cost for the supervised model compared to the unsupervised model for the corresponding "SH=4" results in Table 4.