Literature study on the current use of Deep Neural Networks for Automatic Speech Recognition

Pierre Petitbon
petitbon@kth.se

Abstract

This paper is an exploration of the current state-of-the-art for ASR: Deep Neural Networks. We analyse 3 types of Deep Neural Networks: MLP-HMMs, RBMs and Convolutional Deep Neural Networks. We focus on the MLP-HMMs and try to understand their effectiveness by comparing them to the previous state-of-the-art, GMM-HMMs. We also identify techniques to overcome the shortcomings of MLP-HMMs.

Introduction

Deep neural networks techniques were already known before the 2010s, but they could not be fully used because of the limitations of the previous generations of hardware. The key of DNNs is their depth. Deep architectures give rise to hierarchical learning, which increases the expressive power tremendously. Actually, using only one hidden layer and an arbitrary large number of nodes in it, a multi-layer perceptron is a universal function approximator, meaning that it can approximate arbitrarily well any continuous function on a compact subset of $\mathbb{R}^n$. This is known as the universal approximation theorem. However, the number of required nodes increases a lot with the complexity of the function to approximate, and quickly becomes too large. That is where deep learning shines: with more hidden layers, we can model more complexity with the same total number of nodes.

The problem that we will mostly consider is phone recognition. In other words, we are interested in the task of finding the sequence of phonemes associated with a spoken utterance, without being given the boundaries of the different phonemes. We will only consider the problem of phone classification when discussing Convolutional Deep Neural Networks. In phone classification, the boundaries of the phonemes are assumed to be known (see Figure 1).

1 GMM-HMMs

Here we will provide a relatively short description of GMM-HMMs. The HMM models the sequential structure of speech. The hidden sequence of states in the HMM is the sequence of underlying phonemes, and the observations are features derived from the frames of speech. The features commonly used are MFCCs, including the 0^{th} coefficient. Optionally, the deltas and the accelerations (first and second order derivatives resp.) can be added.

A key property of HMMs is that the observations are conditionally independent given their associated states. However, in speech, this is generally not true, since adjacent frames are actually correlated.

We associate a mixture of gaussians to each state. They represent the emission probabilities in the HMM (see Figure 2). An important point is that the model expressiveness is mostly held in the emission probability distributions, not so much in the transition probabilities.

[1]Image taken from http://jofrhwld.github.io/papers/gulp_2014/#/
[2]Image taken from http://www.mental.sk/temp/ai/csr.html

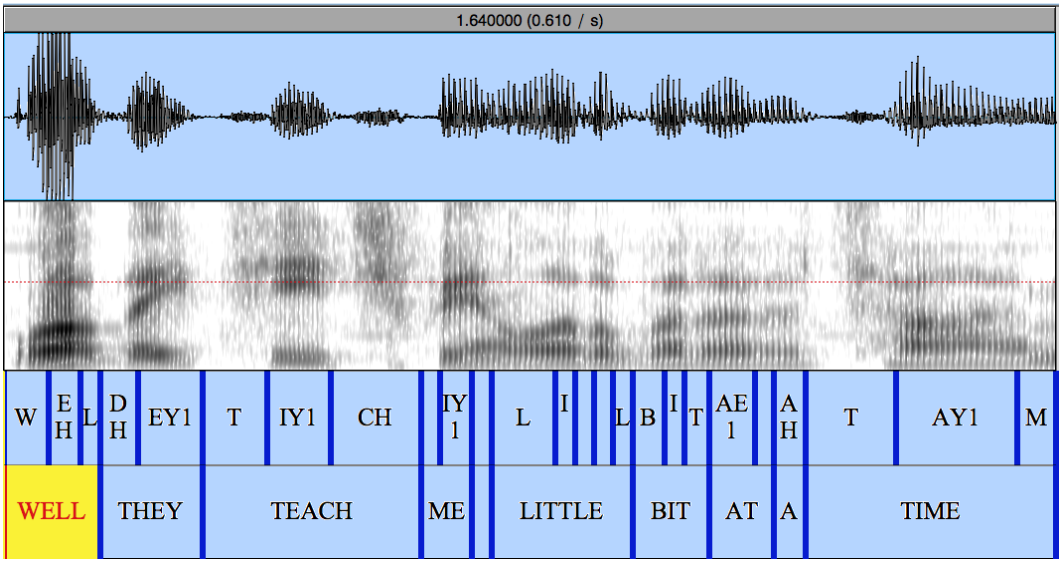


Figure 1: Forced alignment[1]

From top to bottom:

1. Speech signal in time domain.
2. Short-Time Discrete Fourier Transform of the speech signal.
3. Result of forced alignment: phoneme boundaries (with labels for the correct phonemes).
4. Correct words of the sentence, aligned with the signal.

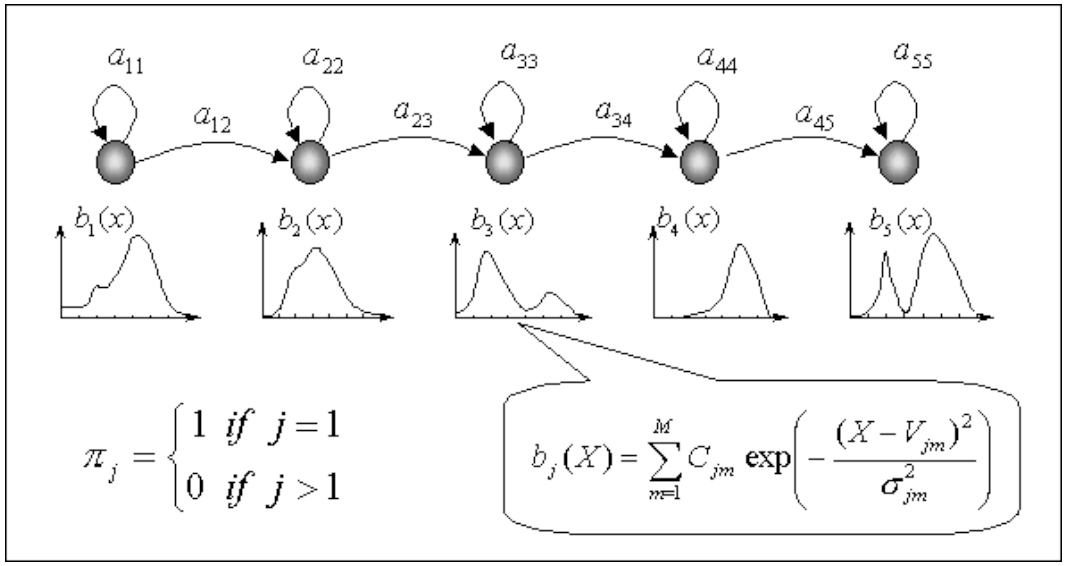


Figure 2: GMM-HMM[2]

- The a_{ij}s represent the transition probabilities.
- The b_{ij}s represent the emission probabilities (mixtures of gaussians).

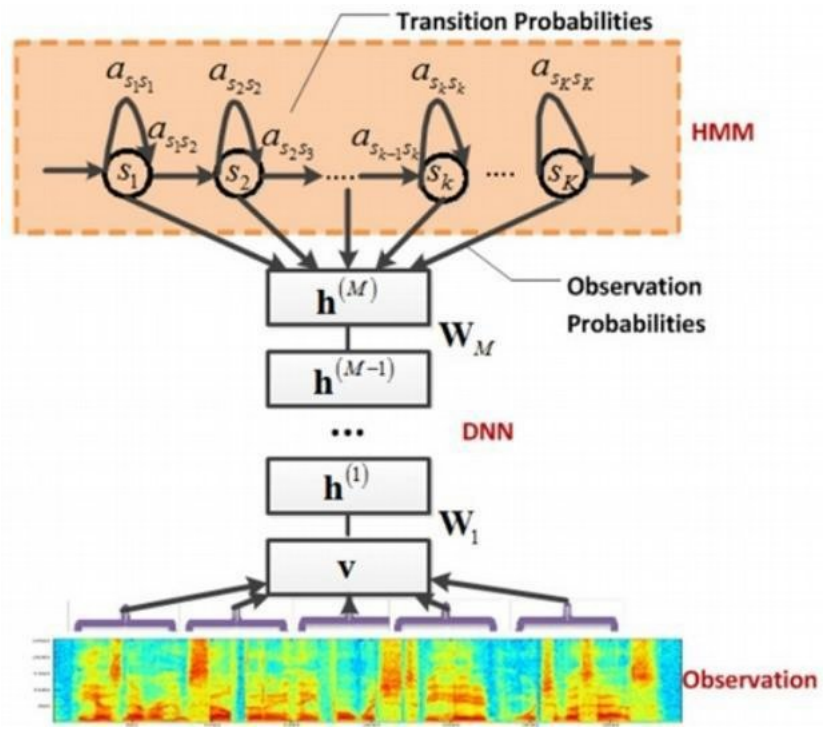


Figure 3: MLP-HMM[3]

- $s_1 \dots s_K$ represent the states (phonemes).
- The $a_{s_i s_j}$s represent the transition probabilities.
- v is the observation vector.
- $h^{(1)} \dots h^{(M)}$ are the activation vectors for the hidden layers of the network.
- $W_1 \dots W_M$ are the weight matrices that connect the layers of the network.
- The output layer $h^{(M)}$ represents the posterior probabilities, from which the observation probabilities can be derived by applying Bayes' rule.

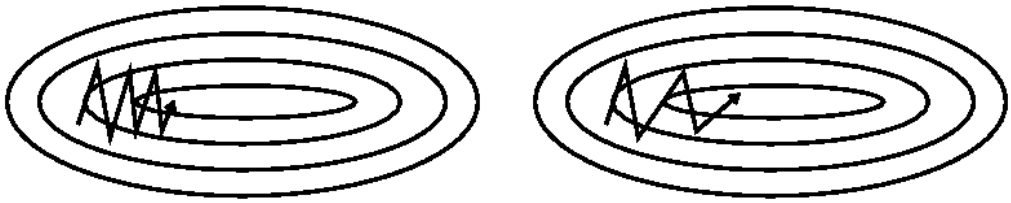
Training is performed using the EM algorithm, called Baum-Welch in the specific case of HMMs. Decoding (finding the most likely state sequence given an observation sequence) relies on the Viterbi algorithm which automatically performs forced alignment (position the boundaries of the phonemes in the utterance).

2 MLP-HMM

The idea of MLP-HMM is to use a Multi-Layer Perceptron instead of Gaussian Mixture Models to approximate the emission probabilities. The structure and basic theory of the MLP is assumed to be known to the reader.

The input of the MLP is a window of MFCC vectors for 9 to 13 adjacent frames. The output is a (discrete) probability distribution over all possible HMM states, in other words, the posterior probabilities (see Figure 3). To model a probability distribution, a softmax layer is used at the end of the network. Also, the loss function used is no longer the traditional Mean Squared Error (MSE), but the cross-entropy, which measures how different two discrete probability distributions are. The target probability distribution associated with a training sample is the deterministic probability distribution corresponding to the target phoneme.

[3]Image taken from `http://research-srv.microsoft.com/pubs/144412/dbn4lvcsr-transaslp.pdf`



(a) Without momentum (b) With momentum

Figure 4: Effect of momentum on gradient descent[4]

It should be noted that in order to train the MLP-HMM, we need to compute the emission probabilities, but the network gives us the posterior probabilities. Thus, we need to apply Bayes' rule to go from the posteriors to the emission probabilities.

Minibatch training is often performed. As its name indicates, it consists in processing the training data in minibatches. We use the term minibatch because the word batch represents the whole training set. The size of the minibatch should be large enough to stabilize learning, but small enough to avoid local minima better.

The MLP-HMM has a number of advantages over its GMM counterpart. First, it does not assume that the observations are conditionally independent, and it models correlation between adjacent frames, thus making use of additional useful information. Also, the MLP has more expressive power than the GMMs for the same number of parameters: it is a more efficient function approximator for the problem at hand [2] (page 2). A possible explanation is that the MLP-HMM builds a shared representation of the input: the weights before the output layer are useful for every output neuron (for every HMM state). On the contrary, GMM-HMMs have a separate representation for each state, which is supposedly less efficient.

But of course, advantages rarely comes without drawbacks. The MLP is very prone to overfitting, and can easily get stuck in a local minima of the loss function. We will explore some solutions to alleviate these problems shortly. A number of hyperparameters have to be chosen. MLPs can be harder to interpret than GMMs, and although the theory behind the MLP is relatively straightforward, it is not yet fully understood why deep learning works so well. Finally, the MLP makes training harder to parallelize on computer clusters [4], but GPU programming has been shown to be quite effective [5].

3 Practical considerations for the MLP

3.1 Stochastic gradient descent

Stochastic gradient descent has already been mentionned as a good way to make the training phase faster. The two extremes are online learning and batch training. The online version consists in training the network one training sample at a time. This has the advantage of making the training rather noisy, which is a good thing to avoid local minima. A physical analogy would be that if you are stuck in a crevasse, being pushed around somewhat randomly can get you out of there. The batch version averages out the weight updates over the whole training set. The advantage is stable learning, but it can easily get stuck in a local minima. Stochastic gradient descent is all about striking a balance between these two strategies [1] (pages 70-72).

3.2 Momentum

Momentum consists in remembering the previous update and combining it to the vector given by gradient descent to make learning faster. This is especially useful in the valleys of the loss function, to avoid the oscillations happening under regular gradient descent [1] (page 73) (see Figure **??**).

[4]Images taken from `http://sebastianruder.com/optimizing-gradient-descent/`

3.3 Regularization techniques

Regularization techniques aim at adding regularity so as to reduce the model complexity and thus reduce overfitting. They are often better solutions than simply reducing the number of layers and/or the number of nodes per layer.

3.3.1 Weight decay

Weight decay belongs to the subset of regularization techniques that add a penalty term to the loss function. This penalty term must have a positive correlation with the complexity of the model. For weight decay, we add the sum of the euclidian norms of the weights. A parameter λ called regularization rate is used to scale the penalty term, making the regularization more or less pronounced.

$$\text{loss} = \text{MSE} + \lambda \sum_{l,i,j} \|\mathbf{W}_{i,j}^{l}\|^2 \tag{1}$$

The main idea behind this technique is that the penalty for larger weights will make the network set small weights to zero. These smaller weights are considered a major source of overfitting, and setting them to zero should improve the generalization capabilities of the network [1] (pages 68-69).

3.3.2 Early stopping

This technique is a simple heuristic that consists in stopping the training once the test error starts increasing, or increases for a given number of consecutive training examples [7].

3.4 Pre-training

Pre-training of Deep Neural Networks is accomplished in an unsupervised and layer-wise fashion. As its name indicates, it occurs before the fine-tuning of backpropagation. It can be achieved using either Restricted Boltzmann Machines or denoising auto-encoders. The unsupervised nature of pre-training allows us to potentially make use of a lot of unlabelled data, which is much more abundant than labelled data. At the beginning of the recent deep learning trend, pre-training was thought to have a very significant role in the success of deep architectures. Two hypothesis were proposed to explain this fact. The first is known as the regularization hypothesis. The idea is that the unsupervised pre-training would constrain the explored weight space and thus reduce model complexity and overfitting [8]. The second hypothesis is based on the analogy of climbing a mountain. The pre-training part would be akin to climbing to the top of the valley, and the fine-tuning would amount to walking along the top of the valley toward a higher point. In other words, the pre-training gives a much better starting point for further optimization [2] (page 4). In fact, this is known as the optimization hypothesis.

However, the importance of pre-training is now more subject to debate, and pre-training is nowhere as essential as was previously thought. However, they remain very useful for smaller datasets [2] (page 14).

4 RBM

The Reduced Boltzmann Machine is an energy-based model. When working with RBMs, the goal is to reach states of low energy. Of course, the energy function is based on a physics analogy. Low-energy states (local minima of the energy function) are the equivalent of stable equilibrium positions in physics.

For training, using an exact method is too computationally expensive, so a fast approximate method called constrative divergence is used instead [6].

Stacking RBMs on top of each other is done to build a Deep Belief Network, which can be used as a feature detector in speech recognition.

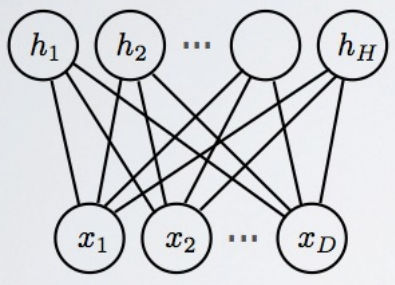


Figure 5: RBM[5]

- $h_1 \dots h_H$ are the hidden nodes.
- $x_1 \dots x_D$ are the visible nodes.

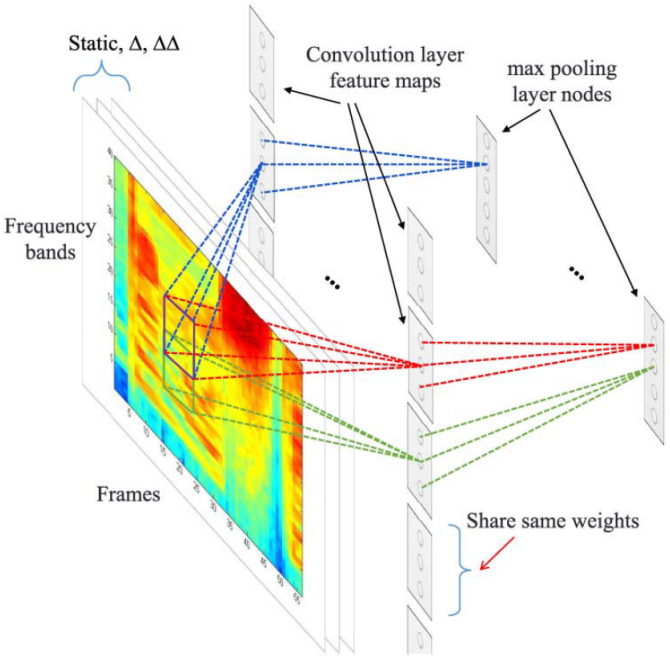


Figure 6: CNN[6]

5 Convolutional Deep Neural Networks

As highlighted in the introduction, we tackle here the phone classification task, meaning that phoneme boundaries are assumed to be known.

Convolutional Deep Neural Networks for speech recognition can be applied in the time domain (across frames) or the frequency domain (for a given frame) or in both. In the time domain, convolutional networks have the property of time invariance. This property is achieved through shared weights, which means that the same set of weights is actually applied on all parts of the speech signal. The shared weights are found in the convolutional layers. The other type of layer found in convolutional networks is the pooling layer, which consists in decreasing the resolution of the input by a given factor, and replacing parts of the input by the maximum or the average value on those parts.

[5]Image taken from `http://eric-yuan.me/rbm/`

[6]Image taken from `http://recognize-speech.com/acoustic-model/knn/comparing-different-architectures/convolutional-neural-networks-cnns`

In the frequency domain, the weight sharing has to be more local. Indeed, very different frequencies often hold very different kinds of information, so they should be processed separately. Thus, shared weights cover only parts of the frequency domain. In the frequency domain, convolutional DNNs have interesting properties, such as speaker invariance with respect to pitch, and an increased noise robustness [1] (page 12).

Convolutional DNNs are an effective way to perform feature identification. Combined with MFCCs, they have been shown to lead to a 0.7% increase in performance for phone classification on the TIMIT dataset, which is a very substantial gain [3].

Discussion and conclusions

DNN-HMMs have very good performance on large vocabulary speech recognition. We have highlighted a number of crucial points to understand their effectiveness. First, the expressiveness of such models stem from the depth and hierarchical nature of their architectures. Their modelling of correlation in neighboring frames is also a key feature, allowing more relevant information to be taken into account.

However, DNN-HMMs also have drawbacks, including the need to set hyperparameters, reduced interpretability and incomplete theoretical understanding.

The pre-training of DNNs is not so essential as we used to think, except for smaller datasets.

Finally, Convolutional DNNs are good feature detectors that can be used to add more useful information to the traditional MFCCs and increase performance even further.

References

[1] Yu, D. & Deng, L. (2015) *Automatic Speech Recognition. A Deep Learning Approach*, chapters 4-6.

[2] Hinton, G. et al. (2012) Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. In *IEEE Signal Processing Magazine, vol. 29, no. 6*, pp. 82-97.

[3] Lee, H. & Pham, P. & Largman, Y. & Ng, AY. (2009) Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*.

[4] Pethick, M. & Liddle, M. & Werstein, P. & Huang, Z. (2003) Parallelization of a Backpropagation Neural Network on a Cluster Computer. In *Parallel and Distributed Computing and Systems*. IASTED/ACTA Press.

[5] Hannun, A. et al. (2014) Deep Speech: Scaling up end-to-end speech recognition.

[6] Hinton, G. (2002) Training products of experts by minimizing contrastive divergence. In *Neural Computation*, vol. 14, pp. 1771–1800.

[7] Bourlard, H. & Morgan, N. (1993) Connectionist Speech Recognition: A Hybrid Approach, Kluwer Academic Publishers, Norwell, MA, USA.

[8] Erhan D. et al. (2010) Why Does Unsupervised Pre-training Help Deep Learning? In *Journal of Machine Learning Research 11*, pp. 625-660.