JOINT TRAINING OF CONVOLUTIONAL AND NON-CONVOLUTIONAL NEURAL NETWORKS

Hagen Soltau, George Saon, and Tara N. Sainath

IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA

ABSTRACT

We describe a simple modification of neural networks which consists in extending the commonly used linear layer structure to an arbitrary graph structure. This allows us to combine the benefits of convolutional neural networks with the benefits of regular networks. The joint model has only a small increase in parameter size and training and decoding time are virtually unaffected. We report significant improvements over very strong baselines on two LVCSR tasks and one speech activity detection task.

Index Terms— Acoustic Modeling, Neural Networks, MLP, CNN

1. INTRODUCTION

While until recently most LVCSR systems were based on Gaussian mixture models (GMM), in the last two years many research groups have seen substantial improvements when switching to neural network acoustic models. The renewed interest in neural networks was sparked by the work of Microsoft [1] where a context dependent neural network outperformed a good GMM baseline on the SWB (Switchboard) task. While the work in [1] used multi layer perceptrons (also called deep neural networks now), other types of neural Nets have become popular as well, such as convolutional neural networks (CNN) [2].

The idea of a CNN is to generate shift invariance to make the model more robust against small changes in the input space and was already discussed by Rumelhart in [3]. He proposed a network that uses only a subset of inputs in the form of localized receptive fields. That network was designed to discriminate between the letters *"T"* and *"C"* and to be *invariant* to translation. In order to achieve shift invariance, the weight learning was changed such that the weight changes were averaged over the receptive fields. For speech recognition problems, invariance against small changes in the temporal domain is important. A time delay neural network (TDNN) [4] applies weight sharing and shift invariance in the temporal domain on a phoneme classification task and performed better than an HMM baseline. Weight sharing and shift invariance in the feature domain was first explored in [5], which used *log-mel* features as input features on a small scale speech task. The work in [6] demonstrated that these ideas also work for larger tasks (Broadcast News and Switchboard).

If CNNs are configured to obtain shift invariance in the feature domain, it places certain constraints on the type of features that can be used. Applying a *maximum* or *average* operator on the outputs of localized windows is meaningful only if the features are topographical. For example, *log-mel* features, which are used in [5], have this property. On the other hand, considerable progress has been made by using more elaborate feature processing for GMM systems. These features can be used directly for regular MLPs and are known to improve results. This motivates our work, where we want to combine the benefits of a CNN (shift invariance) with the benefits of a conventional MLP that can use more advanced features.

Our model is a simple extension of a regular neural network, described in section 3. We compare the performance of our models on two LVCSR tasks. The first set of experiments are done on SWB, a well understood LVCSR task, in section 4. The second task, RATS (Robust Automatic Transcription of Speech), features challenging noisy conditions for multiple languages (section 5). We report results for two RATS sub-tasks, namely keyword search and speech activity detection.

2. RELATED WORK

Closest to our work is the work described in [7], which mentioned a combination of MLPs and CNNs. They reported small improvements (18.9% to 18.6% WER) while almost doubling the parameter size.The difference with our work is that our model is configured such that most layers are shared between the MLP and the CNN and that we do not restrict our model to have the same input features for both MLP and CNN. Indeed, in some preliminary experiments we found that log-mel features are substantially worse than FMLLR features for MLPs and placing the same restriction for MLP features that are used for CNN features will lead to suboptimal results.

Combining the benefits of different models or different features can also be seen as a form of system combination. While our work focuses on neural nets, combining different feature streams was already explored for GMM systems. For example, the authors in [8] combined MLP based posterior features with traditional MFCC features for a GMM based LVCSR system. More recently, [9] showed improvements from combining different NN derived features with a GMM system.

In [10], we experimented with combining different features (FMLLR,FMMI,log-mel,FDLP) for MLP-based acoustic models. While we observed improvements on a $50h$ Broadcast News task, the gains disappeared when scaling up to more training data and larger models. In contrast to these feature combination approaches, we propose a model that allows us to train different types of neural nets jointly.

3. MODEL

While a regular MLP (or CNN) is normally a linear sequence of layers, our model consists of a graph structure of layers. Each layer can receive inputs from multiple layers, and each layer can send its output to multiple layers. The difference with a conventional MLP is minimal: in the forward pass, the outputs of *all* input layers have to be combined (joined). In the backward pass, the gradients of *all* output layers have to be summed up before back-propagating the

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

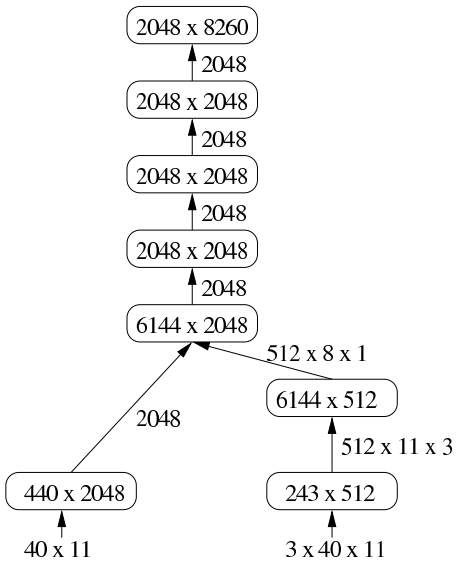

Fig. 1. Joint MLP/CNN model. The matrix dimensions of the weights are marked inside the boxes. The size of the inputs for each layer are marked next to the arrows. The non-convolutional inputs are 40-dim FMLLR features in a context window of 11 frames. For the convolutional part, 40-dim log-mel features are used together with their Δ's and $\Delta\Delta$'s in a context window of 11 frames.

gradient. The graph structure is not restricted to the hidden layers only, the models also allows for multiple input features and outputs.

An example of such a network is shown in Figure 1. In this case, the network consists of two convolutional layers (right side) and a non-convolutional layer (left side) that share four hidden layers plus the output layer.

In the following, we list the benefits of a graph-structured neural network.

3.1. Joint training of MLPs and CNNs

For a conventional MLP, it is easy to use multiple input features by simply combining the feature matrices into one single input matrix. This is not the case for CNNs. CNNs achieve shift invariance by applying a *pooling operation* of the output values. In order to achieve shift invariance in the feature domain, the features have to be *topographical*, such as *log-mel* features. On the other hand, state-of-the-art GMM and MLP based systems make use of more advanced speaker adaptive features such as FMLLR or fMMI features. These features are *not* topographical and therefore can not be used with a CNN. The neural network graph structure allows us to use such features by having CNN and MLP layers in parallel as shown in Figure 1. Since most layers are shared, the joint MLP/CNN configuration shown in Figure 1 has only about 10% more parameters than the corresponding CNN.

3.2. Multi-GPU usage

Another advantage of the graph structure is that we can split one layer into n parallel parts (striping) [11]. For example, a 2048×7000 output layer can be split into two 2048×3500 layers (in parallel). Each matrix multiplication can run in parallel on separate devices. Combining the output of each layer is simply a *cudaMemcpy2D* call where we use the *pitch* parameter to specify the target position of the combined matrix. The nice part here is that *cudaMemcpy2D* can be used directly to copy memory between devices without any extra device management.

SI ML	23.2%
SI bMMI	21.4%
SI fMPE + bMMI	18.9%
SA fMPE + bMMI	14.5%

Table 1. GMM baseline error rate on HUB5-2000

3.3. Multi-task learning

While we have not exploring multi-task learning [12] in this work, another benefit of the graph structure is that it allows us to specify multiple targets. An example of multiple targets is the use of different decision trees as in [13]. Multiple target / output layers for neural networks is a form of multi-task learning and used to improve generalization. For example, speech enhancement features were used in [14]. The graph structure would look like a regular neural net with two parallel output layers, one for the state posteriors (with softmax and cross-entropy) and another one for clean target features (with sigmoid and MSE).

4. EXPERIMENTS ON SWB

4.1. GMM system

The following experiments are carried out on the Switchboard (SWB) task. The training set is the $300h$ SWB-1 corpus, the same as used in [1]. Error rates are reported for the Hub5-2000 test set. First, we used our regular training recipe (see [15] for details) to establish a GMM baseline. The model comes with all standard state-of-the-art techniques including VTLN, FMLLR, MLLR, LDA, STC plus feature and model space discriminative training. The system has 350000 Gaussians with 8260 states and its performance is summarized in Table 1.

4.2. MLP, CNN, and jointly trained MLP/CNN

All neural nets were trained with layer-wise back-propagation. The weights are randomly initialized. The initial step size is 5×10^{-3} and is reduced by half every time the performance on a held-out set does not improve. The training data is randomized at the frame-level for each $30h$ chunk. The mini-batch size is 256 frames. The MLP has six layers (left branch of Figure 1), while the CNN has one more layer (right branch of Figure 1). Each conventional hidden layer has 2048 nodes, while the two convolutional layers use 512 nodes. The MLP uses 40-dim FMLLR features with a temporal context of 11 frames. For the CNN, we use 40-dim VTL-warped log-mel features together with their Δ and $\Delta\Delta$ features. The temporal context for the CNN features is 11 frames.

The configuration of the CNN layers is shown in Table 2. For the first layer, we apply an overlapping window of 9×9 to the log-mel input stream. Pooling operates over 3 output values, a number we based on the findings of [5, 6]. Since we use 40 log-mel's with a temporal context of 11 frames, we get 32×3 windows. After applying a *max* operation over a non-overlapping window of 3×1, we get 11×3 windows. Applying an overlapping window of 4×3 from the second CNN layer, we end up with 8×1 windows that form the input for the conventional layer that fuses the MLP with

CNN #0	Feature domain		Temporal domain	
	Size	Shift	Size	Shift
Windows	9	1	9	1
Pooling	3	3	1	1

CNN #1	Feature domain		Temporal domain	
	Size	Shift	Size	Shift
Windows	4	1	3	1
Pooling	1	1	1	1

Table 2. Configuration for first and second CNN layer

Model	Cross Entropy	Hessian-free sMBR
MLP	13.8%	12.3%
CNN	13.2%	11.8%
MLP/CNN	12.8%	11.2%

Table 3. neural nets on HUB5-2000

the CNN (see Figure 1). The input dimensionality for the first layer is $3 \times 9 \times 9 = 243$ and for the second CNN layer, we have $512 \times 4 \times 3 = 6144$ (number of outputs from previous layer times number of windows).

In Table 3, we summarize the performance of various neural net models. First, the models were trained with standard backpropagation and cross entropy as objective function. The models were then retrained with Hessian-free sequence training as described in [16]. The MLP itself is already substantially better than the best GMM (12.3% vs 14.5%) and we gain another 0.5% improvement by switching to a CNN. The jointly trained MLP/CNN improves the error rate from 11.8% to 11.2%.

4.3. Joint training with I-Vectors

In the next experiment, we add I-Vectors [17] to the jointly trained model. I-Vectors were recently proposed in [18] as a form of speaker adaptation for neural nets. In this work, I-Vectors were used to augment regular FMLLR features to feed in extra speaker information when training neural nets. In [18], conventional MLPs were used for the experiments, and a 5-6% relative improvement was seen on top of MLPs with speaker adaptive features on a SWB task similar to the one used here. Since I-Vector derived features are not topographical, CNNs were not used in [18]. On the other hand, the CNN outperformed the MLP by 0.5% and the jointly trained model is 1.1% better than the MLP, so it is natural to look for ways of how to add I-Vectors to CNNs. The graph structure for our jointly trained model makes it easy for us to do that.

The I-Vectors were generated exactly the same way as described in [18]. A text-independent GMM with 2048 components is trained on the same training corpus that is used to train the neural nets. The frontend for the GMM uses the same features as the MLP: 40-dimensional FMLLR features. We extract 100-dimensional I-Vectors for every speaker and append them to the input stream for the MLP. This brings the total input for the MLP to $11 \times 40 + 100 = 540$ features. The experiments in [18] were done based on an MLP setup with FMLLR features on the same task as here and reported an error rate reduction from 12.5% to 11.9% after sequence training.

The results for the joint CNN/MLP are shown in Table 4. After sequence training, the error rate improves from 11.2% to 10.4% for

Model	Cross Entropy	Hessian-free sMBR
MLP/CNN	12.8%	11.2%
MLP/CNN + I-Vector	12.1%	10.4%

Table 4. Jointly trained MLP/CNN with I-Vectors, Error rates on HUB5-2000

the jointly trained MLP/CNN with I-Vectors. This shows that the improvements on the MLP setup carry over to the more elaborate joint MLP/CNN model and demonstrates the effectiveness of our model.

4.4. Comparison with System Combination

The jointly trained MLP/CNN can be seen as a form of system combination, where the outputs of the first MLP hidden layer get combined with the outputs of the second CNN layer. As a contrast experiment, we wanted to see how much we would gain from combining separate MLP and CNN models. Since ROVER does not work well when combining only two systems, we used *score fusion*, where we average the acoustic scores from both models. The error rate for the system combination of separate MLP (12.3%) and CNN (11.8%) is 11.2% - the same as for the jointly trained model.

We repeated the same experiment including the I-Vector configuration. In this case, the separate models have error rates of 11.9% (MLP+I-Vector) and 11.2% (CNN). Score fusion of both models gives us an error rate of 10.5%, slightly worse than the jointly trained model with 10.4%.

One way to look at this is that we can achieve the full gain of system combination with only one jointly trained model. The benefit of a jointly trained model is that we do not need to train two separate models (and training time for neural nets matters even on GPUs). Furthermore, during decoding we only need to do one acoustic score computation, whereas score fusion will double the acoustic score computation time. Also, the jointly trained model has only about 10% more parameters than the CNN alone, significantly less than the separate MLP and CNN.

5. EXPERIMENTS ON RATS

RATS is a DARPA program focusing on the noise robust processing of speech with several sub-tasks. We report in this paper experiments on two sub-tasks: speech activity detection (SAD) and keyword search (KWS). The data collection consists of re-transmitted clean data over a noisy channel. The "clean" audio data has Callhome type characteristics (telephone conversations), while the noisy data was obtained by transmitting the original audio through a pair of senders and receivers. In total, 8 different transmissions were performed by using different sender and receiver combinations.

5.1. Acoustic Models for Keyword Search

The goal of keyword search (also known as spoken term detection) is to locate a spoken keyword in audio documents. For this task, 300 hours of acoustic training data are available. From our point of view, KWS is essentially LVCSR plus some form of post-processing of lattices to generate a searchable index. The target languages for KWS are Levantine and Farsi. The baseline acoustic models are neural nets and described in detail in [10]. For an overview of the entire KWS system, we refer to [19]. The CNN and joint MLP/CNN

Model	A	B	C	D	E	F	clean
CNN	49.8	72.3	57.3	56.4	73.7	58.2	42.7
MLP/CNN	47.6	69.5	55.1	52.4	72.2	56.6	40.6

Table 5. Levantine RATS LVCSR, Error rates on dev-04

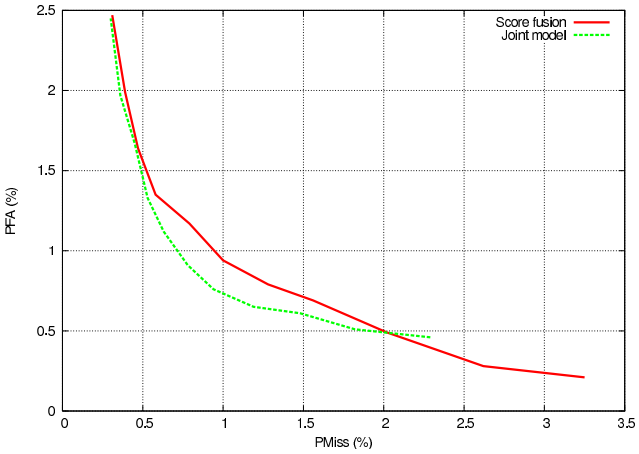

Fig. 2. Comparison between a joint MLP/CNN model and separate models with score fusion for speech activity detection on the RATS task.

configuration is similar to the setup used for the SWB experiments. The only difference is the number of output units, where we use 7000 HMM states for our RATS Levantine system. The results are shown in Table 5 for different noise conditions and demonstrate the effectiveness of the joint training approach.

5.2. Speech Activity Detection

The goal of the speech activity detection (SAD) task is to determine whether a signal contains speech or is just comprised of background noise or music. The performance is measured in terms of the probability of miss (P_{Miss}) and the probability of false accept (P_{FA}) which are defined as the duration of missed speech over the duration of total speech and the duration of false accept (or inserted) speech over the duration of total non-speech, respectively.

In Figure 2 we compare the ROC curves on the DEV1 test set which contains 11 hours of audio for two systems. The first system uses a score-level fusion of two neural nets: an MLP trained on a combination of PLP, voicing and FDLP features and a CNN trained on log-mel spectral features. More details about the models, features and normalizations can be found in [20]. The second system uses a joint MLP/CNN; the MLP part has the same inputs as the separately-trained MLP (PLP, FDLP and voicing) whereas the inputs for the CNN part are given by log-mel spectra (same as for the CNN trained in isolation). As can be seen, the joint model yields a 20% relative improvement in equal error rate over the separate models with score fusion.

6. CONCLUSIONS

We described a simple extension of neural networks, that changes the typical linear sequence of layers to a graph structure. The benefit of the graph structure is that it allows us to train convolutional and regular neural networks jointly. While I-Vectors are not topographical, the joint training approach allows us to leverage I-vectors for convolutional neural networks. Starting with our baseline CNN with an error rate of 11.8%, we reduced the error rate to 10.4%. This is a 10% relative improvement over a very strong baseline. We also demonstrated that our model works across different tasks, such as speech activity detection and LVCSR for RATS keyword search. Furthermore, the proposed neural graph structure allows us to implement other features in an elegant way, such as multitask learning or the parallel use of multiple GPU devices.

7. ACKNOWLEDGMENT

This work was supported by the DARPA RATS Program. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense. Approved for Public Release, Distribution Unlimited.

8. REFERENCES

[1] Frank Seide, Gang Li, and Dong Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Interspeech*, 2011.

[2] Y. Le Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," in *Proc. NIPS*, 1990.

[3] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, 1986.

[4] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition: Neural networks vs hidden Markov models," in *Proc. ICASSP*, 1988.

[5] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural network concepts to hybrid NN-HMM model for speech recognition," in *Proc. ICASSP*, 2012.

[6] T. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Proc. ICASSP*, 2013.

[7] Tara N. Sainath, Brian Kingsbury, Abdel-rahman Mohamed, George Dahl, George Saon, Hagen Soltau, Tomas Beran, Aleksandr Y. Aravkin, and Bhuvana Ramabhadran, "Improvements to deep convolutional neural networks for LVCSR," in *Proc. ASRU*, 2013.

[8] Qifeng Zhu, Andreas Stolcke, Barry Y. Chen, and Nelson Morgan, "Using MLP features in SRIs conversational speech recognition system," in *in Proc. Interspeech*, 2005, pp. 2141–2144.

[9] Pavel Golik, Zoltan Tueske, Ralf Schlueter, and Hermann Ney, "Development of the RWTH Transcription System for Slovenian," in *Interspeech*, Lyon, France, Aug. 2013, pp. 3107–3111.

[10] Hagen Soltau, Hong-Kwang Kuo, Lidia Mangu, George Saon, and Tomas Beran, "Neural Network Acoustic Models for the DARPA RATS Program," in *Proc. Interspeech*, 2013.

[11] Xie Chen, Adam Eversole, Gang Li, Dong Yu, and Frank Seide, "Pipelined back-propagation for context-dependent deep neural networks.," in *Proc. INTERSPEECH*, 2012.

[12] Rich Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.

[13] M. Seltzer and J. Droppo, "Multi-task learning in deep neural networks for improved phoneme recognition," in *Proc. ICASSP*, 2013.

[14] Y. Lu, F. Lu, S. Sehgal, S. Gupta, J. Du, C. Tham, P. Green, and V. Wan, "Multitask learning in connectionist speech recognition," in *Proc. ASSTA*, 2004.

[15] Hagen Soltau, George Saon, and Brian Kingsbury, "The IBM Attila speech recognition toolkit," in *Spoken Language Technology Workshop (SLT), 2010 IEEE*. IEEE, 2010, pp. 97–102.

[16] B. Kingsbury, T. N. Sainath, and H. Soltau, "Scalable minimum Bayes risk training of neural network acoustic models using distributed Hessian-free optimization," in *Proc. Interspeech*, 2012.

[17] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech and Language Processing*, vol. 19, no. 4, May 2011.

[18] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny, "Speaker adaptation of neural network acoustic models using I-Vectors," in *Proc. ASRU*, 2013.

[19] Lidia Mangu, Hagen Soltau, Hong-Kwang Kuo, and George Saon, "The IBM Spoken Term Detection System for the DARPA RATS Program," in *Proc. ASRU*, 2013.

[20] George Saon, Samuel Thomas, Hagen Soltau, Sriram Ganapathy, and Brian Kingsbury, "The IBM speech activity detection system for the DARPA RATS program," in *Proc. Interspeech*, 2013.