INVESTIGATION OF DENSELY CONNECTED CONVOLUTIONAL NETWORKS WITH DOMAIN ADVERSARIAL LEARNING FOR NOISE ROBUST SPEECH RECOGNITION

Chia Yu Li, Ngoc Thang Vu

Institute for Natural Language Processing, University of Stuttgart, Germany
{licu,thangvu}@ims.uni-stuttgart.de

Abstract: We investigate densely connected convolutional networks (DenseNets) and their extension with domain adversarial training for noise robust speech recognition. DenseNets are very deep, compact convolutional neural networks which have demonstrated incredible improvements over the state-of-the-art results in computer vision. Our experimental results reveal that DenseNets are more robust against noise than other neural network based models such as deep feed forward neural networks and convolutional neural networks. Moreover, domain adversarial learning can further improve the robustness of DenseNets against both, known and unknown noise conditions.

1 Introduction

In the last years, automatic speech recognition (ASR) performance has been significantly improved through the use of neural networks with deep structures [1, 2, 3]. Various neural network architectures have been developed to improve ASR performance. They are variations of time delay neural networks (TDNNs) [4], convolutional neural networks (CNNs) [5] recurrent neural networks (RNNs) [6], and their combinations [7]. Among them, very deep CNNs demonstrated impressive performance [8, 9, 10] especially in noisy conditions [8].

Recently in the computer vision research community, densely connected convolutional networks (DenseNets) have obtained significant improvements over the state-of-the-art networks on four highly competitive object recognition benchmark tasks [11]. The idea is to introduce shorter connections between layers close to the input and those close to the output which alleviate the vanishing-gradient problem. Furthermore, DenseNets requires fewer parameters than traditional CNNs with the same deep structure [11]. In [12], we showed that DenseNets can be used for acoustic modeling achieving impressive performance.

In this paper, we explore noise robustness of DenseNets and their extension with domain adversarial learning. This method was originally proposed by Ganin et al. [13] for unsupervised domain adaptation in natural language processing and was then applied to deep feed forward neural networks (DNNs) for noise robust speech recognition [14, 15]. However to our best knowledge, domain adversarial learning has never been examined with a complex network like DenseNets before. Our experimental results on noisy data demonstrate that DenseNets can effectively improve the noise robustness of the system outperforming other neural based models. Using domain adversarial learning can further improve their robustness against both, known and unknown noise conditions.

2 Methods

2.1 DenseNets Acoustic Models

In this subsection, we first describe DenseNets and review the method how to use DenseNets for acoustic modeling [12]. The key idea of DenseNets is the introduction of shorter connections between layers close to the input and those close to the output which alleviate the vanishing-gradient problem. For acoustic modeling, DenseNets take the unadapted features 40-dimensional log Mel filterbank as input and predict the context dependent HMM states (senones) [12].

Given an input x_0 and a CNN with N layers, where each layer n is equipped with a nonlinear transformation $H_n(\cdot)$ which is the composition of three consecutive operations: batch normalization, followed by a ReLU and a 3×3 convolution, DenseNets introduce direct connections from any layer to all subsequent layers. The output of nth layer is:

$$x_n = H_n([x_0, x_1, x_2, ..., x_{n-1}]) \tag{1}$$

where $[x_0, x_1, x_2, ..., x_{n-1}]$ refers to the concatenation of the feature maps yielded in all the previous layers. Fig. 1 illustrates the dense connectivity structure, in which each layer takes all preceding feature-maps as input. This structure is called *dense block*.
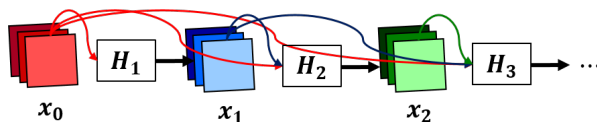

Figure 1 – A 3-layer *dense block*, in which each layer takes all preceding feature-maps as input

DenseNets consist of multiple *dense blocks*, connected in series and separated by *transition layers*. Each transition layer consists of a 1×1 convolution layer and a 2×2 average pooling layer. Fig. 2 illustrates how these *dense blocks* and *transition layers* are composed in DenseNets. Note that pooling is only performed outside of *dense blocks*.
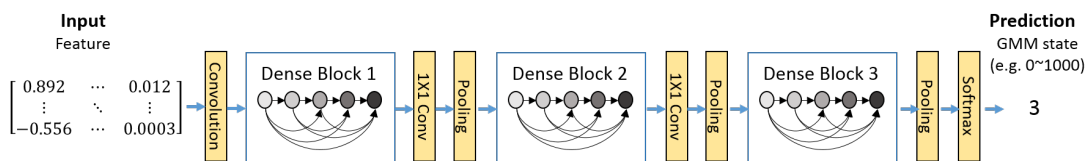

Figure 2 – A DenseNet architecture with three dense blocks connected via transition layers

Furthermore, DenseNets reduce the number of feature-maps by 1×1 convolution layer in the *transition layer* to improve model compactness. For example, if a *dense block* has y feature-maps, the transition layer generates $\lfloor \theta y \rfloor$ output feature-maps, where θ is the compression factor and the range is $0 < \theta \leq 1$. The *growth rate* of DenseNets is the number of channels in their convolution layers. By equation (1), the nth layer within a dense block has $k \times (n-1) + k_0$ input feature-maps, where k_0 is the number of input channels and k (the model's growth rate) is the number of channels for subsequent convolution layers. DenseNets have better performance when k is a small integer, e.g. $k = 12$ [11].
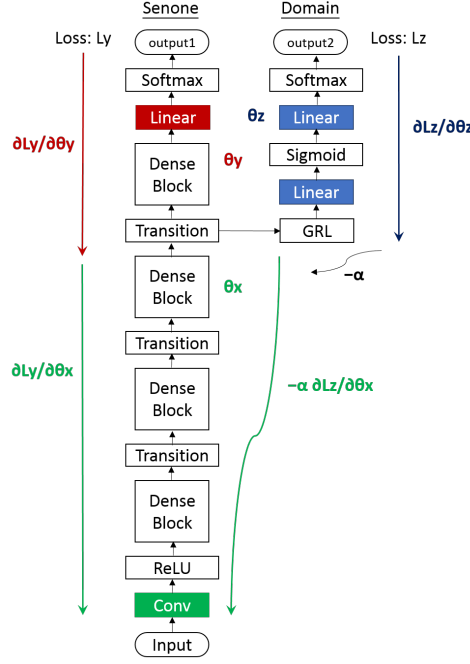
Figure 3 – An architecture of DenseNets with domain adversarial learning which consists of there sub-networks: features extractor sub-network (x), senone classification sub-network (y) and domain classification sub-network (z)

2.2 Domain Adversarial Learning

In this subsection, we introduce the extension of DenseNets with domain adversarial learning [13]. In this context, noisy conditions act as domain information.

The overall architecture of the domain adversarial learning of DenseNets is shown in Fig 3. It consists three sub-networks: the sub-network (y) is for senone classification, the sub-network (z) is for domain classification and the share-network (x) is the share part of the two tasks. The shared-network (x) can be seen as a feature extractor to convert an input vector to its latent representation. Each output sub-network acts as a classifier to calculate posterior probabilities of classes given the this latent representation [14, 15]. In the domain adversarial learning, the representation is learned adversarially to the domain classification and friendly to the senone classification, so that domain-dependent information to the senone classifier is removed from the representation.

Let $\theta_x, \theta_y, \theta_z$ denote the parameters of the share-network (x), sub-network (y) and sub-network (z), respectively. The cross-entropy loss functions for the senone classifier and domain classifier are defined as

$$\mathscr{L}_y(\theta_x, \theta_y) = -\sum_i log P(y_i|x_i; \theta_x, \theta_y) \tag{2}$$

$$\mathscr{L}_z(\theta_x, \theta_z) = -\sum_i log P(z_i|x_i; \theta_x, \theta_z) \tag{3}$$

The parameters are updated as

$$\theta_y \leftarrow \theta_y - \varepsilon \frac{\partial \mathscr{L}_y}{\partial \theta_y} \tag{4}$$

$$\theta_z \leftarrow \theta_z - \varepsilon \frac{\partial \mathscr{L}_z}{\partial \theta_z} \tag{5}$$

$$\theta_x \leftarrow \theta_x - \varepsilon\left(\frac{\partial \mathcal{L}_y}{\partial \theta_x} - \lambda \frac{\partial \mathcal{L}_z}{\partial \theta_x}\right) \tag{6}$$

where λ is the gradient reversal coefficient which is a positive scalar parameter to adjust the strength of the regularization.

The first layer of the sub-network (z) is the gradient reversal layer (GRL) as proposed in [14]. In the forward propagation phase, the GRL just passes the input to the output as follows:

$$\xi_{out} \leftarrow \xi_{in} \tag{7}$$

where ξ_{in} and ξ_{out} represent input and output vectors of the layer, respectively. In the backward propagation phase, the GRL reverses the gradient by multipling it with λ as follows:

$$\frac{\partial \mathcal{L}}{\partial \xi_{in}} \leftarrow -\lambda \frac{\partial \mathcal{L}}{\partial \xi_{out}} \tag{8}$$

Hence the shared-network (x) is trained adversarially to the sub-network (y) for the domain classification. When the training is finished, the output of the entire network (x, y) for the senone classification is used for decoding.

3 Setup

Two experiments are conducted in this paper. The goal for the first experiment is to explore DenseNets' robustness at different levels of noise. We compare the baseline models, which are deep neural networks (DNNs), CNNs and TDNNs, and DenseNets on noise corrupted DARPA 1000-words English language Resource Managemen (RM). The second experiment examines the effectiveness of domain adversarial learning, in which we compare the performance of DenseNets, DenseNets-AD and the best baseline model TDNNs on noise corrupted RM and Aurora 4 Task (Aurora4).

3.1 Resources

3.1.1 Data

The noise corrupted RM is made by artificially adding different types of noise at different values of SNR [1] to RM [16] using the large-scale open-source acoustic simulator developed in [17]. It contains 1,993 noisy conditions: 1,500 are used for training and 493 for testing. We created three noise corrupted data sets with different SNRs: Data-1 (SNR from 0 to 4), Data-2 (SNR from 0 to 8) and Data-3 (SNR from 0 to 12). Figure 4 shows the data distribution of Data-1, Data-2 and Data-3. For example, 19.9% utterances in Data-1 are adding noise (randomly chosen) at SNR=0, 20.3% of them are at SNR=1, 19.6% of them are at SNR=2, 21.2% of them are at SNR=3 and 19% of them are at SNR=4. Two noise corrupted test sets are used in this paper. In the "known-noise test set" (KNN), the noise is randomly picked from 1,500 training noise and added to the utterance at the same range of SNR used in the training set. On the contrary in the "unknown-noise test set" (UKN), the noise is selected from 493 testing noises and added to the utterance.

The Aurora 4 task, which is a medium vocabulary task speech recognition task, is based on the Wall Street Journal (WSJ0) dataset [18]. It contains 16 kHz speech data in the presence of six additive noises (car, crowd of people, restaurant, street, airport and train station) with

---

[1]Signal-to-Noise ratio (SNR) is defined as the ratio of the power of a signal to the power of noise ($\frac{P_{signal}}{P_{noise}}$)
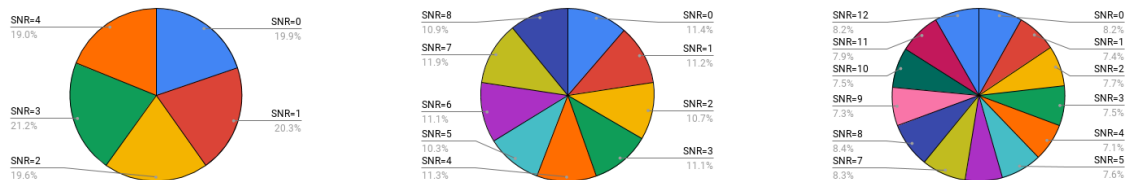
Figure 4 – The data composition of Data-1, Data-2 and Data-3 from left to right.

linear convolutional channel distortions. The multi-condition training set with 7138 utterances from 83 speakers includes a combination of clean utterance and utterance corrupted by one of six different noises at 10-20 dB SNR. 3,569 utterances are from the primary Sennheiser microphone and 3,569 utterances are from the secondary microphone. The test data is made using the same types of noise and microphones, and these can be classified into five test-conditions: clean, noisy, clean with channel distortion, noisy with channel distortion, and all of them which will be referred to as A, B, C, D and Average respectively.

3.1.2 Baseline systems

The baseline models in our experiments are DNNs, CNNs and TDNNs. DNNs take the 40-dimensional log Mel filterbank features as input and contain six hidden layers with sigmoid activation functions and one fully-connected output layer with a softmax activation. Each hidden layer has 1024 units. CNNs are composed of two convolution layers and max-pooling layers, and four affine layers with sigmoid activation function. Each affine layer contains 1024 units and is trained using 40-dimensional log Mel filterbank features with the first and the second time derivatives. Both DNNs and CNNs use the same context window of five and batch size of 256. The best TDNNs in Kaldi use in addtion iVector for speaker adapted systems. They contain five weight layers with different context specifications (subsampling). Furthermore, the TDNNs recipe applies data augmentation technique which does speed perturbation of the training data in order to emulate vocal tract length perturbations and speaking rate perturbation. All the ASR systems are built up with the Kaldi speech recognition toolkit [19]. The acoustic models except TDNNs are implemented with PDNN (A Python Deep learning toolkit) [20], Theano [21], Lasagne [22] and DenseNets source code [11].

3.1.3 Hyperparameters for DenseNets and DenseNets-AD

The architecture of DenseNets in this paper is the same as the best model in the previous work [12]: the first layer is 3×3 convolution which is followed by 4 dense blocks. Each block contains 14 3×3 convolutional layers. Each dense block except the last one is followed by a transition which consists of 1×1 convolution and 2×2 average pooling. The depth is 65, the growth rate is 12 and the compression ratio is 0.5. All the convolutional layer use kernel size $3X3$. The architecture of DenseNets-AD is mentioned in figure 3 except the shared layer is only the first convolutional layer and the following layers are training for senone (phoneme) recognition task. The gradient reversal coefficient λ is 0.5. The input features for both models is 40-dimensional log Mel filterbank features with the first and the second time derivatives.

4 Results and Discussion

Figure 5 shows the WERs of all the baseline models and DenseNets on noise corrupted RM test set at different ranges of SNR. Note that all the models are examined on the noise corrupted test
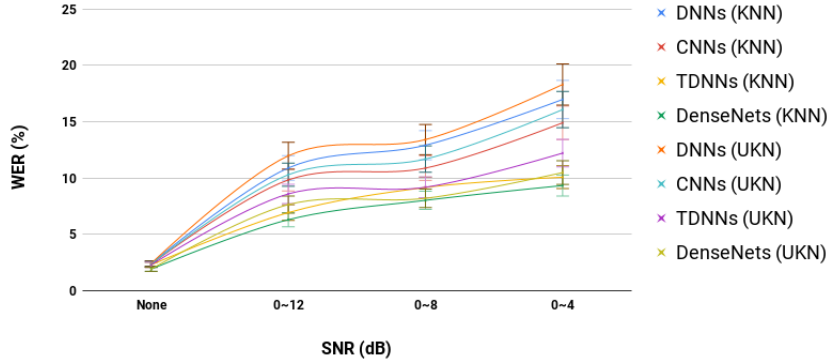
Figure 5 – The WERs of TDNN, DenseNets and DenseNets-AD on noisy test sets at different SNR ranges; DNNs(KNN) means the WER of DNNs on known-noise test set (KNN) and DNNs(UKN) means the WER of DNNs on unknown-noise test set (UKN).

SNR	System	KNN	UKN
	TDNNs	6.93	8.59
0-12	DenseNets	6.30	7.64
	DenseNets-AD	6.11	6.97
	TDNNs	9.16	9.18
0-8	DenseNets-AD	8.02	8.20
	DenseNets-AD	7.84	7.95
	TDNNs	10.07	12.21
0-4	DenseNets	9.33	10.48
	DenseNets-AD	8.64	9.68

Table 1 – The WERs of TDNN, DenseNets and DenseNets-AD on KNN and UKN test sets. Where KNN means known-noise test set and UKN means uknow-noise test set.

set at the same range of SNR as the training. For example, the model are trained on the noise corrupted training set at the SNR range from 0 to 4 and tested on the noise corrupted test set at the same SNR range. The experimental results in figure 5 show that the WERs of DNNs and CNNs increase when the SNR decreases. However, TDNNs and DenseNets are relatively stable when reducing SNR. Overall, DenseNets outperform the baseline models on all the test sets.

Table 1 and Table 2 show the comparison between TDNNs, DenseNets and densely connected convolutional networks with domain adversarial learning (DenseNets-AD) on the noise corrupted RM and Aurora4. As expected, the WERs of three models increase when the SNR decreases. However, DenseNets-AD achieves best performance on both KNN and UKN test sets at all three SNR ranges. One of the reason is that TDNNs and DenseNets recognize noise as speech when the noise becomes severe. Table 3 shows one example of TDNNs and DenseNets failing to distinguish the noise and speech while DenseNets-AD was unaffected.

5 Conclusions

This paper investigated noise robustness of DenseNets and their extension with domain adversarial learning. Our experimental results demonstrated that DenseNets are more robust against noise than other types of neural networks. Furthermore, we showed that applying domain adversarial learning improved the performance of DenseNets and improved model generalization.

System	A	B	C	D	Average
TDNNs	3.47	7.44	10.14	21.91	13.57
DenseNets	3.57	7.29	7.12	16.56	11.53
DenseNets-AD	3.58	6.58	6.76	16.42	10.21

Table 2 – WERs of TDNNs, DenseNets and DenseNets-AD on Aurora4 test set with A, B, C, D conditions. (A: clean and Sennheiser mic, B: Sennheiser mic and noise added, C: clean and 2nd mic, D: 2nd mic and noise added)

Reference	LIST FULL LOCATION DATA FOR TRACK FFF088
TDNNs	LIST FULL LOCATION DATA FOR TRACK FFF088 TO EIGHT
DenseNets	LIST FULL LOCATION DATA FOR TRACK FFF088 IN THE EIGHT
DenseNets-AD	LIST FULL LOCATION DATA FOR TRACK FFF088

Table 3 – The references and the ASR outputs from TDNNs, DenseNets and DenseNets-AD of an utterance when adding dumpster truck noise at SNR=1

References

[1] HINTON, G., L. DENG, D. YU, G. DAHL, A. MOHAMED, N. JAITLY, A. SENIOR, V. VANHOUCKE, P. NGUYEN, T. SAINATH, and B. KINGSBURY: *Deep neural networks for acoustic modeling in speech recognition*. In *IEEE Signal Process. Mag.*, vol. 29(6), pp. 82–97. 2012.

[2] SEIDE, F., G. LI, and D. YU: *Conversational speech transcription using context-dependent deep neural networks*. In *Proc. of the Interspeech*. 2011.

[3] DAHL, G. E., D. YU, L. DENG, and A. ACERO: *Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition*. *IEEE Transactions on audio, speech, and language processing*, 2012.

[4] WAIBEL, A., T. HANAZAWA, G. HINTON, K. SHIKANO, and K. J. LANG: *Phoneme recognition using time-delay neural networks*. In *Readings in speech recognition*. 1990.

[5] ABDEL-HAMID, O., A.-R. MOHAMED, H. JIANG, and G. PENN: *Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition*. In *Proc. of ICASSP*. 2012.

[6] GRAVES, A., A.-R. MOHAMED, and G. HINTON: *Speech recognition with deep recurrent neural networks*. In *Proc. of the ICASSP*. 2013.

[7] LECUN, Y. and Y. BENGIO: *Convolutional networks for images, speech, and time series*. In *The Handbook of Brain Theory and Neural Networks*. 1995.

[8] QIAN, Y. and P. C. WOODLAND: *Very deep convolutional neural networks for robust speech recognition*. In *Proc. of the IEEE SLT*. 2016.

[9] YU, D., W. XIONG, J. DROPPO, A. STOLCKE, G. YE, J. LI, and G. ZWEIG: *Deep convolutional neural networks with layer-wise context expansion and attention*. In *Proc. of Interspeech*. 2016.

[10] Xiong, W., L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke: *The microsoft 2017 conversational speech recognition system*. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5934–5938. IEEE, 2018.

[11] Huang, G., Z. Liu, L. Van Der Maaten, and K. Q. Weinberger: *Densely connected convolutional networks*. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[12] Li, C.-Y. and N. T. Vu: *Densely connected convolutional networks for speech recognition*. In *Proc. of the 13th ITG conference on Speech Communication*. 2018.

[13] Ganin, Y., E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky: *Domain-adversarial training of neural networks*. *The Journal of Machine Learning Research*, 2016.

[14] Shinohara, Y.: *Adversarial multi-task learning of deep neural networks for robust speech recognition*. In *Proc. of the Interspeech*. 2016.

[15] Denisov, P., N. T. Vu, and M. Ferras: *Unsupervised domain adaptation by adversarial learning for robust speech recognition*. In *Proc. of the 13th ITG conference on Speech Communication*. 2018.

[16] Price, P., W. M. Fisher, J. Bernstein, and D. S. Pallett: *The DARPA 1000-word resource management database for continuous speech recognition*. In *Proc. of the ICASSP*. 1988.

[17] Ferras, M., S. Madikeri, P. Motlicek, S. Dey, and H. Bourlard: *A large-scale open-source acoustic simulator for speaker recognition*. *IEEE Signal Processing Letters*, 2016.

[18] Parihar, N., J. Picone, D. Pearce, and H.-G. Hirsch: *Performance analysis of the aurora large vocabulary baseline system*. In *Proc. of the 12th European Signal Processing Conference*. 2004.

[19] Povey, D., A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz et al.: *The kaldi speech recognition toolkit*. In *Proc. of the IEEE ASRU*. 2011.

[20] Miao, Y.: *Kaldi+pdnn: Building dnn-based ASR systems with kaldi and PDNN*. In *arXiv e-prints:1401.6984*. 2014.

[21] Theano Development Team: *Theano: A Python framework for fast computation of mathematical expressions*. In *arXiv e-prints:1605.02688*. 2016.

[22] Dieleman, S. and et al.: *Lasagne: First release*. 2015.