REGULARIZING DNN ACOUSTIC MODELS WITH GAUSSIAN STOCHASTIC NEURONS

Hao Zhang, Yajie Miao, Florian Metze

Language Technologies Institute, School of Computer Science, Carnegie Mellon University
Pittsburgh, PA, USA
{haoz1,ymiao,fmetze}@cs.cmu.e du

ABSTRACT

Dropout and DropConnect can be viewed as regularization methods for deep neural network (DNN) training. In DNN acoustic modeling, the huge number of speech samples makes it expensive to sample the neuron mask (Dropout) or the weight mask (DropConnect) repetitively from a high dimensional distribution. In this paper we investigate the effect of Gaussian stochastic neurons on DNN acoustic modeling. The pre-Gaussian stochastic term can be viewed as a variant of Dropout/DropConnect and the post-Gaussian stochastic term generalizes the idea of data augmentation into hidden layers. Gaussian stochastic neurons can give improvement on large data sets where Dropout tends to be less useful. Under the low resource condition, its performance is comparable with Dropout, but with a lower time complexity during fine-tuning.

Index Terms— DNN acoustical model, Dropout, Stochastic neuron

1. INTRODUCTION

Context-depndent deep neural network hidden Markov models (CD-DNN-HMMs) have shown much better performance than traditional state-of-the-art GMM-HMM models on automatic speech recognition (ASR) tasks [1, 2]. While DNNs demonstrate great power on modeling the posterior distribution of HMM states given speech frames, their non-convex objective function and large number of parameters make them hard to optimize. Traditional methods like adding an $L2$ or $L1$ penalty on the network weights can be used for regularization. Dropout training is proposed in [3], where the output of each hidden unit is randomly set to zero with a pre-defined probability during forward propagation. Experiments show that it improves generalization performance on unseen testing data. The authors of [4] proposed to set weights to zero randomly instead of the activation of hidden unit. This idea is called DropConnect, which can be viewed as a generalization of Dropout. [5, 6] introduced low-rank matrix factorization to constrain the parameter matrix in two low dimensional subspace. Besides regularization, this can further reduce the number of parameters. This method reduces training time but does not yield improvements in word error rate (WER). Though Dropout and DropConnect offer good regularization, repetitively sampling from a high dimensional Bernoulli distribution significantly slows down the fine-tuning process. In DNN acoustic modeling of ASR, the huge number of training samples further aggravates the problem.

It is well-known that there is a connection between artificially corrupted training data and regularization. [7] showed that training with features that have been corrupted by additive Gaussian noise is equivalent to a form of $L2$ regularization when the noise is low. Stochastic neuron networks [8, 9] are built by introducing random variations into the network, either by giving the neurons stochastic transfer functions, or by giving the connection stochastic weights. Dropout can therefore be viewed as binary noise which is multiplied after the neuron non-linearity.

In previous work [10], Dropout training was accelerated by integrating a Gaussian approximation of its objective function. In this paper, we explore Gaussian stochastic neurons for regularizing DNN acoustic model training. Our idea differs from [10], in that we still do sampling but in a less time consuming way. The basic idea is to introduce a pre-activation and a post-activation Gaussian additive noise during forward propagation. Additive noise is cheaper in terms of computation than the multiplication in Dropout. Furthermore, if we draw one pair (pre and post) of Gaussian samples as noise perturbations and share it among all the units in a hidden layer, the sampling cost will be reduced greatly. We investigate the effectiveness of Gaussian stochastic neurons in terms of model regularization and training acceleration. In our experiments, Gaussian stochastic neuron DNNs yield good regularization and they reduce the training time greatly compared with Dropout. We illustrate the theoretical relation between Dropout and Gaussian stochastic neurons by referring to previous work in [10].

2. CD-DNN-HMM TRAINING AND DROPOUT

In the CD-DNN-HMM framework, we train a DNN with a softmax output layer to classify input speech frames to context-depedent tied states. In each hidden layer, the DNN computes the activation of hidden units given the outputs from the previous layer. When using the sigmoid activation

function, the outputs of the i-th layer can be computed as follows:

$$\boldsymbol{h}_i = \sigma(\boldsymbol{W}_i \boldsymbol{h}_{i-1} + \boldsymbol{b}_i), 1 \leq i \leq L \quad (1)$$

where $\boldsymbol{h}_0 = \boldsymbol{o}_t$ is the observation, $\boldsymbol{W}_i$ is the weight matrix that connects the $(i-1)$-th layer and i-th layer, $\boldsymbol{b}_i$ is the bias vector at the i-th layer, and $\sigma(x) = (1 + exp(-x))^{-1}$ is the element-wise sigmoid function. The output layer produces an estimate of the posterior probability $P(s|\boldsymbol{o}_t)$ of each tied state given the observation $\boldsymbol{o}_t$:

$$P(s|\boldsymbol{o}_t) = \frac{exp(\boldsymbol{W}_{L+1}\boldsymbol{h}_L + \boldsymbol{b}_{L+1})}{\sum_{s'} exp(\boldsymbol{W}_{L+1}\boldsymbol{h}_L + \boldsymbol{b}_{L+1})} \quad (2)$$

The DNN is trained by stochastic gradient descent (SGD) with momentum, where a cross-entropy cost function over the training set is optimized. Pre-training methods such as Stacked de-noising Auto-encoder (SdA) and Restricted Boltzmann Machines (RBM) can be used before fine-tuning to set up a good parameter initialization.

Dropout is introduced into deep neural network training as a form of regularization for fully connected network layers [3, 11]. In Dropout training, the j-th element of a layer's output is set to zero with probability p_j, or kept intact with probability $(1 - p_j)$. The hidden layer outputs after applying Dropout could be represented as:

$$\boldsymbol{h}_i = \sigma(\boldsymbol{W}_i \boldsymbol{h}_{i-1} + \boldsymbol{b}_i)\boldsymbol{D}_z, 1 \leq i \leq L \quad (3)$$

where $z = \{z_j\}_{j=1...m}$, and each $z_j \sim$ Bernoulli(p_j). $\boldsymbol{D}_z$ =diag$(z) \in R^{m \times m}$. In each training mini-batch, z_j is sampled for each neuron in the network. In addition, the final weight parameters need to be compensated for Dropout training as:

$$\overline{\boldsymbol{W}}_i = (1-p)\boldsymbol{W}_i \quad (4)$$

where p is the dropout probability. In DropConnect, the masking matrix $\boldsymbol{D}_z$ is no longer diagonal and it is multiplied with the weight matrix $\boldsymbol{W}_i$.

3. GAUSSIAN STOCHASTIC NEURON

3.1. Model description

Dropout training can be viewed as injecting binary noise into neurons by multiplication with the neuron activation. In Gaussian stochastic neurons, noise samples are drawn from a pre-defined Gaussian distribution and injected into the neurons by addition. It can be formularized as:

$$\boldsymbol{h}_i = \sigma(\boldsymbol{W}_i \boldsymbol{h}_{i-1} + \boldsymbol{b}_i + \boldsymbol{\delta_{pre}}) + \boldsymbol{\delta_{post}}, 1 \leq i \leq L \quad (5)$$

where we have the pre-activation noise term

$$\boldsymbol{\delta_{pre}} = \{\delta^j_{pre}\}_{j=1...m} \quad (6)$$

and the post-activation term

$$\boldsymbol{\delta_{post}} = \{\delta^j_{post}\}_{j=1...m} \quad (7)$$

We discuss two ways to generate samples for $\boldsymbol{\delta_{pre}}$ and $\boldsymbol{\delta_{post}}$.

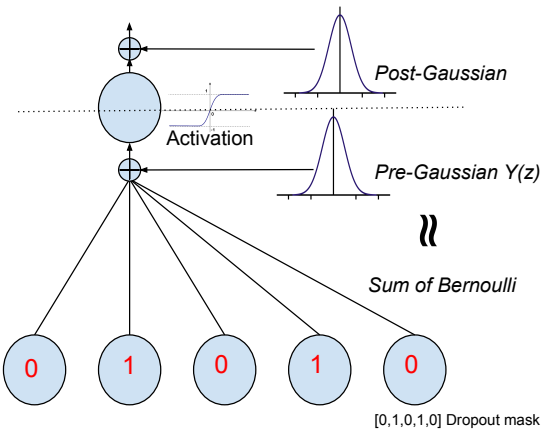

Fig. 1. Illustration of Gaussian stochastic neuron.

3.1.1. *Untied Gaussian stochastic neurons*

In untied Gaussian stochastic neurons (UGSN), all the δ^j_{pre} are independently draw from the same Gaussian distribution, and all the δ^j_{post} are drawn from another Gaussian distribution. Namely:

$$\delta^j_{pre} \sim N(\mu_{pre}, \sigma^2_{pre}), \delta^j_{post} \sim N(\mu_{post}, \sigma^2_{post}) \quad (8)$$

We do not save on the number of sampling operation compared to Dropout but the noise perturbations are added instead of multiplied onto the neuron.

3.1.2. *Tied Gaussian stochastic neurons*

In tied Gaussian stochastic neurons (TGSN), we simply tie all the elements $\delta^j_{pre}, j = 1...m$ together in each hidden layer, and similarly for δ^j_{post}. So we have:

$$\delta^1_{pre} = \delta^2_{pre} = ... = \delta^m_{pre} \sim N(\mu_{pre}, \sigma^2_{pre}) \quad (9)$$

Compared with the untied case, we only need to draw one sample for each hidden layer and share that sample among its neurons. Hence we can reduce the number of sampling by a great magnitude.

3.2. Intuition and justification of Gaussian stochastic neurons

The pre-Gaussian noise perturbation is added before the activation function. There is a close relationship between Dropout and the pre-activation Gaussian term $\boldsymbol{\delta_{pre}}$. [10] investigated how to do fast Dropout by sampling from or integrating a Gaussian approximation of Dropout's objective. Let the input to a neuron in Dropout training be written as:

$$Y(\boldsymbol{z}) = \boldsymbol{W}_i^{j,\cdot} \boldsymbol{D}_z \boldsymbol{h}_{i-1} \quad (10)$$

	TFE	VFE	WER	FTT
Baseline	0.434	0.449	19.3%	20.5h (17 epochs)
Dropout	0.448	0.450	19.5%	62h (17 epochs)
UGSN	0.424	0.453	18.7%	30h (16 epochs)
TGSN	0.423	0.458	**18.5%**	**22.5h** (17 epochs)

Table 1. SWB 110 hours setup on eval2000 test set, Training Frame Error (TFE), Validation Frame Error (VFE), Word Error Rate (WER) and Fine-Tuning Time (FTT)

	TFE	VFE	WER	FTT
Baseline	0.389	0.430	4.5%	11h (19 epochs)
Dropout	0.393	0.427	4.57%	35h (16 epochs)
UGSN	0.372	0.422	4.24%	18h (18 epochs)
TGSN	0.366	0.430	**4.04%**	**14.8h** (19 epochs)

Table 2. Result on WSJ 80 hours setup with eval92 test set

where $\boldsymbol{W}_i^{j,\cdot}$ is the j-th row of the weight matrix $\boldsymbol{W}_i$. The random variable Y_z is a weighted sum of the Bernoulli random variables $\{z_j\}_{j=1...m}$. Usually the hidden layer size would be in the hundreds, and according to central limit theorem, $Y(\boldsymbol{z})$ can be well approximated by a single Gaussian distribution (see Figure 1), namely:

$$Y(\boldsymbol{z}) \approx \eta = \mu_\eta + \sigma_\eta^2 \epsilon \tag{11}$$

where $\epsilon \sim N(0,1)$, $\mu_\eta = \sum_{l=1}^{m} p_l h_l w_l$, and $\sigma_\eta^2 = \sum_{l=1}^{m} p_l(1-p_l)(h_l w_l)^2$. In our Gaussian stochastic neurons, we use $\mu_{pre} = \sum_{l=1}^{m} h_l w_l$, which is just the value propagated from the previous layer. We keep σ_{pre} constant for simplicity of computation.

It is worth analyzing tied and untied pre-activation Gaussian terms in a bit more detail. In Dropout training, the sampled mask is applied to the $(i-1)$-th layer and hence the "random dropout" effect on the i-th layer is the same for all the neurons in that layer. Therefore, the tied pre-activation term can be viewed as an approximation to Dropout. In DropConnect training, instead of sampling mask vectors for neurons, a mask matrix is sampled for the weight matrix. In this case, the dropout effect on each neuron in the upper layer is independent, so untied pre-activation terms are a variant of DropConnect.

The post-activation noise term is a Gaussian with $\mu_{post} = 0$ and a small variance σ_{post}. In previous ASR research, researchers tried to increase data variability by transforming or distorting input features dynamically during training [12]. This is usually done by choosing a random Gaussian warping factor during Vocal Tract Length Perturbation (VTLP) in the front-end feature extraction. While VTLP introduces variation on the input layer, δ_{post} introduces distortion in each hidden layer. However, the magnitude of distortion needs to be controlled carefully, to prevent negative effects on convergence.

4. EXPERIMENTS

4.1. CD-DNN-HMM baseline and Dropout system

We first investigate Gaussian stochastic neuron DNNs on two large data sets: the Switchboard (SWB) 110 hours of training setup described in [13] and the DARPA Wall Street Journal (WSJ) 5000-word database. Evaluation is conducted on the SWB eval2000 (Hub5'00) testing set and the WSJ eval92 testing set. We build DNN acoustic models with the Kaldi+PDNN recipe [14]. The GMM-HMM systems are built with standard Kaldi recipe [15] and we obtain DNN training labels from the SAT-GMM model in the fMLLR feature space. The SAT-GMM model achieves 25% and 6.15% word error rate on SWB and WSJ, respectively. On SWB, we build a neural network with 6 hidden layers and each layer contains 1024 units. The input to this DNN is 11 neighboring frames of 40-dimensional fMLLR features. We first pre-train each hidden layer with de-noising auto-encoders and then do fine-tuning, which is to optimize the cross-entropy objective with an exponentially decaying learning rate schedule. The learning rate starts from an initial value 0.08 and remains unchanged for 8 epochs. Then the learning rate is halved in each epoch until the frame accuracy on the validation set stops improving. A momentum of 0.5 is used for fast convergence and we use a mini-batch size of 256 for SGD. On WSJ, we use a DNN with 5 hidden layer and extract same dimension of filter-bank features as input. We keep other setting identical. All our experiments are conducted on a Tesla K20m GPU card.

Next we conduct Dropout training on top of the baseline system. For Dropout DNN, fine-tuning uses the same decay schedule, but starts with a much larger learning rate. We set the initial learning rate to 1 in our experiments. We only perform Dropout on the hidden layer units, with a global dropout probability of 0.2. On the SWB and WSJ evaluation sets, we find that Dropout does not give any improvement on training frame accuracy or WER. We hypothesize that Dropout regularization becomes less effective on large training data sets. However, the fine-tuning time of Dropout is twice more than CD-DNN-HMM baseline. The baseline DNN and Dropout DNN results are listed in Table 1 for SWB and 2 for WSJ. Baseline DNN achieved a WER of 19.3% on the SWB eval2000 set and 4.5% on the WSJ eval92 set. Dropout DNN achieved slightly worse WERs of 19.5% and 4.57%.

4.2. CD-DNN-HMM with Gaussian stochastic neuron

We evaluate the performance of tied/untied Gaussian stochastic neurons respectively. Fine-tuning of Gaussian stochastic neuron DNNs follows the same decay schedule as the baseline. We find that although Dropout is relatively insensitive to a large initial learning rate, it is important to choose a good initial learning rate for stochastic neuron DNNs. In our exper-

	TFE	VFE	WER	FTT
Baseline	0.570	0.607	68.9%	55min (20 epochs)
Dropout	0.543	0.595	**66.6%**	4h16min (17 epochs)
UGSN	0.513	0.607	**66.7%**	**1h52min** (18 epochs)
TGSN	0.508	0.613	67.1%	1h28min (18 epochs)

Table 3. Result on Tagalog 10 hours setup with 2 hours development set

	TFE	VFE	WER	FTT
Baseline	0.558	0.576	50.0%	14h (23 epochs)
Dropout	0.560	0.573	50.0%	40h(19 epochs)
UGSN	0.552	0.578	49.3%	17h (17 epochs)
TGSN	0.546	0.581	**49.2%**	**15h** (18 epochs)

Table 4. Result on Tagalog 80 hours setup with 2 hours development set

iments, when the initial learning rate is below 0.4, the DNN converges to a poor local minimum and does not give any improvement on the training frame accuracy. When the initial learning rate is greater than 0.8, the training frame error blows up easily and training will crash. Hence we choose 0.6 as the initial learning rate both for the untied and tied cases. We empirically choose hyper parameter of the distributions of the noise in stochastic neuron. In the training of all stochastic neuron DNNs, we set $\sigma_{pre} = 0.15$ and $\sigma_{post} = 0.15$.

4.2.1. Untied Gaussian stochastic neuron

UGSN reduced WER by 0.6% (3% relative) on SWB and 0.26% (5.7% relative) on WSJ compared to the baseline DNN, as shown in Tables 1 and 2. UGSN did not reduce the number of sampling operations during training. Surprisingly, we found that the UGSN DNN fine-tuning is much faster than Dropout DNN and it converges to a lower training frame error with a comparable validation frame error, which means overfitting does not occur. We suspect the reason might be that multiplicative noise is more expensive than additive noise in terms of computation. It also might due to implementation issues. We need to further investigate this point in our future work.

4.2.2. Tied Gaussian stochastic neuron

On both the SWB and WSJ evaluation sets, TSGN yields a higher improvement than untied case. It reduced the WER to 18.5% (4% relative gains) and 4.04% (10.2% relative gains), respectively. Because in each layer only one Gaussian is sampled and shared among all hidden units, the number of sampling operation is greatly reduced in TSGN DNN training. We see that the training time is less than the untied case and for SWB, it is only 2 hours more than the baseline.

4.3. Experiments on BABEL DATA

We further evaluated Gaussian stochastic neuron DNNs under low resource conditions. Here we use the BABEL corpus collected in the IARPA BABEL research program. We conduct our experiment on the Tagalog corpus (IARPA-babel106-v0.2f) with 10 hours (Limited language pack) training data and 80 hours training data (Full language pack) respectively. We follow a similar procedure of building a DNN baseline system as in the WSJ experiment. The result of 10 hours training data setup is shown in Table 3. We see that UGSN yields better training frame error and the same WER compared with Dropout DNN. However, it only takes less than half of the time used in the Dropout setup. TGSN achieved a WER of 67.1%, 0.4% worse than the untied case, but there was still a 1.8% absolute improvement over the baseline. As we expected, the training time can be further reduced compared to the untied case. Table 4 shows the result with 80 hours training data. We observed similar phenomena as in SWB and WSJ. Although the baseline WER is high, Dropout still does not improve system performance on 80 hours training data setup. With Gaussian stochastic neuron, we can achieve a lower WER of 49.2% with much less training time.

5. CONCLUSION AND FUTURE WORK

In this paper, we investigated the use of Gaussian stochastic neurons to regularize deep neural network training in acoustic modeling. TGSN can be viewed as an approximation of Dropout and UGSN performs similarly to DropConnect. We observe on large data sets that Dropout DNNs tend to be less useful, but Gaussian stochastic neuron DNNs can still improve system performance. In the situation of limited training data, UGSN can yield the same performance as Dropout DNNs while saving a lot time during the fine-tuning.

In future work, we plan to compare Gaussian stochastic neuron DNNs with DropConnect and also DNN trained by directly approximating the Dropout objective function as proposed in [10]. It is also interesting to apply Gaussian stochastic neuron regularization on Max-out networks [16], where Dropout is used along with Max-out activation to prevent overfitting. Finally, we need to further investigate the reason why UGSN DNNs can be trained faster than Dropout DNNs. Namely, we need to investigate if the difference is inherent to the training procedure or an implementation issue.

6. ACKNOWLEDGEMENTS

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI1053575. Specifically, it used the Blacklight system at the Pittsburgh Supercomputing Center (PSC).

7. REFERENCES

[1] Frank Seide, Gang Li, Xie Chen, and Dong Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 24–29.

[2] George E Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 30–42, 2012.

[3] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[4] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 1058–1066.

[5] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6655–6659.

[6] Jian Xue, Jinyu Li, and Yifan Gong, "Restructuring of deep neural network acoustic models with singular value decomposition.," in *INTERSPEECH*, 2013, pp. 2365–2369.

[7] Chris M Bishop, "Training with noise is equivalent to tikhonov regularization," *Neural computation*, vol. 7, no. 1, pp. 108–116, 1995.

[8] Yoshua Bengio, Nicholas Léonard, and Aaron Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[9] Max Garzón and Luz Gloria Torres, "Stochastic neural networks," *Revista Colombiana de Estadistica*, 1991.

[10] Sida Wang and Christopher Manning, "Fast dropout training," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 118–126.

[11] Yajie Miao and Florian Metze, "Improving low-resource cd-dnn-hmm using dropout and multilingual dnn training.," in *INTERSPEECH*, 2013, pp. 2237–2241.

[12] Navdeep Jaitly and Geoffrey E Hinton, "Vocal tract length perturbation (vtlp) improves speech recognition," in *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[13] Shakti P Rath, Daniel Povey, and Karel Veselỳ, "Improved feature processing for deep neural networks.," in *Proc. Interspeech*, 2013.

[14] Y. Miao, "Kaldi+pdnn: Building dnn-based asr systems with kaldi and pdnn.," in *arXiv:1401.6984*, 2014.

[15] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *Proc. ASRU*, 2011, pp. 1–4.

[16] Yajie Miao, Florian Metze, and Shourabh Rawat, "Deep maxout networks for low-resource speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 398–403.