CSLT TECHNICAL REPORT-20170003 [2017 年 08 月 30 日]

Long-term DropConnect in Speech Recognition (in Chinese)

Yanqing Wang[1,2*], Zhiyuan Tang[1,3] and Dong Wang[1,2]

*Correspondence:
wangyanqingchn@163.com
[1]Center for Speech and Language Technology, Research Institute of Information Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China
Full list of author information is available at the end of the article

Abstract

本报告是基于之前有关 Connection Sparseness 的工作的进一步研究。笔者在之前工作的基础上，提出一些探索性的问题，并给这些问题以初步的答案，同时又发现对未经预训练的神经网络进行随机的权重裁剪后训练也可以得到不错的效果，便以此为出发点做了一些有关 Long-term DropConnect 新的设想和尝试。

Keywords: sparse attribute; sparse structure; value; randomly prune; connection-pruning; speech recognition; sparse deep neural network

1 Introduction

本报告是基于之前有关 Connection Sparseness 的工作 [1]的进一步研究。笔者在之前工作的基础上，提出一些探索性的问题，并通过实验给这些问题以初步的答案。之后，更进一步做了一些设想和尝试。

因为实验分支较多，为了方便读者把握本报告的脉络，我在此处直接将本报告囊括的工作和结论进行最精要的概述：

（1）Connection Sparseness 的工作告诉我们，即使我们的网络只保留3%的权值（又：连接，connection），也可以达到相对不错的效果。但是我们发现，如果用有同样多权值的全连接网络进行语音识别的任务，则根本无法将网络训练至收敛。这说明了网络稀疏性的重要性。

（2）我们在 pre-train 一个网络并且对其进行 connection-pruning 之后，将它的稀疏性网络结构直接应用到一个随机初始化的网络，并对这个新的网络进行 retrain，发现这个新的网络仍然能达到不错的效果。说明对于一个进行了 pre-train 和 connection-pruning 之后的网络，其网络的稀疏性结构（structure）比连接的具体值（value）重要的多。

（3）我们对一个随机初始化的网络进行随机的 connection-pruning（即：不借鉴任何pre-train的网络），也仍然可以达到不错的效果，说明网络的稀疏属性（sparse attribute）比其具体的稀疏性结构（structure）更重要。综合（2）和（3），我们认为重要性可以进行如下排序：$sparse\ attribute > structure > value$。

（4）基于之前的三点发现，我们想尝试着与业内已有的科研工作结合起来，利用稀疏性深度神经网络对语音识别的效果进行提高。我们想借鉴 drop connect [2]的思路，利用（3）的发现，实现 long-time dropout（即对几个随机初始化、随机进行 connection-pruning 的网络进行训练后，将它们集成（make an ensemble））。但是，以上尝试并未得到很好的效果。

（5）之后，仿照（4）的方法，我用几个完全、互斥的网络做了 ensemble，得到了不错的效果，但是从原理上想，似乎仍然有不够合理的地方。

当然，以上只是对所有工作精要的概述，读者可按照下面介绍的全文脉络，对这些工作进行更加深入的了解。

在第2部分中，我将基于有关 Connection Sparseness 的工作提出一些探索性的问题，作为接下来的探索工作的出发点。在第3p部分中，我将对于业内相关的探索进行概述。在第4到第6 部分中，结合具体实验对这些问题给出了初步的答案和结论。在第7部分中，基于这些结论，描述了进一步的设想和简单的尝试，并分别在第8和第9给了初步的结论和对下一步工作的展望和想法。最后，为了帮助读者复现实验和进行下一步的探索，在第10 部分中，从具体实践的角度，说明了这些方案的实现方法。

2 Motivation

在之前关于 Connection Sparseness 的工作 [1]中，我们首先预训练（pre-train）一个深度神经网络（Deep Neural Network, DNN），然后根据其权值的大小进行权值裁剪（connection-pruning），在此之后我们重新训练新得到的稀疏深度神经网络（Sparse Deep Neural Network），进而完成语音识别（Speech Recognition）的任务。在此基础上，我们考虑：

（1）“稀疏”的重要性：在有关 Connection Sparseness 的工作 [1]中，我们看到，我们的网络即使在 connection-pruning 之后只保留了很少（3%）的权值，仍然可以得到一个可以接受的结果（虽然不及 baseline）。那么，用同样多（原网络权值数目的3%）的权值数目的全连接网络进行语音识别，能否达到相似的效果？如果不可以，那么我们可以说网络的稀疏性在这之中扮演了重要的作用。

（2）Value V.S. Structure：在 pre-train 后的 connection-pruning 过程中，我们保留了 pre-train 之后的网络中（我们认为）较为重要的权值，同时也构造出了新的稀疏网络结构。在进行了 connection-pruning 之后的新网络中，到底是权值（初始值）比较重要，还是网络结构比较重要？

（3）pre-train 的必要性：如果是网络结构比初始值更加重要，那么一个比较好的稀疏性网络结构，一定要依赖于事先 pre-train 的网络吗？

3 Related Work

稀疏性深度学习近些年来备受关注 [3]。有关稀疏神经网络的研究目前主要集中在网络裁剪（Pruning）和矩阵分解 [4,5] 两个方向。而网络裁剪可以分为神经元裁剪（Node Sparseness） [6]和权值裁剪（Connection Sparseness） [7]两个类别。对于 Connection Sparseness 的工作，除却最简单的设定阈值进行裁剪的方案 [7]，也有人在采用 Optimal Brain Damage （OBD）的方案 [8]，试图能更加准确的选择出需要裁剪的权重。

为了缓解网络的过拟合问题，Dropout 方案很早就被提出 [9]。之后，又有了采用了相似思路的 DropConnect 方案 [2]。事实上，Dropout 和 DropConnect 方案也与稀疏性深度神经网络（Sparse Deep Neural Network）有着内在的联系，因为这两种方案本质上是对进行了随机裁剪的神经网络进行了集成（make an ensemble）[10]。

4 探索1：“稀疏”的重要性

1. 实验想法与设计

既然在 Connection Sparseness 的相关工作中，我们只保留了一个网络3%的连接权值，仍然可以达到不错的效果（WER：11.26% / baseline 8.51%），那么如果我们用同样多（原网络权值数目的3%）的权值数目的全连接网络进行语音识别，能否达到相似的效果？

因原网络的结构为：输入360维数据，经过6个2000维的 Sigmoid 激活函数，最终输出3430维数据。所以，97% 稀疏度的网络的总权值数目为：(360*2000+2000*2000*5+2000*3430)*3% 个。为了只将改变网络的稀疏性（将稀疏性网络改为全连接网络），保证网络结构等其他参数不变，所以用下式计算新的（全连接）网络的激活函数的维度：

设新的网络的激活函数维度为 n 个，列方程为：

$$(360*2000+2000*2000*5+2000*3430)*3\% = 360*n+n*n*5+n*3430$$

解得：x=176.99（取177）

构造上述网络（方法参见第10部分第1点），而后进行网络的训练即可。

2. 实验结果及结论

新的全连接网络无法在训练过程中收敛，最终的 WER 为100%。这说明网络的稀疏性一定程度上保证了网络可以在训练过程中收敛，表现了稀疏性的重要性。

5 探索2：Value V.S. Structure

1. 实验想法与设计

在 pre-train 后的 connection-pruning 过程中，我们保留了 pre-train 之后的网络中（我们认为）较为重要的权值，同时也构造出了新的稀疏网络结构。因为构造出的新的网络继承了这两方面的属性，我们需要思考，在这个新的网络中，到底是继承的权值（作为其初始值）比较重要，还是继承的网络结构比较重要？

我们采用控制变量法，保留新网络的稀疏性网络结构，但对其权值进行随机初始化，而后对它进行 retrain。如果这样的网络仍然有之前网络的性能，那么就可以证明网络的稀疏性结构的重要性，以及其所继承的权值的无关紧要。

2. 实验结果及结论

我们将进行过 pre-train 和 connection-pruning 的网络的结构直接应用到一个随机初始化的网络（等价于：对 connection-pruning 之后的网络保留结构，同时对权值进行随机初始化）（方法参见第10部分第2点），对这个网络进行 retrain，查看效果。

在笔者的实验中，只保留网络结构但随机初始化权值之后的网络，与之前的网络具有相同的性能（均为11.26%），这直接说明网络结构是重要的，而权值（初始值）对 retrain 后的网络却没有多大影响。

6 探索3：pre-train的必要性

1. 实验想法与设计

既然网络结构比初始值更加重要，那么一个比较好的稀疏性网络结构，一定要依赖于事先pre-train的网络吗？如果我们只是对一个随机初始化后的网络进行随机

的 connection-pruning 和 retrain ，会有什么样的效果？

2. 实验结果及结论

笔者对一个随机初始化的网络进行了随机的 connection-pruning（稀疏度为97%）（方法参见第10部分第3点）。进行了四组实验，四组实验的最终 WER 分别为12.50%，12.09%，12.61%，12.50%，笔者认为，这与探索2中的性能（WER 为11.26%）并没有太大的差距，说明网络的稀疏属性（sparse attribute）应该引起我们高度的重视，这一网络本身的属性比其结构似乎更加重要。

综合探索2、探索3的结论，笔者认为，对于一个稀疏深度神经网络来说，按照对其最终（充分训练之后）性能影响的程度，可以对它的几个属性进行排序：$sparse\ attribute > structure > value$。

7 设想和尝试

一、将多个经过 random connection-pruning 并训练后的网络进行集成

1. 实验想法与设计

尽管业内之前往往基于一个经过 pre-training 的网络进行 connection-pruning（如 weight-cutting [7] 和 OBD [8]）进行 retrain，但我们发现一个网络即时没有经过 pre-training 和被裁剪的权值的选择，仍然可以在随机的 connection-pruning（我们称之为random connection-pruning）之后得到一个不错的效果。将这与有关 dropConnect 的工作 [2]相结合，对我们的思考可以有一定的启发。

DropConnect 的思路本质上是对多个随机 prune 掉一些权值的网络进行集成（make an ensemble），我们是否也可以借鉴这样的思路，对几个经过 random connection-pruning 单独进行 retrain，再做一个它们的 ensemble，看看是否能达成不错的效果（方法参见第10部分第4 点）。

2. 实验结果及结论

笔者做了几组实验（稀疏度不同），使用的 ensemble 方法是最简单的 major vote 方案（又：unweighted average）。但是从结果来看，在使用了 ensemble 的方法之后，无论是 WER 还是对语音噪音的鲁棒性，都没有明显的提升。实验结果如表1到3 所示。

二、将多个互斥的网络进行集成

1. 实验想法与设计

表 1 做ensemble前后的 WER（稀疏度：75%）

单一网络的稀疏度（%）	单一网络（%）	Ensemble （%）
75	9.15	9.17
75	9.10	
75	9.26	
75	9.35	

表 2 做ensemble前后的 WER（稀疏度：80%）

单一网络的稀疏度（%）	单一网络（%）	Ensemble （%）
80	9.44	9.19
80	9.28	
80	9.44	
80	9.28	

笔者尝试生成了四个完全、互斥的网络（方法参见第10部分第5点）。“完全、互斥”的意思是：这四个网络的稀疏度均为75%，但如果将四个网络各层的linear_params（又：转移矩阵，连接矩阵，权值）相加，则会得到一个全连接的网络。也就是说，把全连接网络中的 connection 分成4 等份，则会得到四个“完全、互斥”的网络。

我们从直觉上认为，如果两个模型因为某些差异（如：神经网络的网络结构）而能够学习到不同的特征，那么用它们做 ensemble 可能会得到优于单一网络的结果。所以，笔者想探究上述这样的互斥的网络是否会因稀疏性结构（structure）上的互补性，而在 making an ensemble 后会有比较好的效果。

需要注意的是，正因为之前我们发现 random connection-pruning 的效果不差，这样的实验才有其内在的合理性。反之，如果我们认为对网络的性能有很大的影响的是网络具体的稀疏性结构（structure），而不是网络的稀疏属性（sparse attribute），那么这样的实验就是不合理的——四个完全、互斥的网络，势必会有相差甚远的稀疏性结构（structure），也就必然导致其中某（几）个单一网络的性能的不如人意。

2. 实验结果及结论

如表4所示。

表 3 做ensemble前后的 WER（稀疏度：97%）（注：本组实验增加了对20db 噪音的鲁棒性的探究）

单一网络的稀疏度（%）	单一网络（%）	加入噪音后的单一网络（%）	Ensemble（%）	加入噪音后的Ensemble（%）
97	12.56	48.44	11.82	46.98
97	12.29	46.24		
97	12.05	48.86		
97	12.45	49.36		

表 4 对完全互斥的稀疏度为75%四个网络做ensemble前后的 WER

单一网络的稀疏度（%）	单一网络（%）	Ensemble （%）
75	9.10	9.04
75	9.31	
75	9.15	
75	9.10	

这样的尝试得到了还不错的效果。首先，emsemble 的 WER 确实优于每一个单一网络。其次，与 baseline 的 WER（8.81%）相差不多。（注：这里的 WER of Baseline 与之前提到的8.51%不同，是因为笔者为了加快实验速度，对测试集进行了抽样。）但是，这样的实验结果是否只是一次偶然，还有待于进一步验证。

但是，这样的实验设计其实仍然有不合理的地方。我们知道，这样的形式上的“互斥”事实上并不能很好的反应网络结构（structure）的差异。正如我们所知道的，如果一个转移矩阵做了行或列的交换，这个网络仍是原先的网络，变化的只是形式上的数学表达。所以，转移矩阵中 0 的位置互补，并不能说明这几个网络的结构是有很大差异的。在后续的任务中，我们应当找到一个利用转移矩阵衡量两个网络结构差异的方法，进而利用结构差异大的几个网络做 ensemble。

8 Conclusion

一个稀疏性神经网络有几个基本的属性可能会影响它的性能：稀疏属性（sparse attribute）（最基本、最本质的属性）、稀疏性结构（sparse structure）、初始值（value）。笔者认为重要性的次序依次为：$sparse\ attribute >> sparse\ structure > value$。

借鉴 DropConnect [2]的思路，我们提出 Long-term DropConnect 的方案：把几个随机初始化、随机进行 connection-pruning 的网络进行集成（make an ensemble），但是效果不是很理想。然而，当我们把多个完全互斥的网络进行集成时，似乎可以得到不错的效果。我们可以在验证了这一发现的基础上，想出对结构上的“互斥”更为合理的数学定义，进而改进这一 ensemble方案，提升深度神经网络的性能。

9 Future Work

1. 继续验证：对几个完全、互斥的网络进行集成（make an ensemble）的方案是否会带来神经网络性能必然的提升。

2. Emsemble 的方案有很多种 [11]，笔者现在采用的是最基本一种：major vote / unweighted average，读者可以进行其他ensemble 方案的实现和比较。

3. 因为在第7部分中提到的理论上的缺陷，我们可以想出对结构上的“互斥”更为合理的数学定义，进而改进这一 ensemble 方案，提升深度神经网络的性能。

10 实现方案

1. 如何配置 Deep Neural Network（DNN）或 Time Delay Neural Network（TDNN）的结构

利用 Kaldi ASR tookit 中 nnet3 提供的 run_tdnn.sh 脚本，我们可以轻松实现 DNN 和 TDNN 网络结构的配置。run_tdnn.sh 脚本调用了 train_tdnn.sh 脚本，修改传入这一脚本的参数，即可实现网络结构的配置。

（1）时延的配置：修改传入 train_tdnn.sh 的 splice-indexes 参数，即可实现对 TDNN 的层间时延的配置。如果除第一个参数外的其他参数均为0，则说明 TDNN 已经没有时延，变为普通的 DNN。

（2）激活函数的种类和维度的配置：将 sigmoid-dim 参数传入 train_tdnn.sh 脚本，即可将激活函数（从默认的 Pnorm）修改为 Sigmoid，并且设置该激活函数的维度。其他几种激活函数（Tanh，ReLU）方法类似。

2. 如何在保持稀疏性神经网络的稀疏性结构不变的前提下，随机初始化 connections

我们的任务等价于：参考这个已有的网络初始化一个新的网络。可以利用笔者基于Kaldi环境开发的命令：nnet3-init-sparse。说明如下：

用法：nnet3-init-sparse –binary=false [config-in] [raw-nnet-in] [raw-nnet-out]

我们需要传入这个新的网络的 config 文件（设定了网络结构，参考 [12]。必须与传入的 raw-nnet-in 的结构完全一致），传入原有的稀疏性网络，便会仿照 raw-nnet-in 的稀疏性结构，输出随机初始化的网络 raw-nnet-out 。

3. random connection-pruning 的实现方法

按照下列步骤进行即可（以稀疏度97%为例）。

（1）随机初始化一个网络。

（2）利用 [1]中提供的 connection-pruning方法（选用 Pct-Prune 方案），裁剪掉97%的权值（默认是绝对值较小的数值）。

（3）利用上文进行过说明的命令 nnet3-init-sparse，用（2）中裁剪后的网络作原始网络，生成一个新的网络，这个新的网络就等价于随机初始化后，又随机裁剪了97% connections 的网络。

用这样的方案，既保证了网络结构的随机性（由于第1步初始化时数值的随机性），又保证了网络初始值的随机性（由于第3步利用 nnet3-init-sparse 进行初始化时数值的随机性）。

4. 如何make an ensemble

笔者采用的方案是直接修改 final.mdl 文件。

（1）把几个子网络（subnet）的 final.mdl 中的 component 和 component-node 进行重命名。

（2）按照相应的次序把这些子网络的 mdl 文件 merge 到同一个新文件中。

（3）在 nnet-simple-component.cc 和 nnet-simple-component.h 中将 Kaldi 的 NoOpComponent 引入求和机制，并添加实现除法的 component（AvgComponent）（这两个 component 用于取几个子网络输出的均值）。编译。

（4）将（修改后的）NoOpComponent 和新添加的 AvgComponent 也添加至新的 mdl 文件中。

注：读者可参照笔者的 component 文件[1] 和实现 major vote 的文件夹[2] 进行 component 的添加和 ensembling 的实现。

5. 如何构造四个完全互斥的网络

按照下列步骤进行即可。

（1）利用“3. random connection-pruning 的实现方法”中提供的方法生成一个随机裁剪了75% connections 的网络，作为第一个子网络（nnet_1）。

（2）利用笔者开发的 nnet3-init-sparse-opposite（Kaldi-based Command）生成一个稀疏性结构与 nnet_1 完全相反的网络 nnet_tmp_1 （注：nnet3-init-sparse-opposite 与前文介绍的 nnet3-init-sparse 用法相同，只是生成的网络结构相反）。将此网络的50% 的值砍掉，得到一个新的稀疏度75%的网络 nnet_tmp_2，利用 nnet3-init-sparse（Kaldi-based Command）将此网络的结构应用到一个随机初始化的网络，即第二个子网络 nnet_2。

（3）利用笔者开发的 nnet3-add（Kaldi-based Command）将 nnet_1 与 nnet_2 相加，得到 nnet_tmp_3（稀疏度50%）。再仿照步骤（2），利用 nnet3-init-sparse-opposite（Kaldi-based Command）生成与 nnet_tmp_3 结构相反的网络 nnet_tmp_4。将 nnet_tmp_4 的25%的权值砍掉，得到 nnet_tmp_5。而后利用nnet3-init-sparse（Kaldi-based Command）将此网络的结构应用到一个随机初始化的网络，即第三个子网络 nnet_3。

---

[1]https://github.com/wyq730/CSLT-Sparse-DNN-Toolkit/tree/master/Supplement_for_Kaldi_Source_Code/src/nnet3

[2]https://github.com/wyq730/CSLT-Sparse-DNN-Toolkit/tree/master/CSLT_DNN_Ensembling_Toolkit

（4）将 nnet_1，nnet_2，nnet_3 相加，得到 nnet_tmp_6。利用 nnet3-init-sparse-opposite（Kaldi-based Command），生成与 nnet_tmp_6 结构相反的网络，便得到第四个子网络 nnet_4。

此过程虽然比较繁琐，但是可以在保证这几个子网络的结构完全互斥之外，完全保证它们结构的随机性和初始值的随机性。当然，为了利用已有的命令快速实现功能，此方案必然不是最优的方案。读者可自行优化。

注：读者可参考笔者生成四个完全互斥网络的方案[3] 进行实现。可以重点参考其中 gnrt.sh 脚本的思路（该脚本展示了上述过程）进行复现。

[3]https://github.com/wyq730/CSLT-Sparse-DNN-Toolkit/tree/master/CSLT_Exclusive_DNN_Toolkit

Acknowledgement

This research was supported by the National Science Foundation of China (NSFC) under the project No. 61371136.

Author details

[1]Center for Speech and Language Technology, Research Institute of Information Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China. [2]Center for Speech and Language Technologies, Division of Technical Innovation and Development, Tsinghua National Laboratory for Information Science and Technology, ROOM 1-303, BLDG FIT, 100084 Beijing, China. [3]Chengdu Institute of Computer Applications, Chinese Academy of Sciences, 610041 Chengdu, China.

References

1. Yanqing Wang, Zhiyuan Tang, and Dong Wang, "Connection sparseness in speech recognition based on kaldi asr toolkit," Tech. Rep., CSLT, RIIT, Tsinghua University, 2017.
2. L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International Conference on Machine Learning*, 2013, pp. 1058–1066.
3. Dong Wang, Qiang Zhou, and Amir Hussain, *Deep and Sparse Learning in Speech and Language Processing: An Overview*, Springer International Publishing, 2016.
4. 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 *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6655–6659.
5. J Xue, J Li, and Y Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," pp. 2365–2369, 01 2013.
6. Tianxing He, Yuchen Fan, Yanmin Qian, Tian Tan, and Kai Yu, "Reshaping deep neural network for fast decoding by node-pruning," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 245–249.
7. Anders Krogh and John A. Hertz, "A simple weight decay can improve generalization," in *International Conference on Neural Information Processing Systems*, 1991, pp. 950–957.
8. C Liu, Z Zhang, and D Wang, "Pruning deep neural networks by optimal brain damage," pp. 1092–1095, 01 2014.
9. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
10. Zhi Hua Zhou, Jianxin Wu, and Wei Tang, *Ensembling neural networks: many could be better than all*, Elsevier Science Publishers Ltd., 2002.
11. Cheng Ju, Aurelien Bibaut, and Mark J Van, der Laan, "The relative performance of ensemble methods with deep convolutional neural networks for image classification," 2017.
12. Zhiyuan Tang and Dong Wang, "How to config kaldi nnet3 (in chinese)," Tech. Rep., CSLT, RIIT, Tsinghua University, 2016.