CSLT TECHNICAL REPORT-20170002 [2017 年 07 月 01 日]

Connection Sparseness in Speech Recognition Based on Kaldi ASR Toolkit (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

本组实验作为“语音识别中的稀疏性深度学习”项目（国家自然科学基金面上项目）的一部分，主要进行对DNN中全连接层的裁剪（Prune）操作，从而对DNN 全连接层的权值进行探究。本实验采用两类Prune方法（Value Prune和Pct Prune），选用多种激活函数（Pnorm，Sigmoid，Tanh，Rectifier），对Prune 后的模型进行两种性能（重训练前和重训练后）的考察，最终得到了一些初步结论。

Keywords: speech recognition; deep neural network; sparse neural network; connection sparseness; prune; retrain

1 介绍

本报告将首先于第第3部分介绍实验中所使用的数据集并进行必要的说明。第4部分主要介绍整个实验的脉络和流程，并说明两类（共五种）Prune 方案的具体机理。第5部分详细的说明了整个实验的具体操作，读者可依照此部分（并利用提供的脚本）复现整套实验，并进行相关的改进。第6部分介绍了提供的代码文件/文件夹[1]的接口和使用方法，这些文件能够方便读者进行实验的复现。第7部分对所做的实验进行了描述，并在第8部分给出了实验的具体结果。据此，在第9部分对实验结果进行了制图、分析，并给出了相应的结论。第10部分简要说明了实验未完成的部分和下一步的工作。

2 Related Work

稀疏性深度学习近些年来备受关注[1]。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[2, 3] 两个方向。而网络裁剪可以分为神经元裁剪（Node Sparseness）[4]和权值裁剪（Connection Sparseness）[5, 6]两个类别。

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

3 实验数据集

本部分介绍用于本次实验的数据（集），并进行必要的说明。

本实验采用kaldi中的wsj数据集。

另外，为了方便实验，将dev93和eval92两个数据集按dev93->eval92 的顺序进行合并，取名为dev93_eval92数据集。因此，在“kaldi/egs/wsj/s5/data”文件夹下，生成Fbank 特征后，按照数据格式准备好训练数据集train_si284_hires和测试数据集test_dev93_eval92_hires，两个文件的目录为：

```
/work7/wangyanqing/kaldi/egs/wsj/s5/data/train_si284_hires
/work7/wangyanqing/kaldi/egs/wsj/s5/data/test_dev93_eval92_hires
```

4 实验原理

本部分从宏观的角度介绍实验的基本任务、流程和原理。

一、实验流程图

本实验主要探究对神经网络中各连接层的linear_params_ （中较小的权值）进行prune操作后神经网络的性能变化。主要分为两大部分，在第1部分中，对神经网络进行prune 后直接进行再解码；在第2部分中，对神经网络进行prune 后先进行retrain （又：refine，重训练），再进行解码。下面分别进行介绍。

1. pruning task without retraining

实验流程图如图1所示。

说明：

（1）首先训练初始模型（初始模型的编号x=1,2,3...）。

（2）在某初始模型的基础上依照不同策略进行prune操作（不同的prune策略的编号n=1,2,3...），得到新的模型文件。

（3）利用新的模型文件进行解码，得到Result（wer）。

2. pruning task with retraining

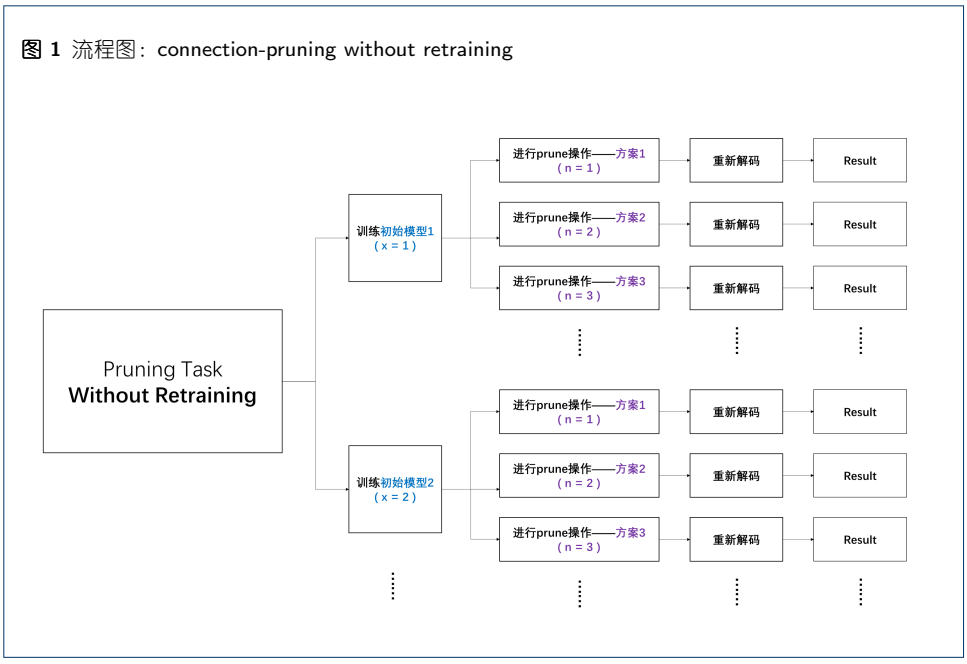
实验流程图如图2所示。

说明：

（1）首先训练初始模型（初始模型的编号x=1,2,3...）。

（2）在某初始模型的基础上依照不同策略进行prune操作（不同的prune策略的编号n=1,2,3...），得到新的模型文件。

（3）重训练retrain。



（4）利用新的模型文件进行解码，得到Result（wer）。

二、prune方案

在此实验中，设置了多种不同的prune方案。上述流程图中，相同的x下，不同的n即对应对同一初始模型的不同的prune方案。

1. Value Prune（指定值方案）

1.1 Abs Value Prune （绝对值裁剪）

设置一绝对值阈值，将linear_params_中所有绝对值小于该值的权重设为0。

1.2 Positive Value Prune（正值裁剪）

设置一正值阈值，将linear_params_中所有绝对值小于该值的**正值**权重设为0。

1.3 Negative Value Prune （负值裁剪）

设置一负值阈值（用绝对值设置），将linear_params_中所有绝对值小于该值的**负值**权重设为0。

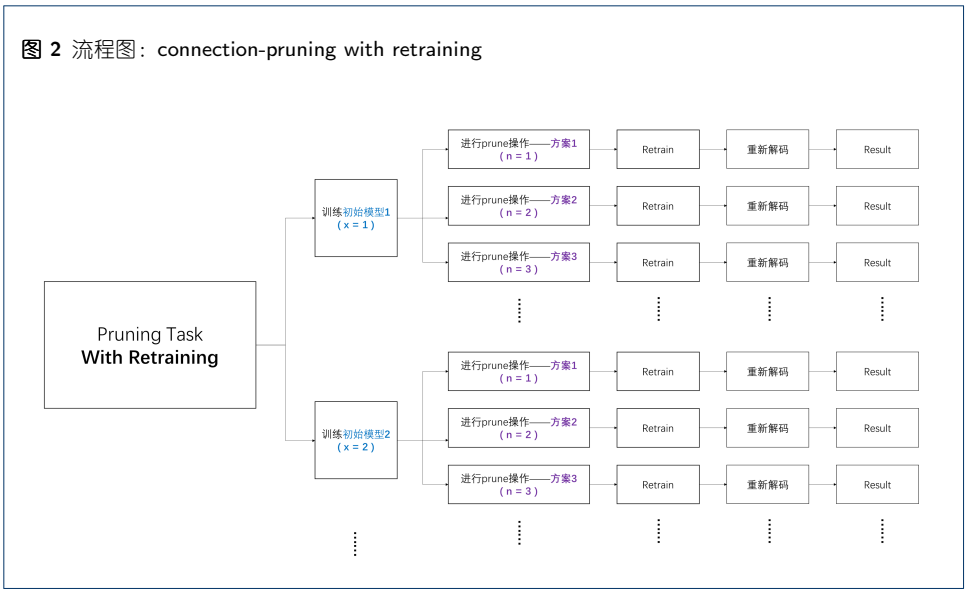
2. Pct Prune （指定百分比方案）

2.1 All-layer Pct Prune（多层百分比裁剪）

设置一百分比（a%），对**所有层**的linear_params_进行如下操作：

将所有正值中，最小的前a%个权重设置为0；

将所有负值中，绝对值最小的前a%个权重设置为0。



2.2 One-layer Pct Prune（单层百分比裁剪）

设置一百分比（a%），对**某一指定层**的linear_params_进行2.1中所述的操作。

5 实验操作与复现

本部分主要介绍实验的基本流程。另外，按照本部分的步骤，读者可以利用所给的代码文件复现各组实验。所使用到的各脚本的接口和功能将于下一部分介绍，在本部分中不着重介绍。

一、Pruning Task Without Retraining

说明：用x作为标识不同组实验的符号，每组实验的区别在于初始模型的不同（如：不同的激活函数、中间层维数等），分别对应如下逻辑关系图的一个分支：

1. 完成wsj的baseline和数据准备工作

（1）利用“kaldi/egs/wsj/s5/”下的run.sh进行完default执行的部分。

（2）对“data/test_eval92_hires”和“data/test_dev93_hires”下的数据进行Fbank特征的提取。可利用脚本：

```
/work7/wangyanqing/kaldi/egs/wsj/s5/run_feat.sh
```

（3）将上述两个文件夹内的对应文件合并（按照dev93，eval92的顺序），放置到一个新的文件夹data/test_dev93_eval92_hires/ 下。

（4）建立文件夹“data/train_si284_hires”，利用（2）中生成的特征，在该文件夹内按照格式准备好训练tdnn（nnet3）需要的训练数据。

2. 训练初始模型

使用nnet3的run_tdnn.sh脚本（位置：/work7/wangyanqing/kaldi/egs/wsj/s5/run_tdnn.sh）训练tdnn模型。注意设置各参数：

参数	含义
stage	若设置为8，则脚本实现训练&解码；若设置为9，脚本只实现解码。故在此应设置为8
dir	应设置为nnet_tdnn_a_x（x可以是1,2,3...具体含义参考前文）

3. 进行pruning task的准备工作

为了使用现有的脚本将后续的工作简化，我们在这里需要做一些准备工作。

（1）完成tdnn的训练和解码任务后，在“kaldi/egs/wsj/s5/exp/nnet3/”新生成了“nnet_tdnn_a_template_x”文件夹。将文件夹“nnet_tdnn_a_template”内的所有内容添加（复制）到新生成的“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_template_x”文件夹下。

（2）在“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_x/”下，将解码文件夹“decode_tgpr...”移动到decode_baseline文件夹下。

（3）在“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_x/prune”下，将上一层目录的final.mdl文件移动到baseline下，并重命名为final_baseline.mdl。

（4）在“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_template_x/prune/baseline”下，将final_baseline.mdl复制为非二进制的形式（final_v_baseline.mdl）。

（5）将下列文件中的x设置为正确的值：

```
nnet_tdnn_a_x/combine_best_wer.sh
nnet_tdnn_a_x/prune/run_several.sh
nnet_tdnn_a_x/prune/run_several_pct.sh
nnet_tdnn_a_x/prune/sparse_rate/combine_sparse_rate.sh
```

设置的方法是：用vim打开各个文件后，执行下述命令（假设1为x的实际值）：

```
:1,$s/nnet_tdnn_a_x/nnet_tdnn_a_1/g
```

（6）统计在“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_template_x/final.mdl”文件中各层的linear_params_（转移矩阵）所占的行数，并按照统计结果修改下列

文件中对应的各行号：

```
nnet_tdnn_a_x/prune/prune_template/prune.awk
nnet_tdnn_a_x/prune/prune_template/sparse_rate_layer.sh
nnet_tdnn_a_x/prune/prune_template/sparse_rate_total.awk
nnet_tdnn_a_x/prune/prune_template_pct/prune.awk
nnet_tdnn_a_x/prune/prune_template_pct/sparse_rate_layer.sh
nnet_tdnn_a_x/prune/prune_template_pct/sparse_rate_total.awk
nnet_tdnn_a_x/prune/prune_template_pct/split.awk
```

（7）在“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_template_x/prune”下，将run_several.sh和run_several_p
中“run_tdnn_x.sh”的x 值改为真实值。

（8）在“kaldi/egs/wsj/s5/”下，将run_tdnn_x.sh中的stage 参数改为9，因为在此任务之后的部分中，只需要重新解码，不再需要训练过程。

4. 进行prune和解码的任务（支持多组任务同时部署）

下面对不同方式的prune操作分别进行说明，大同小异。

（1）如果Value Prune的方案（详见第三部分）进行prune：

① 在“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_template_x/prune”下，使用如下命令生成prune文件夹：

```
bash copy_template.sh n
```

其中，n是该组实验内的编号。

② 进入刚生成的“prune_n”文件夹，修改“strategy_n”文件，在此说明本组实验的prune操作的策略。具体方式如下：

参数	含义	e.g.
第一个参数	被裁剪的层	all, affine1
第二个参数	阈值	abs<0.1 0<value<0.1 -0.1<value<0

③ 修改同目录下的“prune.awk”文件，在此实现本组实验的prune操作。具体方式如下：将连续的三行注释中有用的一行取消注释，修改阈值。

④ 可以多次重复①、②、③，同时部署多个实验

⑤ 返回上一级目录“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_template_x/prune”，修改run_several.sh中的几个参数：

参数	含义
max_id	实验id（即n）的最大值
for x in后面的部分（两处）	本次部署的实验id（即n）

⑥ 运行run_several.sh即可

（2）如果采用All-layer Pct Prune的方案（详见第三部分）进行prune：

① 在“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_template_x/prune”下，使用如下命令生成prune文件夹：

```
bash copy_template_pct.sh n
```

其中，n是该组实验内的编号。

② 进入刚生成的“prune_n”文件夹，修改“strategy_n”文件，在此说明本组实验的prune操作的策略。具体含义如下：

参数	含义	e.g.
第一个参数	被裁剪的层	all, affine1
第二个参数	pct	pct
第三个参数	阈值（稀疏度）	all:40%

③ 修改同目录下的“find_pct.py”文件，在此指定本组实验的prune 操作的百分比的值。具体方式如下：修改文件首部pct参数。

④ 可以多次重复①、②、③，同时部署多个实验

⑤ 返回上一级目录“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_template_x/prune”，修改run_several_pct.sh中的几个参数：

参数	含义
max_id	实验id（即n）的最大值
for x in后面的部分（两处）	本次部署的实验id（即n）

⑥ 运行run_several_pct.sh即可

（3）如果采用One-layer Pct Prune的方案（详见第三部分）进行prune：

① 在“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_template_x/prune”下，使用如下命令生成prune文件夹：

```
bash copy_template_pct.sh n
```

其中，n是该组实验内的编号。

② 进入刚生成的“prune_n”文件夹，修改“strategy_n”文件，在此说明本组实验的prune操作的策略。具体含义如下：

参数	含义	e.g.
第一个参数	被裁剪的层	all, affine1
第二个参数	pct	pct
第三个参数	阈值（稀疏度）	affine1:40%

③ 修改同目录下的“prune.awk”文件，在此指定需要对哪一层进行prune操作。具体方式如下：删除多余的if分支。

④ 修改同目录下的“find_pct.py”文件，在此设置百分比的阈值，具体方式如下：修改文件首部的pct值

⑤ 修改同目录下的“prune_pct.sh”文件，在此设置百分比的阈值，具体方式如下：修改为p_1,2,3...和n_1,2,3...赋值的部分，以及为prune.awk传参的部分。

⑥ 可以多次重复①-⑤，同时部署多个实验

⑦ 返回上一级目录“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_template_x/prune”，修改run_several_pct.sh中的几个参数：

参数	含义
max_id	实验id（即n）的最大值
for x in后面的部分（两处）	本次部署的实验id（即n）

⑧ 运行run_several_pct.sh即可

5. 查看结果

在执行完第3步后，在“kaldi/egs/wsj/s5/exp/nnet3/nnet_tdnn_a_template_x/prune/result”目录下，“strategy”、“sparse_rate”、“best_wer”分别记录了各个组的prune策略、prune后每层和整个模型的稀疏度、以及最终解码的wer。
为了便于复制到excel中进行处理，同时生成了“result_format”文件，该文件的每一行对应于每一个实验的strategy、sparse_rate、best_wer的结果。下面以其中一行为例，说明每一行各个数值的含义：
result_format中的一行：

```
all abs 0.04 29.7574 30.9761 28.1156 23.3632 31.0085 32.9020 33.8023 32.9039
8.64
```

此行各个参数的含义是：

参数	含义
all abs 0.04	对所有层（all）进行prune操作；采用Abs Prune方案进行prune；设置的绝对值阈值为0.04
29.7574	整个模型的稀疏度为29.7574%
30.9761 ... 32.9039	affine1, final-affine, affine2, affine3...affine6的稀疏度分别是30.9761%...32.9039%
8.64	最终解码的正确率为8.64%

二、Pruning Task With Retraining

在进行此项任务前，修改了Kaldi的源代码（可参考笔者添加的 NaturalGradientAffineComponent[2]），使值为0 的权值不再更新。另外，在此不再赘述与“一、Pruning Task Without Retraining”重复的部分。

1. 完成wsj的baseline和数据准备工作

参看“一、Pruning Task Without Retraining”中的对应部分。

2. 训练初始模型

参看“一、Pruning Task Without Retraining”中的对应部分。

3. 进行pruning task的准备工作

参看“一、Pruning Task Without Retraining”中的对应部分。也可只prune，不解码：注释掉run_several.sh或run_several_pct.sh中的相关语句即可。

4. 进行prune和解码的任务（支持多组任务同时部署）

参看“一、Pruning Task Without Retraining”中的对应部分。

5. retrain

（1）将“prune_n”文件夹中新生成的模型文件“final_new.mdl”复制到“nnet_tdnn_a_x”文件夹下，并重命名为20.mdl。

（2）为了顺利进行retrain，需要首先运行（run_tdnn.sh中调用的）train_tdnn.sh中生成0.mdl之前的部分，主要是为了生成egs文件。（此步只需要在n1时进行一次，之后不再需要）

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

图 3 文件结构

```
combine_best_wer.sh
new_decode_file.sh
decode_$n/
decode_baseline/ --- view_best_wer.sh
prune/           --- baseline/
                 --- prune_template/
                     --- sparse_rate.awk
                     --- sparse_rate_total.awk
                     --- sparse_rate_layer.sh
                     --- strategy
                     --- prune.awk
                     --- calc_sparse_rate_format.sh
                 --- prune_template_pct/
                     --- sparse_rate.awk
                     --- sparse_rate_total.awk
                     --- sparse_rate_layer.sh
                     --- strategy
                     --- prune.awk
                     --- split.awk
                     --- prune_pct.sh
                     --- count.py
                     --- find_pct.py
                 --- sparse_rate/
                     --- combine_sparse_rate.sh
                 --- strategy/
                     --- combine.awk
                     --- combine.sh
                 --- results/
                     --- combine_format.sh
                     --- deal_best_wer.awk
                     --- deal_strategy.awk
                 --- copy_template.sh
                 --- copy_template_pct.sh
                 --- prune_$n/
```

（3）将run_tdnn.sh中的stage参数设置为20，注释掉解码的相关代码，进行重训练。

（4）重训练到一定程度时（不必全部训完），将新生成的最后一个mdl文件（如40.mdl）重命名为final.mdl

6. 对新的模型进行再解码

（1）将run_tdnn.sh文件的stage参数设置为9（只进行解码），并运行。

（2）在“nnet_tdnn_a_x”文件夹下，找到decode文件夹（decode_tgpr...），在里面可以找到本次实验的结果（wer）。保存备份后，可以进行下一个实验。

6 代码文件及生成文件

在本部分中，主要介绍辅助pruning task的主要代码文件（夹）的接口、功能，以及各个生成文件（夹）的含义。读者可结合笔者上传的工具包[3]进行了解。

一、文件结构

如图3所示。

二、文件介绍（以“nnet_tdnn_a_template”为 work directory）

如表1和表2所示。

表 1 文件介绍（1）

文件	接口/功能/含义
decode_baseline/	放置baseline的decode文件夹(decode_tgpr_dev93_eval92)
view_best_wer.sh	查看该实验的best_wer
combine_best_wer.sh	按照指定格式，在“nnet_tdnn_a_x”下生成“best_wer”文件，该文件中记录了各个实验的结果(wer)
new_decode_file.sh	生成“decode_n”文件夹，将第n个实验的decode文件夹移动至该文件夹下
decode_n/	待生成，放置第n个实验的decode文件夹
baseline/	备份baseline的mdl文件
prune_n/	待生成，保存某一个prune 实验的策略，并具体实现该策略
prune_template/	prune_n的模板，用于使用Value Prune方案进行prune的任务
prune_template_pct/	prune_n的模板，用于使用Pct Prune方案进行prune的任务
sparse_rate_.awk	被sparse_rate_layer.sh调用。需从外部传入参数：start_line和end_line。在给定上述参数的条件下，统计文件中start_line （含）和end_line （含）之间的所有数字数目和0 的数目，并依此计算sparse_rate，并以两种不同格式分别写入文件sparse_rate 和sparse_rate_format
sparse_rate_total.awk	被sparse_rate_layer.sh调用。需在文件内部设置参数：is_lines_to_prune（标识linear_params_所在的行数)。在给定上述参数的条件下，计算所有层的所有数字数目和0的数目，并依此计算sparse_rate，并以两种不同格式分别写入文件sparse_rate 和sparse_rate_format
sparse_rate_layer.sh	需在文件内部设置参数：多个start_line和end_line。调用上述两个awk文件进行操作，具体实现统计整个模型以及每一层的sparse_rate的操作。
strategy	记录每个实验的prune方案

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

表 2 文件介绍（2）

prune_template中的prune.awk	需在文件内部设置参数：is_lines_to_prune。具体实现Value Prune方案的prune操作，可选择abs、positive、negative 三种prune方式（参见前文），方式为取消该种prune 方式的注释
prune_template_pct中的prune.awk	需在文件内部设置参数：is_line_to_prune_{1,2,3...}；需从外部传入的参数：每层的正阈值（p{1,2,3...}）和负阈值（n{1,2,3...}）。被prune_pct.sh 调用。根据传入的正负阈值对每一层进行prune操作
split.awk	需在文件内部设置参数：is_line_to_prune_{1,2,3...}。被prune_pct.sh调用。将mdl文件中每层的linear_params_保存为单独的文件（layer{1,2,3...}），方便后续处理
prune_template/	prune_n的模板，用于使用Value Prune方案进行prune的任务
count.py	需要文件：由split.awk生成的layer{1,2,3...}。统计每层的linear_params_矩阵的维数
find_pct.py	需在文件内部设置参数：pct（百分比）。被prune_pct.sh调用。按照前文“Pct Prune方案”中的描述，找到各层相应百分比的正值阈值和负值阈值，保存至文件trs
prune_pct.sh	在Pct Prune方案中，（调用了split.awk，find_pct.py，prune.awk）具体实现prune操作
strategy/	存放各个实验的prune方案
sparse_rate/	存放各个实验的（整体及每层的）sparse_rate
results/	**重要，**存放各个实验的strategy（prune 方案），sparse_rate，wer，因最终结果在result_format 中可以参考前文对result_format 文件进行解读
copy_template.sh	使用模板prune_template，生成相应的prune_n文件夹，用于使用Value Prune方案进行prune 的情况
copy_template_pct.sh	使用模板prune_template_pct，生成相应的prune_n文件夹，用于使用Pct Prune方案进行prune的情况
run_several.sh	**重要，**用于部署使用Value Prune 方案的一个或多个实验。用于编辑好prune_n文件夹之后。运行完毕后，结果可在results/ 文件夹下查看
run_several_pct.sh	**重要，**用于部署使用Pct Prune方案的一个或多个实验。用于编辑好prune_n文件夹之后。运行完毕后，结果可在results/文件夹下查看

7 实验描述

一、Pruning Task Without Retraining

1. 使用Value Prune方案进行Prune

1.1 Abs Prune

分别使用如下几组激活函数，设定绝对值阈值进行prune，之后进行decode 。

激活函数	维度
Pnorm	input=2000; output=250
Sigmoid	2000
Sigmoid	1000
Tanh	2000
Tanh	1000
Rectifier	2000
Rectifier	1000

1.2 Positive Prune

分别使用如下几组激活函数，设定positive threshold进行prune，之后进行decode。

激活函数	维度
Pnorm	input=2000; output=250
Sigmoid	2000
Sigmoid	1000
Tanh	2000
Tanh	1000
Rectifier	2000
Rectifier	1000

1.3 Negative Prune

分别使用如下几组激活函数，设定negative threshold进行prune，之后进行decode 。

激活函数	维度
Pnorm	input=2000; output=250
Sigmoid	2000
Sigmoid	1000
Tanh	2000
Tanh	1000
Rectifier	2000
Rectifier	1000

2. 使用Pct Prune方案进行Prune

2.1 分别使用如下几组激活函数，设定一百分比（即目标模型的sparse_rate）进行prune，之后进行decode 。

激活函数	维度
Pnorm	input=2000; output=250
Sigmoid	2000

2.2 使用2000维Sigmoid激活函数，对各单层进行One-layer Pct Prune操作（保持单层的稀疏度为一定值），之后进行decode

二、Pruning Task With Retraining

此部分中，笔者使用Abs Prune进行Prune。
分别使用如下几组激活函数，设定绝对值阈值进行prune，之后进行retrain，最后进行decode 。

激活函数	维度
Sigmoid	2000
Tanh	2000
Rectifier	2000

8 实验结果

一、Pruning Task Without Retraining

1. 各激活函数下，用Abs Prune方案进行Prune

1.1 参数：Pnorm（input：2000，output：250），No Retraining：如表3所示。

表 3 Pnorm 2000/250 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse rate	best_wer_tgpr
17	all	abs	0.01	35.7423	8.8600
10	all	abs	0.05	26.6452	8.3260
30	all	abs	0.0600	31.3210	8.5400
16	all	abs	0.07	35.7423	8.8469
9	all	abs	0.08	39.9287	10.4570
15	all	abs	0.09	43.9048	25.1044
8	all	abs	0.1000	47.6445	73.9055
7	all	abs	0.1500	63.4790	99.9559

1.2 参数：Sigmoid（2000），No Retraining：如表4所示。

表 4 Sigmoid 2000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse rate	best_wer_tgpr
14	all	abs	0.0100	13.1212	8.5200
5	all	abs	0.0150	19.5313	8.6100
2	all	abs	0.0200	25.7619	8.6800
8	all	abs	0.0250	31.7763	8.7800
11	all	abs	0.0300	37.5343	8.8100
23	all	abs	0.0350	43.0024	8.9400
24	all	abs	0.0400	48.1639	9.4500
1	all	abs	0.0500	57.5236	62.6600
18	all	abs	0.3500	99.8414	99.9600
19	all	abs	0.4000	99.9035	99.9600

1.3 参数：Sigmoid（1000），No Retraining：如表5所示。

表 5 Sigmoid 1000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer
1	all	abs	0.0400	29.7574	8.64
4	all	abs	0.0600	43.2075	8.78
7	all	abs	0.0800	55.1066	12.08
13	all	abs	0.1000	65.2604	86.94
9	all	abs	0.1500	83.0400	100

1.4 参数：Tanh（2000），No Retraining：如表6所示。

表 6 Tanh 2000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer_tgpr
10	all	abs	0.0225	28.8587	8.9100
9	all	abs	0.0250	31.8820	8.9600
8	all	abs	0.0275	34.8477	8.9100
1	all	abs	0.0300	37.7547	8.9700
4	all	abs	0.0325	0.0000	8.9300
5	all	abs	0.0350	43.3688	9.0200
6	all	abs	0.0375	46.0694	9.1000
7	all	abs	0.0400	48.6952	9.1900
31	all	abs	0.0450	53.7199	9.4100
26	all	abs	0.0500	58.4279	9.7100
23	all	abs	0.0600	66.8432	14.9000
20	all	abs	0.0800	79.7444	89.2000

1.5 参数：Tanh（1000），No Retraining：如表7所示。

表 7 Tanh 1000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer
16	all	abs	0.0800	62.2630	16.29
15	all	abs	0.0900	67.4199	48.48
14	all	abs	0.1000	71.9367	91.66
13	all	abs	0.1700	90.3070	100

1.6 参数：Rectifier（2000），No Retraining：如表8所示。

表 8 Rectifier 2000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer_tgpr
1	all	abs	0.0600	53.8950	9.2000
2	all	abs	0.0550	50.3026	9.0300
3	all	abs	0.0650	57.2891	9.9400
4	all	abs	0.0700	60.4945	11.0400

1.7 参数：Rectifier（1000），No Retraining：如表9所示。

表 9 Rectifier 1000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer
9	all	abs	0.0700	46.8778	9.12
10	all	abs	0.0800	52.3027	9.44
11	all	abs	0.0900	57.3241	10.28
12	all	abs	0.1000	61.9416	13.95
15	all	abs	0.1200	69.9865	29.64
16	all	abs	0.1400	76.5635	87.68

2. 各激活函数下，进行Positive Prune

2.1 参数：Pnorm（input：2000，output：250），NoRetraining：如表10所示。

表 10 Pnorm 2000/250 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse rate	best_wer_tgpr
49	all	+	0.0250	6.8878	8.2200
51	all	+	0.0260	7.1543	8.2900
33	all	+	0.0270	7.4196	8.3400
48	all	+	0.0280	7.6838	8.3500
46	all	+	0.0290	7.9459	8.5900
21	all	+	0.0300	8.2089	8.9800
44	all	+	0.0310	8.4679	9.9600
42	all	+	0.0320	8.7283	13.2300
28	all	+	0.0330	8.9857	23.5500
40	all	+	0.0340	9.2408	43.4700
27	all	+	0.0350	9.4953	71.0900
25	all	+	0.0400	10.7511	100.0000
23	all	+	0.0500	13.1516	100.0000

2.2 参数：Sigmoid（2000），NoRetraining：如表11所示。

表 11 Sigmoid 2000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse rate	best_wer_tgpr
15	all	+	0.0100	6.5478	8.8400
6	all	+	0.0150	9.7336	9.4000
3	all	+	0.0200	12.8286	10.9000
9	all	+	0.0250	15.8069	13.1600
12	all	+	0.0300	18.6544	16.6800
12	all	+	0.0400	23.9105	79.7500

2.3 参数：Sigmoid（1000），NoRetraining：如表12所示。

表 12 Sigmoid 1000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer
2	all	+	0.0300	11.2540	9.4
5	all	+	0.0600	21.4785	17.43
8	all	+	0.0800	27.3353	76.76
12	all	+	0.1000	32.3126	97.24

2.4 参数：Tanh（2000），NoRetraining：如表13所示。

表 13 Tanh 2000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer_tgpr
16	all	+	0.0200	12.8974	8.8600
15	all	+	0.0250	15.9457	8.8400
2	all	+	0.0300	18.8816	8.9300
11	all	+	0.0325	20.3036	8.9100
12	all	+	0.0350	21.6870	8.9600
13	all	+	0.0375	23.0355	9.1000
14	all	+	0.0400	24.3497	9.7900
34	all	+	0.0425	25.6267	21.2600
32	all	+	0.0450	26.8625	48.7900
35	all	+	0.0475	28.0606	78.2200
27	all	+	0.0500	29.2173	91.9900
24	all	+	0.0600	33.4265	96.5800
21	all	+	0.0800	39.8751	96.6000

2.5 参数：Tanh（1000），NoRetraining：如表14所示。

表 **14** Tanh 1000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer
1	all	+	0.0500	21.4241	8.96
3	all	+	0.0700	28.2121	9.69
7	all	+	0.0900	33.7019	21.05
8	all	+	0.1000	35.9603	67.08
9	all	+	0.1100	37.9376	95.22
5	all	+	0.1200	39.6462	95.3

2.6 参数：Rectifier（2000），NoRetraining：如表15所示。

表 **15** Rectifier 2000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer_tgpr
13	all	+	0.0400	18.4608	12.3200
15	all	+	0.0450	20.4063	21.5300
16	all	+	0.0500	22.2602	52.2600
5	all	+	0.0550	24.0203	88.9100
17	all	+	0.0550	24.0203	88.9100
6	all	+	0.0600	25.6946	98.1500
7	all	+	0.0650	27.2688	98.8300
8	all	+	0.0700	28.7583	98.7600

2.7 参数：Rectifier（1000），NoRetraining：如表16所示。

表 **16** Rectifier 1000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer
1	all	+	0.0400	13.1351	8.63
7	all	+	0.0450	14.5843	8.86
2	all	+	0.0500	15.9841	8.84
3	all	+	0.0600	18.6663	9.25
13	all	+	0.0700	21.1837	10.05
17	all	+	0.0900	25.6778	32.5400
18	all	+	0.1000	27.6746	82.8000
19	all	+	0.1100	29.5020	97.2500
20	all	+	0.1200	31.1839	99.3700

3. 各激活函数下，进行Negative Prune

3.1 参数：Pnorm（input：2000；output：250），NoRetraining：如表17所示。

表 **17** Pnorm 2000/250 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse rate	best_wer_tgpr
50	all	-	0.0250	6.9676	8.3000
52	all	-	0.0260	7.2389	8.3600
34	all	-	0.0270	7.5109	8.4700
47	all	-	0.0290	8.0517	8.6400
22	all	-	0.0300	8.3229	8.7300
45	all	-	0.0310	8.5927	9.2900
43	all	-	0.0320	8.8645	13.2300
29	all	-	0.0330	9.1317	15.2000
39	all	-	0.0350	9.6597	48.2800
26	all	-	0.0400	10.9585	100.0000

3.2 参数：Sigmoid（2000），NoRetraining：如表18所示。

表 **18** Sigmoid 2000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse rate	best_wer_tgpr
50	all	-	0.0250	6.9676	8.3000
52	all	-	0.0260	7.2389	8.3600
34	all	-	0.0270	7.5109	8.4700
47	all	-	0.0290	8.0517	8.6400
22	all	-	0.0300	8.3229	8.7300
45	all	-	0.0310	8.5927	9.2900
43	all	-	0.0320	8.8645	13.2300
29	all	-	0.0330	9.1317	15.2000
39	all	-	0.0350	9.6597	48.2800
26	all	-	0.0400	10.9585	100.0000

3.3 参数：Sigmoid（1000），NoRetraining：如表19所示。

表 **19** Sigmoid 1000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer
3	all	-	0.0200	7.6134	8.6
10	all	-	0.0300	11.3288	24.5
11	all	-	0.0400	14.9444	100
6	all	-	0.0600	21.7289	100

3.4 参数：Tanh（2000），NoRetraining：如表20所示。

表 **20** Tanh 2000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer_tgpr
19	all	-	0.0200	12.8905	8.9300
18	all	-	0.0250	15.9363	8.9600
3	all	-	0.0300	18.8731	9.0100
17	all	-	0.0350	21.6818	9.1200
36	all	-	0.0425	25.6222	28.8800
33	all	-	0.0450	26.8574	76.8700
37	all	-	0.0475	28.0563	94.6500
28	all	-	0.0500	29.2105	96.1200
25	all	-	0.0600	33.4167	98.9100

3.5 参数：Tanh（1000），NoRetraining：如表21所示。

表 **21** Tanh 1000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer
2	all	-	0.0500	21.4451	8.97
4	all	-	0.0700	28.2322	9.07
10	all	-	0.0900	33.7180	19.63
11	all	-	0.1000	35.9765	84.95
12	all	-	0.1100	37.9535	96.51
6	all	-	0.1200	39.6580	98.36

3.6 参数：Rectifier（2000），NoRetraining：如表22所示。

表 **22** Rectifier 2000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer_tgpr
14	all	-	0.0400	19.9364	87.0300
18	all	-	0.0450	22.1456	99.9700
19	all	-	0.0500	24.2652	100.0000
9	all	-	0.0550	26.2822	100.0000

3.7 参数：Rectifier（1000），NoRetraining：如表23所示。

表 **23** Rectifier 1000 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse_rate	best_wer
4	all	-	0.0400	15.2215	8.99
8	all	-	0.0450	17.0668	10.83
5	all	-	0.0500	18.8706	14.6
23	all	-	0.0550	20.6407	29.5600
6	all	-	0.0600	22.3810	91.52
21	all	-	0.0600	22.3810	91.5200
14	all	-	0.0700	25.6941	100

4. 各激活函数下，进行All-layer Pct Prune

4.1 参数：Pnorm（input：2000；output：250），NoRetraining：如表24所示。

表 24 Pnorm 2000/250 NoRetrain

group	pruned layer	pruning method	pruning threshold	sparse rate	best_wer_tgpr
55	all	pct	0.1000	10.0000	8.4000
58	all	pct	0.1300	13.0000	8.4200
54	all	pct	0.1500	15.0000	8.3500
56	all	pct	0.1700	17.0000	8.3700
57	all	pct	0.1800	18.0000	8.3500
53	all	pct	0.2000	20.0000	8.3300
59	all	pct	0.2200	22.0000	8.4000
60	all	pct	0.2600	26.0000	8.5900
61	all	pct	0.2665	26.6500	8.6300
63	all	pct	0.3000	30.0000	9.0400
62	all	pct	0.3993	39.9300	18.8700

4.2 参数：Sigmoid（2000），NoRetraining：如表25所示。

表 25 Sigmoid 2000 NoRetrain

group	pruned layer	pruning method	Pruning threshold	sparse rate	best_wer_tgpr
22	all	pct	0.3500	35.0000	9.0400
17	all	pct	0.4000	40.0000	9.4500
20	all	pct	0.4500	45.0000	10.2000
21	all	pct	0.5000	50.0000	12.1400

5. 使用2000维Sigmoid函数作激活函数，对单层进行One-layer Pct Prune操作

Prune策略	整个模型的稀疏度	wer
affine1的稀疏度：70%	1.8274	9.07
affine2的稀疏度：70%	10.1523	67.77
affine3的稀疏度：70%	10.1523	9.5
affine4的稀疏度：70%	10.1523	8.91
affine5的稀疏度：70%	10.1523	8.52
affine6的稀疏度：70%	10.1523	8.83
final-affine的稀疏度：70%	17.4112	21.32

二、Pruning Task With Retraining

在各激活函数下，进行Abs Prune。

详见“实验结论”部分中第“四”部分中的结果。

9 实验结论

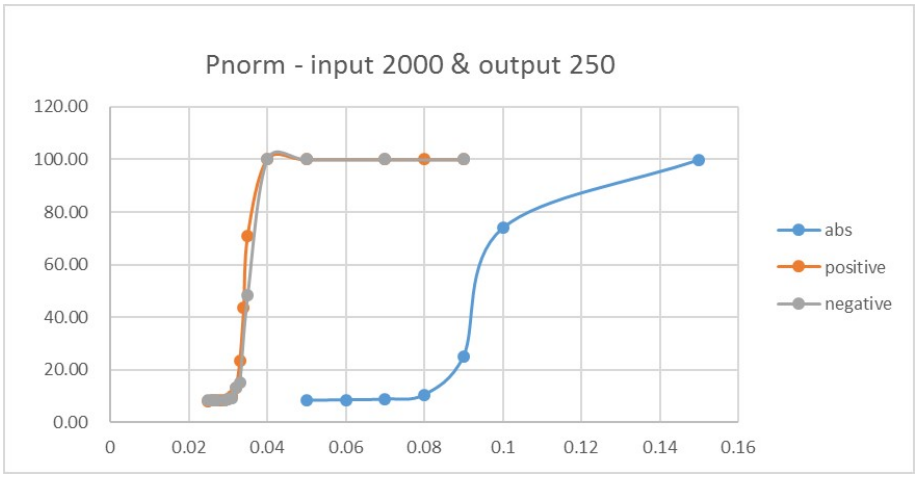
一、模型中正值、负值对于模型的影响程度

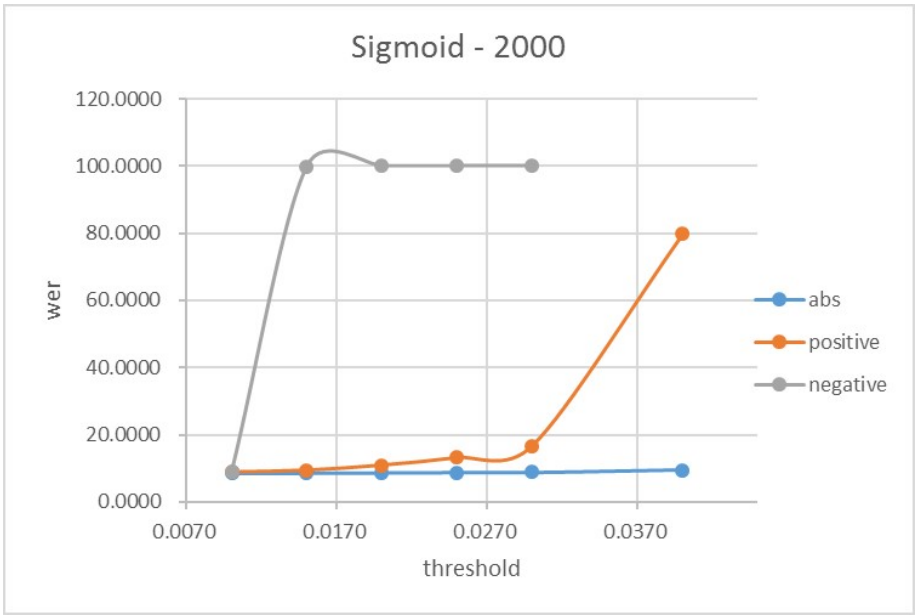
在这里，通过对比三种策略（Abs Prune，Positive Prune，Negative Prune）对于模型的影响程度。

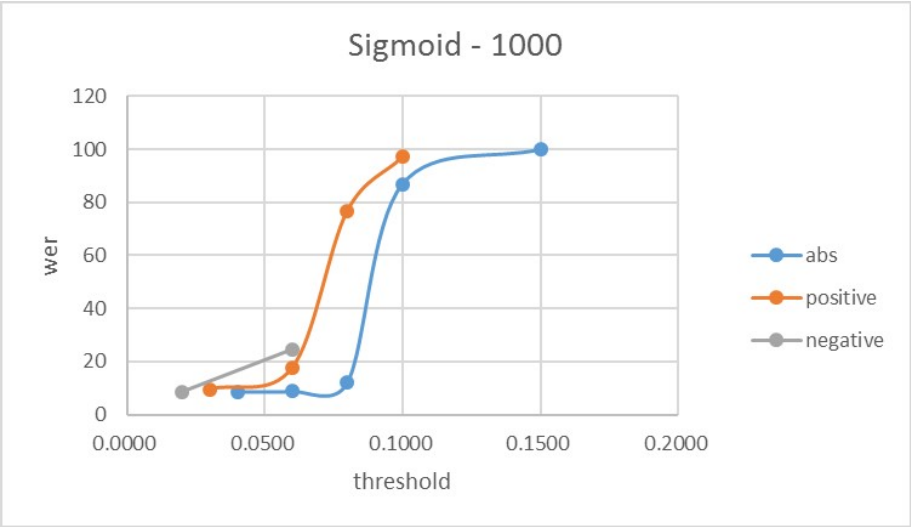
各激活函数下的图形：

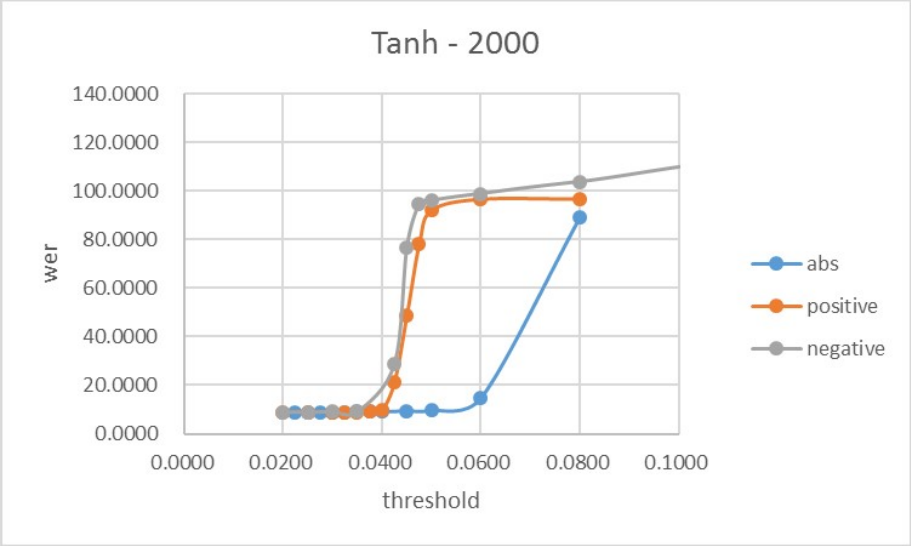
说明：以下各图中数据点是准确的，但因数据点不够密集，瞄点形成的平滑曲线的形状不是理想情况，可供参考。

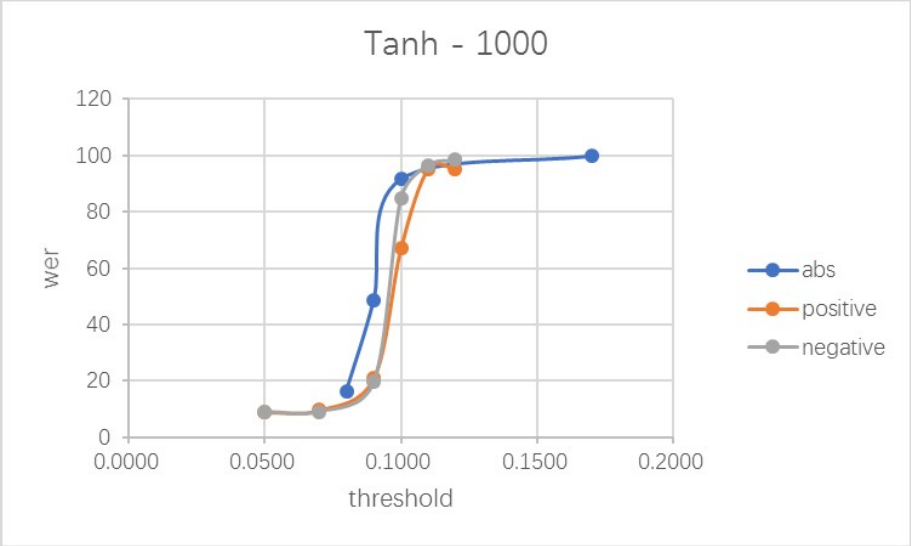
1. 结果图（下列各图的标题中，第一个参数表示激活函数的类别，第二个参数表示激活函数的维度）

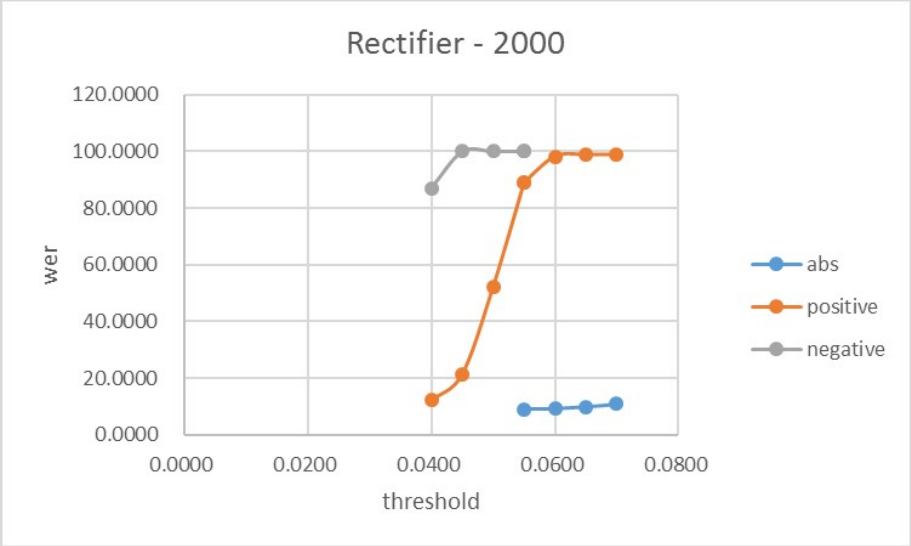


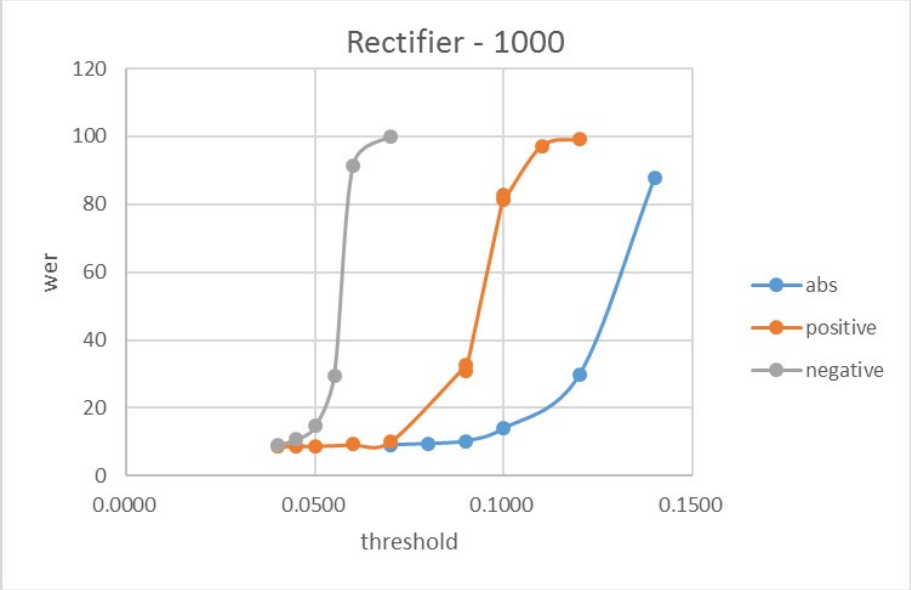



Sigmoid - 1000
120
100
80
60
40
20
0
wer
0.0000
0.0500
0.1000
0.1500
0.2000
threshold
abs
positive
negative


Tanh - 2000
140.0000
120.0000
100.0000
80.0000
60.0000
40.0000
20.0000
0.0000
wer
0.0000
0.0200
0.0400
0.0600
0.0800
0.1000
threshold
abs
positive
negative


Tanh - 1000
120
100
80
60
40
20
0
wer
0.0000
0.0500
0.1000
0.1500
0.2000
threshold
abs
positive
negative


Rectifier - 2000
120.0000
100.0000
80.0000
60.0000
40.0000
20.0000
0.0000
wer
0.0000
0.0200
0.0400
0.0600
0.0800
threshold
abs
positive
negative


Rectifier - 1000
120
100
80
60
40
20
0
wer
0.0000
0.0500
0.1000
0.1500
threshold
abs
positive
negative

2. 实验结论

2.1 网络中正负值的对称性

几乎所有实验表明，使用Abs Prune的操作不仅可以得到几乎两倍于Positive/Negative Prune的稀疏度，而且可以得到好于Positive Prune/Nagative Prune的wer值。另外，从“实验结果”部分可以知道，同一个绝对值区间内的正值和负值的数量相近。这暗示着在网络中，（绝对值相同的）Positive value 和Negative value是相互关联的，我们可以猜测：这种对称性对于网络的性能具有很大意义，当我们通过只进行Positive Prune或Negative Prune来破坏了这种对称性时，网络的性能急剧下降。
但应当注意，在这些组实验中也有一组例外，即上文1.5部分的图形（激活函数为1000维Tanh）。我们可以猜测，这种网络的对称性与网络的形状/容量/宽度/激活函数有着某种程度的依赖。这有待于进一步的探究。

2.2 网络中正负值对于模型的影响比较

几乎所有实验表明，在所设定阈值的绝对值相同的情况下，Positive Prune的操作和Negative Prune得到了相近的稀疏度，但使用Positive Prune 的操作可以得到比Negative Prune 更好的wer。我们可以猜测，Negative value在网络中扮演着比Positive value更为重要的角色。
但应当注意，如果使用Pnorm激活函数，会得到相反的结论。我们可以猜测，这个结论对于激活函数有着一定程度的依赖，这有待于进一步的探究。

二、各层对模型的影响程度

在这部分中，对affine1,affine2...affine6,final-affine分别进行单层的Prune操作，保持各组实验的单层稀疏度为定值（70%）。探究在单层稀疏度一致的情况下，对哪层进行Prune会最大程度的影响模型的性能。

1. 实验结果

1.1 实验参数
激活函数：Sigmoid（2000维）。
Retrain：No
1.2 结果
现将“实验结果”中第一部分的第5部分重列如下：

Prune策略	整个模型的稀疏度	wer
affine1的稀疏度：70%	1.8274	9.07
affine2的稀疏度：70%	10.1523	67.77
affine3的稀疏度：70%	10.1523	9.5
affine4的稀疏度：70%	10.1523	8.91
affine5的稀疏度：70%	10.1523	8.52
affine6的稀疏度：70%	10.1523	8.83
final-affine的稀疏度：70%	17.4112	21.32

2. 实验结论

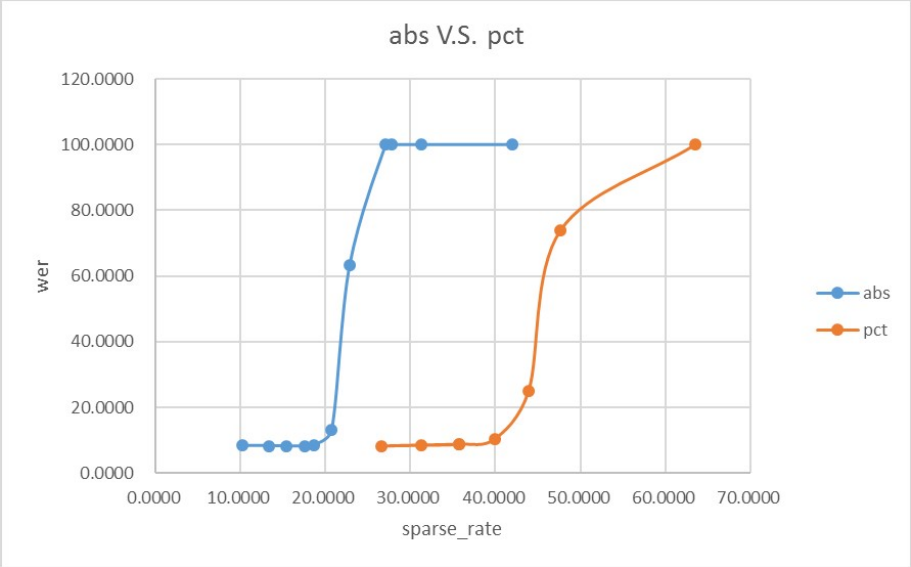
当对各连接层分别单独进行Prune操作，且保持单层的稀疏度一致时，对affine2进行Prune对于模型的性能影响最大，且这种影响不可归因于Prune掉的值过多的原因（因为对affine2进行prune后，整个模型的sparse_rate并不高）。由此得出初步的结论：在本模型中，affine2相比于其他几层，扮演更重要的角色。

但此结论仍应从不同维度（更换激活函数的种类和维度，更换Prune的百分比阈值……）进行进一步的验证。

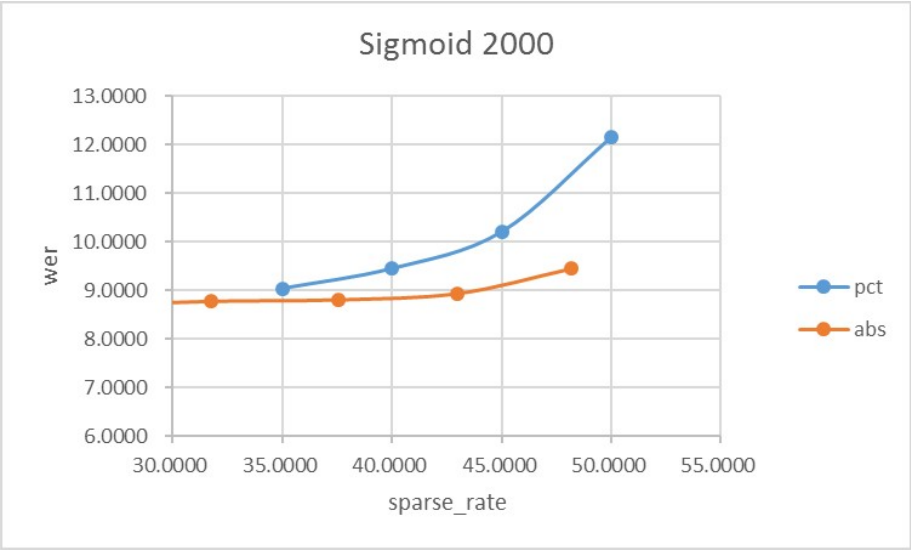
三、对比Value Prune和Pct Prune两种方式

1. 对比表格

1.1 Pnorm 2000/250



1.2 Sigmoid 2000



2. 实验结论

目前业界主要采用的是Abs Prune的方案。但我个人认为，Pct Prune的方案是针对各层更加均匀的Prune方案（各层的稀疏度一致）。

在上述两种激活函数下，得到的结论是不同的：在Pnorm 激活函数下，Abs Prune的效果更佳；但在Sigmoid激活函数下，Pct Prune的效果更佳。

四、对比Retrain前后的模型性能

1. 相关结果（下面各图的标题中，第一个参数表示激活函数的类别，第二个参数表示激活函数的维度）

表 26 Sigmoid 2000

pruned layer	pruning method	tr	sparse rate	wer	
				no retrain	after retrain
all	abs	0.0500	57.5236	62.66	8.9000
all	abs	0.1000	86.3390	100.00	9.5200
all	abs	0.1300	93.4277	100.00	10.3300
all	abs	0.1500	95.9358	100.00	11.2600
all	abs	0.1700	97.4409	100.00	13.3700
all	abs	0.2000	98.6525	100.00	31.8000
all	abs	0.2200	99.0829	100.00	41.4600
all	abs	0.3500	99.8414	99.96	96.1600

表 **27** Tanh 2000

pruned layer	pruning method	tr	sparse rate	wer	
				no retrain	after retrain
all	abs	0.1500	96.8415	100	14.13
all	abs	0.1700	97.9992	100	17.55
all	abs	0.2000	98.8709	100	67.68
all	abs	0.2200	99.1798	100	96.38
all	abs	0.2400	99.3840	100	96.26
all	abs	0.1400	95.9457	100	96.72

表 **28** Rectifier 2000

pruned layer	pruning method	tr	sparse rate	wer	
				no retrain	after retrain
all	abs	0.1500	89.9823	100.0000	9.4200
all	abs	0.1700	92.9586	100.0000	10.1100

2. 结论

在retrain之后，模型的性能相较于retrain之前得到大幅提升。用Sigmoid（2000）作为激活函数时，当稀疏度达到97%时仍能得到可以接受的结果（wer）。

使用不同的激活函数时，retrain之后的模型的性能表现不一。不同的激活函数如何影响最终模型（即retrain之后的模型）这有待于进一步的探究。

10 下一步的工作

1. 使用retrain之后的模型对“实验结论”部分中的“一”“二”“三”中提到的结论进行验证。

2. 对prune后的模型的连接值（除prune掉的值）进行随机初始化，retrain之后进行decode。探究模型结构与模型取值的重要性。

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. Dong Wang, Qiang Zhou, and Amir Hussain, *Deep and Sparse Learning in Speech and Language Processing: An Overview*, Springer International Publishing, 2016.
2. 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.
3. J Xue, J Li, and Y Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," pp. 2365–2369, 01 2013.
4. 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.
5. 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.
6. C Liu, Z Zhang, and D Wang, "Pruning deep neural networks by optimal brain damage," pp. 1092–1095, 01 2014.