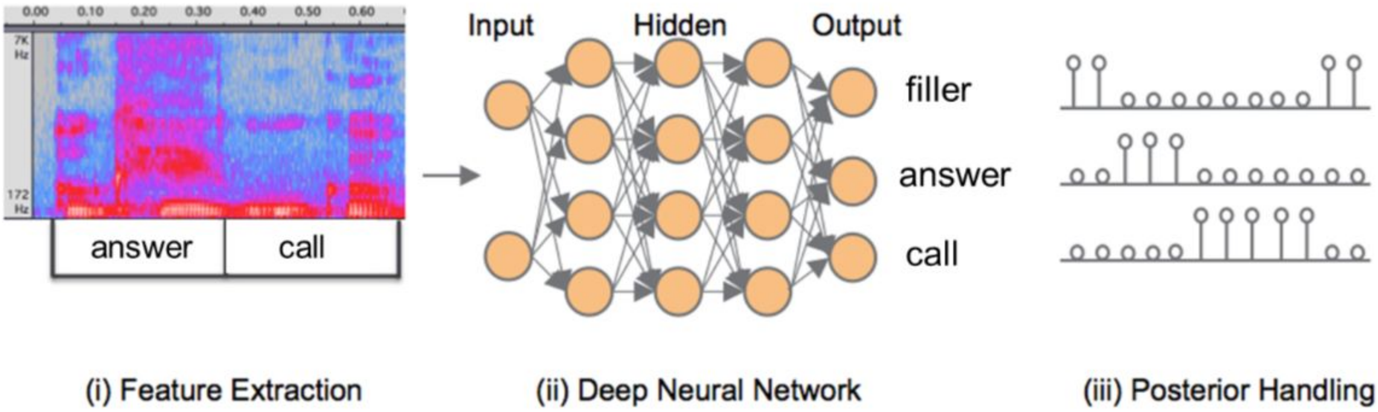


微控制器上的关键词检测技术

水木八刀
信号处理，语音识别

Hello Edge: Keyword Spotting on Microcontrollers

简要介绍

本文是ARM和Stanford合作的论文，在Google speech commands dataset上针对不同大小的资源限制进行了一系列实验，结论是depthwise separable convolutional neural network (DS-CNN) 性能最好，相比同等参数量的DNN结构有约10%的准确度提升。
本文也在同样数据集上正面比较了近几年small footprint KWS领域比较重要的几篇文章中的结构和性能（包括准确率，存储需求，计算量）， 这几篇文章分别来自Google, Baidu, 和Amazon，各自文章中的结果都基于私有数据。

> [5] Guoguo Chen, Carolina Parada, and Georg Heigold. Small-footprint keyword spotting using deep neural networks. In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, pages 4087—4091. IEEE, 2014.
> [6] Tara N Sainath and Carolina Parada. Convolutional neural networks for small-footprint key- word spotting. In Sixteenth Annual Conference of the International Speech Communication Association, 2015.
> [7] Sercan O Arik, Markus Kliegl, Rewon Child, Joel Hestness, Andrew Gibiansky, Chris Fougner, Ryan Prenger, and Adam Coates. Convolutional recurrent neural networks for small-footprint keyword spotting. arXiv preprint arXiv:1703.05390, 2017.
> [8] Ming Sun, Anirudh Raju, George Tucker, Sankaran Panchapagesan, Gengshen Fu, Arindam Mandal, Spyros Matsoukas, Nikko Strom, and Shiv Vitaladevuni. Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting. In Spoken Language Technology Workshop (SLT), 2016 IEEE, pages 474—480. IEEE, 2016.

数据集介绍

Google speech commands dataset 包含6.5w 1s长度的音频，共有30个关键词，每个音频对应一个关键词的语音，有数千人录制。
检测任务为给定一段音频，将其正确分类为如下12类中的一种：

Yes”, “No”, “Up”, “Down”, “Left”, “Right”, “On”, “Off”, “Stop”, “Go”, “silence”， “unknown”

data augmentation:
加背景噪声
100ms以内随机时移

典型KWS系统流程

典型KWS系统如figure1所示，由于测试时输入是连续语音流，因此通常还需要一个后处理过程。后处理过程通常采用简单的平滑策略，神经网络的结构对于最终性能起决定作用。

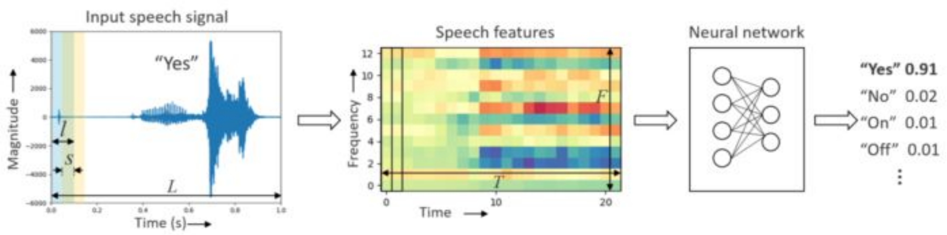
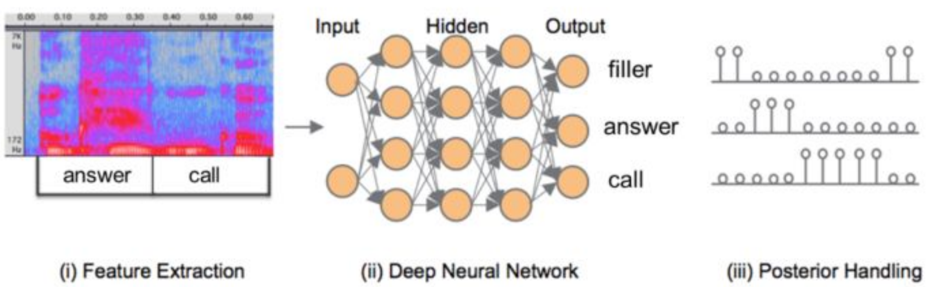


Figure 1: Keyword spotting pipeline.

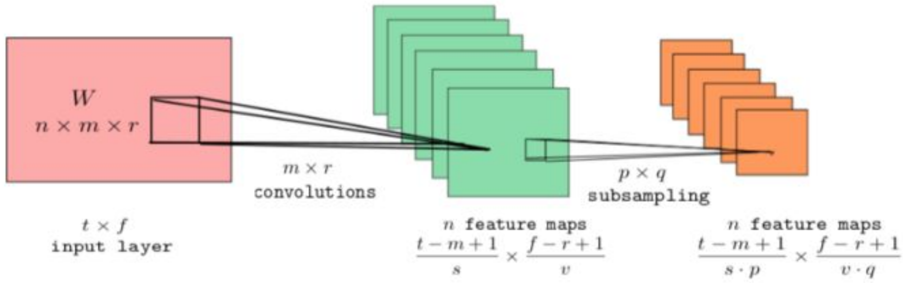
KWS中的神经网络结构

DNN



CNN

t–f语谱图作为输入，通常为灰度图（通道为1），也有文章同时提取多种特征拼成彩色图



RNN

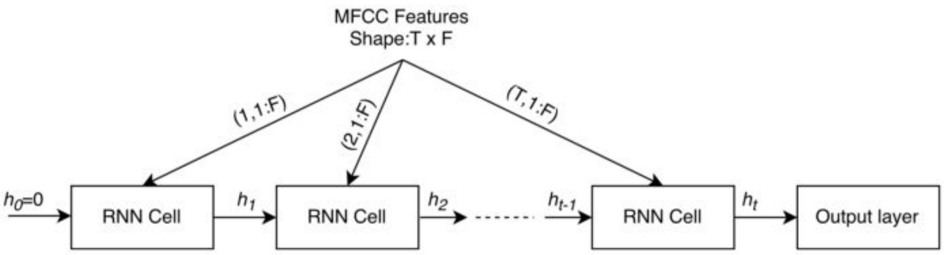


Figure 2: Model architecture of RNN.

CRNN

CNN for local temporal/spatial correlation

RNN for global temporal dependencies

GRU as base cell, less parameters and better convergence than lstm

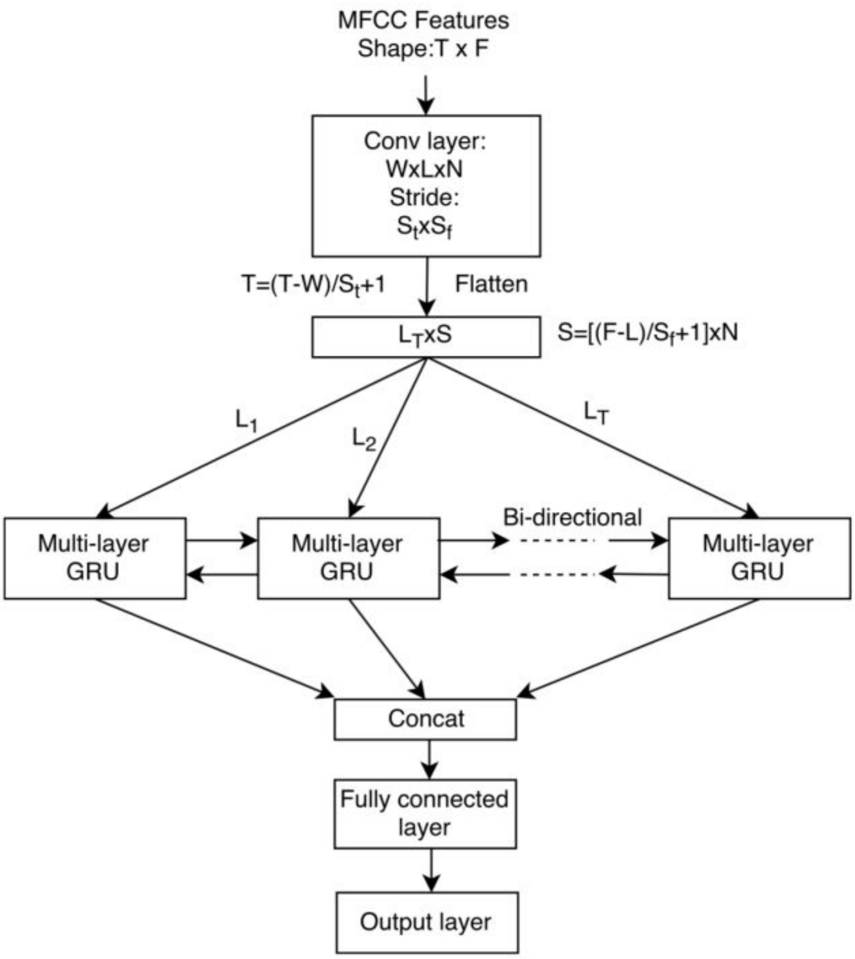


Figure 3: Model Architecture of CRNN.

Depthwise Separable Convolutional Neural Network (DS-CNN)

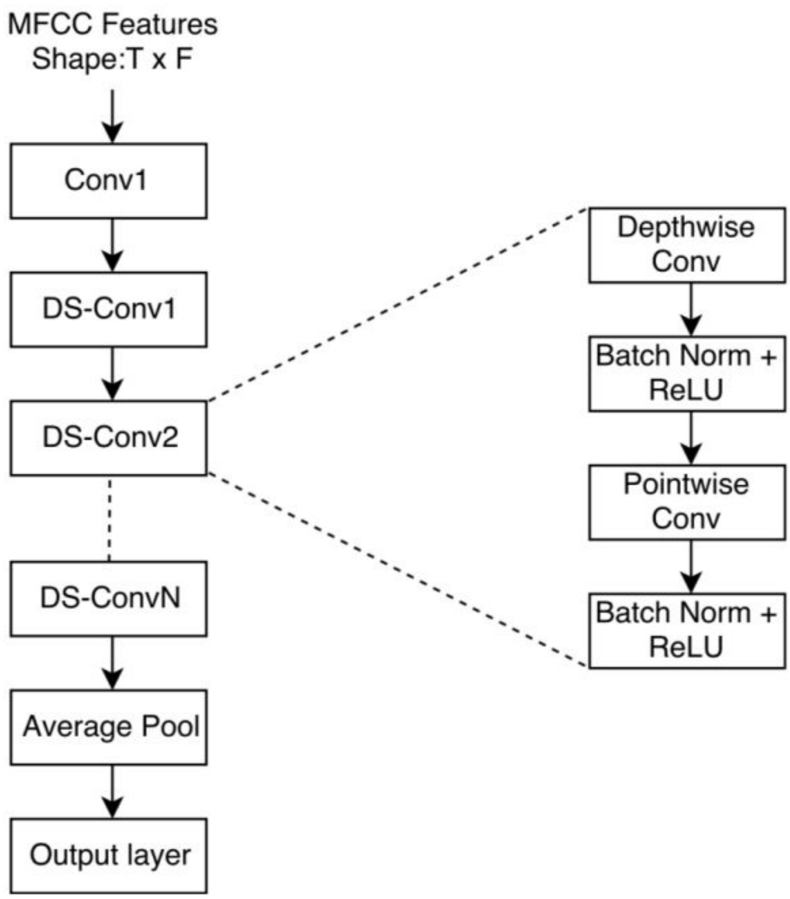
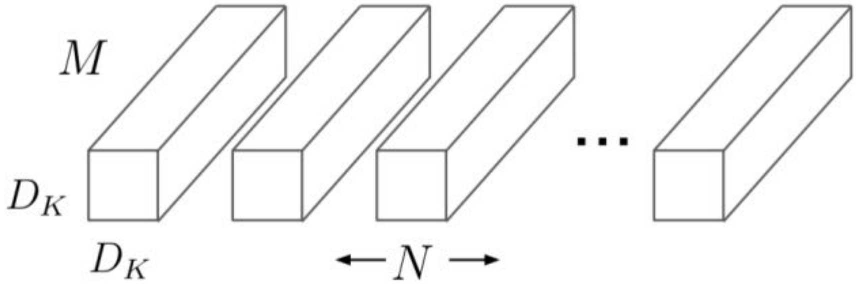
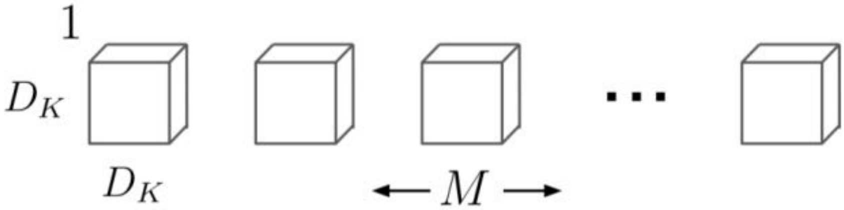


Figure 4: Depthwise separable CNN architecture.

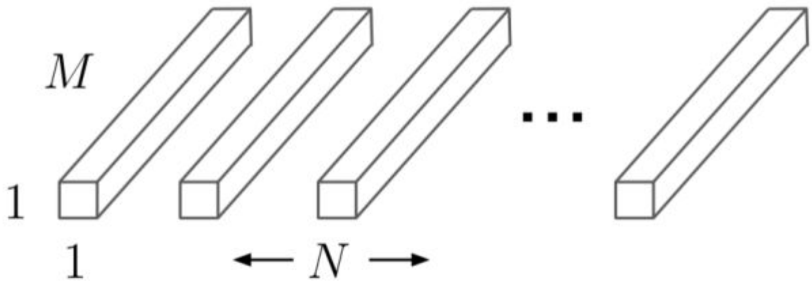
> DS-CNN details



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

> [10] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

基本参数配置

参数	配置
loss	CE
optimizer	Adam
batch size	100
learning rate	[5e-4, 1e-4]
iter	[20K, 10K]

train : valid : test = 80 : 10 : 10， 最终性能由test确定

实验结果

典型结构性能对比

采用40维MFCC, 40ms帧长，20ms帧移，1s音频共有1960(49x40)个特征，评估存储时，假定采用8-bit 权重和激活值

NN Architecture	Accuracy	Memory	Operations
DNN [5]	84.3%	288 KB	0.57 MOps
CNN-1 [6]	90.7%	556 KB	76.02 MOps
CNN-2 [6]	84.6%	149 KB	1.46 MOps
LSTM [8]	88.8%	26 KB	2.06 MOps
CRNN [7]	87.8%	298 KB	5.85 MOps

当然，由于各自文章的结果都是在不同的数据集和资源限制下优化的，直接在google commands dataset上也不是很公平，但是还是能发现一些**insight**：

DNN准确率不高，但是计算量最小，适合算力受限的场景
CNN相比DNN准确率更高，但是也需要消耗更多的算力或存储
LSTM和CRNN则保证较高准确率的同时在存储和算力之间取得很好的平衡

> CNN-1 and CNN-2 details

type	m	r	n	p	q	Par.	Mul.
conv	20	8	64	1	3	10.2K	4.4M
conv	10	4	64	1	1	164.8K	5.2M
lin	-	-	32	-	-	65.5K	65.5K
dnn	-	-	128	-	-	4.1K	4.1K
softmax	-	-	4	-	-	0.5K	0.5K
Total	-	-	-	-	-	244.2K	9.7M

Table 1: CNN Architecture for `cnn-trad-fpool3`

model	m	r	n	s	v	Params	Mult
(a)	32	8	**186**	1	4	47.6K	428.5K
(b)	32	8	**336**	1	8	86.6K	430.1K

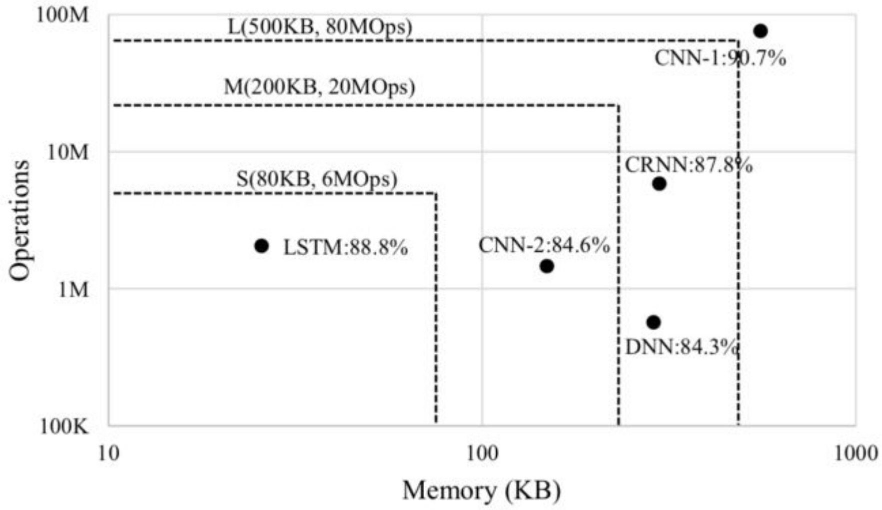
Table 3: CNN for (a) `cnn-one-fstride4` and (b) `cnn-one-fstride8`

> [6] Tara N Sainath and Carolina Parada. Convolutional neural networks for small-footprint key- word spotting. In Sixteenth Annual Conference of the International Speech Communication Association, 2015.

按资源需求对神经网络分类

NN size	NN memory limit	Ops/inference limit
Small (S)	80 KB	6 MOps
Medium (M)	200 KB	20 MOps
Large (L)	500 KB	80 MOps

ops/inference limit 假定1s infer 10次。这里并不是基于流式音频处理, 而是每隔 100ms 直接 infer 1s 数据给出检测结果，相邻的1s数据窗有90%的重叠。 由于数据增强时给训练样本做了100ms以内随机时移，因此每隔100ms做infer是合理的。 对一张1s的图直接infer出一个softmax结果，现场使用时，比如1s数据则infer 了10次，平滑可以在这10个结果中做。这样做比流式效率更高，因为1s仅infer 10次，而不是每帧都infer。本文对应的代码里并没有给出具体的平滑策略，而只是把每次infer的值打印出来。



寻找最优网络结构

特征提取的超参数

MFCC维数 F 和 帧移 S

最优结果对应的 F=10， S=20 ms

模型超参数

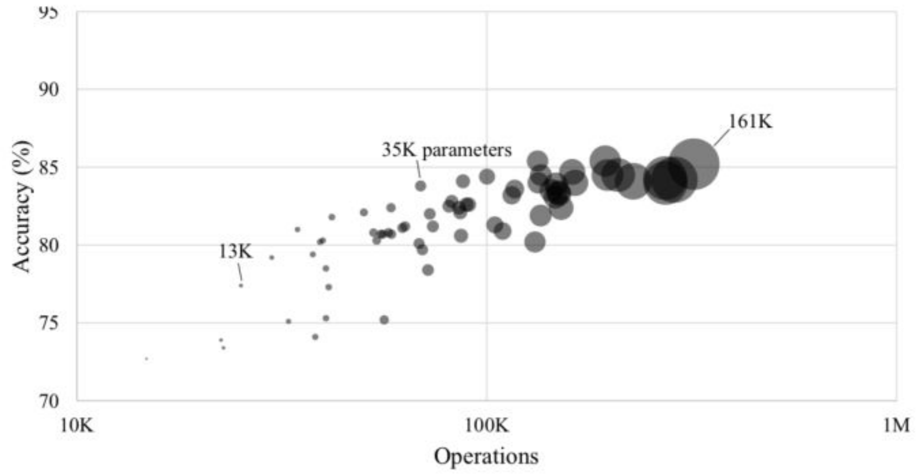
默认采用ReLU激活函数，卷积层和全连接层后均接Batch-Norm, 循环层后均接Layer-Norm

NN model	Model hyperparameters
DNN	Number of fully-connected (FC) layers and size of each FC layer
CNN	Number of Conv layers: features/kernel size/stride, linear layer dim., FC layer size
Basic LSTM	Number of memory cells
LSTM	Number of memory cells, projection layer size
GRU	Number of memory cells
CRNN	Conv features/kernel size/stride, Number of GRU and memory cells, FC layer size
DS-CNN	Number of DS-Conv layers, DS-Conv features/kernel size/stride

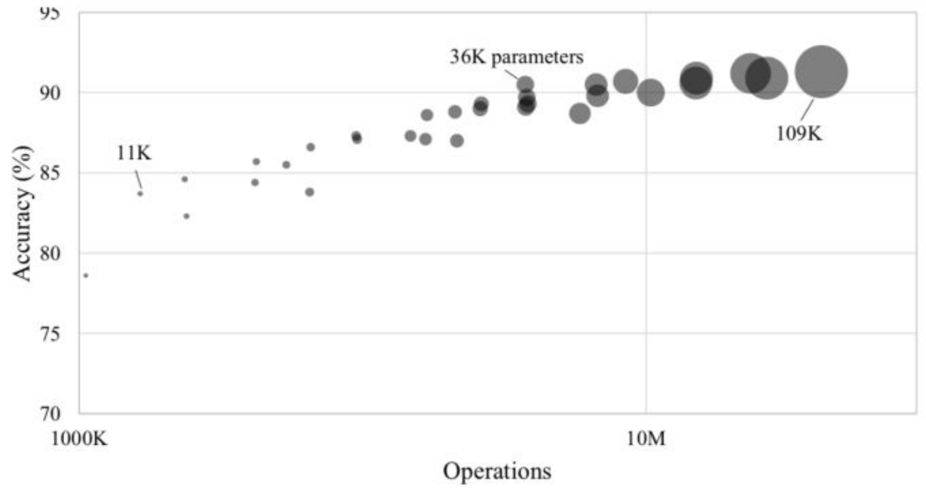
Table 4: Neural network hyperparameters used in this study.

人工选择+遍历搜索

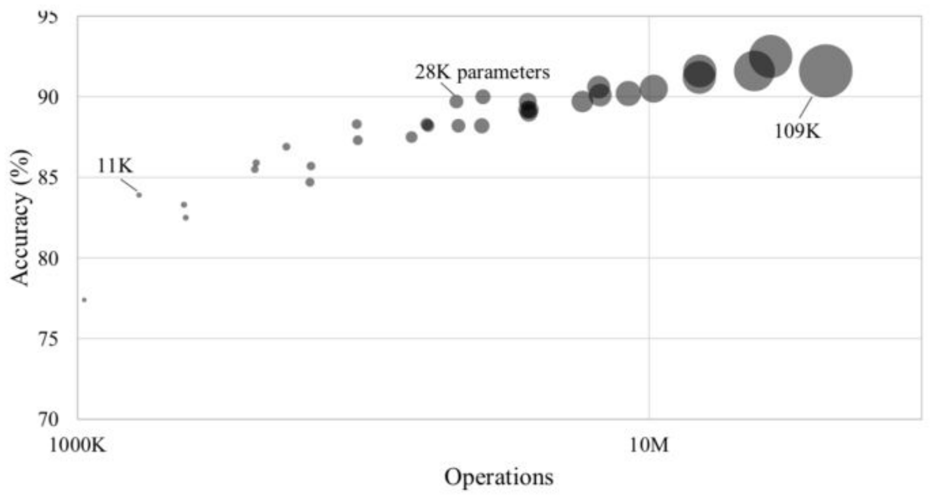
We iteratively perform exhaustive search of feature extraction hyperparameters and NN model hyperparameters followed by manual selection to narrow down the search space. The final best



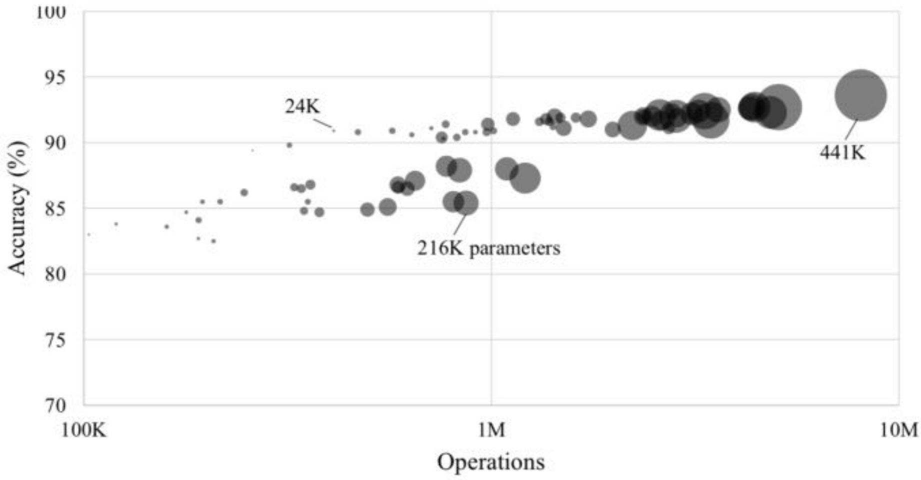
(a) DNN



(b) Basic LSTM



(c) LSTM



(d) CRNN

最优结果：

NN model	S(80KB, 6MOps)			M(200KB, 20MOps)			L(500KB, 80MOps)		
	Acc.	Mem.	Ops	Acc.	Mem.	Ops	Acc.	Mem.	Ops
DNN	84.6%	80.0KB	158.8K	86.4%	199.4KB	397.0K	86.7%	496.6KB	990.2K
CNN	91.6%	79.0KB	5.0M	92.2%	199.4KB	17.3M	92.7%	497.8KB	25.3M
Basic LSTM	92.0%	63.3KB	5.9M	93.0%	196.5KB	18.9M	93.4%	494.5KB	47.9M
LSTM	92.9%	79.5KB	3.9M	93.9%	198.6KB	19.2M	94.8%	498.8KB	48.4M
GRU	93.5%	78.8KB	3.8M	94.2%	200.0KB	19.2M	94.7%	499.7KB	48.4M
CRNN	94.0%	79.7KB	3.0M	94.4%	199.8KB	7.6M	95.0%	499.5KB	19.3M
DS-CNN	94.4%	38.6KB	5.4M	94.9%	189.2KB	19.8M	95.4%	497.6KB	56.9M

Table 5: Summary of best neural networks from the hyperparameter search. The memory required for storing the 8-bit weights and activations is shown in the table.

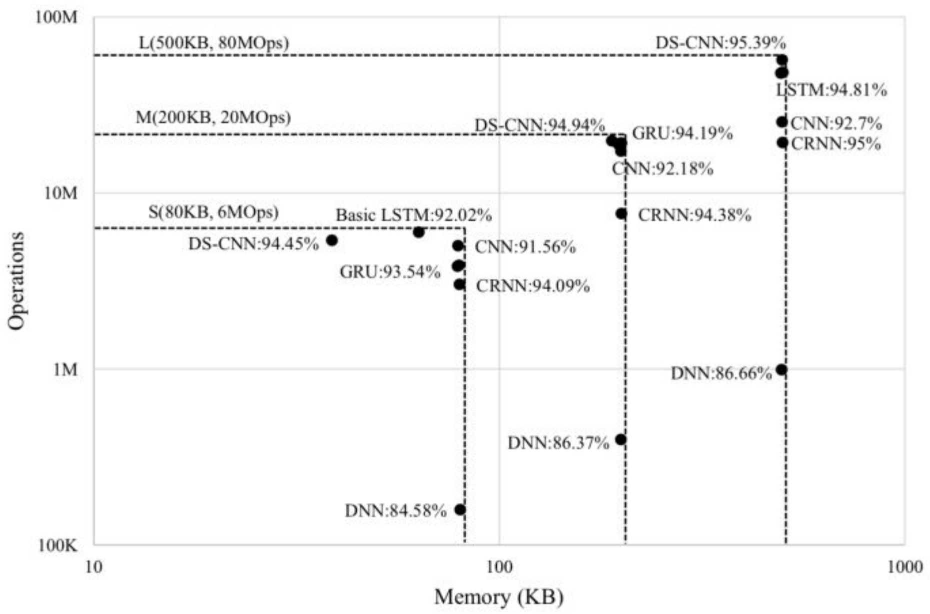


Figure 6: Memory vs. Ops/inference of the best models described in Table 5.

最优结果对应的超参数集：

FC(n) 表示n个神经元的全连接层

L(n) 表示n个神经元的Low-rank线性层

C(num_filters, kenerl_size_time, kernel_size_freq, stride_time, stride_freq)

LSTM(n), GRU(n) 表示n个cell的LSTM或GRU结构

DSC(num_filters, kerne率方向的卷积核大小相同，步长也相同

Model	S	NN model hyperparameters	Memory	Ops	Train	Val.	Test
DNN	40	FC(144)-FC(144)-FC(144)	80.0KB	158.8K	91.5%	85.6%	84.6%
DNN	40	FC(256)-FC(256)-FC(256)	199.4KB	397.1K	95.4%	86.7%	86.4%
DNN	40	FC(436)-FC(436)-FC(436)	496.6KB	990.2K	97.8%	88.0%	86.7%
CNN	20	C(28,10,4,1,1)-C(30,10,4,2,1)-L(16)-FC(128)	79.0KB	5.0M	96.9%	91.1%	91.6%
CNN	20	C(64,10,4,1,1)-C(48,10,4,2,1)-L(16)-FC(128)	199.4KB	17.3M	98.6%	92.2%	92.2%
CNN	20	C(60,10,4,1,1)-C(76,10,4,2,1)-L(58)-FC(128)	497.8KB	25.3M	99.0%	92.4%	92.7%
Basic LSTM	20	LSTM(118)	63.3KB	5.9M	98.2%	91.5%	92.0%
Basic LSTM	20	LSTM(214)	196.5KB	18.9M	98.9%	92.0%	93.0%
Basic LSTM	20	LSTM(344)	494.5KB	47.9M	99.1%	93.0%	93.4%
LSTM	40	LSTM(144), Projection(98)	79.5KB	3.9M	98.5%	92.3%	92.9%
LSTM	20	LSTM(280), Projection(130)	198.6KB	19.2M	98.8%	92.9%	93.9%
LSTM	20	LSTM(500), Projection(188)	498.8KB	4.8M	98.9%	93.5%	94.8%
GRU	40	GRU(154)	78.8KB	3.8M	98.4%	92.7%	93.5%
GRU	20	GRU(250)	200.0KB	19.2M	98.9%	93.6%	94.2%
GRU	20	GRU(400)	499.7KB	48.4M	99.2%	93.9%	93.7%
CRNN	20	C(48,10,4,2,2)-GRU(60)-GRU(60)-FC(84)	79.8KB	3.0M	98.4%	93.6%	94.1%
CRNN	20	C(128,10,4,2,2)-GRU(76)-GRU(76)-FC(164)	199.8KB	7.6M	98.7%	93.2%	94.4%
CRNN	20	C(100,10,4,2,1)-GRU(136)-GRU(136)-FC(188)	499.5KB	19.3M	99.1%	94.4%	95.0%
DS-CNN	20	C(64,10,4,2,2)-DSC(64,3,1)-DSC(64,3,1)-DSC(64,3,1)-DSC(64,3,1)-AvgPool	38.6KB	5.4M	98.2%	93.6%	94.4%
DS-CNN	20	C(172,10,4,2,1)-DSC(172,3,2)-DSC(172,3,1)-DSC(172,3,1)-DSC(172,3,1)-AvgPool	189.2KB	19.8M	99.3%	94.2%	94.9%
DS-CNN	20	C(276,10,4,2,1)-DSC(276,3,2)-DSC(276,3,1)-DSC(276,3,1)-DSC(276,3,1)-DSC(276,3,1)-AvgPool	497.6KB	56.9M	99.3%	94.3%	95.4%

Table 7: Summary of hyperparameters of the best models described in Table 5.

神经网络量化

32-bit float 权重和激活值 采用8-bit int 定点表示

$$v = -B_7 \times 2^{7-N} + \sum_{i=0}^{6} B_i \times 2^{i-N}$$

N为小数点后的小数部分的位数。
神经网络中每层使用同样的N，但不同层可以使用不同的N。
N的选取通过逐层优化使得量化后准确率损失最小。比如，先保持后面所有参数为浮点，仅将第一层权重量化，找到最优N使得准确率损失最小，然后固定第一层，优化第二层，以此类推
注意到是直接将训练好的浮点参数量化，**无需重新训练**
可以看到，量化后性能没有损失，**测试集上甚至略微提升**

NN model	32-bit floating point model accuracy			8-bit quantized model accuracy		
	Train	Val.	Test	Train	Val.	Test
DNN	97.77%	88.04%	86.66%	97.99%	88.91%	87.60%
Basic LSTM	98.38%	92.69%	93.41%	98.21%	92.53%	93.51%
GRU	99.23%	93.92%	94.68%	99.21%	93.66%	94.68%
CRNN	98.34%	93.99%	95.00%	98.43%	94.08%	95.03%

代码

本文对应tensorflow DS-CNN code

调用slim接口

```
def _depthwise_separable_conv(inputs,
                              num_pwc_filters,
                              sc,
                              kernel_size,
                              stride):
  """ Helper function to build the depth-wise separable convolu
  """
  # skip pointwise by setting num_outputs=None
  depthwise_conv = slim.separable_convolution2d(inputs,
                                                 num_outputs=Nor
                                                 stride=stride,
                                                 depth_multiplie
                                                 kernel_size=ke
                                                 scope=sc+'/dept
  bn = slim.batch_norm(depthwise_conv, scope=sc+'/dw_batch_norr
  pointwise_conv = slim.convolution2d(bn,
                                      num_pwc_filters,
                                      kernel_size=[1, 1],
                                      scope=sc+'/pointwise_conv
```

```
    bn = slim.batch_norm(pointwise_conv, scope=sc+'/pw_batch_norr

    return bn
```

mxnet DS-CNN code

找到mobileNet的mxnet实现，可以看到DS-CNN直接由两个卷积实现，因此低版本mxnet应该也可以支持

```
def Conv_DPW(data, depth=1, stride=(1, 1), name='', idx=0, suf

    conv_dw = Conv(data, num_group=depth, num_filter=depth, kerne

    conv = Conv(conv_dw, num_filter=depth * stride[0], kernel=(1,

    return conv
```

特征提取部分

找到tensorflow提取mfcc源码，可以看到filterbank默认只有40个，也即40维MFCC只是对40维FBANK通过DCT进行去相关，10维MFCC则还有降维（因为配置参数里仅配置dct_coef维度，并没有filterbank数的配置）

```
namespace tensorflow {

const double kDefaultUpperFrequencyLimit = 4000;
const double kDefaultLowerFrequencyLimit = 20;
const double kFilterbankFloor = 1e-12;
const int kDefaultFilterbankChannelCount = 40;
const int kDefaultDCTCoefficientCount = 13;
```