基于 FPGA 的递归神经网络加速器的研究进展

高琛，张帆

（国家数字交换系统工程技术研究中心，河南 郑州 450002）

摘　要：递归神经网络(RNN)近些年来被越来越多地应用在机器学习领域，尤其是在处理序列学习任务中，相比 CNN 等神经网络性能更为优异。但是 RNN 及其变体，如 LSTM、GRU 等全连接网络的计算及存储复杂性较高，导致其推理计算慢，很难被应用在产品中。一方面，传统的计算平台 CPU 不适合处理 RNN 的大规模矩阵运算；另一方面，硬件加速平台 GPU 的共享内存和全局内存使基于 GPU 的 RNN 加速器的功耗比较高。FPGA 由于其并行计算及低功耗的特性，近些年来被越来越多地用来做 RNN 加速器的硬件平台。对近些年基于 FPGA 的 RNN 加速器进行了研究，将其中用到的数据优化算法及硬件架构设计技术进行了总结介绍，并进一步提出了未来研究的方向。

关键词：递归神经网络；FGPA；加速器

中图分类号：TP391.1

文献标识码：A

doi: 10.11959/j.issn.2096−109x.2019034

Survey of FPGA based recurrent neural network accelerator

GAO Chen, ZHANG Fan

National Digital Switching System Engineering and Technological Research Center, Zhengzhou 450002, China

Abstract: Recurrent neural network(RNN) has been used wildly used in machine learning field in recent years, especially in dealing with sequential learning tasks compared with other neural network like CNN. However, RNN and its variants, such as LSTM, GRU and other fully connected networks, have high computational and storage complexity, which makes its inference calculation slow and difficult to be applied in products. On the one hand, traditional computing platforms such as CPU are not suitable for large-scale matrix operation of RNN. On the other hand, the shared memory and global memory of hardware acceleration platform GPU make the power consumption of GPU-based RNN accelerator higher. More and more research has been done on the RNN accelerator of the FPGA in recent years because of its parallel computing and low power consumption performance. An overview of the researches on RNN accelerator based on FPGA in recent years is given. The optimization algorithm of software level and the architecture design of hardware level used in these accelerator are summarized and some future research directions are proposed.

Key words: recurrent neural network, FPGA, accelerator

收稿日期：2019−02−20；**修回日期：**2019−04−15

通信作者：高琛，616414829@qq.com

基金项目：国家自然科学基金资助项目（No.61572520）；国家自然科学基金创新研究群体资助项目（No.61521003）

Foundation Items: The National Natural Science Foundation of China (No.61572520); The National Natural Science Foundation for Creative Research Groups of China (No.61521003)

论文引用格式：高琛, 张帆. 基于 FPGA 的递归神经网络加速器的研究进展[J]. 网络与信息安全学报, 2019, 5(4): 1-13.

GAO C, ZHANG F. Survey of FPGA based recurrent neural network accelerator[J]. Chinese Journal of Network and Information Security, 2019, 5(4): 1-13.

1　引言

在当今社会，人工智能和机器学习领域的研究已经取得了显著成就。现有的一些研究表明神经网络在这些领域相比传统算法的性能更好。然而，在自然语言处理领域，对机器来说，理解、学习人类语言仍然是一件很困难的事情，这成为一个比较有挑战性的问题[1]。研究者提出了很多模型来解决这个问题，其中比较有代表性的是递归神经网络（RNN，recurrent neural network）及其变体，这些模型在语音建模[2-3]、机器翻译[4]、语音识别[5]、情景分析[6]等任务中取得了非常好的性能。总体来说，RNN 在语音特征识别及提取任务中的性能非常好，使其在人工智能领域中应用越来越广泛。

但 RNN 的计算及存储复杂度比较高，特别是为了取得更好的性能，研究人员往往会将 RNN 模型设计得越来越大、越来越复杂[7]，这导致模型参数量及计算复杂度增加，从而使 RNN 的训练及推理过程变得很慢。

RNN 模型的训练及推理运算中包含大量复杂的矩阵运算，这导致在 CPU 上对 RNN 进行计算十分不高效，因此选择一个合适的硬件加速平台对 RNN 模型的运算进行加速显得非常重要。一个典型的 CPU 每秒可以执行 10~100 G 次浮点操作，电源效率通常低于 1 GOP/J，这使 CPU 很难满足云计算应用的高性能需求或者移动终端应用的低能耗需求[8]。与 CPU 不同，GPU 由于其强大的并行处理能力，在大规模计算领域扮演着越来越重要的角色[9]。GPU 高效的并行计算能力可以让 RNN 模型实现高性能的训练及推理计算，另外，深度学习框架 Caffe 和 Tensorflow[10]也为 GPU 提供了很多快速简易的接口。但由于 GPU 在计算时访问共享内存和全局内存，并且有较低的指令驱动执行模式，因此基于 GPU 的设计往往需要消耗大量的能源来获取显著计算速度的提升[11]。专用集成电路 ASIC 能获得更高的功效，但是开发周期比较长，成本比较高[8]。

近些年来，FPGA 越来越多地被应用在高性能人工智能计算领域。由于其灵活的可配置性、低能耗及并行计算等特性，基于 FPGA 的硬件加速计算近年来成为一个热点，因此，通过 FPGA 对 RNN 模型实现高性能计算是一个可行的方案。相比于 GPU，基于 FPGA 的 RNN 加速器往往能获得更高的能源效率[12]，所以 FPGA 更适合用作 RNN 加速的平台。

由于 RNN 模型的参数量比较大，一般的 FPGA 芯片不足以完全存储其权重参数；另外，由于 RNN 模型的复杂性，将 RNN 模型映射到 FPGA 上，设计合理的加速器架构进行加速计算比较困难。目前，比较流行的神经网络架构 Caffe 和 Tensorflow 也没有支持 FPGA 的 API，这使基于 FPGA 的 RNN 加速器的设计变得很不灵活通用。

现有很多研究旨在解决上述提到的问题。这些研究提出了各种各样高能效、高速灵活的 RNN 加速器计算及存储架构，对 RNN 加速器设计的研究推动很大。为了将 RNN 模型映射到 FPGA 上，他们运用数据优化方法在软件层面对 RNN 的数据进行压缩优化，结合 FPGA 硬件层面的并行计算架构、流水线设计、数据存储模式等架构，进一步提高内存访问效率及吞吐量，降低了计算及内存访问能耗，实现了对 RNN 模型高效的加速计算。

本文对现有研究中提到的基于 FPGA 的 RNN 加速器设计技术进行了总结归纳，包括所用到的软件层面的数据优化方法及硬件层面的架构设计，将其进行细致的比较，对其优缺点进行了详细评估。为了进一步提高 RNN 加速器的计算精度，提高计算速度以及通用性，本文提出未来可

基于 FPGA 的深度架构设计、层内层间流水线结合设计及可重构 RNN 加速器设计等几个方面进行研究。

2　背景

自然语言处理正在成为一个跨学科的研究热点，作为自然语言处理的核心，语言建模是一个重要的部分。这其中比较有代表性的模型是 RNN，尤其是在处理语音识别、机器翻译、词性标注解析等任务中，RNN 的性能更为优异。为了取得更好的性能，研究人员提出了各种各样优异的 RNN 变体，这导致模型参数量及计算复杂度增加。因此，越来越多的研究者选择在 FPGA 等硬件平台上设计高效的计算架构，对这些 RNN 模型进行加速运算。

2.1　RNN 及其变体

在自然语言处理领域应用比较广泛的一个典型模型是 RNN [13]。与卷积神经网络（CNN）不同，RNN 是一种通过隐层节点间的连接，来捕捉序列化数据中信息的全连接神经网络，可以对序列化的数据进行学习，并成功应用在语音识别、自然语言处理、情景分析等领域。RNN 可以保存上下文的状态，甚至能够学习任意长度的上下文并存储、学习、表达相关信息。

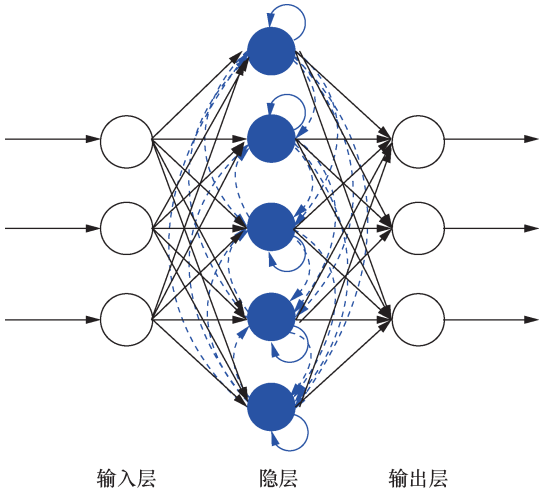


图 1　节点资源分配情况

近年来，RNN 已被广泛用于自然语言处理任务中，并且已经显示出很强的学习和预测序列数据的能力。图 1 展现了标准 RNN 的架构，RNN 的节点循环连接，其当前隐藏状态是下一时刻的输入。状态的更新用式(1)描述。

$$h_t = \sigma(Wx_t + Uh_{t-1} + b) \tag{1}$$

其中，$x_t \in R^M$ 和 $h_t \in R^N$ 分别是在时间 t 时刻的输入和隐层状态。$W \in R^{M\times N}$、$U \in R^{N\times N}$ 及 $b \in R^N$ 是输入向量、隐层状态的权值以及神经元的偏差。σ 是神经元的激活函数，如 sigmoid 或线性单位函数 ReLU。一般的 RNN 架构在训练的过程中存在很严重的梯度消失及梯度爆炸现象，通过增加门控单元可以解决这个问题，相应的预测准确率也由于门控单元的加入而得到提高。

LSTM（long short-term memory）算法最早由 Hochreiter S 等[14]于 1997 年提出。LSTM 最开始主要是为了解决 RNN 梯度消失及梯度爆炸问题而提出的，这个模型有效解决了传统 RNN 不能建立长时间依赖的问题。LSTM 算法扩展了一个基本的 RNN 的隐层单元，增加了 3 个乘法单元（称为输入、输出和忘记门）用来控制信息的流入、流出及保存。LSTM 模型结构如图 2 所示。

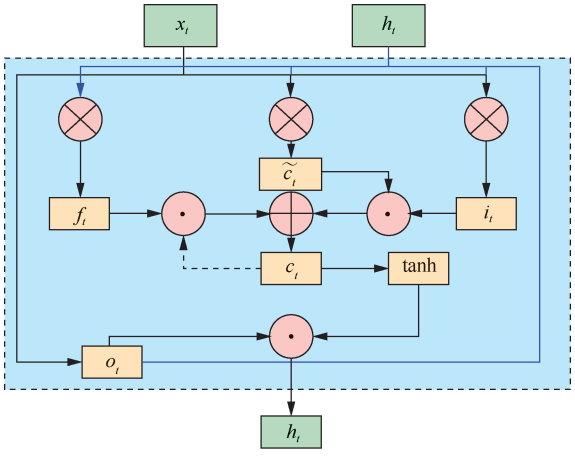


图 2　LSTM 模型

LSTM 中的忘记门 f_t 决定丢弃哪些历史信息，输入门 i_t 决定要更新的有用新信息。通过这

2 个门创建一个新的候选值向量，这个向量会被加入新的细胞状态中存储起来用作下一次计算使用。输出门 o_t 确定最终的输出 h_t。

Cho 等[15]在 2014 年提出的 GRU（gated recurrent unit）对 LSTM 进行了优化，移除了隐层存储单元，降低了模型复杂度，使每个循环单元能够自适应地捕获序列数据之间的依赖关系。GRU 模型（如图 3 所示）具有 2 个门——重置门和更新门，更新门 u_t 控制带入当前状态中新的信息量的程度，更新门的值越大，说明带入新的信息越多。重置门 r_t 控制忽略前一时刻状态信息的程度，重置门的值越小，说明忽略得越多。

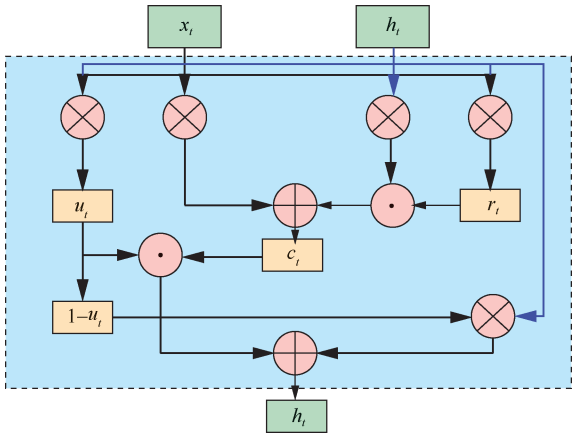


图 3　GRU 模型

与 LSTM 单元类似，GRU 也具有门控单元来调控信息的学习与遗忘。GRU RNN 具有与 LSTM RNN 类似的预测精度，但计算复杂度较低。GRU 与 LSTM 共有的一个特征就是在从 $t-1$ 时刻到 t 时刻，增加了门控单元来控制信息的流入与流出，而传统的 RNN 总是用一个新的向量来代替当前的状态而并没有保留上一时刻的相关信息。

研究者近年来一直对 RNN 模型及其变体进行优化以求取得更高性能的模型。Wojciech 等[16]在 LSTM RNN 的训练过程中，通过对非循环连接部分使用 dropout 算法随机地删除一般的神经元连接，显著地减少了 LSTM 对各种任务的过拟合。

本文在理论上将常用的 RNN 模型参数量做了对比。假设输入向量长度为 M、隐层节点个数为 N，忽略模型的常量，将层数为 1 层的不同 RNN 模型隐层节点计算所需的网络权重参数列在表 1 中。从表 1 中可以明显看出，相比于标准的 RNN 模型，LSTM 及 GRU 模型的参数量较大。这意味着理论上，在计算出一个输出向量之前，LSTM 与 GRU 模型的所需运算量更大，模型更复杂。

表 1　节点信息

变体	输入信息	隐层信息	总参数量
标准 RNN	$M\times N$	$N\times N$	$MN\times N^2$
LSTM	$4M\times N$	$4N\times N$	$4MN\times N^2$
GRU	$3M\times N$	$3N\times N$	$3MN\times N^2$

另外，在实际应用中，Cho 等[15]用典型数据集对 GRU、LSTM 和 RNN 模型的实际性能进行了评估。他们发现 LSTM 及 GRU 模型在所评估的实验数据集上性能优于传统的 RNN 模型。在 polyphonic 和 Ubisoft A 数据集中 GRU 的性能最好；在 Ubisoft B 数据集中 LSTM 模型性能更为优异。LSTM 及 RNN 在不同的数据集中各有优势，在实际应用中需要根据具体的数据集来选择更为合适的 RNN 模型。

2.2　基于 FPGA 的 RNN 加速器

随着 RNN 模型设计得越来越复杂，其训练及推理中的矩阵运算量越来越大，这导致 RNN 在 CPU 上的运算变得不高效。FPGA 由于其灵活的可配置性、低能耗及并行计算等特性，很适合用来对 RNN 复杂的矩阵运算进行加速。

近年来，研究人员基于 FPGA 开发了很多针对 RNN 的优化算法及架构，设计了很多高性能的 RNN 加速器。文献[17-19]使用 HLS 将 RNN

映射到 FPGA 上来简化设计；Li 等[20]提出的 Independently RNN 算法将隐层节点变得独立，IndRNN 可以构造成深层神经网络；文献[21-22]设计了简单的可配置的 RNN 阵列及精度计算机制，提高了 RNN 加速器的通用性。但这些研究涉及的范围比较窄，在实际研究中并不多，本节主要介绍典型的基于 FPGA 的 RNN 硬件加速架构。

对于一个典型的基于 FPGA 的 RNN 加速器，其系统架构如图 4 所示。RNN 加速器主要包括 CPU 控制部分及 FPGA 逻辑运算部分。FPGA 芯片通常通过一个 PCIe/AXI 接口与 CPU 部分相连。FPGA 逻辑运算部分主要包含 RNN 加速器模块、数据控制模块及存储模块。具体地，数据控制模块主要负责数据的读取与流出；RNN 加速器模块包括阵列计算、激活函数查找表等单元，主要负责对 RNN 模型进行加速运算，包括矩阵向量乘法计算、激活函数计算、向量乘法计算等。存储模块主要用来临时存储计算的中间结果及计算所需的网络权重参数、输入向量等。

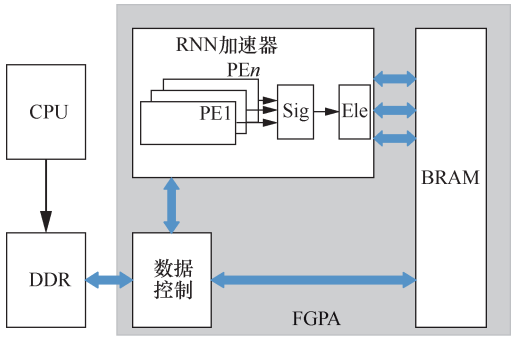


图 4　典型的 RNN 加速器架构

通过分析目标算法的结构，将 RNN 算法中的矩阵向量乘法运算映射至 FPGA 上进行并行计算，进一步结合流水线设计，可提高 RNN 加速器的吞吐量，提高 RNN 的计算速度；通过对数据进行预处理并设计高效数据存储模式，可降低 RNN 的能耗。FPGA 开发人员可以在硬件上实现逻辑运算的部分，在 CPU 上实现对整个加速器必要的配置与控制，以获得高效的运算。

3　性能评估指标

在 FPGA 上对 RNN 进行加速运算可取得更高的计算速度和更低的能耗。通过对现有 RNN 加速器进行归纳总结，本文发现其中用到的硬件加速设计技术包括数据量化、数据压缩、阵列计算、流水线设计等，也是针对 RNN 中的计算速度及能耗两方面进行优化。具体体现在硬件上，概况来说，衡量这些 RNN 加速器性能的指标有吞吐量、功率效率两方面。

在理想情况下，加速器的峰值计算性能可以通过每个时钟周期可执行的计算（乘法或者加法）次数来衡量。对于 RNN 这种大型的神经网络来说，大量的计算资源及能量消耗在矩阵运算过程中，所以一个 RNN 加速器的性能可以由矩阵运算部分的性能来衡量。一个矩阵运算部分的吞吐量可由式(2)[23]决定。其中，P 为处理单元的数量，f 为时钟频率。

$$PERF_{\text{perk}} = 2P^2 f \tag{2}$$

由式(2)可以看出，处理单元的数量和时钟频率都可以影响 RNN 加速器的吞吐量。对于一个确定的 FPGA 芯片来说，片上可利用的逻辑计算资源有限，因此，对片上资源的利用率越高，意味着可综合出的参与计算的处理单元的个数越多，能达到的峰值计算性能就越高。另外，通过提高 RNN 加速器的时钟频率来增加计算单元每秒可执行计算数的方式，需要综合考虑系统及硬件平台后设定。在实际设计中，为了提高吞吐量，大多数设计的加速器会采用流水线架构[3,24,25]，这样的低延迟 RNN 加速器可应用在更多的实时处理设备中。

能源效率是衡量 RNN 加速器单位能耗成本内实际性能的一个指标。能源效率的计算公式如

下所示。

$$Energy\_Efficiency = \frac{Operation_{\text{total}}}{E_{\text{total}}} \tag{3}$$

$$E_{\text{total}} = E_{\text{operation}} + E_{\text{RAM}} + E_{\text{other}} \tag{4}$$

式(4)中第一项代表总的计算能耗，第二项表示总的内存访问能耗，第三项表示其他能耗。提高神经网络能效的一个方法是减少 RNN 加速器的总能耗 E_{total}，对 RNN 来说，前两项能耗占总能耗的大多数，因此对于一个确定的神经网络，降低计算能耗或者降低内存访问的能耗等可以提高 RNN 加速器的能源效率。

4 RNN 加速器架构设计

高效的计算架构及存储访问系统往往能够提高 RNN 加速器的计算性能。为了提高系统的吞吐量，降低系统的计算及内存访问能耗，加速器的架构设计显得十分重要。本节主要在硬件层面上介绍基于 FPGA 的 RNN 硬件加速器的设计技术，并将这些技术分为 3 个方面：阵列处理单元设计、流水线设计及存储系统设计。

4.1 阵列处理单元设计

阵列处理单元设计可以直接影响到加速器的吞吐量，FPGA 片上资源的利用率及处理单元的时钟频率都是影响阵列处理单元设计的直接因素。由于 FPGA 逻辑及存储资源有限，所以设计的处理单元越多，对加速器的性能提升越大。在资源有限的情况下，一个很好的方法就是降低处理单元的大小。第 3 节提到的低位宽数据量化技术就是一种能直接减小处理单元大小、降低处理单元对片上资源消耗的方法。合理地把这些处理单元设计成阵列进行高效的并行运算是提高 RNN 加速器数据吞吐量的关键。

Guan 等[24]提出了一个基于 FPGA 的 LSTM RNN 加速器。他们将 32 bit 浮点计算处理单元设计成阵列，并采用线性函数逼近法近似激活函数 sigmoid，优化了计算性能及数据通信的需求，与 CPU 相比获得了 20.18 倍的计算速度提升。文献[25-27]采用 16 bit 定点处理单元阵列的方式来加速 LSTM RNN 的推理运算。实验表明，16 bit 定点运算的处理单元并没有降低模型的性能，并且能取得更高的计算速度和更低的能源消耗。

Lee 等[28]将训练好的 LSTM RNN 模型的权重参数使用 6 bit 定点量化，进一步降低了阵列处理单元的大小，实现了低延迟和高吞吐量的矩阵向量运算。他们以仅 9.24 W 的功耗将计算速度提升 4.12 倍，与高端的 GPU 设计相比，功率效率高出 10 倍左右。

4.2 流水线设计

在 RNN 中，全连接层的计算和存储需求在整个神经网络中占据了绝大部分，所以有必要将全连接层的计算进行仔细拆分，在 FPGA 上展开并设计成流水线以提高计算效率。

Peng 等[11]将 RNN 推理计算拆分成 $Wx(t)$、$Uh(t-1)$、sigmoid、$c(t)$、$o(t)$ 几个部分并设计成流水运算模式。Wang 等[29]给出了层间流水线的设计方式。他们利用隐层计算时间相同的特性将多层 LSTM 的计算进行拆分并设计成流水线。当前隐层计算完毕后，将计算结果送入层间 Buffer 中缓存起来，然后从上一层 Buffer 中读取新的数据进行计算，经过多个时钟周期就可以实现连续不断地对输入向量的流水运算，提高了神经网络的吞吐量，加速了 RNN 的推理运算。

4.3 存储系统设计

由于 FPGA 的片上 BRAM 与大型神经网络模型的参数量相比太小，因此，常见的设计往往使用 DDR 来存储 RNN 的权重参数，但这样会造成频繁的内存数据访问，给 RNN 加速器带来额外的内存访问能耗，降低了加速器的能源效率。

使用第 3 节所介绍的模型压缩算法将模型的参数量进行大规模压缩后，可以将模型的权重参数存储到片上存储器，简化 RNN 加速器内存访问系统的设计，并降低额外的内存访问能耗；在计算时仅仅需要从 DDR 中读出输入向量送入 RNN 加速器中与片上存储的权重参数进行乘累加运算，提高了加速器的计算速度，降低了计算延迟。

文献[25-28]所设计的 RNN 加速器充分利用了数据压缩算法，将 RNN 权重参数量化压缩后存储到 FPGA 片上的存储资源中。实验表明，他们所设计的加速器相比在 DDR 中存储权重参数的 RNN 加速器，具有更高的能源效率。在对能耗比较敏感的移动端或者嵌入式应用中，可结合相应的数据压缩算法将权值存储在片上存储器中。

5　RNN 加速器优化算法

在 FPGA 上设计高效的计算架构及存储访问系统能够提高 RNN 加速器的计算性能，这些设计往往依赖于 RNN 加速器所需的数据格式及数据量。近年来，针对 RNN 数据结构及数据量的优化算法越来越多，这些研究基于以上两点优化 RNN 加速器的计算模式、降低 RNN 模型的参数数量，实现了更加高效的 RNN 加速器系统。

已有研究[30-32]将 RNN 模型映射到 FPGA 上，并通过并行运算取得了不错的效果，但没有任何对模型的优化方法。一方面，针对 RNN 模型的优化算法往往能够设计出高效且更快速的神经网络加速器；另一方面，大型、复杂的 RNN 模型通常能够取得更高的性能，这意味着研究者要根据实际情况平衡模型性能与硬件的计算速度及能耗。

现有一些基于 RNN 数据优化算法的研究可以很好地平衡模型性能与硬件性能。针对数据结构，使用低位宽定点数据量化等技术对激活函数参数或权重参数进行量化，简化 RNN 模型的计算以提高计算速度、降低计算能耗；针对 RNN 模型的数据量，对 RNN 进行裁剪，应用稀疏矩阵乘法、矩阵参数循环分块、矩阵分解等方法，旨在降低 RNN 加速器的计算及存储的复杂度，实现高效的计算、降低内存访问能耗。接下来，本节将从数据格式优化及数据量压缩算法 2 个层间，详细介绍这些现有 RNN 加速器的数据优化算法。

5.1　数据结构优化技术

RNN 模型的参数量一般比较大，导致在对 RNN 模型进行推理计算时耗时比较多；另外，RNN 模型的激活函数及权重参数一般是浮点数，但文献[28]研究表明在 RNN 的推理运算中低位宽的定点乘法比浮点乘法更高效，且选择适当的低位宽参数几乎不会对 RNN 的性能造成损失。

近些年来，有不少研究针对 RNN 的数据结构进行优化。用 FPGA 对 RNN 进行加速运算最常用的一种方法就是针对模型数据进行低位宽量化。低位宽数据量化技术能够降低 RNN 硬件加速器的内存访问需求及存储需求，减少内存访问、硬件运算的能耗和片上资源的消耗。数据量化技术总体来说分为线性数据量化及非线性数据量化。

5.1.1　线性数据量化

线性数据量化一个很直接的方法就是将激活函数及全连接层的权重参数量化成低位宽定点数。线性数据量化技术比较简便，相应地也会造成数据精度损失并增加计算结果的错误率，且线性量化技术在数据位宽较低时性能损失较大。

Lee 等[28]使用 6 bit 定点量化训练好的 LSTM RNN 模型的权重参数，并存储在 FPGA 片上的存储资源中以减少内存访问能耗。与浮点参数运算相比，该方法仅仅以增加 1.5%单词错误率的微小成本，将存储空间需求压缩了 75%；与高端的

GPU 系统相比，功率效率提高了 10 倍左右。Joas 等[33]用 18 bit 的定点数将 LSTM RNN 模型权重参数进行了量化，结合大量的并行运算设计，他们所设计的 LSTM RNN 加速器的计算速度相比 CPU 提升了 251 倍，与 Chang 等[26]设计的 RNN 加速器相比提升了 14 倍。

5.1.2　非线性数据量化

非线性数据量化是将权重参数通过函数映射成相应的二进制码，再通过查找表的方式用参数映射后的二进制码进行计算。非线性量化的目的是将权重等参数进行压缩，但可以将 RNN 模型的数据压缩到更低的位宽，相应的非线性量化技术应用通用性较差，需根据参数数据情况而定。

Skin 等[34]研究分析了 LSTM 模型参数，对参数量化不敏感的层使用 2 bit，对参数量化敏感的层使用 3~5 bit 定点数据量化后进行再训练，在模型性能不变的前提下将参数的数量压缩了 90%，同时存储空间的需求仅仅为浮点数据模型的 16.75%。

5.2　数据量压缩算法

为了取得更高的模型性能，研究者往往把模型设计得特别大，造成对硬件设备要求提高。另外，大规模 RNN 的训练非常不高效，并且消耗大量的存储资源，因此近些年来针对 RNN 模型参数量的数据压缩算法的应用越来越多，包括网络剪枝算法、稀疏矩阵乘法、Delta RNN 算法、低秩矩阵近似算法、循环分块矩阵算法。这些算法可以降低 RNN 的参数量，从而降低加速器的计算及存储需求，使 RNN 加速器更加高效的运作。

5.2.1　网络剪枝算法

网络剪枝算法就是很直接地将参数矩阵中的冗余参数置零后再将 RNN 剪枝。对已训练的 RNN 参数进行观察可以知道，RNN 权重参数中往往包含很多冗余的小参数或者零元素，这些参数对最终的计算结果影响不大。因此可以对训练数据分析，将这些冗余参数去除，提高模型的计算速度。

Han 等[35]将 RNN 进行剪枝，减少了神经网络存储及计算需求。整个工作分 3 步进行：第一步训练 LSTM RNN 并寻找冗余的连接；第二步将冗余网络连接剪枝；第三步将剪枝后的网络进行再训练对权重进行微调。压缩后的 LSTM RNN 模型参数仅为原始 LSTM 的 7.69%～11.1%。

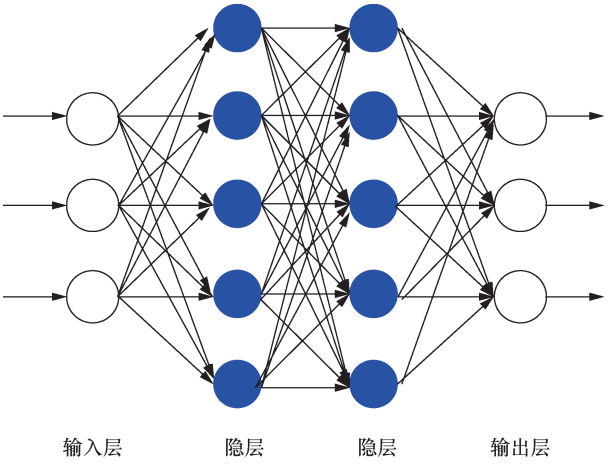


图 5　网络剪枝前

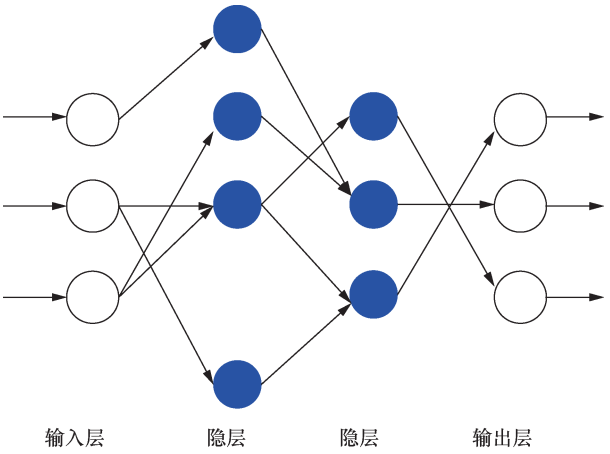


图 6　网络剪枝后

Han 等[36]提出一种负载均衡感知的剪枝方法，通过设置一个冗余门限将权值小于门限的冗余网络连接去除，将 LSTM RNN 模型的大小压缩了 95%，几乎没有影响模型的预测准确率。网络

剪枝方法中冗余门限的设置需要非常慎重，否则会造成模型性能的损失。

5.2.2　稀疏矩阵乘法

对大多数高性能的 RNN 模型计算而言，使用稀疏矩阵乘法来替代稠密矩阵乘法运算是一个很常用的优化算法。

Ali 等[37]使用 CCBV 紧凑型列向量稀疏索引方法，使用一维数组将稀疏矩阵的所有非零数据映射成列参数形式，使参数量降低为原始模型的 18.87%～33.33%，并减少了 40%～66.7%的计算延迟，实现了低延迟的高效流水运算。

稀疏矩阵乘法可以提高 RNN 模型的性能，但非零数据的存储使内存数据访问不规则。Jeremy 等[38]为了解决这个问题，提出了 CISR 数据编码算法。他们将每个时钟周期从内存接收到的总宽度的数据划分成不同的槽，每个槽中的数据直接送到相应的数据通道中进行运算。这种直连接的方式降低了稀疏矩阵乘法硬件实现的复杂度。

5.2.3　Delta RNN 算法

如式(5)和式(6)所示，与稀疏矩阵向量乘法及网络剪枝不同，Delta 算法不再关注参数矩阵的冗余，转而关注输入向量的冗余。

$$\Delta \boldsymbol{x}_t = \boldsymbol{x}_t - \boldsymbol{x}_{t-1} \tag{5}$$

$$h_t = \sigma(W\Delta \boldsymbol{x}_t + U\Delta h_{t-1} + b) \tag{6}$$

Neil 等[39]将当前时刻的输入向量 $\boldsymbol{x}(t)$ 与前一时刻的输入向量 $x(t-1)$ 的变化值 $\Delta x(t)$ 作为新的输入向量送到 GRU 中，并设定了一个门限 θ，$\Delta x(t)$ 中小于门限 θ 的参数被置为 0。在 TIDIGITS 数据库上的实验表明，通过引用 Delta RNN 算法降低了 GRU 模型的参数量，GRU 模型可仅以 $\frac{1}{9}$ 的计算能耗完成运算，且不会造成模型精度的损失。

Gao 等[40]在 FPGA 上将 Delta RNN 算法应用 GRU 中进行加速运算。他们设计了两层 GRU 网络并将参数存储在片上以减少内存的访问，最终获得了 1.2 Top/s 的高吞吐量和 164 Gop/(s·W)$^{-1}$ 的能源效率，与传统应用稀疏矩阵乘法的 RNN 模型相比计算速度提升了 5.6 倍。

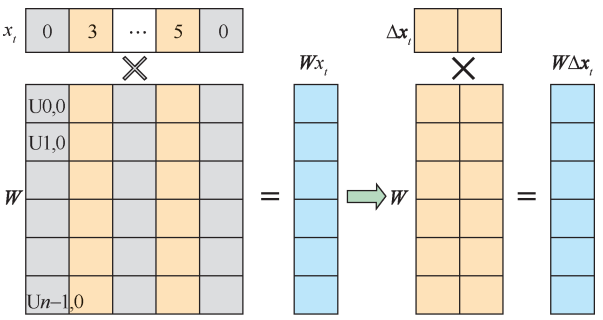


图 7　Delta RNN 算法流程

5.2.4　低秩矩阵近似算法

低秩矩阵近似的方法可以避免稀疏矩阵及网络剪枝方法引起的内存读取不规则问题。低秩矩阵近似算法核心思想是将一个 $m \times n$ 的矩阵 $\boldsymbol{W}$ 用 2 个相乘矩阵 $\boldsymbol{A}_{m\times r}\boldsymbol{B}_{r\times n}$ 来近似。当秩足够小时，矩阵 $\boldsymbol{A}_{m\times r}$ 和 $\boldsymbol{B}_{r\times n}$ 的总参数量比矩阵 $\boldsymbol{W}_{m\times n}$ 的参数量要少得多。

Kingsbury 等[41]将低秩矩阵近似算法应用到 DBN 底层网络中，将 DBN 网络的权重进行压缩；Xue 等[42]在 DNN 中使用低秩矩阵近似算法对权重矩阵进行重新构造，很大限度上压缩了 DNN 模型的大小且几乎没有造成模型性能的损失；Qiu 等[43]对 CNN 全连接层的权重参数使用低秩分解近似的方法将参数压缩，取得了不错的效果。

低秩矩阵近似算法在 RNN 硬件加速方面颇有成效。文献[44-45]使用低秩矩阵近似和参数共享算法优化了 RNN 及 LSTM 的权重参数。他们分析了循环架构中的冗余，在不影响模型性能的前提下，对 RNN 权重矩阵使用低秩矩阵近似算法将权重矩阵进行替代，成功将 RNN LSTM 模型的参数量进行了压缩，提高了 RNN 加速器的性能。

5.2.5　循环分块矩阵算法

循环分块矩阵算法通过压缩 RNN 权重矩阵的参数量以降低内存占用及访问时间。如式(7)所

示，循环分块矩阵算法将权重矩阵进行分块，使用每个块矩阵中的一行或一列对块矩阵参数进行编码表示。循环分块矩阵是 Toeplize 矩阵中的一种特殊情况，与常规的矩阵向量乘法相比，当使用循环分块矩阵算法时，可以将计算复杂度 $O(n^2)$ 降低到 $O(n)$。

$$\boldsymbol{W}=\begin{bmatrix} w_1 & w_2 & \cdots & w_n \\ w_2 & w_3 & \cdots & w_1 \\ \vdots & \vdots & \vdots & \vdots \\ w_n & w_1 & \cdots & w_{n-1} \end{bmatrix} \tag{7}$$

Li 等[46]在 RNN 模型中使用循环分块矩阵算法对 RNN 模型的参数进行了编码压缩，将分块矩阵的计算使用 FFT 和 IFFT 进行优化。实验表明，压缩后的 RNN 模型在 FPGA 上的加速运算几乎没有造成模型性能的损失，在能源效率方面相比文献[36]取得了 35.7 倍的提升。

Wang 等[47]使用循环分块矩阵算法，并结合非线性激活函数和数据量化优化算法，压缩了 LSTM 模型 95%的参数量，降低了内存资源的使用。Wang 等[29]提出了一个基于 FPGA 的 C-LSTM 框架自动优化和实现 LSTM RNN 的各种变体。C-LSTM 通过循环分块矩阵算法将 LSTM 模型的参数进行压缩，并使用 FFT 算法对矩阵运算进行了优化，解决了传统压缩算法的不规则计算及内存读取问题。

6 总结与评估

计算架构、存储访问系统及针对 RNN 的数据结构及数据量的优化算法，可以提升 RNN 加速器的计算速度及能效。尽管这些方法各有不同，但都是针对 RNN 计算及能效进行的优化。为了清晰地对比各方法的优劣，本节按照硬件架构设计、数据格式优化、数据压缩优化 3 个方面对第 4 节和第 5 节所介绍的方法进行总结评估。

6.1 硬件架构设计

硬件架构的设计往往依赖于数据处理后的数据结构形式及数据量，并且所针对的 RNN 模型中的问题有所不同，本文综合参考文献中用的硬件优化技术，对已有的架构设计技术进行了总结评估。

1) 矩阵阵列计算单元主要针对 RNN 中的矩阵向量计算模块进行加速。通过并行计算矩阵的多行或者多列数据，以提高 RNN 加速器的吞吐量。但阵列计算单元需根据数据格式，设计相应位宽的浮点或者定点乘法器、加法器。

2) 流水线设计主要针对 RNN 模型的复杂运算进行加速。将 RNN 计算部分拆分并设计成流水计算，以降低 RNN 加速器的计算延迟。

3) 数据存储系统设计主要针对 RNN 的参数，结合数据压缩算法，降低 RNN 计算及内存访问能耗。在实际应用中，可根据 RNN 参数规模选择片上存储模型参数，以进一步提高计算速度。

总体来说，RNN 硬件架构设计，如阵列计算、流水线计算、存储系统设计技术，可进一步降低 RNN 加速器的计算延迟及能耗，提高 RNN 加速器的计算速度及能效。在实际应用中，可根据实际模型数据情况选择合适的硬件架构设计或合理地搭配使用。

6.2 数据格式优化技术

线性量化和非线性量化技术都可以将 RNN 的参数进行压缩，加速处理单元的计算速度。为了方便对比各方法的优劣，本文将参考文献中用到的数据量化技术及相应的加速器性能列在表 1 中。

从表 1 中可以简要看出，非线性量化技术对 RNN 模型数据的压缩性能更好，但无法判断在提升模型性能上哪种数据量化方法更好。

尽管非线性量化对模型的压缩效果更好，但往往受限于特定的参数数据。相对而言，线性量

表 1　　节点信息

文献	模型	对比平台	量化方法	数据压缩倍数	计算速度提升
文献[28]	LSTM	NVIDIA GeForce Titan X	Fixed-point 6	4 倍	4.12 倍
文献[33]	LSTM	CORE i7-3770k	Fixed-point 17	—	251 倍
文献[26]	LSTM	ARM Cortex-A9 CPU	Fixed-point 16	—	21 倍
文献[34]	LSTM	—	非线性量化	5~9 倍	—

化技术在神经网络硬件加速领域更加简便通用，并可结合相应的数据压缩算法进一步优化 RNN 加速器。

6.3　数据压缩算法

针对 RNN 模型数据的压缩算法种类比较多，但没有一个简要对比来评估这些算法的优劣区别。本文将一些典型的数据压缩算法做了简要对比，所有的压缩算法及性能提升来自文献[35-40, 44, 46-47]。尽管这些压缩算法针对的数据集有所不同，这样简单的对比有失偏颇，但图 8 仍能展示出不同压缩算法对模型数据的压缩性能。

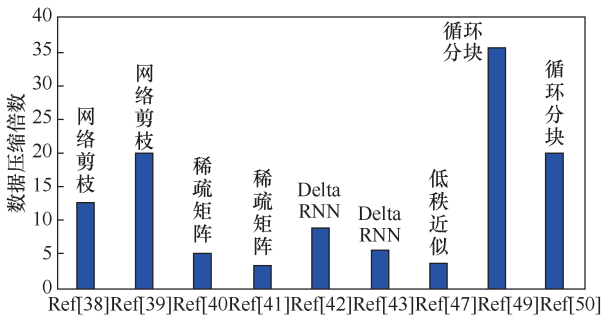


图 8　压缩算法对比

从图 8 中可以看出，循环分块矩阵能够对模型进行更好的压缩。使用循环分块矩阵算法可以将计算复杂度 $O(n^2)$ 降低到 $O(n)$。

循环分块矩阵受限于参数矩阵的形状，并不通用；网络剪枝方法同样可以对模型进行很好的压缩，但剪枝后的网络往往会造成模型预测精度上的损失；稀疏矩阵、DeltaRNN 算法在压缩模型数据、提高模型的计算速度及能源效率的同时会带来内存数据的不规则访问，给存储系统的设计带来麻烦；低秩近似矩阵可以避免内存数据不规则访问问题，用 2 个低秩矩阵相乘来近似参数矩阵可以降低模型的参数量，但秩的选取比较困难。

总体来说，以上这些算法一方面可将 RNN 进行压缩，减少 RNN 的数据量，降低内存数据访问能耗；另一方面可降低计算次数，从而提高系统的计算速度，使 RNN 加速器更加高效地工作。这些数据压缩算法各有优势缺点，在实际应用中需要针对不同的数据库及实验平台进行选择。

7　结束语

现有研究针对数据层面的数据量化、数据压缩算法及硬件层面的计算及存储架构设计技术，使 RNN 的实际性能在 FPGA 上得到了显著提升，但还有几个问题亟待解决。

1) 现有 RNN 加速器的深度一般为 1-2 层，未来是否可通过构建合理的架构，在 FPGA 有限的资源上搭建深层 RNN 以提高模型的计算精度有待于进一步研究。

2) 现有 RNN 加速器流水线技术仅应用在隐层节点的计算或者层间计算中，未来将隐层内流水线推理运算与层间流水线设计结合到一块，模型的性能是否会提升，有待于在 FPGA 上进行进一步验证。

3) 现有 RNN 加速器都是针对不同 RNN 模型的专用加速器，这导致其应用场景十分受限，未来是否可在 FPGA 上实现加速器模型的动态可配

置，即可以根据需求及应用场景动态配置计算单元构建 LSTM、GRU 等 RNN 变体，来自适应处理不同的自然语言处理任务有待于进一步研究。

4) 现有 RNN 加速器优化算法发展比较快、种类比较多，但至今没有公正客观的硬件性能对比实验来评估这些算法的优劣。未来是否可在其他设计方法相同的前提下，进行实验来评估这些优化算法硬件实现的设计复杂度及对 RNN 性能的提升，有待于进一步研究。

参考文献：

[1] HAO Y, QUIGLEY S. The implementation of a deep recurrent neural network language model on a Xilinx FPGA[J]. arXiv Preprint arXiv:1710.10296, 2017.

[2] SAK H, SENIOR A, BEAUFAYS F. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition[J]. arXiv Preprint arXiv:1402.1128, 2014.

[3] MIKOLOV T, KARAFIAT M, BURGET L, et al. Recurrent neural network based language model[C]//Eleventh Annual Conference of the International Speech Communication Association. 2010.

[4] CHO K, VAN -MERRIENBOER B, GULCEHRE C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv Preprint arXiv:1406.1078, 2014.

[5] GRAVES A, MOHAMED A, HINTON G. Speech recognition with deep recurrent neural networks[C]//2013 IEEE International Conference on. Acoustics, speech and signal processing (icassp). 2013: 6645-6649.

[6] BYEONW, BREUEL T M, RAUE F, et al. Scene labeling with LSTM recurrent neural networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 3547-3555.

[7] ZHANG Y, WANG C, GONG L, et al. A power-efficient accelerator based on FPGA for LSTM network[C]//2017 IEEE International Conference on Cluster Computing (CLUSTER). 2017: 629-630.

[8] GUO K, ZENG S, YU J, et al. A survey of FPGA-based neural network accelerator[J]. arXiv preprint arXiv:1712.08934, 2017.

[9] HWANG K, SUNG W. Single stream parallelization of generalized LSTM-like RNNs on a GPU[J]. arXiv Preprint arXiv:1503.02852, 2015.

[10] ABADI M, AGARWAL A, BARHAM P, et al. Tensorflow: large-scale machine learning on heterogeneous distributed systems[J]. arXiv preprint arXiv:1603.04467, 2016.

[11] OUYANG P, YIN S, WEI S. A fast and power efficient architecture to parallelize LSTM based RNN for cognitive intelligence applications[C]//The 54th Annual Design Automation Conference 2017. ACM, 2017: 63.

[12] NURVITADHI E, SIM J, SHEFFIELD D, et al. Accelerating recurrent neural networks in analytics servers: comparison of FPGA, CPU, GPU, and ASIC[C]//2016 26th International Conference on Field Programmable Logic and Applications (FPL).2016: 1-4.

[13] HOPFIELD J J. Neural networks and physical systems with emergent collective computational abilities[J]. Proceedings of the National Academy of Sciences, 1982, 79(8): 2554-2558.

[14] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[15] CHUNG J, GULCEHRE C, CHO K H, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. arXiv Preprint arXiv:1412.3555, 2014.

[16] ZAREMBA W, SUTSKEVER I, VINYALS O. Recurrent neural network regularization[J]. arXiv Preprint arXiv:1409.2329, 2014.

[17] RYBALKIN V, PAPPALARDO A, GHAFFAR M M, et al. FINN-L: library extensions and design trade-off analysis for variable precision LSTM networks on FPGAs[J]. arXiv Preprint arXiv:1807.04093, 2018.

[18] RYBALKIN V, WEHN N, YOUSEFI M R, et al. Hardware architecture of bidirectional long short-term memory neural network for optical character recognition[C]//The Conference on Design, Automation & Test in Europe. European Design and Automation Association. 2017: 1394-1399.

[19] GUAN Y, LIANG H, XU N, et al. FP-DNN: an automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates[C]//2017 IEEE 25th Annual International Symposium on Field-programmable Custom Computing Machines (FCCM). 2017: 152-159.

[20] LI S, LI W, COOK C, et al. Independently recurrent neural network (indrnn): building a longer and deeper RNN[C]//IEEE Conference on Computer Vision and Pattern Recognition. 2018: 5457-5466.

[21] HAJDUK Z. Reconfigurable FPGA implementation of neural networks[J]. Neuro Computing, 2018, 308: 227-234.

[22] LIU B, DONG W, XU T, et al. E-ERA: an energy-efficient reconfigurable architecture for RNN using dynamically adaptive approximate computing[J]. IEICE Electronics Express, 2017, 14(15): 20170637-20170637.

[23] 宋翔, 周凡, 陈耀武, 等. 基于 FPGA 的实时双精度浮点矩阵乘法器设计[J]. 浙江大学学报(工学版), 2008, 42(9):1611-1615.
SONG X, ZHOU F, CHEN Y W, et al. Design of real time double precision floating point matrix multiplier based on FPGA [J]. Journal of ZheJiang University,2008,42(9):1611-1615.

[24] GUAN Y, YUAN Z, SUN G, et al. FPGA-based accelerator for long short-term memory recurrent neural networks[C]//IEEE Design Automation Conference (ASP-DAC). 2017: 629-634.

[25] CHANG A X M, CULURCIELLO E. Hardware accelerators for recurrent neural networks on FPGA[C]//2017 IEEE International Symposium on. Circuits and Systems (ISCAS). 2017: 1-4.

[26] CHANG A X M, MARTINI B, CULURCIELLO E. Recurrent neural networks hardware implementation on FPGA[J]. arXiv Preprint arXiv:1511.05552, 2015.

[27] LI S, WU C, LI H, et al. Fpga acceleration of recurrent neural network based language model[C]//2015 IEEE 23rd Annual International Symposium on Field-programmable Custom Computing Machines. IEEE, 2015: 111-118.

[28] LEE M, HWANG K, PARK J, et al. FPGA-based low-power

speech recognition with recurrent neural networks[C]//2016 IEEE International Workshop on.Signal Processing Systems (SiPS). 2016: 230-235.

[29] WANG S, LI Z, DING C, et al. C-LSTM: enabling efficient LSTM using structured compression techniques on FPGAs[C]// ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2018: 11-20.

[30] ZHANG Y, WANG C, GONG L, et al. Implementation and optimization of the accelerator based on FPGA hardware for LSTM network[C]//IEEE International Symposium on Parallel and Distributed Processing with Applications and IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC). 2017: 614-621.

[31] LIAO Y, LI H, WANG Z. Based real-time processing architecture for recurrent neural network[C]//International Conference on Intelligent and Interactive Systems and Applications. 2017: 705-709.

[32] SALCIC Z, BERBER S, SECKER P. FPGA prototyping of RNN decoder for convolutional codes[J]. EURASIP Journal on Advances in Signal Processing, 2006, 2006(1): 015640.

[33] FERREIRA J C, FONSECA J. An FPGA implementation of a long short-term memory neural network[C]//2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig). 2016: 1-8.

[34] SHIN S, HWANG K, SUNG W. Fixed-point performance analysis of recurrent neural networks[J]. arXiv Preprint arXiv:1512.01322, 2015.

[35] HAN S, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural network[C]//Advances in Neural Information Processing Systems. 2015: 1135-1143.

[36] HAN S, KANG J, MAO H, et al. Ese: efficient speech recognition engine with sparse LSTM on FPGA[C]//ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2017: 75-84.

[37] ALI S M, SHAOJUN W, NING M, et al. A bandwidth in-sensitive low stall sparse matrix vector multiplication architecture on reconfigurable FPGA platform[C]//13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI). 2017: 171-176.

[38] FOWERS J, OVTCHAROV K, STRAUSS K, et al. A high memory bandwidth FPGA accelerator for sparse matrix-vector multiplication[C]//IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). 2014: 36-43.

[39] NEIL D, LEE J H, DELBRUCK T, et al. Delta networks for optimized recurrent network computation[J]. arXiv Preprint arXiv: 1612.05571, 2016.

[40] GAO C, NEIL D, CEOLINI E, et al. DeltaRNN: a power-efficient recurrent neural network accelerator[C]//ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2018: 21-30.

[41] KINGSBURY B E D, SAINATH T N, SINDHWANI V. Low-rank matrix factorization for deep belief network training with high-dimensional output targets[P]. 2016-2-16.

[42] XUE J, LI J, GONG Y. Restructuring of deep neural network acoustic models with singular value decomposition[C]//Interspeech. 2013: 2365-2369.

[43] QIU J, WANG J, YAO S, et al. Going deeper with embedded FPGA platform for convolutional neural network[C]//ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2016: 26-35.

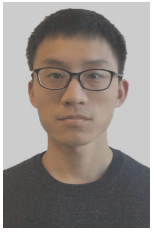
[44] LU Z, SINDHWANI V, SAINATH T N. Learning compact recurrent neural networks[J]. arXiv Preprint arXiv:1604.02594, 2016.

[45] RIZAKIS M, VENIERIS S I, KOURIS A, et al. Approximate FPGA-based LSTM under computation time constraints[J]. arXiv Preprint arXiv:1801.02190, 2018.

[46] LI Z, WANG S, DING C, et al. Efficient recurrent neural networks using structured matrices in FPGA[J]. arXiv Preprint arXiv:1803.07661, 2018.

[47] WANG Z, LIN J, WANG Z. Accelerating recurrent neural networks: a memory-efficient approach[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2017, 25(10): 2763-2775.

[作者简介]



高琛（1994− ），男，河南郑州人，国家数字交换系统工程技术研究中心硕士生，主要研究方向为 FPGA 硬件加速及人工智能芯片。

张帆（1981− ），男，河南郑州人，博士，国家数字交换系统工程技术研究中心副研究员、硕士生导师，主要研究方向为主动防御、芯片设计技术、高性能计算。