# An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling

# 通用卷积网络和循环网络用于序列建模的实证评估

Shaojie Bai J. Zico Kolter Vladlen Koltun

邵杰 白 J. Zico 科尔特 Vladlen 科尔图

# Abstract

# 摘要

For most deep learning practitioners, sequence modeling is synonymous with recurrent networks. Yet recent results indicate that convolutional architectures can outperform recurrent networks on tasks such as audio synthesis and machine translation. Given a new sequence modeling task or dataset, which architecture should one use? We conduct a systematic evaluation of generic convolutional and recurrent architectures for sequence modeling. The models are evaluated across a broad range of standard tasks that are commonly used to benchmark recurrent networks. Our results indicate that a simple convolutional architecture outperforms canonical recurrent networks such as LSTMs across a diverse range of tasks and datasets, while demonstrating longer effective memory. We conclude that the common association between sequence modeling and recurrent networks should be reconsidered, and convolutional networks should be regarded as a natural starting point for sequence modeling tasks. To assist related work, we have made code available athttp://github.com/locuslab/TCN.

对于大多数深度学习从业者来说，序列建模与循环网络是同义的。然而，最近的研究结果表明，在音频合成和机器翻译等任务上，卷积架构可以超越循环网络。面对一个新的序列建模任务或数据集，应该使用哪种架构呢？我们对通用的卷积和循环架构进行了系统的评估，用于序列建模。这些模型在一系列常用于评估循环网络的标准化任务上进行了评估。我们的结果表明，一个简单的卷积架构在多种不同的任务和数据集上超过了典型的循环网络，如LSTMs，同时表现出更长的有效记忆。我们得出结论，应该重新考虑序列建模与循环网络之间的常见联系，卷积网络应该被视为序列建模任务的天然起点。为了协助相关工作，我们已经将代码公开在http://github.com/locuslab/TCN。

# 1. Introduction

# 1. 引言

Deep learning practitioners commonly regard recurrent architectures as the default starting point for sequence modeling tasks. The sequence modeling chapter in the canonical textbook on deep learning is titled "Sequence Modeling: Recurrent and Recursive Nets" (Goodfellow et al., 2016), capturing the common association of sequence modeling and recurrent architectures. A well-regarded recent online course on "Sequence Models" focuses exclusively on recurrent architectures .

深度学习从业者通常将循环架构视为序列建模任务的默认起点。深度学习标准教科书中的序列建模章节标题为“序列建模：循环和递归网络”（Goodfellow等人，2016），反映了序列建模与循环架构之间的常见联系。一个备受推崇的最近在线课程“序列模型”专门讲述了循环架构 。

On the other hand, recent research indicates that certain convolutional architectures can reach state-of-the-art accuracy in audio synthesis, word-level language modeling, and machine translation (van den Oord et al., 2016; Kalchbrenner et al., 2016; Dauphin et al., 2017; Gehring et al., 2017a;b). This raises the question of whether these successes of convolutional sequence modeling are confined to specific application domains or whether a broader reconsideration of the association between sequence processing and recurrent networks is in order.

另一方面，近期研究指出，某些卷积架构在音频合成、词级语言建模和机器翻译方面可以达到最先进的效果（van den Oord et al., 2016; Kalchbrenner et al., 2016; Dauphin et al., 2017; Gehring et al., 2017a;b）。这引出了一个问题，即卷积序列建模的成功是否仅限于特定的应用领域，或者是否需要对序列处理与循环网络之间的关联进行更广泛的重新考虑。

We address this question by conducting a systematic empirical evaluation of convolutional and recurrent architectures on a broad range of sequence modeling tasks. We specifically target a comprehensive set of tasks that have been repeatedly used to compare the effectiveness of different recurrent network architectures. These tasks include polyphonic music modeling, word- and character-level language modeling, as well as synthetic stress tests that had been deliberately designed and frequently used to benchmark RNNs. Our evaluation is thus set up to compare convolutional and recurrent approaches to sequence modeling on the recurrent networks’ "home turf".

我们通过在广泛的序列建模任务上对卷积和循环架构进行系统的实证评估来回答这个问题。我们特别针对一组用于反复比较不同循环网络架构效果的综合任务集。这些任务包括多声部音乐建模、词级和字符级语言建模，以及为了评估RNN而特意设计并经常使用的合成压力测试。因此，我们的评估旨在在循环网络的“主场”上比较卷积和循环方法对序列建模的效果。

To represent convolutional networks, we describe a generic temporal convolutional network (TCN) architecture that is applied across all tasks. This architecture is informed by recent research, but is deliberately kept simple, combining some of the best practices of modern convolutional architectures. It is compared to canonical recurrent architectures such as LSTMs and GRUs.

为了表示卷积网络，我们描述了一个通用的时序卷积网络（TCN）架构，该架构被应用于所有任务。这个架构借鉴了近期的研究，但故意保持简单，结合了现代卷积架构的一些最佳实践。它与标准的循环架构如LSTMs和GRUs进行了比较。

The results suggest that TCNs convincingly outperform baseline recurrent architectures across a broad range of sequence modeling tasks. This is particularly notable because the tasks include diverse benchmarks that have commonly been used to evaluate recurrent network designs (Chung et al., 2014; Pascanu et al., 2014; Jozefowicz et al., 2015; Zhang et al., 2016). This indicates that the recent successes of convolutional architectures in applications such as audio processing are not confined to these domains.

结果表明，TCN在广泛的序列建模任务上明显优于基线循环架构。这尤其值得注意的是，因为这些任务包括通常用于评估循环网络设计的多种不同的基准（Chung et al., 2014; Pascanu et al., 2014; Jozefowicz et al., 2015; Zhang et al., 2016）。这表明卷积架构在音频处理等应用中的近期成功并不仅限于这些领域。

To further understand these results, we analyze more deeply the memory retention characteristics of recurrent networks. We show that despite the theoretical ability of recurrent architectures to capture infinitely long history, TCNs exhibit substantially longer memory, and are thus more suitable for domains where a long history is required.

为了进一步理解这些结果，我们深入分析了循环网络的记忆保持特性。我们表明，尽管循环架构在理论上能够捕捉无限长的历史，但TCN表现出更长的记忆，因此更适合需要长历史记录的领域。

Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA Intel Labs, Santa Clara, CA, USA. Correspondence to: Shaojie Bai shaojieb@cs.cmu.edu>, J. Zico Kolter <zkolter@cs.cmu.edu>, Vladlen Koltun <vkoltun@gmail.edu>.

卡内基梅隆大学，机器学习系，匹兹堡，PA，美国 卡内基梅隆大学，计算机科学系，匹兹堡，PA，美国 英特尔实验室，圣克拉拉，CA，美国。通讯作者：Shaojie Bai shaojieb@cs.cmu.edu>, J. Zico Kolter <zkolter@cs.cmu.edu>, Vladlen Koltun <vkoltun@gmail.edu>。

To our knowledge, the presented study is the most extensive systematic comparison of convolutional and recurrent architectures on sequence modeling tasks. The results suggest that the common association between sequence modeling and recurrent networks should be reconsidered. The TCN architecture appears not only more accurate than canonical recurrent networks such as LSTMs and GRUs, but also simpler and clearer. It may therefore be a more appropriate starting point in the application of deep networks to sequences.

据我们所知，本研究是迄今为止对卷积网络和循环网络在序列建模任务上最全面的系统性比较。结果表明，应当重新考虑序列建模与循环网络之间的常见关联。TCN架构不仅比标准的循环网络如LSTMs和GRUs更准确，而且更简单、更清晰。因此，在将深度网络应用于序列时，它可能是更合适的起点。

# 2. Background

# 2. 背景

Convolutional networks (LeCun et al., 1989) have been applied to sequences for decades (Sejnowski & Rosenberg, 1987; Hinton, 1989). They were used prominently for speech recognition in the 80 s and 90 s (Waibel et al., 1989; Bottou et al., 1990). ConvNets were subsequently applied to NLP tasks such as part-of-speech tagging and semantic role labelling (Collobert & Weston, 2008; Col-lobert et al., 2011; dos Santos & Zadrozny, 2014). More recently, convolutional networks were applied to sentence classification (Kalchbrenner et al., 2014; Kim, 2014) and document classification (Zhang et al., 2015; Conneau et al., 2017; Johnson & Zhang, 2015; 2017). Particularly inspiring for our work are the recent applications of convolutional architectures to machine translation (Kalchbrenner et al., 2016; Gehring et al., 2017a;b), audio synthesis (van den Oord et al., 2016), and language modeling (Dauphin et al., 2017).

卷积网络（LeCun等人，1989年）数十年来一直被应用于序列（Sejnowski & Rosenberg，1987年；Hinton，1989年）。它们在80年代和90年代被突出用于语音识别（Waibel等人，1989年；Bottou等人，1990年）。随后，ConvNets被应用于自然语言处理任务，如词性标注和语义角色标注（Collobert & Weston，2008年；Collobert等人，2011年；dos Santos & Zadrozny，2014年）。最近，卷积网络被应用于句子分类（Kalchbrenner等人，2014年；Kim，2014年）和文档分类（Zhang等人，2015年；Conneau等人，2017年；Johnson & Zhang，2015年；2017年）。特别激发我们工作灵感的是最近将卷积架构应用于机器翻译（Kalchbrenner等人，2016年；Gehring等人，2017a;b），音频合成（van den Oord等人，2016年）和语言建模（Dauphin等人，2017年）的应用。

Recurrent networks are dedicated sequence models that maintain a vector of hidden activations that are propagated through time (Elman, 1990; Werbos, 1990; Graves, 2012). This family of architectures has gained tremendous popularity due to prominent applications to language modeling (Sutskever et al., 2011; Graves, 2013; Hermans & Schrauwen, 2013) and machine translation (Sutskever et al., 2014; Bahdanau et al., 2015). The intuitive appeal of recurrent modeling is that the hidden state can act as a representation of everything that has been seen so far in the sequence. Basic RNN architectures are notoriously difficult to train (Bengio et al., 1994; Pascanu et al., 2013) and more elaborate architectures are commonly used instead, such as the LSTM (Hochreiter & Schmidhuber, 1997) and the GRU (Cho et al., 2014). Many other architectural innovations and training techniques for recurrent networks have been introduced and continue to be actively explored (El Hihi & Bengio, 1995; Schuster & Paliwal, 1997; Gers et al., 2002; Koutnik et al., 2014; Le et al., 2015; Ba et al., 2016; Wu et al., 2016; Krueger et al., 2017; Merity et al., 2017; Campos et al., 2018). Multiple empirical studies have been conducted to evaluate the effectiveness of different recurrent architectures. These studies have been motivated in part by the many degrees of freedom in the design of such architectures. Chung et al. (2014) compared different types of recurrent units (LSTM vs. GRU) on the task of polyphonic music modeling. Pas-canu et al. (2014) explored different ways to construct deep RNNs and evaluated the performance of different architectures on polyphonic music modeling, character-level language modeling, and word-level language modeling. Joze-fowicz et al. (2015) searched through more than ten thousand different RNN architectures and evaluated their performance on various tasks. They concluded that if there were "architectures much better than the LSTM", then they were "not trivial to find". Greff et al. (2017) benchmarked the performance of eight LSTM variants on speech recognition, handwriting recognition, and polyphonic music modeling. They also found that "none of the variants can improve upon the standard LSTM architecture significantly". Zhang et al. (2016) systematically analyzed the connecting architectures of RNNs and evaluated different architectures on character-level language modeling and on synthetic stress tests. Melis et al. (2018) benchmarked LSTM-based architectures on word-level and character-level language modeling, and concluded that "LSTMs outperform the more recent models".

循环网络是专门用于序列建模的模型，它们维持一个隐藏激活向量，该向量随时间传播（Elman, 1990; Werbos, 1990; Graves, 2012）。这类架构因其在语言建模（Sutskever et al., 2011; Graves, 2013; Hermans & Schrauwen, 2013）和机器翻译（Sutskever et al., 2014; Bahdanau et al., 2015）中的突出应用而获得了极大的流行。循环建模的直观吸引力在于，隐藏状态可以作为序列到目前为止所看到的所有内容的表示。基本的RNN架构训练起来非常困难（Bengio et al., 1994; Pascanu et al., 2013），因此通常使用更复杂的架构，如LSTM（Hochreiter & Schmidhuber, 1997）和GRU（Cho et al., 2014）。已经引入了许多其他循环网络的架构创新和训练技术，并且仍在积极研究（El Hihi & Bengio, 1995; Schuster & Paliwal, 1997; Gers et al., 2002; Koutnik et al., 2014; Le et al., 2015; Ba et al., 2016; Wu et al., 2016; Krueger et al., 2017; Merity et al., 2017; Campos et al., 2018）。已经进行了多项实证研究来评估不同循环架构的有效性。这些研究部分是由这些架构设计中许多自由度所激发的。Chung et al. (2014) 在多声部音乐建模任务上比较了不同类型的循环单元（LSTM 与 GRU）。Pascanu et al. (2014) 探索了构建深度RNN的不同方法，并评估了不同架构在多声部音乐建模、字符级语言建模和词级语言建模上的性能。Joze-fowicz et al. (2015) 筛选了超过一万种不同的RNN架构，并评估了它们在各种任务上的性能。他们得出结论，如果存在“比LSTM好得多的架构”，那么它们“并不容易找到”。Greff et al. (2017) 对八种LSTM变体在语音识别、手写识别和多声部音乐建模上的性能进行了基准测试。他们还发现“没有一种变体能够显著改进标准LSTM架构”。Zhang et al. (2016) 系统地分析了RNN的连接架构，并在字符级语言建模和合成压力测试上评估了不同的架构。Melis et al. (2018) 对基于LSTM的架构在词级和字符级语言建模上进行了基准测试，并得出结论“LSTM超过了较新的模型”。

Other recent works have aimed to combine aspects of RNN and CNN architectures. This includes the Convolutional LSTM (Shi et al., 2015), which replaces the fully-connected layers in an LSTM with convolutional layers to allow for additional structure in the recurrent layers; the Quasi-RNN model (Bradbury et al., 2017) that interleaves convolutional layers with simple recurrent layers; and the dilated RNN (Chang et al., 2017), which adds dilations to recurrent architectures. While these combinations show promise in combining the desirable aspects of both types of architectures, our study here focuses on a comparison of generic convolutional and recurrent architectures.

其他近期的工作旨在结合RNN和CNN架构的方面。这包括卷积LSTM（Shi等人，2015年），它将LSTM中的全连接层替换为卷积层，以允许循环层中有额外的结构；准RNN模型（Bradbury等人，2017年），它在卷积层和简单的循环层之间交替；以及扩张RNN（Chang等人，2017年），它在循环架构中添加扩张。虽然这些组合在结合两种架构的可取方面上显示出希望，但我们这里的研究主要集中在比较通用卷积和循环架构。

While there have been multiple thorough evaluations of RNN architectures on representative sequence modeling tasks, we are not aware of a similarly thorough comparison of convolutional and recurrent approaches to sequence modeling. (Yin et al. (2017) have reported a comparison of convolutional and recurrent networks for sentence-level and document-level classification tasks. In contrast, sequence modeling calls for architectures that can synthesize whole sequences, element by element.) Such comparison is particularly intriguing in light of the aforementioned recent success of convolutional architectures in this domain. Our work aims to compare generic convolutional and recurrent architectures on typical sequence modeling tasks that are commonly used to benchmark RNN variants themselves (Hermans & Schrauwen, 2013; Le et al., 2015; Joze-fowicz et al., 2015; Zhang et al., 2016).

虽然已有对RNN架构在代表性序列建模任务上的多次彻底评估，但我们不清楚是否有同样彻底的比较卷积和循环方法在序列建模上的研究。（Yin等人（2017年）报告了卷积网络和循环网络在句子级别和文档级别分类任务上的比较。相比之下，序列建模需要能够逐元素合成整个序列的架构。）这种比较在考虑到最近卷积架构在该领域的成功时尤其引人入胜。我们的工作旨在比较通用卷积和循环架构在典型序列建模任务上的表现，这些任务通常用于对RNN变体本身进行基准测试（Hermans & Schrauwen，2013年；Le等人，2015年；Joze-fowicz等人，2015年；Zhang等人，2016年）。

# 3. Temporal Convolutional Networks

# 3. 时空卷积网络

We begin by describing a generic architecture for convolutional sequence prediction. Our aim is to distill the best practices in convolutional network design into a simple architecture that can serve as a convenient but powerful starting point. We refer to the presented architecture as a temporal convolutional network (TCN), emphasizing that we adopt this term not as a label for a truly new architecture, but as a simple descriptive term for a family of architectures. (Note that the term has been used before (Lea et al., 2017).) The distinguishing characteristics of TCNs are: 1) the convolutions in the architecture are causal, meaning that there is no information "leakage" from future to past; 2) the architecture can take a sequence of any length and map it to an output sequence of the same length, just as with an RNN. Beyond this, we emphasize how to build very long effective history sizes (i.e., the ability for the networks to look very far into the past to make a prediction) using a combination of very deep networks (augmented with residual layers) and dilated convolutions.

我们首先描述了一种用于卷积序列预测的通用架构。我们的目标是提炼卷积网络设计中最佳实践，将其浓缩为一个简单但强大的起点架构。我们将所提出的架构称为时间卷积网络（TCN），强调我们采用这个术语并不是作为一个全新架构的标签，而是作为一个用于描述一系列架构的简单描述性术语。（请注意，这个术语之前已经被使用过（Lea et al., 2017）。）TCN的独特特征是：1）架构中的卷积是因果的，意味着未来到过去没有信息“泄露”；2）该架构可以接受任意长度的序列并将其映射到相同长度的输出序列，就像RNN一样。除此之外，我们强调如何通过结合非常深的网络（增强残留层）和扩张卷积来构建非常长的有效历史大小（即网络能够回溯到很远过去进行预测的能力）。

Our architecture is informed by recent convolutional architectures for sequential data (van den Oord et al., 2016; Kalchbrenner et al., 2016; Dauphin et al., 2017; Gehring et al., 2017a;b), but is distinct from all of them and was designed from first principles to combine simplicity, autoregressive prediction, and very long memory. For example, the TCN is much simpler than WaveNet (van den Oord et al., 2016) (no skip connections across layers, conditioning, context stacking, or gated activations).

我们的架构受到最近用于序列数据的卷积架构的启发（van den Oord et al., 2016; Kalchbrenner et al., 2016; Dauphin et al., 2017; Gehring et al., 2017a;b），但与它们都不同，并且是从基本原则出发设计的，以结合简洁性、自回归预测和非常长的记忆。例如，TCN比WaveNet（van den Oord et al., 2016）简单得多（没有跨层的跳过连接、条件、上下文堆叠或门控激活）。

Compared to the language modeling architecture of Dauphin et al. (2017), TCNs do not use gating mechanisms and have much longer memory.

与Dauphin et al.（2017）的语言建模架构相比，TCN不使用门控机制，并且具有更长的记忆。

# 3.1. Sequence Modeling

# 3.1. 序列建模

Before defining the network structure, we highlight the nature of the sequence modeling task. Suppose that we are given an input sequence , and wish to predict some corresponding outputs at each time. The key constraint is that to predict the output for some time , we are constrained to only use those inputs that have been previously observed: . Formally, a sequence modeling network is any function that produces the mapping

在定义网络结构之前，我们强调序列建模任务的本质。假设我们给定了一个输入序列 ，并且希望在每个时刻预测一些相应的输出 。关键约束是，为了预测某个时刻 的输出 ，我们只能使用之前观察到的输入： 。正式地说，序列建模网络是任何产生映射的函数 。

if it satisfies the causal constraint that depends only on and not on any "future" inputs . The goal of learning in the sequence modeling setting is to find a network that minimizes some expected loss between the actual outputs and the predictions, , where the sequences and outputs are drawn according to some distribution. This formalism encompasses many settings such as autoregressive prediction (where we try to predict some signal given its past) by setting the target output to be simply the input shifted by one time step. It does not, however, directly capture domains such as machine translation, or sequence-to-sequence prediction in general, since in these cases the entire input sequence (including "future" states) can be used to predict each output (though the techniques can naturally be extended to work in such settings).

如果它满足因果约束，即 只依赖于 而不依赖于任何“未来”的输入 。在序列建模设置中，学习的目标是找到一个网络 ，它能够最小化实际输出和预测之间的期望损失 ，其中序列和输出是根据某个分布抽取的。这种形式包括了许多设置，例如自回归预测（我们尝试根据信号的过去来预测信号），通过将目标输出设置为简单地将输入向前移动一个时间步。然而，它并没有直接捕捉到诸如机器翻译或一般的序列到序列预测这样的领域，因为在这些情况下，可以使用整个输入序列（包括“未来”状态）来预测每个输出（尽管这些技术自然可以扩展到在这些设置中工作）。

# 3.2. Causal Convolutions

# 3.2. 因果卷积

As mentioned above, the TCN is based upon two principles: the fact that the network produces an output of the same length as the input, and the fact that there can be no leakage from the future into the past. To accomplish the first point, the TCN uses a 1D fully-convolutional network (FCN) architecture (Long et al., 2015), where each hidden layer is the same length as the input layer, and zero padding of length (kernel size -1) is added to keep subsequent layers the same length as previous ones. To achieve the second point, the TCN uses causal convolutions, convolutions where an output at time is convolved only with elements from time and earlier in the previous layer.

如上所述，TCN基于两个原则：网络产生的输出与输入长度相同，以及未来不能泄露到过去。为了实现第一个原则，TCN使用了一种一维全卷积网络（FCN）架构（Long等人，2015年），其中每个隐藏层与输入层长度相同，并且添加了长度为（卷积核大小-1）的零填充，以保持后续层与前一层长度相同。为了达到第二个原则，TCN使用因果卷积，即输出在时间 只与前一层的 及更早的元素进行卷积。

To put it simply: causal convolutions.

简而言之： 因果卷积。

Note that this is essentially the same architecture as the time delay neural network proposed nearly 30 years ago by Waibel et al. (1989), with the sole tweak of zero padding to ensure equal sizes of all layers.

注意，这本质上与Waibel等人（1989年）近30年前提出的时间延迟神经网络架构相同，唯一的调整是使用零填充以确保所有层大小相同。

A major disadvantage of this basic design is that in order to achieve a long effective history size, we need an extremely deep network or very large filters, neither of which were particularly feasible when the methods were first introduced. Thus, in the following sections, we describe how techniques from modern convolutional architectures can be integrated into a TCN to allow for both very deep networks and very long effective history.

这种基本设计的一个主要缺点是，为了达到长的有效历史大小，我们需要一个极深的网络或非常大的滤波器，在方法首次提出时，这两者都不可行。因此，在接下来的章节中，我们描述了如何将现代卷积架构的技术整合到TCN中，以允许构建非常深的网络和非常长的有效历史。

# 3.3. Dilated Convolutions

# 3.3. 膨胀卷积

A simple causal convolution is only able to look back at a history with size linear in the depth of the network. This makes it challenging to apply the aforementioned causal convolution on sequence tasks, especially those requiring longer history. Our solution here, following the work of van den Oord et al. (2016), is to employ dilated convolutions that enable an exponentially large receptive field (Yu & Koltun, 2016). More formally, for a 1-D sequence input and a filter , the dilated convolution operation on element of the sequence is defined as

简单的因果卷积只能回溯与网络深度成线性关系的历史长度。这使得在序列任务上应用上述因果卷积变得具有挑战性，尤其是在需要更长的历史记录的任务上。我们的解决方案，遵循van den Oord等人（2016年）的工作，是使用扩张卷积，它能够拥有指数级大的感受野（Yu & Koltun, 2016）。更正式地说，对于一个1维序列输入 和一个滤波器 ，在序列元素 上的扩张卷积操作 定义为

where is the dilation factor, is the filter size, and accounts for the direction of the past. Dilation is thus equivalent to introducing a fixed step between every two adjacent filter taps. When , a dilated convolution reduces to a regular convolution. Using larger dilation enables an output at the top level to represent a wider range of inputs, thus effectively expanding the receptive field of a ConvNet.其中 是扩张因子， 是滤波器大小， 考虑过去的方向。因此，扩张相当于在每两个相邻滤波器抽头之间引入固定步长。当 时，扩张卷积简化为常规卷积。使用更大的扩张使得顶层输出能够表示更宽范围的输入，从而有效地扩展了卷积网络（ConvNet）的感受野。

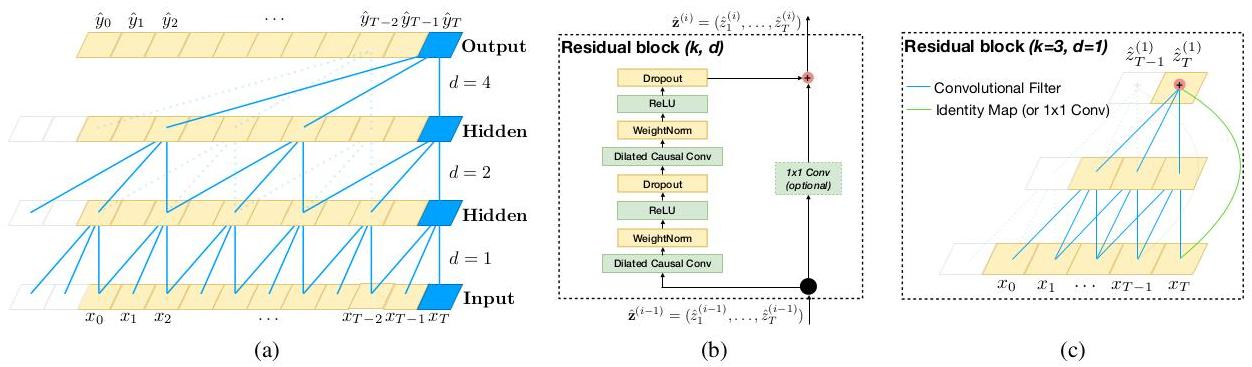


Figure 1. Architectural elements in a TCN. (a) A dilated causal convolution with dilation factors and filter size . The receptive field is able to cover all values from the input sequence. (b) TCN residual block. An convolution is added when residual input and output have different dimensions. (c) An example of residual connection in a TCN. The blue lines are filters in the residual function, and the green lines are identity mappings.

图1. TCN中的架构元素。（a）具有扩张因子 和滤波器大小 的扩张因果卷积。感受野能够覆盖输入序列中的所有值。（b）TCN残差块。当残差输入和输出具有不同维度时，添加一个 卷积。（c）TCN中残差连接的一个示例。蓝色线条是残差函数中的滤波器，绿色线条是恒等映射。

This gives us two ways to increase the receptive field of the TCN: choosing larger filter sizes and increasing the dilation factor , where the effective history of one such layer is . As is common when using dilated convolutions, we increase exponentially with the depth of the network (i.e., at level of the network). This ensures that there is some filter that hits each input within the effective history, while also allowing for an extremely large effective history using deep networks. We provide an illustration in Figure 1(a).

这为我们提供了两种增加TCN感受野的方法：选择更大的滤波器尺寸 和增加扩张因子 ，其中这样一个层的有效历史是 。和使用扩张卷积时通常的做法一样，我们随着网络深度的增加指数性地增加 （即，网络级别 上的 ）。这确保了在有效历史内的每个输入都有一个滤波器能够触及，同时使用深度网络也能实现非常大的有效历史。我们在图1(a)中提供了说明。

# 3.4. Residual Connections

# 3.4. 残差连接

A residual block (He et al., 2016) contains a branch leading out to a series of transformations , whose outputs are added to the input of the block:

残差块（He等人，2016）包含一个分支，该分支导向一系列的转换 ，其输出被加到块的输入 上：

This effectively allows layers to learn modifications to the identity mapping rather than the entire transformation, which has repeatedly been shown to benefit very deep networks.

这实际上允许层学习对恒等映射的修改，而不是整个转换，这已经反复证明对于非常深的网络是有益的。

Since a TCN’s receptive field depends on the network depth as well as filter size and dilation factor , stabilization of deeper and larger TCNs becomes important. For example, in a case where the prediction could depend on a history of size and a high-dimensional input sequence, a network of up to 12 layers could be needed. Each layer, more specifically, consists of multiple filters for feature extraction. In our design of the generic TCN model, we therefore employ a generic residual module in place of a convolutional layer. The residual block for our baseline TCN is shown in Figure 1(b). Within a residual block, the TCN has two layers of dilated causal convolution and non-linearity, for which we used the rectified linear unit (ReLU) (Nair & Hinton, 2010). For normalization, we applied weight normalization (Salimans & Kingma, 2016) to the convolutional filters. In addition, a spatial dropout (Srivastava et al., 2014) was added after each dilated convolution for regularization: at each training step, a whole channel is zeroed out.

由于 TCN 的感受野取决于网络深度 、滤波器大小 以及扩张因子 ，因此稳定更深更大 TCN 变得重要。例如，在预测可能依赖于大小为 的历史记录和高维输入序列的情况下，可能需要多达 12 层的网络。具体来说，每一层都由多个用于特征提取的滤波器组成。在我们设计的通用 TCN 模型中，因此用一个通用残差模块代替卷积层。我们基线 TCN 的残差块如图 1(b)所示。在残差块内部，TCN 有两层扩张因果卷积和非线性，我们使用了修正线性单元（ReLU）（Nair & Hinton, 2010）作为非线性激活函数。为了归一化，我们对卷积滤波器应用了权重归一化（Salimans & Kingma, 2016）。此外，在每个扩张卷积之后添加了空间dropout（Srivastava et al., 2014）进行正则化：在每次训练步骤中，一个完整的通道将被置零。

However, whereas in standard ResNet the input is added directly to the output of the residual function, in TCN (and ConvNets in general) the input and output could have different widths. To account for discrepant input-output widths, we use an additional convolution to ensure that element-wise addition receives tensors of the same shape (see Figure 1(b, c)).

然而，在标准 ResNet 中，输入直接加到残差函数的输出上，而在 TCN（以及一般的卷积网络 ConvNets）中，输入和输出可能有不同的宽度。为了解决输入输出宽度不一致的问题，我们使用了一个额外的 卷积，以确保逐元素加法 接收到的张量具有相同的形状（见图 1(b, c)）。

# 3.5. Discussion

# 3.5. 讨论

We conclude this section by listing several advantages and disadvantages of using TCNs for sequence modeling.

我们通过列举使用 TCN 进行序列建模的几个优点和缺点来结束本节。

* Parallelism. Unlike in RNNs where the predictions for later timesteps must wait for their predecessors to complete, convolutions can be done in parallel since the same filter is used in each layer. Therefore, in both training and evaluation, a long input sequence can be processed as a whole in TCN, instead of sequentially as in RNN.
* 并行性。与RNNs不同，在RNNs中，后期时间步的预测必须等待前驱时间步完成，卷积可以在并行中进行，因为每个层都使用相同的滤波器。因此，在训练和评估过程中，TCN可以一次性处理整个长输入序列，而不是像RNN那样顺序处理。
* Flexible receptive field size. A TCN can change its receptive field size in multiple ways. For instance, stacking more dilated (causal) convolutional layers, using larger dilation factors, or increasing the filter size are all viable options (with possibly different interpretations). TCNs thus afford better control of the model’s memory size, and are easy to adapt to different domains.
* 灵活的感受野大小。TCN可以通过多种方式改变其感受野大小。例如，堆叠更多扩张（因果）卷积层，使用更大的扩张因子，或者增加滤波器大小都是可行的选项（可能有不同的解释）。因此，TCN可以更好地控制模型的内存大小，并且易于适应不同的领域。
* Stable gradients. Unlike recurrent architectures, TCN has a backpropagation path different from the temporal direction of the sequence. TCN thus avoids the problem of exploding/vanishing gradients, which is a major issue for RNNs (and which led to the development of LSTM, GRU, HF-RNN (Martens & Sutskever, 2011), etc.).
* 稳定的梯度。与循环架构不同，TCN的反向传播路径与序列的时间方向不同。因此，TCN避免了梯度爆炸/消失的问题，这是RNN的一个主要问题（也是LSTM、GRU、HF-RNN（Martens & Sutskever, 2011）等发展的原因）。
* Low memory requirement for training. Especially in the case of a long input sequence, LSTMs and GRUs can easily use up a lot of memory to store the partial results for their multiple cell gates. However, in a TCN the filters are shared across a layer, with the backpropagation path depending only on network depth. Therefore in practice, we found gated RNNs likely to use up to a multiplicative factor more memory than TCNs.
* 训练时低内存要求。特别是在长输入序列的情况下，LSTMs和GRUs可以轻易地使用大量内存来存储其多个细胞门的中间结果。然而，在TCN中，滤波器在层内共享，反向传播路径仅依赖于网络深度。因此实际上，我们发现门控RNNs可能比TCN使用更多的内存。
* Variable length inputs. Just like RNNs, which model inputs with variable lengths in a recurrent way, TCNs can also take in inputs of arbitrary lengths by sliding the 1D convolutional kernels. This means that TCNs can be adopted as drop-in replacements for RNNs for sequential data of arbitrary length.
* 可变长度输入。就像RNNs一样，它们以循环方式对可变长度的输入进行建模，TCNs也可以通过滑动1D卷积核接受任意长度的输入。这意味着TCN可以作为RNNs的即插即用替代品，用于任意长度的序列数据。

There are also two notable disadvantages to using TCNs.

使用TCN还有两个值得注意的缺点。

* Data storage during evaluation. In evaluation/testing, RNNs only need to maintain a hidden state and take in a current input in order to generate a prediction. In other words, a "summary" of the entire history is provided by the fixed-length set of vectors , and the actual observed sequence can be discarded. In contrast, TCNs need to take in the raw sequence up to the effective history length, thus possibly requiring more memory during evaluation.
* 评估过程中的数据存储。在评估/测试过程中，RNN仅需要维持一个隐藏状态并接收当前输入 以生成预测。换句话说，固定长度的向量集合 提供了整个历史的“摘要”，而实际观察到的序列可以被丢弃。相比之下，TCN需要接收直到有效历史长度的原始序列，因此在评估过程中可能需要更多的内存。
* Potential parameter change for a transfer of domain. Different domains can have different requirements on the amount of history the model needs in order to predict. Therefore, when transferring a model from a domain where only little memory is needed (i.e., small and ) to a domain where much longer memory is required (i.e., much larger and ), TCN may perform poorly for not having a sufficiently large receptive field.
* 领域迁移中的潜在参数变化。不同的领域对模型预测所需的历 史量有不同的要求。因此，当将一个模型从只需要少量内存的领域（即，小的 和 ）迁移到一个需要更长记忆的领域（即，大得多的 和 ）时，由于没有足够大的感受野，TCN可能会表现不佳。

# 4. Sequence Modeling Tasks

# 4. 序列建模任务

We evaluate TCNs and RNNs on tasks that have been commonly used to benchmark the performance of different RNN sequence modeling architectures (Hermans & Schrauwen, 2013; Chung et al., 2014; Pascanu et al., 2014; Le et al., 2015; Jozefowicz et al., 2015; Zhang et al., 2016). The intention is to conduct the evaluation on the "home turf" of RNN sequence models. We use a comprehensive set of synthetic stress tests along with real-world datasets from multiple domains.

我们在一系列已被广泛用于评估不同RNN序列建模架构性能的任务上评估TCN和RNN（Hermans & Schrauwen, 2013; Chung et al., 2014; Pascanu et al., 2014; Le et al., 2015; Jozefowicz et al., 2015; Zhang et al., 2016）。我们的意图是在RNN序列模型的“主场”进行评估。我们使用了一组全面的合成压力测试以及来自多个领域的真实世界数据集。

The adding problem. In this task, each input consists of a length- sequence of depth 2, with all values randomly chosen in , and the second dimension being all zeros except for two elements that are marked by 1 . The objective is to sum the two random values whose second dimensions are marked by 1 . Simply predicting the sum to be 1 should give an MSE of about 0.1767 . First introduced by Hochreiter & Schmidhuber (1997), the adding problem has been used repeatedly as a stress test for sequence models (Martens & Sutskever, 2011; Pascanu et al., 2013; Le et al., 2015; Arjovsky et al., 2016; Zhang et al., 2016).

加法问题。在此任务中，每个输入由长度为 的深度为2的序列组成，所有值随机选择在 范围内，第二个维度除了两个标记为1的元素外，其余均为零。目标是计算第二维度标记为1的两个随机值的和。简单预测和为1应该得到大约0.1767的平均平方误差（MSE）。由Hochreiter & Schmidhuber (1997) 首次提出，加法问题已被反复用作序列模型的压力测试（Martens & Sutskever, 2011; Pascanu et al., 2013; Le et al., 2015; Arjovsky et al., 2016; Zhang et al., 2016）。

Sequential MNIST and P-MNIST. Sequential MNIST is frequently used to test a recurrent network’s ability to retain information from the distant past (Le et al., 2015; Zhang et al., 2016; Wisdom et al., 2016; Cooijmans et al., 2016; Krueger et al., 2017; Jing et al., 2017). In this task, MNIST images (LeCun et al., 1998) are presented to the model as a sequence for digit classification. In the more challenging P-MNIST setting, the order of the sequence is permuted at random (Le et al., 2015; Arjovsky et al., 2016; Wisdom et al., 2016; Krueger et al., 2017).

序列MNIST和P-MNIST。序列MNIST常用于测试循环网络保留远期过去信息的能力（Le et al., 2015; Zhang et al., 2016; Wisdom et al., 2016; Cooijmans et al., 2016; Krueger et al., 2017; Jing et al., 2017）。在此任务中，MNIST图像（LeCun et al., 1998）作为数字分类的 序列呈现给模型。在更具挑战性的P-MNIST设置中，序列的顺序被随机置换（Le et al., 2015; Arjovsky et al., 2016; Wisdom et al., 2016; Krueger et al., 2017）。

Copy memory. In this task, each input sequence has length . The first 10 values are chosen randomly among the digits , with the rest being all zeros, except for the last 11 entries that are filled with the digit ’ 9 ’ (the first ’ 9 ’ is a delimiter). The goal is to generate an output of the same length that is zero everywhere except the last 10 values after the delimiter, where the model is expected to repeat the 10 values it encountered at the start of the input. This task was used in prior works such as Zhang et al. (2016); Arjovsky et al. (2016); Wisdom et al. (2016); Jing et al. (2017).

复制记忆。在此任务中，每个输入序列的长度为 。前10个值在数字 中随机选择，其余的值都是零，除了最后11个条目被数字 ’9’ 填充（第一个 ’9’ 是分隔符）。目标是生成一个同样长度的输出，除了分隔符后的最后10个值之外，其余位置都是零，模型预计会重复输入序列开头的10个值。此任务在之前的研究中曾被使用，如Zhang等人（2016）；Arjovsky等人（2016）；Wisdom等人（2016）；Jing等人（2017）。

JSB Chorales and Nottingham. JSB Chorales (Allan & Williams, 2005) is a polyphonic music dataset consisting of the entire corpus of 382 four-part harmonized chorales by J. S. Bach. Each input is a sequence of elements. Each element is an 88-bit binary code that corresponds to the 88 keys on a piano, with 1 indicating a key that is pressed at a given time. Nottingham is a polyphonic music dataset based on a collection of 1,200 British and American folk tunes, and is much larger than JSB Chorales. JSB Chorales and Nottingham have been used in numerous empirical investigations of recurrent sequence modeling (Chung et al., 2014; Pascanu et al., 2014; Jozefowicz et al., 2015; Greff et al., 2017). The performance on both tasks is measured in terms of negative log-likelihood (NLL).

JSB Chorales 和 Nottingham。JSB Chorales（Allan & Williams, 2005）是一个多声部音乐数据集，包含了J. S. Bach的全部382首四声部和谐合唱曲。每个输入是一个元素序列。每个元素是一个88位的二进制代码，对应于钢琴上的88个键，1表示在特定时间被按下的键。Nottingham是基于1200首英国和美国民间曲调的一个多声部音乐数据集，比JSB Chorales大得多。JSB Chorales 和 Nottingham已在许多循环序列建模的实证研究中被使用（Chung等人，2014；Pascanu等人，2014；Jozefowicz等人，2015；Greff等人，2017）。在这两个任务上的性能是以负对数似然（NLL）来衡量的。

PennTreebank. We used the PennTreebank (PTB) (Marcus et al., 1993) for both character-level and word-level language modeling. When used as a character-level language corpus, PTB contains characters for training, for validation, and for testing, with an alphabet size of 50 . When used as a word-level language corpus, PTB contains words for training, for validation, and for testing, with a vocabulary size of . This is a highly studied but relatively small language modeling dataset (Miyamoto & Cho, 2016; Krueger et al., 2017; Mer-ity et al., 2017).

PennTreebank。我们使用了PennTreebank（PTB）（Marcus等人，1993年）进行字符级和词汇级的语言建模。当作为字符级语言语料库使用时，PTB包含 个训练字符， 个验证字符，以及 个测试字符，字母表大小为50。当作为词汇级语言语料库使用时，PTB包含 个训练单词， 个验证单词，以及 个测试单词，词汇量为 。这是一个被广泛研究但相对较小的语言建模数据集（Miyamoto & Cho，2016；Krueger等人，2017；Merity等人，2017）。

Wikitext-103. Wikitext-103 (Merity et al., 2016) is almost

Wikitext-103。Wikitext-103（Merity等人，2016年）几乎是

Table 1. Evaluation of TCNs and recurrent architectures on synthetic stress tests, polyphonic music modeling, character-level language modeling, and word-level language modeling. The generic TCN architecture outperforms canonical recurrent networks across a comprehensive suite of tasks and datasets. Current state-of-the-art results are listed in the supplement. means that higher is better. means that lower is better.

表1。在合成压力测试、多声部音乐建模、字符级语言建模和词汇级语言建模中评估TCN和循环架构。通用TCN架构在一系列全面的任务和数据集上超过了标准的循环网络。当前最先进的结果列在补充材料中。 表示数值越高越好。 表示数值越低越好。

| Sequence Modeling Task | Model Size | Models | | | |
| --- | --- | --- | --- | --- | --- |
|  |  | LSTM | GRU | RNN | TCN |
| Seq. MNIST (accuracy ) |  | 87.2 | 96.2 | 21.5 | 99.0 |
| Permuted MNIST (accuracy) | 70K | 85.7 | 87.3 | 25.3 | 97.2 |
| Adding problem | 70K | 0.164 | 5.3e-5 | 0.177 | 5.8e-5 |
| Copy memory (loss) | 16K | 0.0204 | 0.0197 | 0.0202 | 3.5e-5 |
| Music JSB Chorales (loss) |  | 8.45 | 8.43 | 8.91 | 8.10 |
| Music Nottingham (loss) | 1M | 3.29 | 3.46 | 4.05 | 3.07 |
| Word-level PTB (perplexity ) | 13M | 78.93 | 92.48 | 114.50 | 88.68 |
| Word-level Wiki-103 (perplexity) | - | 48.4 | - | - | 45.19 |
| Word-level LAMBADA (perplexity) | - | 4186 | - | 14725 | 1279 |
| Char-level PTB (bpc ) | 3M | 1.36 | 1.37 | 1.48 | 1.31 |
| Char-level text8 (bpc) | 5M | 1.50 | 1.53 | 1.69 | 1.45 |

110 times as large as PTB, featuring a vocabulary size of about . The dataset contains Wikipedia articles (about 103 million words) for training, 60 articles (about words) for validation, and 60 articles for testing. This is a more representative and realistic dataset than PTB, with a much larger vocabulary that includes many rare words, and has been used in Merity et al. (2016); Grave et al. (2017); Dauphin et al. (2017).

是PTB的110倍大，词汇量约为 。该数据集包含 篇维基百科文章（大约1.03亿个单词）用于训练，60篇文章（大约 个单词）用于验证，以及60篇文章 用于测试。这是一个比PTB更具代表性和现实性的数据集，拥有大得多的词汇量，包括许多罕见单词，已在Merity等人（2016年）；Grave等人（2017年）；Dauphin等人（2017年）中使用过。

LAMBADA. Introduced by Paperno et al. (2016), LAM-BADA is a dataset comprising passages extracted from novels, with an average of 4.6 sentences as context, and 1 target sentence the last word of which is to be predicted. This dataset was built so that a person can easily guess the missing word when given the context sentences, but not when given only the target sentence without the context sentences. Most of the existing models fail on LAMBADA (Paperno et al., 2016; Grave et al., 2017). In general, better results on LAMBADA indicate that a model is better at capturing information from longer and broader context. The training data for LAMBADA is the full text of 2,662 novels with more than words. The vocabulary size is about .

LAMBADA。由Paperno等人（2016年）提出，LAM-BADA是一个数据集，包含 从小说中提取的段落，平均包含4.6个句子作为上下文，以及1个目标句子，其最后一个单词需要被预测。这个数据集的构建目的是让一个人在给定上下文句子时能够轻易猜测出缺失的单词，而在只给出没有上下文的目标句子时则不能。大多数现有模型在LAMBADA上失败（Paperno等人，2016年；Grave等人，2017年）。一般来说，在LAMBADA上取得更好的结果表明模型在捕捉更长、更广泛上下文信息的能力更强。LAMBADA的训练数据是2662部小说的全文，包含超过 个单词。词汇量大约为 。

text8. We also used the text 8 dataset for character-level language modeling (Mikolov et al., 2012). text8 is about 20 times larger than PTB, with about 100M characters from Wikipedia (90M for training, 5M for validation, and 5M for testing). The corpus contains 27 unique alphabets.

text8。我们还使用了text 8数据集进行字符级语言建模（Mikolov等人，2012年）。text8大约是PTB的20倍大，包含大约1亿个来自维基百科的字符（训练用9000万，验证用500万，测试用500万）。语料库包含27个独特的字母表。

# 5. Experiments

# 5. 实验

We compare the generic TCN architecture described in Section 3 to canonical recurrent architectures, namely LSTM, GRU, and vanilla RNN, with standard regularizations. All experiments reported in this section used exactly the same TCN architecture, just varying the depth of the network and occasionally the kernel size so that the receptive field covers enough context for predictions. We use an exponential dilation for layer in the network, and the Adam optimizer (Kingma & Ba, 2015) with learning rate 0.002 for TCN, unless otherwise noted. We also empirically find that gradient clipping helped convergence, and we pick the maximum norm for clipping from . When training recurrent models, we use grid search to find a good set of hyperparameters (in particular, optimizer, recurrent drop , learning rate, gradient clipping, and initial forget-gate bias), while keeping the network around the same size as TCN. No other architectural elaborations, such as gating mechanisms or skip connections, were added to either TCNs or RNNs. Additional details and controlled experiments are provided in the supplementary material.

我们将第3节中描述的通用TCN架构与规范循环架构进行比较，即LSTM、GRU和标准RNN，并使用标准正则化。本节报告的所有实验都使用了完全相同的TCN架构，仅改变了网络的深度 ，偶尔改变核大小 ，以便感受野能够覆盖足够多的上下文进行预测。我们为网络中的层 使用指数膨胀 ，并使用Adam优化器（Kingma & Ba, 2015）和0.002的学习率进行TCN训练，除非另有说明。我们还实证发现梯度裁剪有助于收敛，并从 中选择裁剪的最大范数。在训练循环模型时，我们使用网格搜索来找到一组好的超参数（特别是优化器、循环dropout 、学习率、梯度裁剪和初始遗忘门偏置），同时保持网络大小与TCN相近。没有对TCN或RNN添加其他架构细节，如门控机制或跳过连接。补充材料中提供了额外的细节和控制实验。

# 5.1. Synopsis of Results

# 5.1. 结果概要

A synopsis of the results is shown in Table 1. Note that on several of these tasks, the generic, canonical recurrent architectures we study (e.g., LSTM, GRU) are not the state-of-the-art. (See the supplement for more details.) With this caveat, the results strongly suggest that the generic TCN architecture with minimal tuning outperforms canonical recurrent architectures across a broad variety of sequence modeling tasks that are commonly used to benchmark the performance of recurrent architectures themselves. We now analyze these results in more detail.

表1显示了结果的概要。注意，在我们研究的这些任务中，我们研究的通用规范循环架构（例如，LSTM、GRU）并非最先进的。（参见补充材料以获取更多详细信息。）在保留这一前提下，结果表明，经过最少调整的通用TCN架构在广泛用于衡量循环架构性能的序列建模任务上优于规范循环架构。我们现在将更详细地分析这些结果。

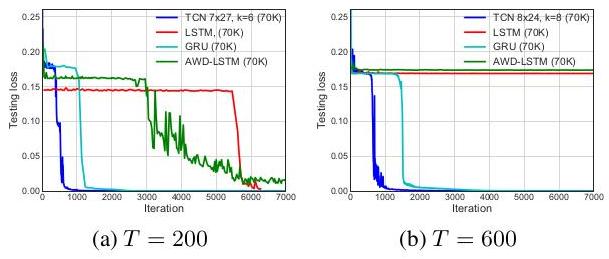


Figure 2. Results on the adding problem for different sequence lengths . TCNs outperform recurrent architectures.

图2. 不同序列长度 的加法问题的结果。TCN优于循环架构。

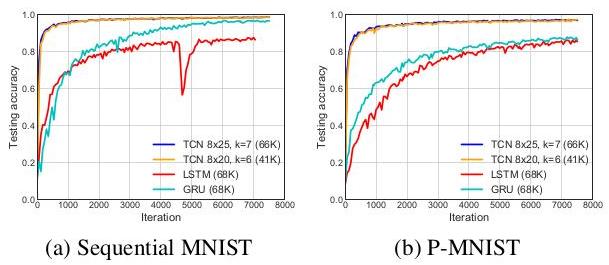


Figure 3. Results on Sequential MNIST and P-MNIST. TCNs outperform recurrent architectures.

图 3。顺序 MNIST 和 P-MNIST 的结果。TCNs 在性能上超过了循环架构。

# 5.2. Synthetic Stress Tests

# 5.2. 合成压力测试

The adding problem. Convergence results for the adding problem, for problem sizes and 600, are shown in Figure 2. All models were chosen to have roughly parameters. TCNs quickly converged to a virtually perfect solution (i.e., MSE near 0). GRUs also performed quite well, albeit slower to converge than TCNs. LSTMs and vanilla RNNs performed significantly worse.

加法问题。加法问题的收敛结果，针对问题大小 和 600，展示在图 2 中。所有模型都被选择为具有大约 个参数。TCNs 快速收敛到一个几乎完美的解决方案（即，MSE 接近 0）。GRUs 也表现相当好，尽管收敛速度比 TCNs 慢。LSTMs 和标准 RNNs 的表现显著较差。

Sequential MNIST and P-MNIST. Convergence results on sequential and permuted MNIST, run over 10 epochs, are shown in Figure 3. All models were configured to have roughly parameters. For both problems, TCNs substantially outperform the recurrent architectures, both in terms of convergence and in final accuracy on the task. For P-MNIST, TCNs outperform state-of-the-art results (95.9%) based on recurrent networks with Zoneout and Recurrent BatchNorm (Cooijmans et al., 2016; Krueger et al., 2017).

顺序 MNIST 和 P-MNIST。顺序和置换 MNIST 上的收敛结果，在 10 个周期内运行，展示在图 3 中。所有模型都被配置为具有大约 个参数。对于这两个问题，TCNs 在收敛速度和任务最终准确性上都大大超过了循环架构。对于 P-MNIST，TCNs 超过了基于循环网络带 Zoneout 和循环批量归一化（Cooijmans et al., 2016; Krueger et al., 2017）的最先进结果（95.9%）。

Copy memory. Convergence results on the copy memory task are shown in Figure 4. TCNs quickly converge to correct answers, while LSTMs and GRUs simply converge to the same loss as predicting all zeros. In this case we also compare to the recently-proposed EURNN (Jing et al., 2017), which was highlighted to perform well on this task. While both TCN and EURNN perform well for sequence length , the TCN has a clear advantage for and longer (in terms of both loss and rate of convergence).

复制记忆。复制记忆任务上的收敛结果展示在图 4 中。TCNs 快速收敛到正确答案，而 LSTMs 和 GRUs 仅收敛到与预测所有零相同的损失。在这种情况下，我们还比较了最近提出的 EURNN（Jing et al., 2017），它被强调在这个任务上表现良好。尽管 TCN 和 EURNN 在序列长度 上都表现良好，但在 和更长的序列长度上，TCN 在损失和收敛率上都有明显优势。

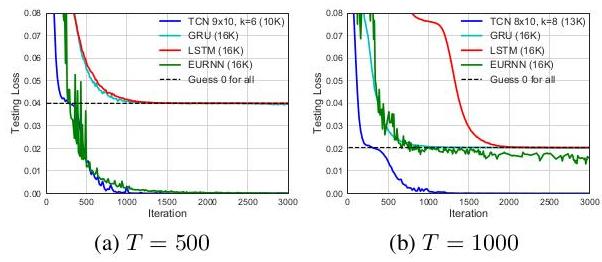


Figure 4. Result on the copy memory task for different sequence lengths . TCNs outperform recurrent architectures.

图 4。不同序列长度 的复制记忆任务结果。TCNs 在性能上超过了循环架构。

# 5.3. Polyphonic Music and Language Modeling

# 5.3. 多声部音乐与语言建模

We now discuss the results on polyphonic music modeling, character-level language modeling, and word-level language modeling. These domains are dominated by recurrent architectures, with many specialized designs developed for these tasks (Zhang et al., 2016; Ha et al., 2017; Krueger et al., 2017; Grave et al., 2017; Greff et al., 2017; Merity et al., 2017). We mention some of these specialized architectures when useful, but our primary goal is to compare the generic TCN model to similarly generic recurrent architectures, before domain-specific tuning. The results are summarized in Table 1.

我们现在讨论多声部音乐建模、字符级语言建模和词级语言建模的结果。这些领域主要由循环架构主导，许多专门为这些任务而开发的设计（Zhang et al., 2016; Ha et al., 2017; Krueger et al., 2017; Grave et al., 2017; Greff et al., 2017; Merity et al., 2017）。在有用时，我们会提到这些专门的架构，但我们的主要目标是将在特定领域调整之前的通用TCN模型与同样通用的循环架构进行比较。结果总结在表1中。

Polyphonic music. On Nottingham and JSB Chorales, the TCN with virtually no tuning outperforms the recurrent models by a considerable margin, and even outperforms some enhanced recurrent architectures for this task such as HF-RNN (Boulanger-Lewandowski et al., 2012) and Diagonal RNN (Subakan & Smaragdis, 2017). Note however that other models such as the Deep Belief Net LSTM perform better still (Vohra et al., 2015); we believe this is likely due to the fact that the datasets are relatively small, and thus the right regularization method or generative modeling procedure can improve performance significantly. This is largely orthogonal to the RNN/TCN distinction, as a similar variant of TCN may well be possible.

多声部音乐。在Nottingham和JSB Chorales数据集上，几乎没有经过调整的TCN模型明显优于循环模型，甚至优于为此任务增强的循环架构，如HF-RNN（Boulanger-Lewandowski et al., 2012）和对角线RNN（Subakan & Smaragdis, 2017）。然而，请注意，其他模型如Deep Belief Net LSTM的表现甚至更好（Vohra et al., 2015）；我们认为这很可能是由于数据集相对较小，因此正确的正则化方法或生成建模过程可以显著提高性能。这一点在很大程度上与RNN/TCN的区别是正交的，因为类似变体的TCN也可能实现。

Word-level language modeling. Language modeling remains one of the primary applications of recurrent networks and many recent works have focused on optimizing LSTMs for this task (Krueger et al., 2017; Merity et al., 2017). Our implementation follows standard practice that ties the weights of encoder and decoder layers for both TCN and RNNs (Press & Wolf, 2016), which significantly reduces the number of parameters in the model. For training, we use SGD and anneal the learning rate by a factor of 0.5 for both TCN and RNNs when validation accuracy plateaus.

词级语言建模。语言建模仍然是循环网络的主要应用之一，许多近期的工作都集中在优化LSTM以完成此任务（Krueger等人，2017；Merity等人，2017）。我们的实现遵循标准做法，为TCN和RNNs的编码器和解码器层绑定权重（Press & Wolf，2016），这显著减少了模型中的参数数量。在训练过程中，我们使用SGD，并在验证准确度停滞时，将TCN和RNNs的学习率以0.5的因子进行退火。

On the smaller PTB corpus, an optimized LSTM architecture (with recurrent and embedding dropout, etc.) outperforms the TCN, while the TCN outperforms both GRU and vanilla RNN. However, on the much larger Wikitext-103 corpus and the LAMBADA dataset (Paperno et al., 2016), without any hyperparameter search, the TCN outperforms the LSTM results of Grave et al. (2017), achieving much lower perplexities.

在较小的PTB语料库上，优化的LSTM架构（带有循环和嵌入丢弃等）优于TCN，而TCN又优于GRU和普通RNN。然而，在更大的Wikitext-103语料库和LAMBADA数据集（Paperno等人，2016）上，未经任何超参数搜索，TCN优于Grave等人（2017）的LSTM结果，实现了更低的困惑度。

Character-level language modeling. On character-level language modeling (PTB and text8, accuracy measured in bits per character), the generic TCN outperforms regularized LSTMs and GRUs as well as methods such as Norm-stabilized LSTMs (Krueger & Memisevic, 2015). (Specialized architectures exist that outperform all of these, see the supplement.)

字符级语言建模。在字符级语言建模（PTB和text8，以每个字符的比特数衡量准确度）上，通用TCN优于正则化的LSTMs和GRUs，以及如Norm-stabilized LSTMs（Krueger & Memisevic，2015）等方法。（存在专门架构优于所有这些，参见补充材料。）

# 5.4. Memory Size of TCN and RNNs

# 5.4. TCN和RNNs的内存大小

One of the theoretical advantages of recurrent architectures is their unlimited memory: the theoretical ability to retain information through sequences of unlimited length. We now examine specifically how long the different architectures can retain information in practice. We focus on 1) the copy memory task, which is a stress test designed to evaluate long-term, distant information propagation in recurrent networks, and 2) the LAMBADA task, which tests both local and non-local textual understanding.

循环架构的理论优势之一是它们无限的内存：理论上能够通过无限长度的序列保留信息。我们现在具体考察不同架构在实际中能够保留信息多长时间。我们关注 1) 复制记忆任务，这是一个压力测试，旨在评估循环网络中的长期、远距离信息传播，以及 2) LAMBADA 任务，它测试本地和非本地文本理解。

The copy memory task is perfectly set up to examine a model’s ability to retain information for different lengths of time. The requisite retention time can be controlled by varying the sequence length . In contrast to Section 5.2, we now focus on the accuracy on the last 10 elements of the output sequence (which are the nontrivial elements that must be recalled). We used models of size for both TCN and RNNs.

复制记忆任务完美地设置了检查模型在不同时间长度保留信息能力的情况。所需的保留时间可以通过改变序列长度 来控制。与第5.2节相比，我们现在关注输出序列最后10个元素的准确性（这些是必须回忆的非平凡元素）。我们为TCN和RNNs使用了大小为 的模型。

The results of this focused study are shown in Figure 5. TCNs consistently converge to accuracy for all sequence lengths, whereas LSTMs and GRUs of the same size quickly degenerate to random guessing as the sequence length grows. The accuracy of the LSTM falls below for , while the GRU falls below for . These results indicate that TCNs are able to maintain a much longer effective history than their recurrent counterparts.

这项聚焦研究的成果显示在图5中。TCN在所有序列长度上始终趋于 的准确性，而相同大小的LSTM和GRU在序列长度 增长时迅速退化到随机猜测。当 时，LSTM的准确性低于 ，而GRU在 时低于 。这些结果表明，TCN能够保持比其循环对应物更长的有效历史。

This observation is backed up on real data by experiments on the large-scale LAMBADA dataset, which is specifically designed to test a model’s ability to utilize broad context (Pa-perno et al., 2016). As shown in Table 1, TCN outperforms LSTMs and vanilla RNNs by a significant margin in perplexity on LAMBADA, with a substantially smaller network and virtually no tuning. (State-of-the-art results on this dataset are even better, but only with the help of additional memory mechanisms (Grave et al., 2017).)

这一观察结果通过在大型LAMBADA数据集上的实验得到了实际数据的支持，该数据集专门设计用于测试模型利用广泛上下文的能力（Pa-perno等人，2016年）。如表1所示，TCN在LAMBADA上的困惑度上以显著优势超过了LSTMs和普通RNNs，且网络规模更小，几乎无需调整。（在这个数据集上的最新成果甚至更好，但需要额外的记忆机制的帮助（Grave等人，2017年）。）

# 6. Conclusion

# 6. 结论

We have presented an empirical evaluation of generic convolutional and recurrent architectures across a comprehensive suite of sequence modeling tasks. To this end, we have described a simple temporal convolutional network (TCN) that combines best practices such as dilations and residual connections with the causal convolutions needed for autoregressive prediction. The experimental results indicate that TCN models substantially outperform generic recurrent architectures such as LSTMs and GRUs. We further studied long-range information propagation in convolutional and recurrent networks, and showed that the "infinite memory" advantage of RNNs is largely absent in practice. TCNs exhibit longer memory than recurrent architectures with the same capacity.

我们提出了一项对通用卷积和循环架构在一系列全面的序列建模任务中的实证评估。为此，我们描述了一个简单的时间卷积网络（TCN），它结合了最佳实践，如扩张和残差连接，以及用于自回归预测所需的因果卷积。实验结果表明，TCN模型在性能上显著优于通用的循环架构，如LSTMs和GRUs。我们进一步研究了卷积和循环网络中的长距离信息传播，并表明在实际中，RNNs的“无限记忆”优势在很大程度上是不存在的。与相同容量的循环架构相比，TCN表现出更长的记忆。

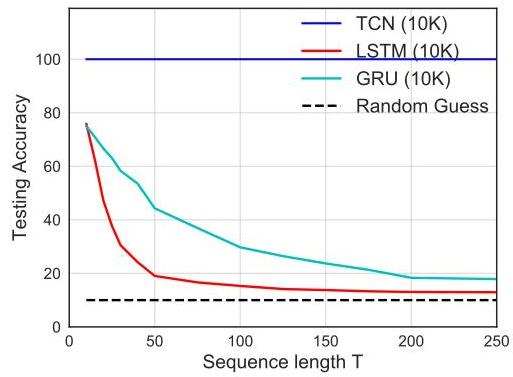


Figure 5. Accuracy on the copy memory task for sequences of different lengths . While TCN exhibits accuracy for all sequence lengths, the LSTM and GRU degenerate to random guessing as grows.

图5. 在不同长度的序列上的复制记忆任务准确性 。当 增长时，TCN对所有序列长度都表现出 的准确性，而LSTM和GRU会退化成随机猜测。

Numerous advanced schemes for regularizing and optimizing LSTMs have been proposed (Press & Wolf, 2016; Krueger et al., 2017; Merity et al., 2017; Campos et al., 2018). These schemes have significantly advanced the accuracy achieved by LSTM-based architectures on some datasets. The TCN has not yet benefitted from this concerted community-wide investment into architectural and algorithmic elaborations. We see such investment as desirable and expect it to yield advances in TCN performance that are commensurate with the advances seen in recent years in LSTM performance. We will release the code for our project to encourage this exploration.

已提出了许多先进的正则化和优化LSTM的方案（Press & Wolf, 2016; Krueger et al., 2017; Merity et al., 2017; Campos et al., 2018）。这些方案显著提高了基于LSTM的架构在某些数据集上达到的准确性。TCN尚未从这种社区范围内的结构化和算法细化投资中受益。我们认为这种投资是值得的，并期望它能带来TCN性能的提升，与近年来LSTM性能的提升相匹配。我们将发布我们项目的代码，以鼓励这种探索。

The preeminence enjoyed by recurrent networks in sequence modeling may be largely a vestige of history. Until recently, before the introduction of architectural elements such as dilated convolutions and residual connections, convolutional architectures were indeed weaker. Our results indicate that with these elements, a simple convolutional architecture is more effective across diverse sequence modeling tasks than recurrent architectures such as LSTMs. Due to the comparable clarity and simplicity of TCNs, we conclude that convolutional networks should be regarded as a natural starting point and a powerful toolkit for sequence modeling.

循环网络在序列建模中的优势可能很大程度上是历史的遗留问题。直到最近，在引入了扩张卷积和残差连接等结构元素之前，卷积架构确实较弱。我们的结果表明，有了这些元素，简单的卷积架构在多种序列建模任务上比LSTM等循环架构更有效。由于TCN的可比性清晰度和简单性，我们得出结论，卷积网络应被视为序列建模的自然起点和强大的工具包。

# References

# 参考文献

Allan, Moray and Williams, Christopher. Harmonising chorales by probabilistic inference. In NIPS, 2005.

Arjovsky, Martin, Shah, Amar, and Bengio, Yoshua. Unitary evolution recurrent neural networks. In ICML, 2016.

Ba, Lei Jimmy, Kiros, Ryan, and Hinton, Geoffrey E. Layer normalization. arXiv:1607.06450, 2016.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. In ICLR, 2015.

Bengio, Yoshua, Simard, Patrice, and Frasconi, Paolo. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 1994.

Bottou, Léon, Soulie, F Fogelman, Blanchet, Pascal, and Liénard, Jean-Sylvain. Speaker-independent isolated digit recognition: Multilayer perceptrons vs. dynamic time warping. Neural Networks, 3(4), 1990.

Boulanger-Lewandowski, Nicolas, Bengio, Yoshua, and Vincent, Pascal. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. arXiv:1206.6392, 2012.

Bradbury, James, Merity, Stephen, Xiong, Caiming, and Socher, Richard. Quasi-recurrent neural networks. In ICLR, 2017.

Campos, Victor, Jou, Brendan, Giró i Nieto, Xavier, Torres, Jordi, and Chang, Shih-Fu. Skip RNN: Learning to skip state updates in recurrent neural networks. In ICLR, 2018.

Chang, Shiyu, Zhang, Yang, Han, Wei, Yu, Mo, Guo, Xiaoxiao, Tan, Wei, Cui, Xiaodong, Witbrock, Michael J., Hasegawa-Johnson, Mark A., and Huang, Thomas S. Dilated recurrent neural networks. In NIPS, 2017.

Cho, Kyunghyun, Van Merriënboer, Bart, Bahdanau, Dzmitry, and Bengio, Yoshua. On the properties of neural machine translation: Encoder-decoder approaches. arXiv:1409.1259, 2014.

Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, and Ben-gio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555, 2014.

Chung, Junyoung, Ahn, Sungjin, and Bengio, Yoshua. Hierarchical multiscale recurrent neural networks. arXiv:1609.01704, 2016.

Collobert, Ronan and Weston, Jason. A unified architecture for natural language processing: Deep neural networks with multitask learning. In ICML, 2008.

Collobert, Ronan, Weston, Jason, Bottou, Léon, Karlen, Michael, Kavukcuoglu, Koray, and Kuksa, Pavel P. Natural language processing (almost) from scratch. JMLR, 12, 2011.

Conneau, Alexis, Schwenk, Holger, LeCun, Yann, and Barrault, Loic. Very deep convolutional networks for text classification. In European Chapter of the Association for Computational Linguistics (EACL), 2017.

Cooijmans, Tim, Ballas, Nicolas, Laurent, César, Gülçehre, Çağlar, and Courville, Aaron. Recurrent batch normalization. In ICLR, 2016.

Dauphin, Yann N., Fan, Angela, Auli, Michael, and Grangier, David. Language modeling with gated convolutional networks. In ICML, 2017.

dos Santos, Cícero Nogueira and Zadrozny, Bianca. Learning character-level representations for part-of-speech tagging. In ICML, 2014.

El Hihi, Salah and Bengio, Yoshua. Hierarchical recurrent neural

networks for long-term dependencies. In NIPS, 1995.

Elman, Jeffrey L. Finding structure in time. Cognitive Science, 14 (2), 1990.

Gehring, Jonas, Auli, Michael, Grangier, David, and Dauphin, Yann. A convolutional encoder model for neural machine translation. In .

Gehring, Jonas, Auli, Michael, Grangier, David, Yarats, Denis, and Dauphin, Yann N. Convolutional sequence to sequence learning. In .

Gers, Felix A, Schraudolph, Nicol N, and Schmidhuber, Jürgen. Learning precise timing with lstm recurrent networks. JMLR, 3, 2002.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. Deep Learning. MIT Press, 2016.

Grave, Edouard, Joulin, Armand, and Usunier, Nicolas. Improving neural language models with a continuous cache. In ICLR, 2017.

Graves, Alex. Supervised Sequence Labelling with Recurrent Neural Networks. Springer, 2012.

Graves, Alex. Generating sequences with recurrent neural networks. arXiv:1308.0850, 2013.

Greff, Klaus, Srivastava, Rupesh Kumar, Koutník, Jan, Steune-brink, Bas R., and Schmidhuber, Jürgen. LSTM: A search space odyssey. IEEE Transactions on Neural Networks and Learning Systems, 28(10), 2017.

Ha, David, Dai, Andrew, and Le, Quoc V. HyperNetworks. In ICLR, 2017.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In CVPR, 2016.

Hermans, Michiel and Schrauwen, Benjamin. Training and analysing deep recurrent neural networks. In NIPS, 2013.

Hinton, Geoffrey E. Connectionist learning procedures. Artificial Intelligence, 40(1-3), 1989.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. Neural Computation, 9(8), 1997.

Jing, Li, Shen, Yichen, Dubcek, Tena, Peurifoy, John, Skirlo, Scott, LeCun, Yann, Tegmark, Max, and Soljačić, Marin. Tunable efficient unitary neural networks (EUNN) and their application to RNNs. In ICML, 2017.

Johnson, Rie and Zhang, Tong. Effective use of word order for text categorization with convolutional neural networks. In HLT-NAACL, 2015.

Johnson, Rie and Zhang, Tong. Deep pyramid convolutional neural networks for text categorization. In .

Jozefowicz, Rafal, Zaremba, Wojciech, and Sutskever, Ilya. An empirical exploration of recurrent network architectures. In ICML, 2015.

Kalchbrenner, Nal, Grefenstette, Edward, and Blunsom, Phil. A convolutional neural network for modelling sentences. In , 2014.

Kalchbrenner, Nal, Espeholt, Lasse, Simonyan, Karen, van den Oord, Aäron, Graves, Alex, and Kavukcuoglu, Koray. Neural machine translation in linear time. arXiv:1610.10099, 2016.

Kim, Yoon. Convolutional neural networks for sentence classification. In EMNLP, 2014.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. In ICLR, 2015.

Koutnik, Jan, Greff, Klaus, Gomez, Faustino, and Schmidhuber, Juergen. A clockwork RNN. In ICML, 2014.

Krueger, David and Memisevic, Roland. Regularizing RNNs by stabilizing activations. arXiv:1511.08400, 2015.

Krueger, David, Maharaj, Tegan, Kramár, János, Pezeshki, Mohammad, Ballas, Nicolas, Ke, Nan Rosemary, Goyal, Anirudh, Bengio, Yoshua, Larochelle, Hugo, Courville, Aaron C., and Pal, Chris. Zoneout: Regularizing RNNs by randomly preserving hidden activations. In ICLR, 2017.

Le, Quoc V, Jaitly, Navdeep, and Hinton, Geoffrey E. A simple way to initialize recurrent networks of rectified linear units. arXiv:1504.00941, 2015.

Lea, Colin, Flynn, Michael D., Vidal, René, Reiter, Austin, and Hager, Gregory D. Temporal convolutional networks for action segmentation and detection. In CVPR, 2017.

LeCun, Yann, Boser, Bernhard, Denker, John S., Henderson, Donnie, Howard, Richard E., Hubbard, Wayne, and Jackel, Lawrence D. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4), 1989.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. Proceedings of the IEEE, .

Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

Marcus, Mitchell P, Marcinkiewicz, Mary Ann, and Santorini, Beatrice. Building a large annotated corpus of English: The Penn treebank. Computational Linguistics, 19(2), 1993.

Martens, James and Sutskever, Ilya. Learning recurrent neural networks with Hessian-free optimization. In ICML, 2011.

Melis, Gábor, Dyer, Chris, and Blunsom, Phil. On the state of the art of evaluation in neural language models. In ICLR, 2018.

Merity, Stephen, Xiong, Caiming, Bradbury, James, and Socher, Richard. Pointer sentinel mixture models. arXiv:1609.07843, 2016.

Merity, Stephen, Keskar, Nitish Shirish, and Socher, Richard. Regularizing and optimizing LSTM language models. arXiv:1708.02182, 2017.

Mikolov, Tomáš, Sutskever, Ilya, Deoras, Anoop, Le, Hai-Son, Kombrink, Stefan, and Cernocky, Jan. Subword language modeling with neural networks. Preprint, 2012.

Miyamoto, Yasumasa and Cho, Kyunghyun. Gated word-character recurrent language model. arXiv:1606.01700, 2016.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted Boltzmann machines. In ICML, 2010.

, Andrew. Sequence Models (Course 5 of Deep Learning Specialization). Coursera, 2018.

Paperno, Denis, Kruszewski, Germán, Lazaridou, Angeliki, Pham, Quan Ngoc, Bernardi, Raffaella, Pezzelle, Sandro, Baroni, Marco, Boleda, Gemma, and Fernández, Raquel. The LAM-BADA dataset: Word prediction requiring a broad discourse context. arXiv:1606.06031, 2016.

Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua. On the difficulty of training recurrent neural networks. In ICML, 2013.

Pascanu, Razvan, Gülçehre, Çaglar, Cho, Kyunghyun, and Bengio, Yoshua. How to construct deep recurrent neural networks. In ICLR, 2014.

Press, Ofir and Wolf, Lior. Using the output embedding to improve

language models. arXiv:1608.05859, 2016.

Salimans, Tim and Kingma, Diederik P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In NIPS, 2016.

Schuster, Mike and Paliwal, Kuldip K. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45 (11), 1997.

Sejnowski, Terrence J. and Rosenberg, Charles R. Parallel networks that learn to pronounce English text. Complex Systems, 1, 1987.

Shi, Xingjian, Chen, Zhourong, Wang, Hao, Yeung, Dit-Yan, Wong, Wai-Kin, and Woo, Wang-chun. Convolutional LSTM network: A machine learning approach for precipitation now-casting. In NIPS, 2015.

Srivastava, Nitish, Hinton, Geoffrey E, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. JMLR, 15(1), 2014.

Subakan, Y Cem and Smaragdis, Paris. Diagonal RNNs in symbolic music modeling. arXiv:1704.05420, 2017.

Sutskever, Ilya, Martens, James, and Hinton, Geoffrey E. Generating text with recurrent neural networks. In ICML, 2011.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In NIPS, 2014.

van den Oord, Aäron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew W., and Kavukcuoglu, Koray. WaveNet: A generative model for raw audio. arXiv:1609.03499, 2016.

Vohra, Raunaq, Goel, Kratarth, and Sahoo, JK. Modeling temporal dependencies in data using a DBN-LSTM. In Data Science and Advanced Analytics (DSAA), 2015.

Waibel, Alex, Hanazawa, Toshiyuki, Hinton, Geoffrey, Shikano, Kiyohiro, and Lang, Kevin J. Phoneme recognition using time-delay neural networks. IEEE Transactions on Acoustics, Speech, and Signal Processing, 37(3), 1989.

Werbos, Paul J. Backpropagation through time: What it does and how to do it. Proceedings of the IEEE, 78(10), 1990.

Wisdom, Scott, Powers, Thomas, Hershey, John, Le Roux, Jonathan, and Atlas, Les. Full-capacity unitary recurrent neural networks. In NIPS, 2016.

Wu, Yuhuai, Zhang, Saizheng, Zhang, Ying, Bengio, Yoshua, and Salakhutdinov, Ruslan R. On multiplicative integration with recurrent neural networks. In NIPS, 2016.

Yang, Zhilin, Dai, Zihang, Salakhutdinov, Ruslan, and Cohen, William W. Breaking the softmax bottleneck: A high-rank RNN language model. ICLR, 2018.

Yin, Wenpeng, Kann, Katharina, Yu, Mo, and Schütze, Hinrich. Comparative study of CNN and RNN for natural language processing. arXiv:1702.01923, 2017.

Yu, Fisher and Koltun, Vladlen. Multi-scale context aggregation by dilated convolutions. In ICLR, 2016.

Zhang, Saizheng, Wu, Yuhuai, Che, Tong, Lin, Zhouhan, Memi-sevic, Roland, Salakhutdinov, Ruslan R, and Bengio, Yoshua. Architectural complexity measures of recurrent neural networks. In NIPS, 2016.

Zhang, Xiang, Zhao, Junbo Jake, and LeCun, Yann. Character-level convolutional networks for text classification. In NIPS, 2015.

# An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling Supplementary Material

# 序列建模中通用卷积和循环网络的实证评估 补充材料

# A. Hyperparameters Settings

# A. 超参数设置

# A.1. Hyperparameters for TCN

# A.1. TCN的超参数

Table 2 lists the hyperparameters we used when applying the generic TCN model on various tasks and datasets. The most important factor for picking parameters is to make sure that the TCN has a sufficiently large receptive field by choosing and that can cover the amount of context needed for the task.

表2列出了我们在将通用TCN模型应用于各种任务和数据集时使用的超参数。选择参数最重要的因素是确保通过选择 和 使得TCN具有足够大的感受野，能够覆盖任务所需的上下文数量。

As discussed in Section 5, the number of hidden units was chosen so that the model size is approximately at the same level as the recurrent models with which we are comparing. In Table 2, a gradient clip of N/A means no gradient clipping was applied. In larger tasks (e.g., language modeling), we empirically found that gradient clipping (we randomly picked a threshold from ) helps with regularizing TCN and accelerating convergence.

如第5节所述，选择隐藏单元的数量是为了使模型大小大约与我们所比较的循环模型相当。在表2中，梯度裁剪值为N/A表示未应用梯度裁剪。在更大的任务（例如，语言建模）中，我们经验性地发现，梯度裁剪（我们从 中随机选择一个阈值）有助于规范TCN并加速收敛。

All weights were initialized from a Gaussian disitribution . In general, we found TCN to be relatively insensitive to hyperparameter changes, as long as the effective history (i.e., receptive field) size is sufficient.

所有权重均从高斯分布 中初始化。总的来说，我们发现只要有效历史（即，感受野）大小足够，TCN对超参数变化相对不敏感。

# A.2. Hyperparameters for LSTM/GRU

# A.2. LSTM/GRU的超参数

Table 3 reports hyperparameter settings that were used for the LSTM. These values are picked from hyperparameter search for LSTMs that have up to 3 layers, and the optimizers are chosen from {SGD, Adam, RMSprop, Adagrad }. For certain larger datasets, we adopted the settings used in prior work (e.g., Grave et al. (2017) on Wikitext-103). GRU hyperparameters were chosen in a similar fashion, but typically with more hidden units than in LSTM to keep the total network size approximately the same (since a GRU cell is more compact).

表3报告了用于LSTM的超参数设置。这些值是从具有最多3层的LSTM的超参数搜索中选取的，优化器则从 {SGD, Adam, RMSprop, Adagrad } 中选择。对于某些更大的数据集，我们采用了之前工作中的设置（例如，Grave等人（2017年）在Wikitext-103上的设置）。GRU的超参数也是以类似方式选择的，但通常比LSTM有更多的隐藏单元，以保持总的网络大小大致相同（因为GRU单元更加紧凑）。

# B. State-of-the-Art Results

# B. 最先进的结果

As previously noted, the generic TCN and LSTM/GRU models we used can be outperformed by more specialized architectures on some tasks. State-of-the-art results are summarized in Table 4. The same TCN architecture is used across all tasks. Note that the size of the state-of-the-art model may be different from the size of the TCN.

如前所述，我们使用的通用TCN和LSTM/GRU模型在某些任务上可能会被更专业的架构超越。最先进的结果在表4中总结。所有任务都使用相同的TCN架构。请注意，最先进模型的大小可能与TCN的大小不同。

# C. Effect of Filter Size and Residual Block

# C. 滤波器大小和残差块的影响

In this section we briefly study the effects of different components of a TCN layer. Overall, we believe dilation is required for modeling long-term dependencies, and so we mainly focus on two other factors here: the filter size used by each layer, and the effect of residual blocks.

在本节中，我们简要研究 TCN 层不同组件的影响。总体而言，我们认为扩张是建模长期依赖所必需的，因此我们主要关注这里的两个其他因素：每层使用的滤波器大小 ，以及残差块的影响。

We perform a series of controlled experiments, with the results of the ablative analysis shown in Figure 6. As before, we kept the model size and depth exactly the same for different models, so that the dilation factor is strictly controlled. The experiments were conducted on three different tasks: copy memory, permuted MNIST (P-MNIST), and Penn Treebank word-level language modeling. These experiments confirm that both factors (filter size and residual connections) contribute to sequence modeling performance.

我们进行了一系列受控实验，消融分析的结果显示在图 6 中。与之前一样，我们为不同的模型保持了完全相同的模型大小和深度，以便严格地控制扩张因子。实验在三项不同的任务上进行：复制记忆，置换 MNIST（P-MNIST）和 Penn Treebank 词级语言建模。这些实验证实了这两个因素（滤波器大小和残差连接）都有助于序列建模性能。

Filter size . In both the copy memory and the P-MNIST tasks, we observed faster convergence and better accuracy for larger filter sizes. In particular, looking at Figure 6a, a TCN with filter size only converges to the same level as random guessing. In contrast, on word-level language modeling, a smaller kernel with filter size of works best. We believe this is because a smaller kernel (along with fixed dilation) tends to focus more on the local context, which is especially important for PTB language modeling (in fact, the very success of -gram models suggests that only a relatively short memory is needed for modeling language).

滤波器大小 。在复制记忆和 P-MNIST 任务中，我们观察到较大滤波器尺寸的收敛速度更快且准确度更高。特别是观察图 6a，使用滤波器大小 的 TCN 仅收敛到与随机猜测相同的水平。相比之下，在词级语言建模中，较小的核，滤波器大小为 的效果最好。我们认为这是因为较小的核（以及固定的扩张）倾向于更多地关注局部上下文，这对于 PTB 语言建模尤为重要（实际上， -gram 模型的巨大成功表明，建模语言只需要相对较短的记忆）。

Residual block. In all three scenarios that we compared here, we observed that the residual function stabilized training and brought faster convergence with better final results. Especially in language modeling, we found that residual connections contribute substantially to performance (See Figure 6f).

残差块。在我们比较的所有三种场景中，我们观察到残差函数稳定了训练过程，并带来了更快的收敛速度和更好的最终结果。特别是在语言建模中，我们发现残差连接对性能的贡献很大（见图 6f）。

# D. Gating Mechanisms

# D. 门控机制

One component that had been used in prior work on convolutional architectures for language modeling is the gated activation (van den Oord et al., 2016; Dauphin et al., 2017). We have chosen not to use gating in the generic TCN model. We now examine this choice more closely.

在之前关于卷积架构用于语言建模的工作中，一个被使用的组件是门控激活（van den Oord等人，2016年；Dauphin等人，2017年）。我们选择在通用的TCN模型中不使用门控。我们现在更仔细地研究这个选择。

Table 2. TCN parameter settings for experiments in Section 5.

表2. 第5节实验中TCN的参数设置。

| TCN SETTINGS | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Dataset/Task | Subtask |  |  | Hidden | Dropout | Grad Clip | Note |
| The Adding Problem |  | 6 | 7 | 27 | 0.0 | N/A |  |
|  | 7 | 7 | 27 |
|  | 8 | 8 | 24 |
| Seq. MNIST | - | 7 | 8 | 25 | 0.0 | N/A |  |
| 6 | 8 | 20 |
| Permuted MNIST | - | 7 | 8 | 25 | 0.0 | N/A |  |
| 6 | 8 | 20 |
| Copy Memory Task |  | 6 | 9 | 10 | 0.05 | 1.0 | RMSprop 5e-4 |
|  | 8 | 8 | 10 |
|  | 8 | 9 | 10 |
| Music JSB Chorales | - | 3 | 2 | 150 | 0.5 | 0.4 |  |
| Music Nottingham | - | 6 | 4 | 150 | 0.2 | 0.4 |  |
| Word-level LM | PTB | 3 | 4 | 600 | 0.5 | 0.4 | Embed. size 600 Embed. size 400 Embed. size 500 |
| Wiki-103 | 3 | 5 | 1000 | 0.4 |
| LAMBADA | 4 | 5 | 500 |
| Char-level LM | PTB | 3 | 3 | 450 | 0.1 | 0.15 | Embed. size 100 |
| text8 | 2 | 5 | 520 |

Dauphin et al. (2017) compared the effects of gated linear units (GLU) and gated tanh units (GTU), and adopted GLU in their non-dilated gated ConvNet. Following the same choice, we now compare TCNs using ReLU and TCNs with gating (GLU), represented by an elementwise product between two convolutional layers, with one of them also passing through a sigmoid function . Note that the gates architecture uses approximately twice as many convolutional layers as the ReLU-TCN.

Dauphin等人（2017年）比较了门控线性单元（GLU）和门控双曲正切单元（GTU）的效果，并在他们的非扩张门控卷积网络中采用了GLU。遵循同样的选择，我们现在比较使用ReLU的TCN和带门控（GLU）的TCN（由两层卷积层之间的逐元素乘积表示，其中一个层还通过sigmoid函数 ）。注意，门控架构使用的卷积层大约是ReLU-TCN的两倍。

The results are shown in Table 5, where we kept the number of model parameters at about the same size. The GLU does further improve TCN accuracy on certain language modeling datasets like PTB, which agrees with prior work. However, we do not observe comparable benefits on other tasks, such as polyphonic music modeling or synthetic stress tests that require longer information retention. On the copy memory task with , we found that TCN with gating converged to a worse result than TCN with ReLU (though still better than recurrent models).

结果显示在表5中，我们保持了模型参数的数量大约相同。GLU进一步提高了TCN在某些语言建模数据集（如PTB）上的准确性，这与之前的工作一致。然而，我们发现在其他任务上，例如需要更长时间信息保留的多声部音乐建模或合成压力测试上，并没有观察到可比较的好处。在复制记忆任务 中，我们发现带门控的TCN比带ReLU的TCN收敛到更差的结果（尽管仍然比循环模型好）。

Table 3. LSTM parameter settings for experiments in Section 5.

表3. 第5节实验中LSTM的参数设置。

| LSTM Settings (Key Parameters) | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Dataset/Task | Subtask |  | Hidden | Dropout | Grad Clip | Bias | Note |
| The Adding Problem |  | 2 | 77 |  | 50 | 5.0 | SGD 1e-3 |
|  | 2 | 77 | 0.0 | 50 | 10.0 | Adam 2e-3 |
|  | 1 | 130 |  | 5 | 1.0 | - |
| Seq. MNIST | - | 1 | 130 | 0.0 | 1 | 1.0 | RMSprop 1e-3 |
| Permuted MNIST | - | 1 | 130 | 0.0 | 1 | 10.0 | RMSprop 1e-3 |
| Copy Memory Task |  | 1 | 50 | 0.05 | 0.25 | - | RMSprop/Adam |
|  | 1 | 50 | 1 |
|  | 3 | 28 | 1 |
| Music JSB Chorales | - | 2 | 200 | 0.2 | 1 | 10.0 | SGD/Adam |
| Music Nottingham | - | 3 | 280 | 0.1 | 0.5 | - | Adam 4e-3 |
| 1 | 500 |  | 1 | - |
| Word-level LM | PTB | 3 | 700 | 0.4 | 0.3 | 1.0 | SGD 30, Emb. 700, etc. |
| Wiki-103 | - | - | - | - | - | Grave et al. (2017) |
| LAMBADA | 1 | - | - |  | - | Grave et al. (2017) |
| Char-level LM | PTB | 2 | 600 | 0.1 | 0.5 | - | Emb. size 120 |
| text8 | 1 | 1024 | 0.15 | 0.5 | - | Adam 1e-2 |

Table 4. State-of-the-art (SoTA) results for tasks in Section 5.

表4. 第5节任务中的最新（SoTA）结果。

| TCN vs. SoTA Results | | | | | |
| --- | --- | --- | --- | --- | --- |
| Task | TCN Result | Size | SoTA | Size | Model |
| Seq. MNIST (acc.) | 99.0 |  | 99.0 |  | Dilated GRU (Chang et al., 2017) |
| P-MNIST (acc.) | 97.2 | 42K | 95.9 | 42K | Zoneout (Krueger et al., 2017) |
| Adding Prob. 600 (loss) |  | 70K | 5.3e-5 | 70K | Regularized GRU |
| Copy Memory 1000 (loss) |  | 70K | 0.011 |  | EURNN (Jing et al., 2017) |
| JSB Chorales (loss) | 8.10 |  | 3.47 | - | DBN+LSTM (Vohra et al., 2015) |
| Nottingham (loss) | 3.07 |  | 1.32 | - | DBN+LSTM (Vohra et al., 2015) |
| Word PTB (ppl) | 88.68 | 13M | 47.7 | 22M | AWD-LSTM-MoS + Dynamic Eval. (Yang et al., 2018) |
| Word Wiki-103 (ppl) | 45.19 |  | 40.4 | >300M | Neural Cache Model (Large) (Grave et al., 2017) |
| Word LAMBADA (ppl) | 1279 | 56M | 138 | >100M | Neural Cache Model (Large) (Grave et al., 2017) |
| Char PTB (bpc) | 1.31 | 3M | 1.22 |  | 2-LayerNorm HyperLSTM (Ha et al., 2017) |
| Char text8 (bpc) | 1.45 | 4.6M | 1.29 | >12M | HM-LSTM (Chung et al., 2016) |

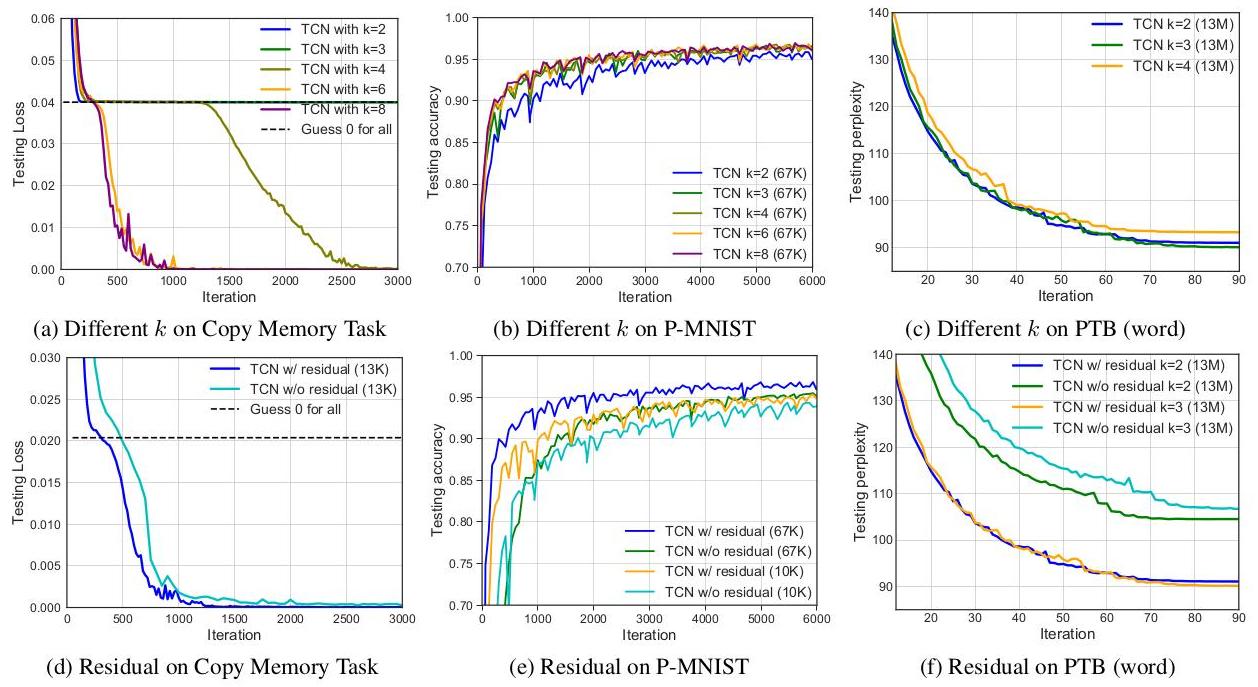


Figure 6. Controlled experiments that study the effect of different components of the TCN model.

图6. 控制实验，研究TCN模型不同组件的效果。

Table 5. An evaluation of gating in TCN. A plain TCN is compared to a TCN that uses gated activations.

表5. TCN中门控的评价。将普通TCN与使用门控激活的TCN进行比较。

| Task | TCN | TCN + Gating |
| --- | --- | --- |
| Sequential MNIST (acc.) | 99.0 | 99.0 |
| Permuted MNIST (acc.) | 97.2 | 96.9 |
| Adding Problem (loss) | 5.8e-5 | 5.6e-5 |
| Copy Memory (loss) | 3.5e-5 | 0.00508 |
| JSB Chorales (loss) | 8.10 | 8.13 |
| Nottingham (loss) | 3.07 | 3.12 |
| Word-level PTB (ppl) | 88.68 | 87.94 |
| Char-level PTB (bpc) | 1.31 | 1.306 |
| Char text8 (bpc) | 1.45 | 1.485 |