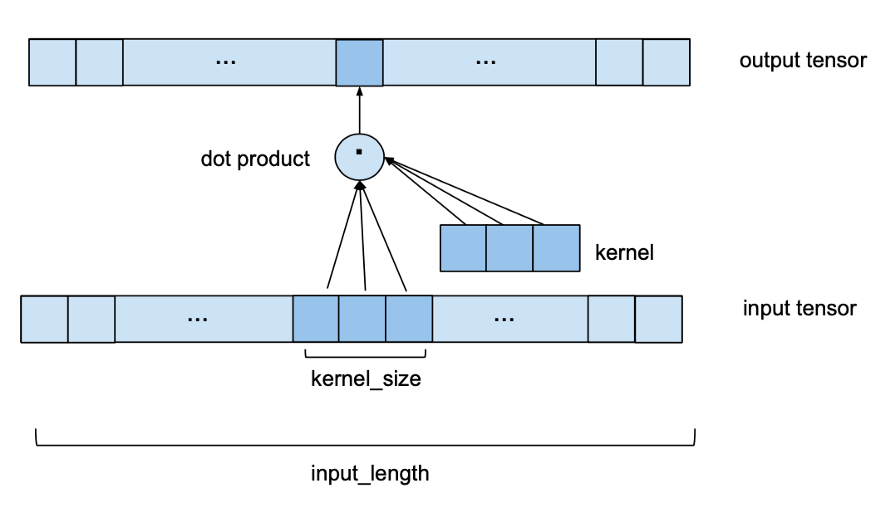
TCN

总结来讲，时间卷积网络是：

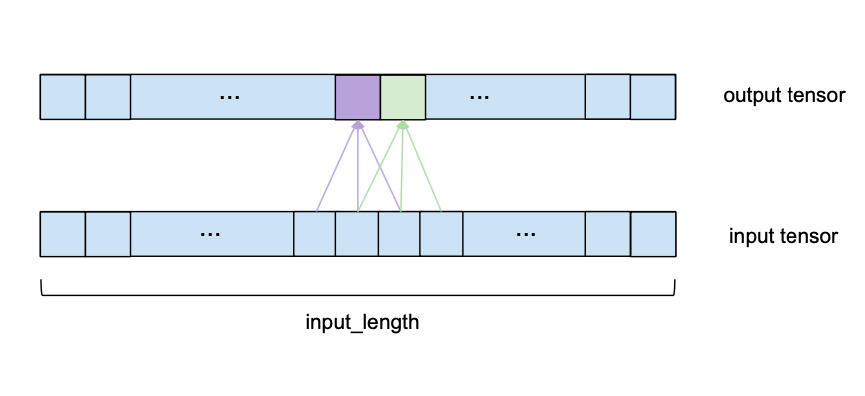
同时用到一维因果卷积和扩张卷积作为标准卷积层，并将每两个这样的卷积层与恒等映射可以封装为一个残差模块（包含了relu函数）。再由残差模块堆叠起深度网络，并在最后几层使用全卷积层代替全连接层。

**一维卷积网络**

单个1D卷积层接收一个输入张量，形状为（batch\_size、input\_length、input\_size）并输出一个张量，形状为（batch\_size、input\_length、output\_size）。为了了解单个层如何将其输入转换为输出，让我们看一下批处理的一个元素(对批处理中的每个元素都进行相同的处理)。让我们从最简单的例子开始，其中input\_size和output\_size都等于1。在这种情况下，我们看到的是一维输入和输出张量。下图显示了输出张量的一个元素是如何计算的。

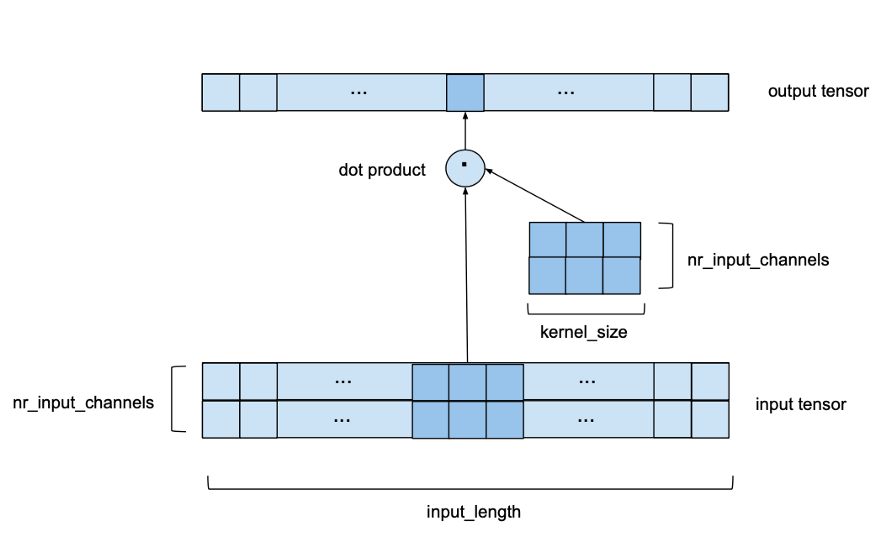


我们可以看到，要计算输出的一个元素，我们需要查看输入的一系列长度为kernel\_size的连续元素。在上面的例子中，我们选择了一个大小为3的kernel\_size。为了得到输出，我们取输入的子序列和相同长度的已学习权值的核向量的点积。输出的下一个元素,相同的应用程序,但kernel\_size窗口的输入序列是由一个元素转移到正确的(对于本预测模型,步幅总是设置为1)。请注意,相同的一组内核权重将被用来计算每输出一个卷积层。下图显示了两个连续的输出元素及其各自的输入子序列。



为了确保输出序列的长度与输入序列相同，应用了一些零填充。

现在让我们看看有多个输入通道的情况，即nr\_input\_channels大于1。在本例中，上面描述的过程对每个输入通道重复，但每次使用不同的核。这计算出nr\_input\_channels中间输出向量和kernel\_size \* nr\_input\_channels的一些内核权值。然后，将所有中间输出向量相加，得到最终输出向量。在某种意义上，这等价于一个2D卷积与形状的输入张量(input\_size, nr\_input\_channels)和形状的核(kernel\_size, nr\_input\_channels)，如下图所示。它仍然是一维的，因为窗口只沿着一个轴移动，但我们确实在每一步都有一个二维卷积因为我们使用了一个二维核矩阵。



在此示例中，我们选择nr\_input\_channels等于 2。现在，我们可以通过kernel\_size内核矩阵沿着nr\_input\_channels宽系列的长度input\_length滑动来nr\_input\_channels，而不是在 1 维输入序列上滑动的内核载体。

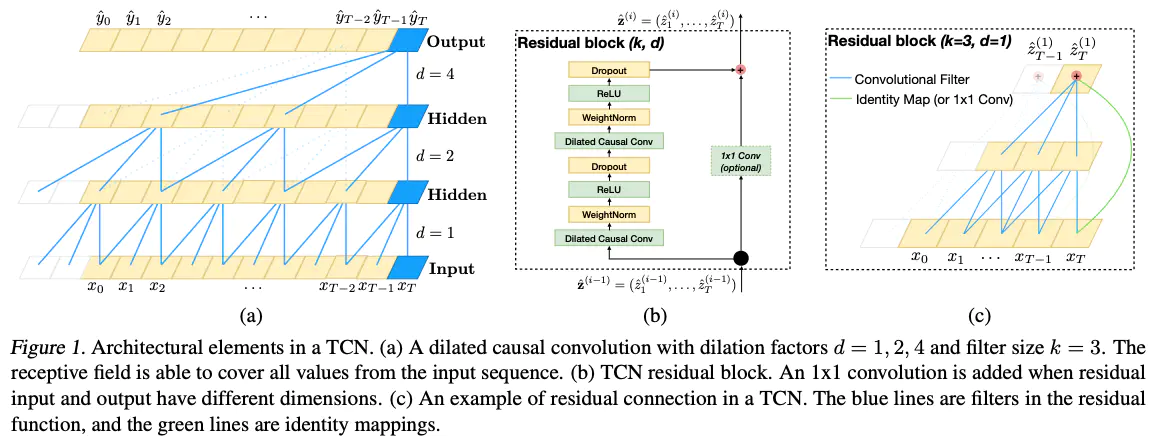
如果nr\_input\_channels和nr\_output\_channels都大于 1，则只需重复使用具有不同内核矩阵的每个输出通道的上述过程。然后，输出向量相互堆叠，形成形状的输出张量（input\_length，nr\_output\_channels）。 在这种情况下，内核权重的数量等于kernel\_size\*nr\_input\_channels\*nr\_output\_channels。

nr\_input\_channels和nr\_output\_channels两个变量取决于网络中层的位置。第一层将具有nr\_input\_channels = input\_size，最后一层将具有nr\_output\_channels = output\_size。所有其他层将使用num\_filters提供的中间通道数。

**扩张卷积**

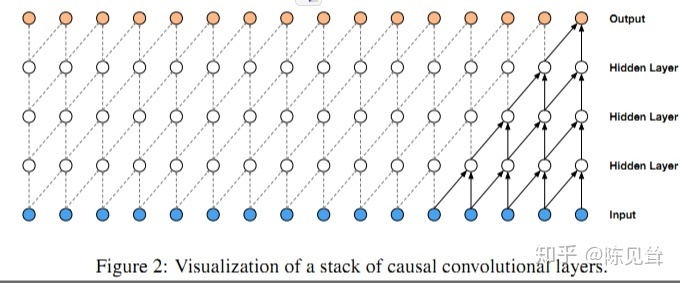
扩张卷积（dilated convolution）与普通的卷积相比，除了卷积核的大小以外，还有一个扩张率(dilation rate)参数，主要用来表示扩张的大小。

扩张卷积与普通卷积的相同点在于，卷积核的大小是一样的，神经网络中的参数数量不变。两者区别区别在于扩张卷积具有更大的感受野。它通过跳过部分输入来使卷积核可以应用于大于卷积核本身长度的区域。等同于通过增加零来从原始卷积核中生成更大的卷积核。



扩张卷积的好处是不做pooling损失信息的情况下，加大了感受野，让每个卷积输出都包含较大范围的信息。

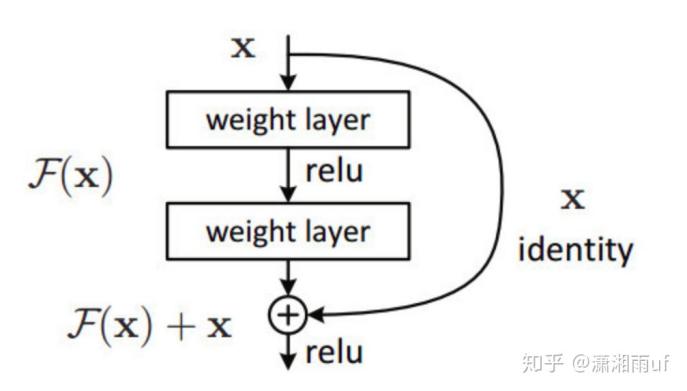
**因果卷积**



因果卷积（Causal Convolution）可以用上图直观表示。 即对于上一层t时刻的值，只依赖于下一层t时刻及其之前的值。和传统的卷积神经网络的不同之处在于，因果卷积不能看到未来的数据，它是单向的结构，不是双向的。也就是说只有有了前面的因才有后面的果，是一种严格的时间约束模型，因此被成为因果卷积。

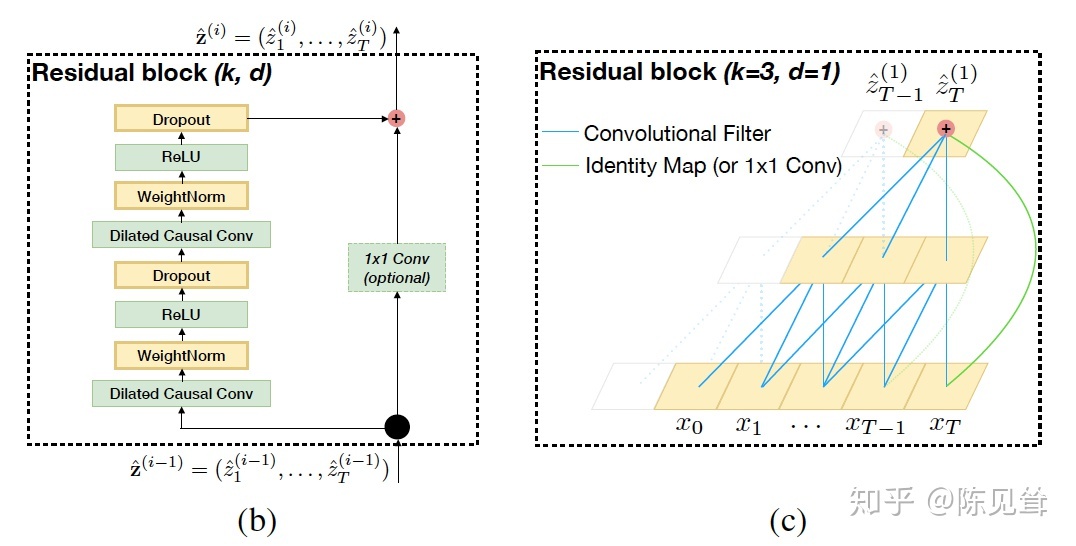
**残差卷积的跳层连接**

残差网络 ResNet 就是经典的跳层连接（skip-connection），如下图所示。



上一层的特征图 x 直接与卷积后的 F(x)对齐加和，变为 F(x)+x （特征图数量不够可用 0 特征补齐，特征图大小不一可用带步长卷积做下采样）。

这样在每层特征图中添加上一层的特征信息，可使网络更深，加快反馈与收敛。

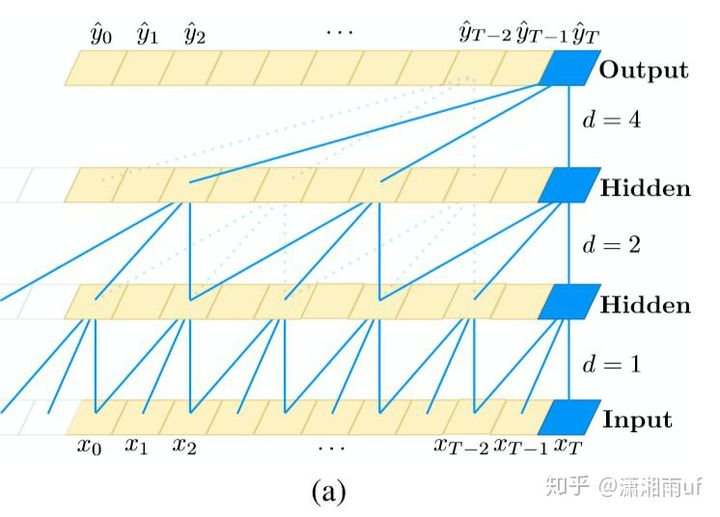


本文构建了一个残差块来代替一层的卷积。如上图所示，一个残差块包含两层的卷积和非线性映射，在每层中还加入了WeightNorm和Dropout来正则化网络。

**全卷积网络**

FCN与经典的CNN在卷积层之后使用全连接层得到固定长度的特征向量进行分类不同，FCN可以接受任意尺寸的输入，采用卷积层对最后一个卷积层的feature map进行上采样, 使它恢复到与输入相同的尺寸，再进行预测。

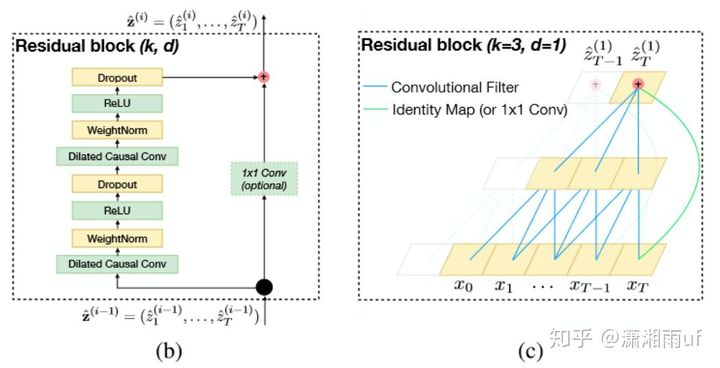
**整体架构**



1. 图为空洞系数 d=1, 2, 4、卷积核大小 k=3 的**扩张卷积**，感受野能覆盖输入序列中的所有值。



上式为扩张卷积的计算公式，d为扩展系数。



（b）为 TCN **残差块**，当残差输入和输出有不同的维度，会往其中添加一个 1x1 的卷积。

（c）为 残差连接的示例，蓝线为残差函数中的卷积核，绿线为恒等映射。

TCN 的卷积层结合了**扩张卷积**与**因果卷积**两种结构。使用因果卷积的目的是为了保证前面时间步的预测不会使用未来的信息，因为时间步 t 的输出只会根据 t-1 及之前时间步上的卷积运算得出。

可以看出，TCN的卷积和普通的**一维卷积**非常类似，只不过最大的不同是用了扩张卷积，随着层数越多，卷积窗口越大，卷积窗口中的空孔会越多。

值得一提的是在 TCN 的**残差模块**内(即图b)有两层扩张卷积和 ReLU 非线性函数，且卷积核的权重都经过了权重归一化。此外TCN 在残差模块内的每个空洞卷积后都添加了 Dropout 以实现正则化。

论文中跳层连接时直接将下层的特征图跳层连接到上层，这样的话对应的通道数channel不一致，所以不能直接做加和操作，于是，为了两个层加和时特征图数量，即通道数数量相同，作者通过用**1×1卷积**进行元素合并来保证两个张量的形状相同。

**TCN的优点**

    （1）并行性。当给定一个句子时，TCN可以将句子并行的处理，而不需要像RNN那样顺序的处理。

    （2）灵活的感受野。TCN的感受野的大小受层数、卷积核大小、扩张系数等决定。可以根据不同的任务不同的特性灵活定制。

    （3）稳定的梯度。RNN经常存在梯度消失和梯度爆炸的问题，这主要是由不同时间段上共用参数导致的，和传统卷积神经网络一样，TCN不太存在梯度消失和爆炸问题。

    （4）内存更低。RNN在使用时需要将每步的信息都保存下来，这会占据大量的内存，TCN在一层里面卷积核是共享的，内存使用更低。

**TCN的缺点**

    （1）TCN 在迁移学习方面可能没有那么强的适应能力。这是因为在不同的领域，模型预测所需要的历史信息量可能是不同的。因此，在将一个模型从一个对记忆信息需求量少的问题迁移到一个需要更长记忆的问题上时，TCN 可能会表现得很差，因为其感受野不够大。

    （2）论文中描述的TCN还是一种单向的结构，在语音识别和语音合成等任务上，纯单向的结构还是相当有用的。但是在文本中大多使用双向的结构，当然将TCN也很容易扩展成双向的结构，不使用因果卷积，使用传统的卷积结构即可。

    （3）TCN毕竟是卷积神经网络的变种，虽然使用扩展卷积可以扩大感受野，但是仍然受到限制，相比于Transformer那种可以任意长度的相关信息都可以抓取到的特性还是差了点。TCN在文本中的应用还有待检验。

**N-BEATS网络**

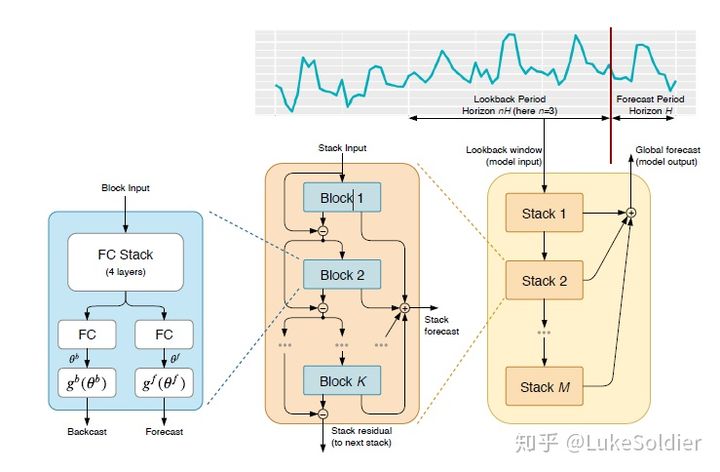


图1：N-BEATS网络结构图

该网络实现单变量多对多预测，网络结构有两个作用，一个叫复原，一个是预测，复原的作用是进行有效的时间序列特征提取，发挥深度学习自动特征学习的优势。整个网络的Block的设计与迭代决策树的思想类似，用网络学习残差，先让神经网络学习简单的规律，再学习复杂的规律。

**Block内部结构**：每个block分为两部分，第一部分用于生成后项展开系数(backward expansion coefficients)和前项展开系数(forward expansion coefficients)，原始时间序列输入经过四层全连接后，经过线性投影层生成后项展开系数和前项展开系数，线性投影层就是简单的线性变换，例如。

第二部分将后项展开系数和前项展开系数通过basis layers输出结果





**Block间连接**：不同bock之间通过残差模块进行衔接。公式如下所示，其实就是下一个block目标是学习上一个block转换得到的残差。



**模型可解释性**：



在可解释性层面，作者提出两种结构，一种generic architecture不依赖于特定经验知识，该变化可以看做在时间的波形重建，由于没有额外的限制在上 ，导致生成的解释性比较差。另一种则通过引入专家经验来满足可解释性要求，即通过引入时间序列的趋势性(trend)和季节性(seasonality)来实现可解释性。具体方法为人为设定与相乘的矩阵使得该矩阵满足趋势性和季节性的要求。这方面带来的启示是，**可以通过设计相乘矩阵的数值形式来实现专家经验的引入。**



**模型集成：**最后通过两种类型的ensemble进行结果的输出,SMAPE, MASE和MAPE, SPAME四种优化目标融合，以及不同时间窗口进行融合。