Paper Notes：An empirical genetic convolution and recurrent network for sequence modeling

整理：ShaoXuan Huang

> Paper程式碼參考實現：https://github.com/locuslab/TCN

1.解決的問題：time series discovery

- TCN：Temporal Convolution Network （時間卷積網絡），這篇2018年出的，算是TCN的開端 (citation: 203)
- 希望建立的是通用解，主要應用在 NLP、語音合成等任務（Sequence Modeling）

2. 如何解

- TCN = 1D FCN (Fully Convolution Network）全連接層 + Causal Network 因果卷積
- 模型設計：
    - (1) 序列建模：Causal Network（定義：yt 只受過去 {Xt, Xt-1, Xt-2, ...} 影響，並不受 {Xt+1, Xt+2, ...} 影響
    - (2) 對歷史有記憶：Dialated Network 擴張卷積、Residual Block 殘差層
    - (3) 透過以下兩個參數可以彈性調整感受野的大小（receptive field size）
        - d ：擴張卷積、 以 2 的指數遞增，決定在層的級別隔幾個空格傳遞至下一層
        - n：filter size、決定每一層之間取幾個值傳到下一層（下圖每一層皆為 k=3）
    - (4) 確保輸入輸出維度一致：Fully Convolution Network

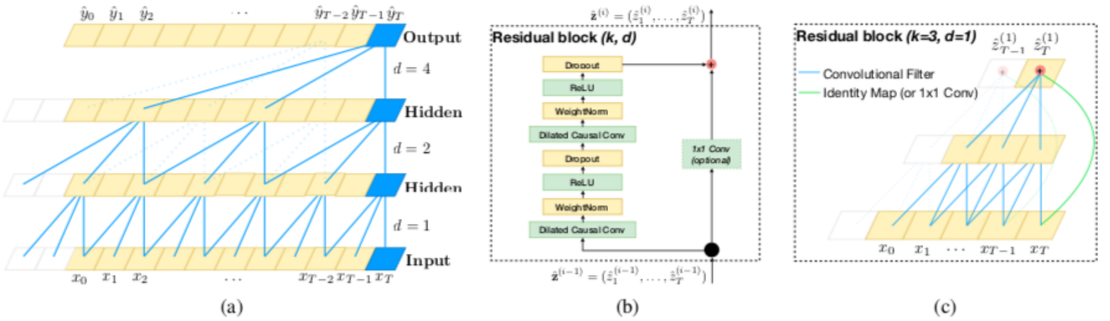

$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ ... $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 ... x_{T-2} x_{T-1} x_T
Residual block (k, d)
Dropout
ReLU
WeightNorm
Dilated Causal Conv
1x1 Conv (optional)
$\hat{\mathbf{z}}^{(i)} = (\hat{z}_1^{(i)}, \ldots, \hat{z}_T^{(i)})$
$\hat{\mathbf{z}}^{(i-1)} = (\hat{z}_1^{(i-1)}, \ldots, \hat{z}_T^{(i-1)})$
Residual block (k=3, d=1)
Convolutional Filter
Identity Map (or 1x1 Conv)
$\hat{z}_{T-1}^{(1)}$ $\hat{z}_T^{(1)}$
x_0 x_1 ... x_{T-1} x_T
(a)
(b)
(c)

Figure 1. Architectural elements in a TCN. (a) A dilated causal convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. The receptive field is able to cover all values from the input sequence. (b) TCN residual block. An 1x1 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.

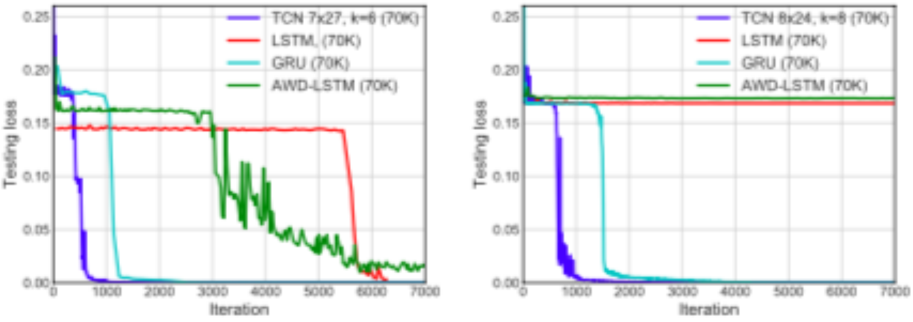
- 模型特點：
    - 其實TCN只是一維卷積變形之後在時序問題上變得適用。
    - 擴張卷積和普通1D卷積最大的不同：越到上層，卷積窗口越大，而卷積窗口中的“空孔”越多；擴張卷積可以做到每一層隱層都和輸入序列大小一樣，計算量降低，感受野足夠大。
    - 因果卷積：時序預測要求對時刻 t 的預測yt 只能通過t時刻之前的輸入x1到xt-1來判別（e.g. 隱馬爾科夫鏈）。
    - TCN還為了提高準確率，還加入了殘差卷積的跳層連接，以及1×1的卷積操作。

      source 参考：時間卷積網絡TCN總結

3. 如何衡量此解法的好壞

論文裏面根據不同的任務，主要都是與 LSTM 與 GRU 比較，TCN 皆取得了較好的成果：

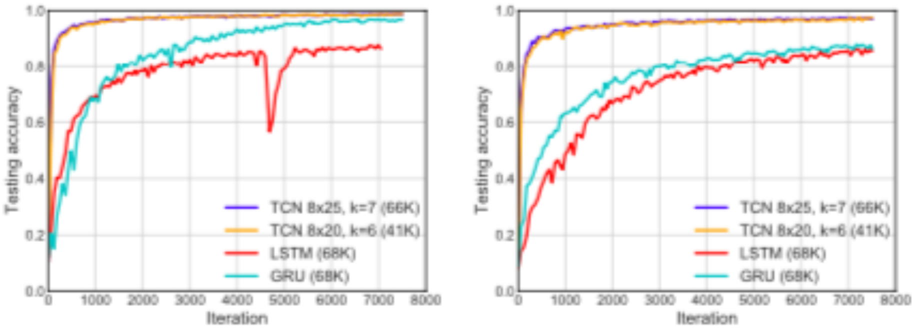
- 隨著序列長度增加，TCN 可以更快收斂



(a) $T = 200$ (b) $T = 600$

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

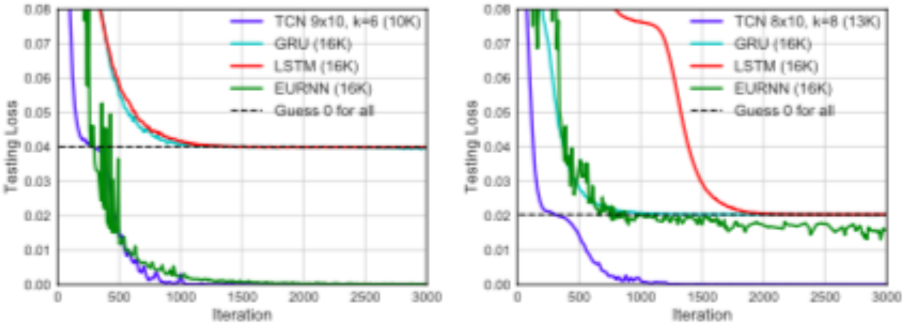
- 隨著迭代次數增加，TCN可以達到較高的準確率 （MNIST & P-MNIST)



(a) Sequential MNIST (b) P-MNIST

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

- 在效能上也可以吃較少的內存：是由於 TCN 基於卷積的關系，有共享參數



(a) $T = 500$ (b) $T = 1000$

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

Python 程式碼

- 使用 keras-tcn repo

Install

```
pip install keras-tcn
from tcn import compiled_tcn
```

[參考來源： Keras-TCN @Github]

TCN Model

```
class TCN:
    """Creates a TCN layer.

        Input shape:
            A tensor of shape (batch_size, timesteps, input_dim).

        Args:
            nb_filters: The number of filters to use in the convolutional layers.
            kernel_size: The size of the kernel to use in each convolutional layer.
            dilations: The list of the dilations. Example is: [1, 2, 4, 8, 16, 32, 64].
            nb_stacks : The number of stacks of residual blocks to use.
            padding: The padding to use in the convolutional layers, 'causal' or 'same'.
            use_skip_connections: Boolean. If we want to add skip connections from input to each residual block.
            return_sequences: Boolean. Whether to return the last output in the output sequence, or the full sequence.
            activation: The activation used in the residual blocks o = Activation(x + F(x)).
            dropout_rate: Float between 0 and 1. Fraction of the input units to drop.
            name: Name of the model. Useful when having multiple TCN.
            kernel_initializer: Initializer for the kernel weights matrix (Conv1D).
            use_batch_norm: Whether to use batch normalization in the residual layers or not.

        Returns:
            A TCN layer.
        """

    def __init__(self,
                 nb_filters=64,
                 kernel_size=2,
                 nb_stacks=1,
                 dilations=[1, 2, 4, 8, 16, 32],
                 padding='causal',
                 use_skip_connections=True,
                 dropout_rate=0.0,
                 return_sequences=False,
                 activation='linear',
                 name='tcn',
                 kernel_initializer='he_normal',
                 use_batch_norm=False):
        self.name = name
        self.return_sequences = return_sequences
        self.dropout_rate = dropout_rate
        self.use_skip_connections = use_skip_connections
        self.dilations = dilations
        self.nb_stacks = nb_stacks
        self.kernel_size = kernel_size
        self.nb_filters = nb_filters
        self.activation = activation
        self.padding = padding
        self.kernel_initializer = kernel_initializer
        self.use_batch_norm = use_batch_norm

        if padding != 'causal' and padding != 'same':
            raise ValueError("Only 'causal' or 'same' padding are compatible for this layer.")

        if not isinstance(nb_filters, int):
            print('An interface change occurred after the version 2.1.2.')
            print('Before: tcn.TCN(x, return_sequences=False, ...)')
            print('Now should be: tcn.TCN(return_sequences=False, ...)(x)')
            print('The alternative is to downgrade to 2.1.2 (pip install keras-tcn==2.1.2).')
            raise Exception()
```

Structure

```
├─ src
|     ├──main.py         # 主要运行程序
└──   └──utils.py        # 工具(分桶, 计算ks与iv值)
```