02456 DEEP LEARNING, DTU COMPUTE, FALL 2018

RNA SUB-CELLULAR LOCALIZATION USING QUASI-RECURRENT NEURAL NETWORKS AND TEMPORAL CONVOLUTIONAL NETWORKS

Alessandro Montemurro [a], Léa Riera [a], Niels Mølgaard Knudsen [b]

[a] DTU Compute, [b] DTU Bioinformatics
Technical University of Denmark

Abbreviations

1. INTRODUCTION

It is becoming increasingly apparent that the subcellular localization of RNA molecules is of huge importance. Localization of mRNA allows for regulation by spatially restricting production of proteins. Extracellular microRNA (miRNA) have recently seen a surge in interest due to them potentially being cancer biomarkers [1]. In mammals, the localization of long non-coding RNAs (lncRNAs) has been shown to be an important regulator of neural cell growth and size [2]. This also means that various neurological diseases, such as epilepsy and some forms of muscular dystrophy, are caused by errors in RNA localization [3, 4].

As a high-throughput way of determining subcellular localization of RNA is not feasible yet, and given that RNA localization plays a much larger role in human health than previously thought, it remains attractive to use the RNA sequence itself for prediction of its subcellular localization. There are many steps involved in localization including e.g. the initial binding of RNA-binding proteins, translational repression during the transport and anchoring of the RNA at the destination [4]. This means that the exact motifs determining the localization of a particular set of sequences can often be very convoluted, and as such advanced sequence analysis methods are necessary.

Deep learning techniques are well fit to tackle these kinds of complex black-box models. In particular, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have seen much use in sequence analysis due to their ability to understand local motifs and sequentiality of data, respectively [5]. In this paper, we use a novel mix of the two networks to predict subcellular localization of RNA.

2. RELATED WORK

Recurrent neural networks (RNNs) have become one of the most commonly used type of neural network for natural language processing tasks (NLP). Performances of RNNs on a range of applications going from sentiment analysis [6] to question answering [7] and word- and character-level language modeling [8] has been demonstrated. However the inability of RNNs to handle problems involving very long sequences, such as biological sequences, has restricted their usage. Unfortunately gated variants including the long short-term memory (LSTM) [9] have not been able to fully deal with this limitation.
Convolutional neural networks (CNNs) [10] are more popular for tasks including image data but has also been used for NLP problems. For example, CNNs were successfully applied to document classification [11] and language modelling [12]. In fact CNNs have been applied to sequences for decade, and recent studies suggest that they should even be consider as a starting point for language modelling tasks [13]. CNNs possessed several advantages over RNNs, including easier parallellization and better capacity to handle long sequences. Extended versions of convolutional layers, such as Dilated convolutions [14], increase the ability of CNNs to handle sequential modeling tasks.
Hybrid architectures were developed to combine advantages of RNNs and CNNs [15] but their ability to fully use the large-scale sequence order information stays limited. Quasi-recurrent neural networks (QRNNs) is a recent type of network that alternate convectional layers with a minimal recurrent pooling function [16]. QRNNs seem really promising for language processing of long sequences, but hadn't been tested yet on biological sequences. We present a bioinformatic application of QRNNs.

In order to take a mathematical approach to language, a numerical way of representing letters, words or sentences is required. Using one-hot representation is an easy but weak solution since the dimensions of the encoded data depends on the size of the vocabulary considered. In contrast, distributed representations aim to represent words as arbitrary low-dimension vectors of real numbers. These are called word embeddings. Word embeddings are often used as input to machine learning algorithms, especially in Deep learning community. *Word2vec* [17] is one of the mos popular method for word embedding, and several extensions were proposed to extend it to biological language [18].

3. METHODS

3.1. One-hot encoding

If time permits, distributed representation will be implemented but as a starting point one-hot encoding vectors will be used to encode RNA sequences given to the network. Let's define $x_s \in \mathbb{R}^5, s \in \{A,T,C,G,N\}$ the one-hot encoding vector of the nucleotide x, with :

$$x_A \begin{pmatrix}1\\0\\0\\0\\0\end{pmatrix} \quad x_T \begin{pmatrix}0\\1\\0\\0\\0\end{pmatrix} \quad x_C \begin{pmatrix}0\\0\\1\\0\\0\end{pmatrix} \quad x_G \begin{pmatrix}0\\0\\0\\1\\0\end{pmatrix} \quad x_N \begin{pmatrix}0\\0\\0\\0\\1\end{pmatrix}$$

Then the input vector $x \in \mathbb{R}^{5n}$ representing a RNA sequence of length n is the concatenation of the nucleotide vectors. For example, with $\oplus$ the concatenation operator :

$$\begin{aligned} x(ATT) &= x_A \oplus x_T \oplus x_T \\ &= (1\,0\,0\,0\,0) \oplus (0\,1\,0\,0\,0) \oplus (0\,1\,0\,0\,0) \\ &= (1\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0) \end{aligned}$$

3.2. LSTM and QRNN

When working with sequential data, the most natural choice of neural network architecture is Recurrent Neural Network. They are network with loops inside, allowing information to persist across the length of the sequence. In particular, well tuned LSTM networks can achieve good performance by learning long-term dependencies.
Quasi-Recurrent Neural Networks try to extract and merge the benefits from standard models. QRNNs operate removing the dependencies between the computations of hidden states and handling dependencies between pooling. In a recurrent fashion, QRNNs allow the output to be modelled in a sequential manner; like CNNs, QRNNs are highly parallelizable, making possible to handle long time dependencies. As a result, QRNNs perform like LSTMs but the computation time dramatically decreases. Figure 1 shows one QRNN layer. Notice how both convolution and pooling operation can proceed in parallel.
Each layer of a QRNN is composed by two elements: first, a convolutional layer is applied; a poling operation is then applied. Convolution can be carried out in parallel, by its nature; the pooling component is fully parallel since no parameters are to train. Given an input sequence $\mathbf{X} \in \mathbb{R}^{T \times n}$, the convolutional component of the QRNN performs convolution in the time-step dimension with a bank of m filters, producing a sequence $\mathbf{Z} \in \mathbb{R}^{T \times m}$ of candidate vectors $\mathbf{z}_t$. The candidate vectors are passed through a hyperbolic tangent non-linearity. One of the key-concept upon which QRNNs are built is that there should be no "leakage" of information from feature to past. This is accomplished using *causal convolutions*. The term *causal* comes from the theory of signals processing: convolution can be seen as a filtering operation, and a filter is said causal if its output does not depend on future inputs. Hence, with a causal convolution the output at time t is convolved only with elements from time t and earlier in the previous layers. Hence, for a filter of width k, each $\mathbf{z}_t$ depends only on $\mathbf{x}_{t-k+1}$ through $\mathbf{x}_t$. Causal convolutions are implemented by padding the input to the left by the filter size minus one.

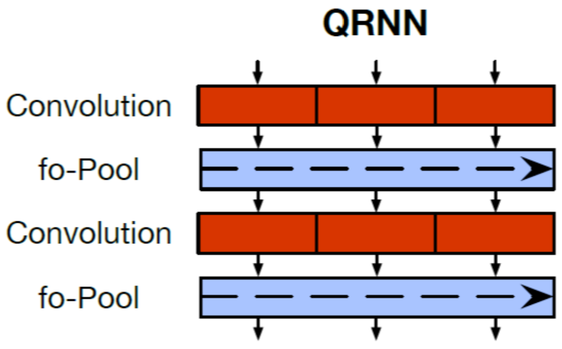


Fig. 1. Diagram showing QRNN computation. [16]

A forget gate $\mathbf{f}_t$ and a output gate $\mathbf{o}_t$ can be computed at each time-step similarly to $\mathbf{z}_t$, activating them with a *sigmoid* function. Thus, the output of the convolutional component will be:

$$\begin{aligned} \mathbf{Z} &= \tanh(\mathbf{W}_z * \mathbf{X}) \\ \mathbf{F} &= \sigma(\mathbf{W}_f * \mathbf{X}) \\ \mathbf{O} &= \sigma(\mathbf{W}_o * \mathbf{X}) \end{aligned}$$

where $\mathbf{W}_z$, $\mathbf{W}_f$, $\mathbf{W}_o$ are the filter banks and $*$ denotes the causal convolution operation.
The second step in the QRNN layer consists is using the gates computed in the convolutions to compute hidden states according to a certain pooling method. One easy option, as presented in [19], can be the *dynamic average pooling* that uses forget and output gate:

$$\begin{aligned} \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{z}_t \\ \mathbf{h}_t &= \mathbf{o}_t \odot \mathbf{c}_t. \end{aligned}$$

In the above equations, $\odot$ represents the Hadamard (element-wise) product. A pooling with both forget and output gate is called *fo*-pooling. The choice of such pooling function makes QRNNs much faster than LSTM: QRNNs don't compute a slow hidden-to-hidden matrix multiplication at each time-step, they rely on a fast element-wise operator.
The essence of QRNNs is to do heavy computation in parallel and minimal sequential processing in the pooling layer, where $\mathbf{z}_t$, $\mathbf{f}_t$ and $\mathbf{o}_t$ don't depend on any previous value.

3.3. Temporal Convolutional Networks and dilated convolutions

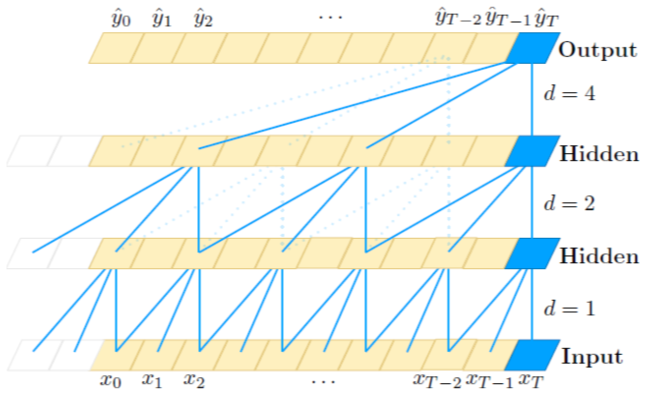
As discussed in Section 3.2, the operation of convolution is very fast compared to recurrences in the network. This idea led to the development of algorithms for sequential modeling with Convolutional Neural Networks. The term used to describe such networks is Temporal Convolutional Networks (TCN): the temporal dependencies are captured with convolutions instead of recurrences. The main characteristic of TCNs is that they are able to look very far into the past using specific configurations for convolutional layers. Temporal convolutional architectures was designed to combine the simplicity, autoregressive properties and very long memory. For example, TCN is much simpler than WaveNet([20]): TCNs don't use gating mechanisms and have much longer memory. [13] Also in TCNs, it is important that no information form the future is considered. Hence, also here causal convolutions are applied.

The main drawback of a convolutional network using causal convolutional layers is that a very deep network is needed to achieve a reasonably long history size; causal convolution is able to look back in the past with a history size that grows only *linearly* with the depth of the network.
This limitation can be avoided employing *dilated* convolutions, where the history size, also called *receptive field*, grows *exponentially* with the numbers of layers in the network ([14]).

In ordinary convolutional layers, the cells in the filter are constrained to be contiguous. The *dilation* is defined as a new hyperparameter of the network that represents the space between cells in the filter. Figure 2 provides an illustration of dilated convolutions. Between the input and the first hidden layer, the dilation coefficient d is set to 1; this corresponds to an usual convolution. When $d = 2$, the filter is still composed by three cells, but they are not one next to the other but they are interspersed by empty cells. In the last layer, it's clear how the receptive field has grown: the output cell is computed using only one filter that covers the entire sequence. For the same sequence, six filters without dilation are needed to cover it all. When working with dilated convolutions, it's a convenient practice to increase d exponentially with the depth of the network, i.e., $d = O(2^i)$ at level i of the network.
More formally, let $x \in \mathbb{R}^n$ be a one-dimensional input sequence; let $f : (0, \dots, k-1) \longrightarrow \mathbb{R}$ a filter. The dilated convolution operation F on an element s of the input sequence is defined as

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d\cdot i} \qquad (1)$$

where d is the dilation fraction, k is the size of the filter and $s - d \cdot i$ represents the portion of the past sequence we look back.



$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ $\cdots$ $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 $\cdots$ x_{T-2} x_{T-1} x_T

Fig. 2. Dilated causal convolution with dilation factors $d = 1, 2, 4$ and a filter size $k = 3$.

4. EXPERIMENTAL SETUP

All the models are implemented in Python v.3.6.5 and the deep learning framework PyTorch v.0.4.1. The training of the algorithm will be done via a GPU NVIDIA® GeForce GTX 1050.

4.1. Dataset

Training, testing and validation data all comes from the RNALocate database [21]. The data is divided in 3 sets, training (70%), validation (10%) and test (20%), with respect to its sequence length distribution. Some sequences appear twice with different localization. In figure 3, a histogram showing the distribution of sequence lengths can be found, along with a bar plot of the categories (that is possible destinations). The skewed distribution of lengths should justify the use of dynamic batching in the network.

5. REFERENCES

[1] "Genomic analysis of RNA localization," *RNA Biology*, vol. 11, no. 8, pp. 1040–1050, 2014.

[2] "Nucleolin-Mediated RNA Localization Regulates Neuron Growth and Cycling Cell Size," *Cell Reports*, vol. 16, no. 6, pp. 1664–1676, 2016.

[3] "Dysregulation of mRNA Localization and Translation in Genetic Disease," *The Journal of Neuroscience*, vol. 36, no. 45, pp. 11418–11426, 2016.

[4] Doo Young Lee, Jangsup Moon, Soon Tae Lee, Keun Hwa Jung, Dong Kyu Park, Jung Seok Yoo, Jun Sang Sunwoo, Jung Ick Byun, Jung Ah Lim, Tae Joon Kim, Ki Young Jung, Manho Kim, Daejong Jeon, Kon Chu, and Sang Kun Lee, "Dysregulation of

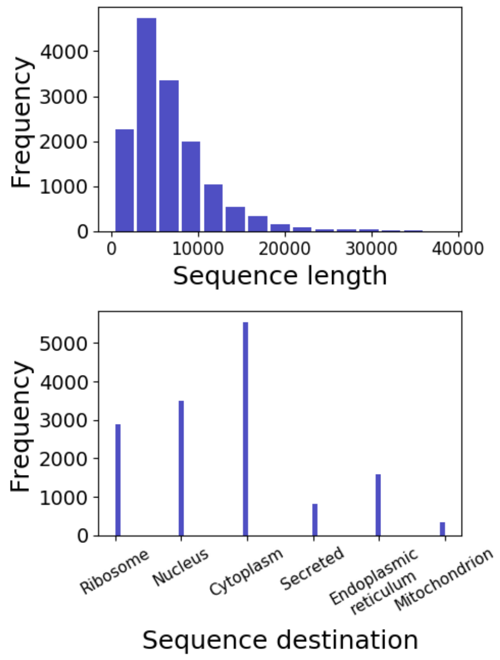


Fig. 3. **Upper**: histogram showing the distribution of sequence lengths in the entire data set. Notice most sequences are around 3000-7000 nucleotides, while very few are above 30000. **Lower**: Bar plot showing the distribution of classes (subcellular localizations). Clearly, some destinations are underrepresented

long non-coding RNAs in mouse models of localization-related epilepsy," *Biochemical and Biophysical Research Communications*, vol. 462, no. 4, pp. 433–440, 2015.

[5] Vanessa Isabell Jurtz, Morten Nielsen, Ole Winther, Søren Kaae Sønderby, Casper Kaae Sønderby, and Alexander Rosenberg Johansen, "An introduction to Deep learning on biological sequence data Examples and solutions," *Bioinformatics*, vol. 31, pp. 31–37, 2016.

[6] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[7] Shayne Longpre, Sabeek Pradhan, Caiming Xiong, and Richard Socher, "A way out of the odyssey: Analyzing and combining recent insights for lstms," *arXiv preprint arXiv:1611.05104*, 2016.

[8] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[9] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun, "Very deep convolutional networks for text classification," *arXiv preprint arXiv:1606.01781*, 2016.

[12] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier, "Language modeling with gated convolutional networks," *arXiv preprint arXiv:1612.08083*, 2016.

[13] Shaojie Bai, J Zico Kolter, and Vladlen Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[14] Fisher Yu and Vladlen Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[15] Jason Lee, Kyunghyun Cho, and Thomas Hofmann, "Fully character-level neural machine translation without explicit segmentation," *arXiv preprint arXiv:1610.03017*, 2016.

[16] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher, "Quasi-recurrent neural networks," *arXiv preprint arXiv:1611.01576*, 2016.

[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[18] Patrick Ng, "dna2vec: Consistent vector representations of variable-length k-mers," *arXiv preprint arXiv:1701.06279*, 2017.

[19] David Balduzzi and Muhammad Ghifary, "Strongly-typed recurrent neural networks," *arXiv preprint arXiv:1602.02218*, 2016.

[20] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio.," in *SSW*, 2016, p. 125.

[21] "RNALocate: A resource for RNA subcellular localizations," *Nucleic Acids Research*, vol. 45, no. D1, pp. D135–D138, 2017.