Neural Edit Operations for Biological Sequences
Satoshi Koide, Keisuke Kawano, Takuro Kutsuna

Presented by
Mohimenul Karim
Overview of The Work

• Two neural network architectures that can treat edit operations in biological sequences:
  • Edit invariant neural networks (EINN) based on differentiable Needleman-Wunsch algorithm
  • Deep CNNs with concatenations

• CNNs can recognize regular expressions without Kleene star

• Experiment on protein secondary structure prediction task
Needleman-Wunsch Algorithm

- Needleman-Wunsch algorithm:
  - Calculates similarity score between two sequences
  - Uses dynamic programming to maximize the score
Differentiable Sequence Alignment

- Needleman-Wunsch (NW) algorithm as a differentiable function
- Score function is defined as the inner product (line 7)
- Softmax function instead of using hard max function (line 10)
Differentiable Sequence Alignment

Differentiate the NW similarity score $s_{NW}(x_{1:m}, y_{1:n}; g)$ with respect to $x_{1:m}, y_{1:n}$ and $g$ (gap cost)

$$\frac{\partial s_{NW}}{\partial x_i} = \sum_{j=1}^{n} Q_{i,j} \exp(H_{i,j}/\gamma) \cdot y_j, \quad \frac{\partial s_{NW}}{\partial y_j} = \sum_{i=1}^{m} Q_{i,j} \exp(H_{i,j}/\gamma) \cdot x_i,$$

where $H_{i,j} := F_{i-1,j-1} + x_i \cdot y_j - F_{i,j}$.

$$\frac{\partial s_{NW}}{\partial g} = P_{m,n}.$$
Differentiable Sequence Alignment

Use dynamic programming to calculate the derivatives

**Algorithm 2:** Calculation of $Q$ (backward). We denote $\varphi_\gamma(a, b) := \exp((a - b)/\gamma)$.

1. $Q \leftarrow 0$; // $(m+2) \times (n+2)$ zero matrix
2. for $i = 1 \ldots m$ do
3.     $F_{i,n+1} \leftarrow \infty$
4.     $F_{m+1,n+1} \leftarrow F_{m,n}$; $Q_{m+1,n+1} \leftarrow 1$
5.     for $j = n+1 \ldots 1$ do
6.         $F_{m+1,j} \leftarrow \infty$
7.         for $i = m \ldots 1$ do
8.             $a \leftarrow \varphi_\gamma(F_{i,j} + x_i \cdot y_j, F_{i+1,j+1})$
9.             $b \leftarrow \varphi_\gamma(F_{i,j} - g, F_{i+1,j})$
10.            $c \leftarrow \varphi_\gamma(F_{i,j} - g, F_{i,j+1})$
11.            $Q_{i,j} \leftarrow aQ_{i+1,j+1} + bQ_{i+1,j} + cQ_{i,j+1}$
12. return $Q$

**Algorithm 3:** Calculation of $P$. We denote $\varphi_\gamma(a, b) := \exp((a - b)/\gamma)$.

1. $P \leftarrow 0$; // $(m+2) \times (n+2)$ zero matrix
2. for $i = 0 \ldots m$ do
3.     $P_{i,0} \leftarrow -i$
4.     for $j = 1 \ldots n$ do
5.         $P_{0,j} \leftarrow -j$
6.         for $i = 1 \ldots m$ do
7.             $a \leftarrow \varphi_\gamma(F_{i-1,j-1} + x_i \cdot y_j, F_{i,j})$
8.             $b \leftarrow \varphi_\gamma(F_{i-1,j-1} - g, F_{i,j})$
9.             $c \leftarrow \varphi_\gamma(F_{i-1,j-1} - g, F_{i,j})$
10.            $P_{i,j} \leftarrow aP_{i-1,j-1} + b(P_{i-1,j-1} - 1) + c(P_{i,j-1} - 1)$
11. return $P$
Edit Invariant Neural Networks (EINN)

- Extends the traditional CNNs by the NW score
- Let,
  - Convolutional filter of kernel size $K$: $w \in \mathbb{R}^{d \times K}$
  - A frame of length $K$ at a certain position in the embedded sequence $X$: $x \in \mathbb{R}^{d \times K}$
- In CNNs the similarity score is calculated using the inner product $w \cdot x$
- Replace the inner product with the proposed $s_{NW}(x, w; g)$
- A generalization of CNNs, because $s_{NW}$ converges to an inner product when $g \to \infty$ (Proposition 1)
Regular Expressions

• Unix-like notations of regular expressions

• Example:
  • /a.b/ represents “a, then any character and then b”
  • /a[bc]a/ represents “a, then b or c and then again a”
  • /(abc|ac)/ represents “abc or ac”

• Considered regular expressions without the Kleene star, R* (R is a regular expression and R* accepts infinite repeats of strings in R)

• Therefore, regular expressions like /ab*/ which represents “a followed by any number of b” are not considered
• x is a string of length L on an alphabet $\Sigma = \{a, b, c\}$
• Assumes one-hot representation for x
• Composed a 1d-convolutional layer with filter matrix w (one-hot representation) and bias b to match a regular expression
• Using a filter, the output of the layer at position i is 1, if the regular expression finds match in the string
• Use ReLU and obtain 1 for matching and 0 otherwise
Simple Regular Expressions With CNNs

\[ w_1 = (e_a, e_b, e_c) \text{ where } e_a \text{ is the one-hot vector of character 'a' and } b_1 = -2 \]
\[ w_2 = (e_a, e_c, 0) \text{ and } b_2 = -1 \]
Deeper CNNs for Complex Regular Expressions

If the shaded cell is 1, then we can detect deletion with the regular expression.
Experiments

- Protein secondary structure prediction
- Dataset:
  - Test: CB513
  - Training: Filtered CB6133 (filtered, if proteins in the original CB6133 have 25% or higher similarity with the proteins in CB513)
- Predict the eight-class secondary structure labels for each position of a given sequence
- Feature vector at each position:
  - One-hot representation of amino acid
  - Position specific scoring matrix (PSSM)
- Zero-padding for constant sequence length
Experiments (Simplified Models)

• Tiny-CNN
• Tiny-EINN (replace Conv-5 layers with EINN convolutional layer)

• Training:
  • One-hot vector for input (did not use PSSM)
  • 2% of training data sampled from the filtered CB6133 dataset
• When the gap cost $g$ is greater than 10, the accuracy is equal to that of CNN
• For $g=2.5$, the accuracy is maximum
Experiments (Simplified Models)

Figure 4: Gapcost $g$ vs accuracy (Tiny-EINN). Tiny-EINN is nearly equivalent to Tiny-CNN for $g > 10$. 
Experiments (Simplified Models)

- Used different sizes of training data (data size: 1%, 2% and 5%)

Figure 5: Effect of data size. The performance gain of EINNs increases as the data size decreases.
Experiments (Deeper Models)

- Stack two ConvBlocks
- At each position, a fully connected layer, batch normalization, dropout and ReLU are applied
- For investigating the effect of EINNs:
  - Replaced the convolutions (shaded in the figure) in the first ConvBlock with EINNs of the same filter and kernel size
Experiments (Deeper Models)

- Observation: Data augmentation improves accuracy
- Get training data by replacing the one-hot vector at random positions with an amino acid drawn from the uniform distribution
- Randomly replaced 15% of the residues
- Improved accuracy by up to 0.8 points
Effect of Depth

• Initially, the shallow stacking of the ConvBlocks
• Then deeper stacking (from 2 blocks to 16 blocks)
• Employed the multitasking technique and simultaneously predict the secondary structure and solvent accessibility
• 12-block CNN*† achieved 71.5% CB513 accuracy (*: with multitasking; †: with data augmentation)
Effect of Network Architecture

- Replaced ConvBlock with the modified ConvBlock
- Replaced the ConvBlocks with the residual blocks
Table 2: Comparison of precisions for the secondary structure prediction on CB513 dataset. Note that these results are for non-ensemble models. (*: with multitasking / †: with data augment.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our 2-block CNN†</td>
<td>69.7</td>
</tr>
<tr>
<td>Our 2-block EINN†</td>
<td>69.8</td>
</tr>
<tr>
<td>Our 2-block CNN*†</td>
<td>69.8</td>
</tr>
<tr>
<td>Our 4-block CNN*†</td>
<td>70.6</td>
</tr>
<tr>
<td>Our 8-block CNN*†</td>
<td>71.2</td>
</tr>
<tr>
<td>Our 12-block CNN*†</td>
<td>71.5</td>
</tr>
<tr>
<td>Our 16-block CNN*†</td>
<td>71.3</td>
</tr>
<tr>
<td>Our 8-block MCNN*†</td>
<td>71.3</td>
</tr>
<tr>
<td>Our 12-block MCNN*†</td>
<td>71.5</td>
</tr>
<tr>
<td>ResNet*† (best result)</td>
<td>71.0</td>
</tr>
<tr>
<td>GSN [26] (2014)</td>
<td>66.4</td>
</tr>
<tr>
<td>DeepCNF [24] (2016)</td>
<td>68.3</td>
</tr>
<tr>
<td>DCRNN [13] (2016)</td>
<td>69.4</td>
</tr>
<tr>
<td>NextCond CNN [2] (2017)</td>
<td>70.3</td>
</tr>
</tbody>
</table>
Conclusion and Comment

• EINN consisting of differentiable NW algorithm
• EINN performs better than CNN
• Deep CNNs can recognize complex regular expressions
• Deep CNNs perform better than the state-of-the-art models

• Small increase in the accuracy against large computation in EINN