Neural Edit Operations for Biological Sequences

Satoshi Koide, Keisuke Kawano, Takuro Kutsuna

Presented by

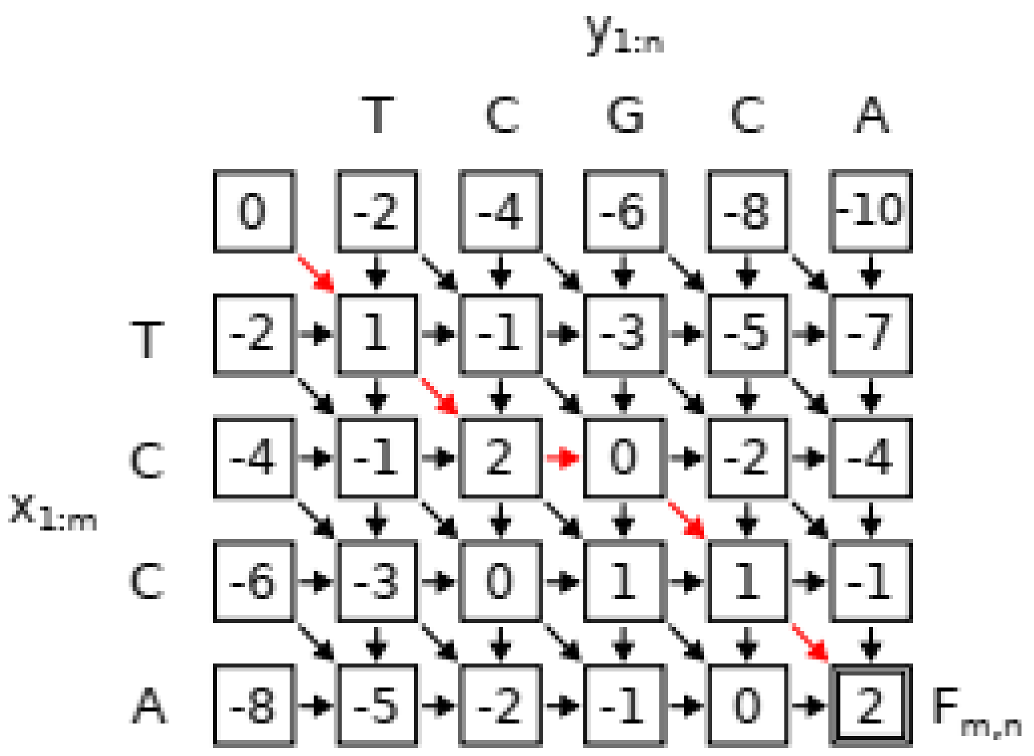
Mohimenul Karim

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

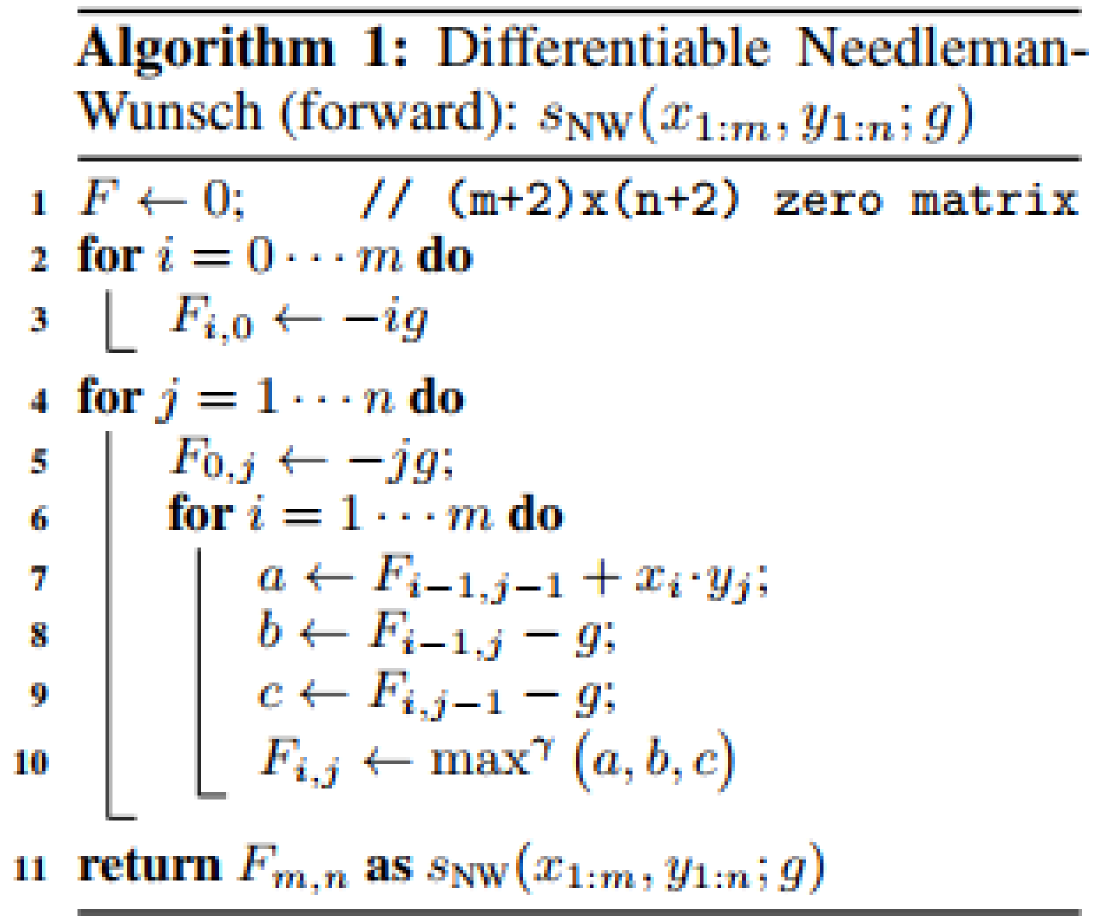
Needleman-Wunsch Algorithm

- 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)



Algorithm 1: Differentiable Needleman-Wunsch (forward): $s_{\mathrm{NW}}(x_{1:m}, y_{1:n}; g)$

```
1  $F \leftarrow 0$;        // (m+2)x(n+2) zero matrix
2  for $i = 0 \cdots m$ do
3      $F_{i,0} \leftarrow -ig$
4  for $j = 1 \cdots n$ do
5      $F_{0,j} \leftarrow -jg$;
6      for $i = 1 \cdots m$ do
7          $a \leftarrow F_{i-1,j-1} + x_i \cdot y_j$;
8          $b \leftarrow F_{i-1,j} - g$;
9          $c \leftarrow F_{i,j-1} - g$;
10         $F_{i,j} \leftarrow \max^{\gamma}(a, b, c)$
11 return $F_{m,n}$ as $s_{\mathrm{NW}}(x_{1:m}, y_{1:n}; g)$
```

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)

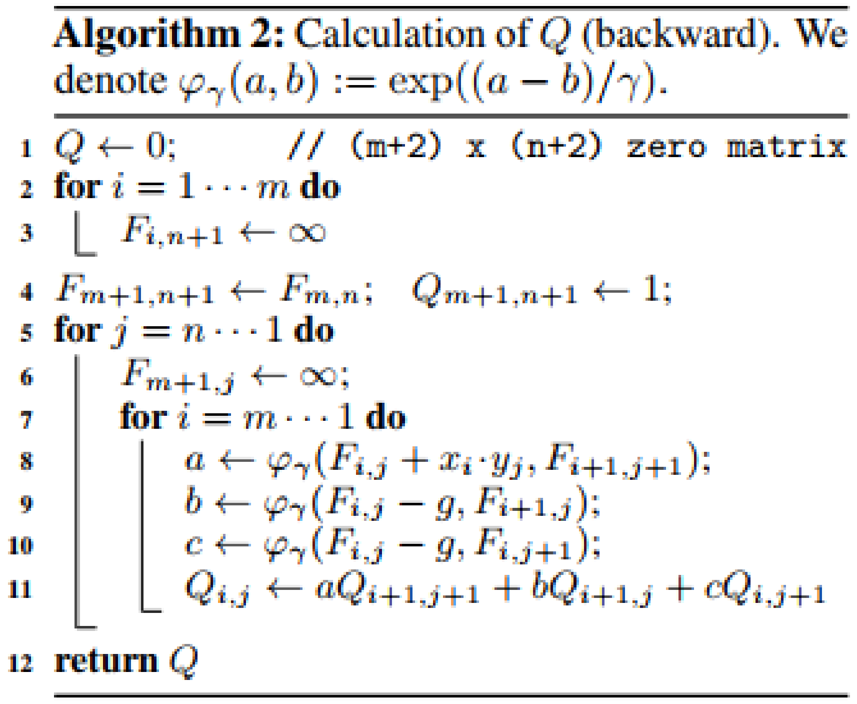
$$\frac{\partial s_{\mathrm{NW}}}{\partial x_i} = \sum_{j=1}^{n} Q_{i,j} \exp(H_{i,j}/\gamma) \cdot y_j, \quad \frac{\partial s_{\mathrm{NW}}}{\partial y_j} = \sum_{i=1}^{m} Q_{i,j} \exp(H_{i,j}/\gamma) \cdot x_i,$$

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

$$\frac{\partial s_{\mathrm{NW}}}{\partial g} = P_{m,n}.$$

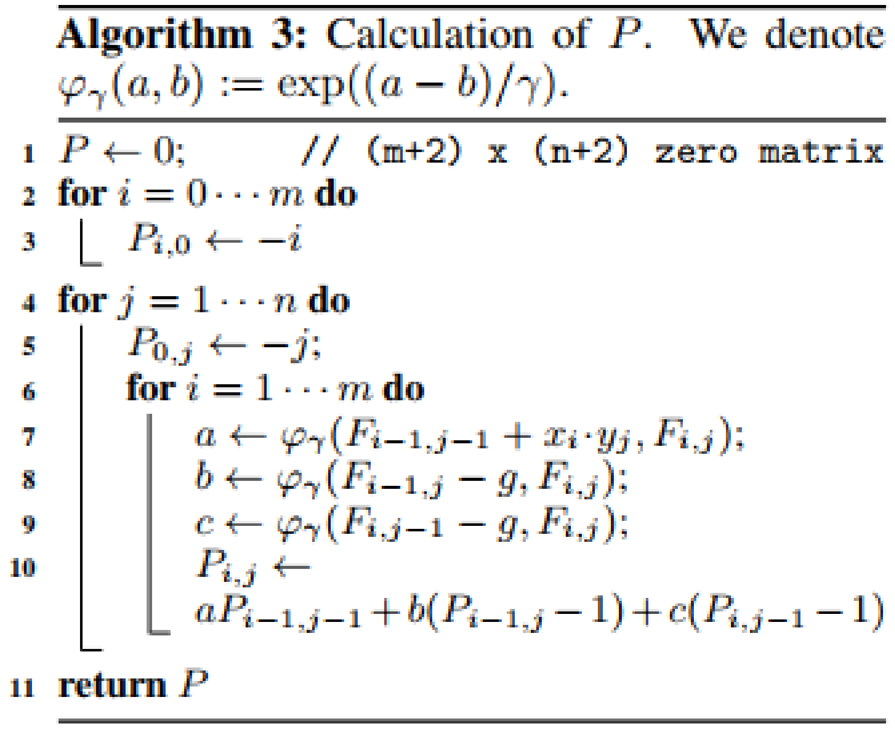
Differentiable Sequence Alignment

Use dynamic programming to calculate the derivatives



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

```
$Q \leftarrow 0$;        // (m+2) x (n+2) zero matrix
for $i = 1 \cdots m$ do
    $F_{i,n+1} \leftarrow \infty$
$F_{m+1,n+1} \leftarrow F_{m,n}$;   $Q_{m+1,n+1} \leftarrow 1$;
for $j = n \cdots 1$ do
    $F_{m+1,j} \leftarrow \infty$;
    for $i = m \cdots 1$ do
        $a \leftarrow \varphi_\gamma(F_{i,j} + x_i \cdot y_j, F_{i+1,j+1})$;
        $b \leftarrow \varphi_\gamma(F_{i,j} - g, F_{i+1,j})$;
        $c \leftarrow \varphi_\gamma(F_{i,j} - g, F_{i,j+1})$;
        $Q_{i,j} \leftarrow aQ_{i+1,j+1} + bQ_{i+1,j} + cQ_{i,j+1}$
return $Q$
```



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

```
$P \leftarrow 0$;        // (m+2) x (n+2) zero matrix
for $i = 0 \cdots m$ do
    $P_{i,0} \leftarrow -i$
for $j = 1 \cdots n$ do
    $P_{0,j} \leftarrow -j$;
    for $i = 1 \cdots m$ do
        $a \leftarrow \varphi_\gamma(F_{i-1,j-1} + x_i \cdot y_j, F_{i,j})$;
        $b \leftarrow \varphi_\gamma(F_{i-1,j} - g, F_{i,j})$;
        $c \leftarrow \varphi_\gamma(F_{i,j-1} - g, F_{i,j})$;
        $P_{i,j} \leftarrow$
        $aP_{i-1,j-1} + b(P_{i-1,j} - 1) + c(P_{i,j-1} - 1)$
return $P$
```

Edit Invariant Neural Networks (EINN)

- Extends the traditional CNNs by the NW score
- Let,
  - Convolutional filter of kernel size K: $w \in R^{dXK}$
  - A frame of length K at a certain position in the embedded sequence X: x $\in R^{dXK}$
- In CNNs the similarity score is calculated using the inner product $w.x$
- 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

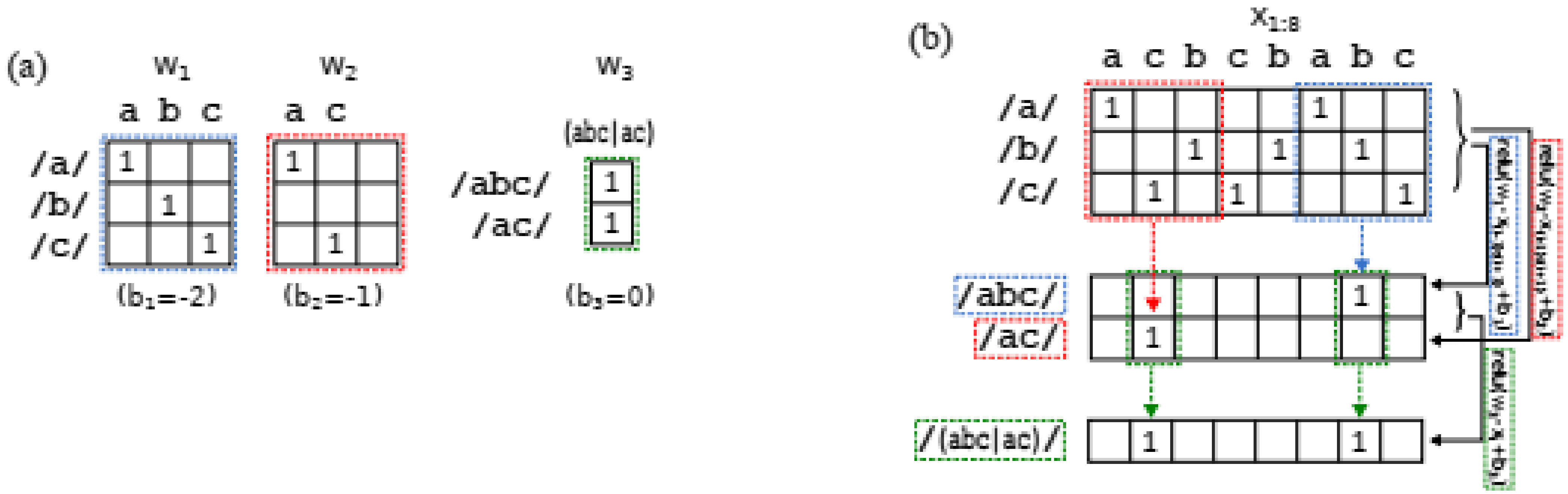
- Unix-like notations of regular expressions
- Example:
  - /a.b/ represents “a, then any character and then b”
  - /a[bc]a/ represents “a, then b or c and then again a”
  - /(abc|ac)/ represents “abc or ac”

- 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

Simple Regular Expressions With CNNs

- x is a string of length L on an alphabet $\sum = \{a, b, c\}$
- 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

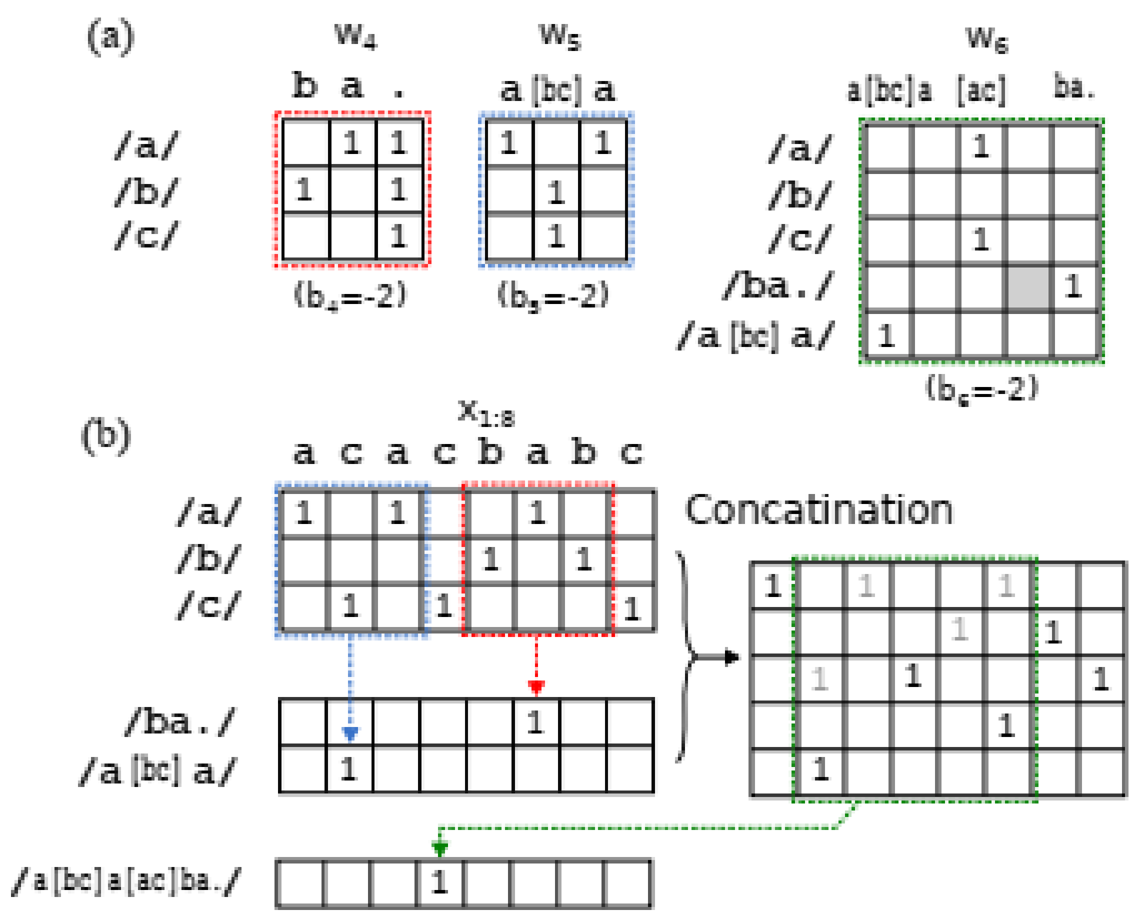
Simple Regular Expressions With CNNs



w_1=(e_a, e_b, e_c) where e_a is the one-hot vector of character 'a' and b_1=-2

w_2=(e_a, e_c, 0) and b_2=-1

Deeper CNNs for Complex Regular Expressions



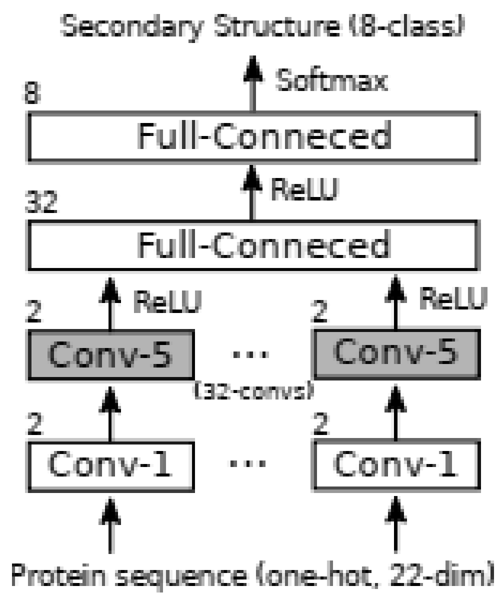
If the shaded cell is 1, then we can detect deletion with the regular expression

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

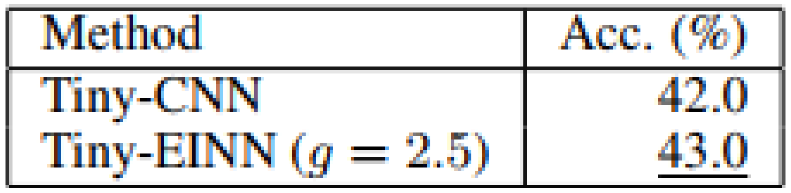
Experiments (Simplified Models)

- 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



(a) Tiny-{CNN/EINN}

Table 1: Test accuracy (CB513).



Method	Acc. (%)
Tiny-CNN	42.0
Tiny-EINN ($g = 2.5$)	<u>43.0</u>

Experiments (Simplified Models)

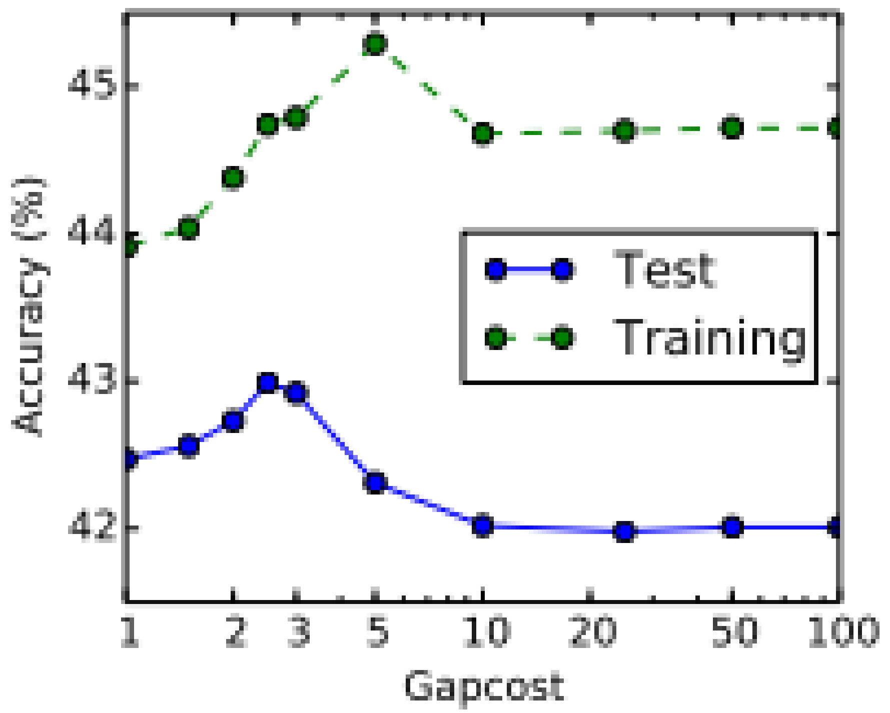


Figure 4: Gapcost g vs accuracy (Tiny-EINN). Tiny-EINN is nearly equivalent to Tiny-CNN for $g > 10$.

Experiments (Simplified Models)

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

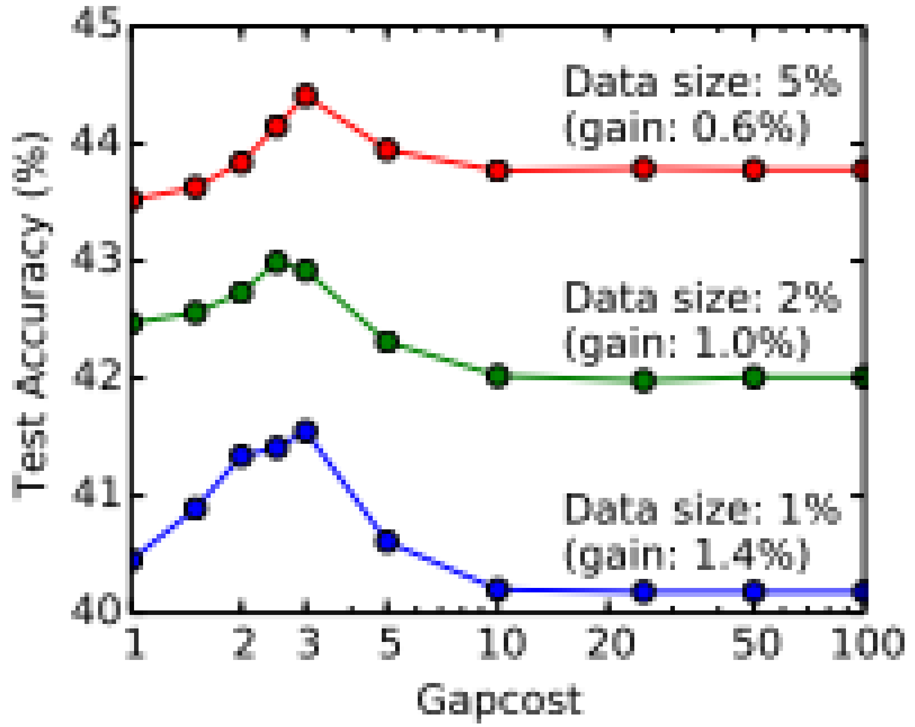
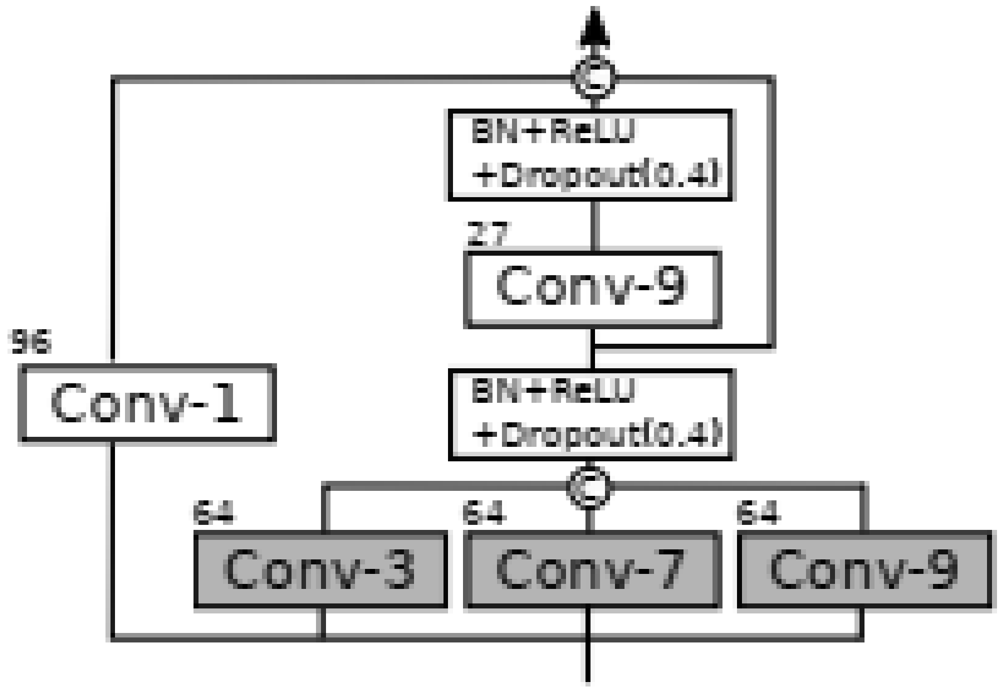


Figure 5: Effect of data size. The performance gain of EINNs increases as the data size decreases.

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



(b) ConvBlock (inspired by [2])

Experiments (Deeper Models)

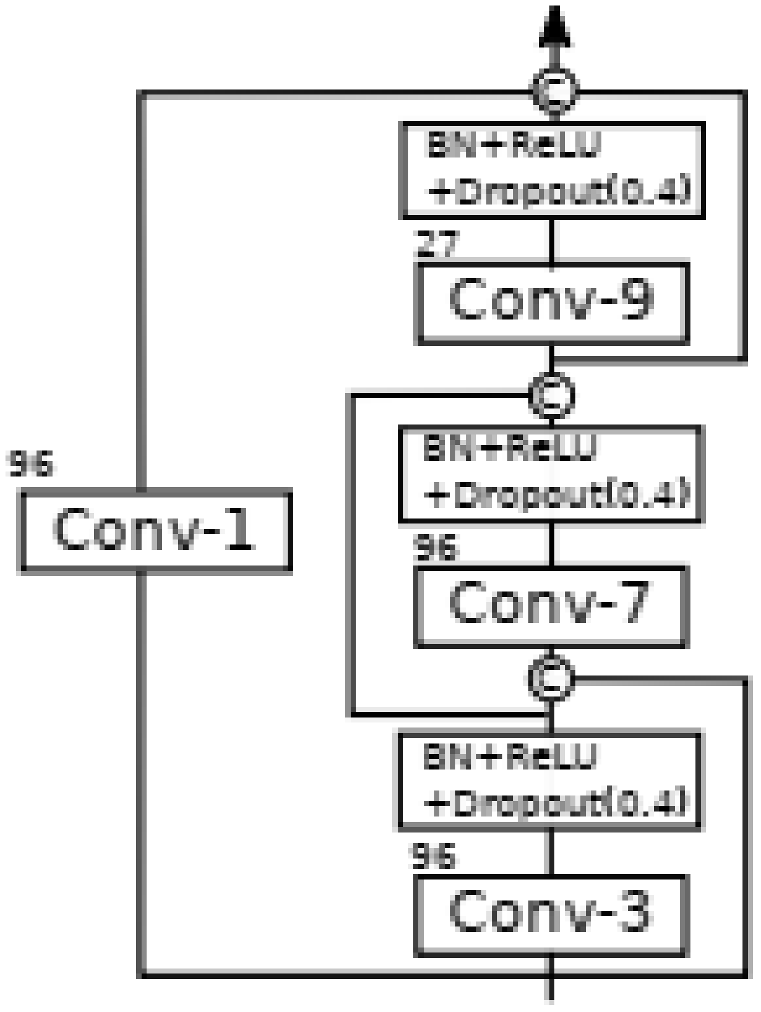
- Observation: Data augmentation improves accuracy
- Get training data by replacing the one-hot vector at random positions with an amino acid drawn from the uniform distribution
- Randomly replaced 15% of the residues
- Improved accuracy by up to 0.8 points

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

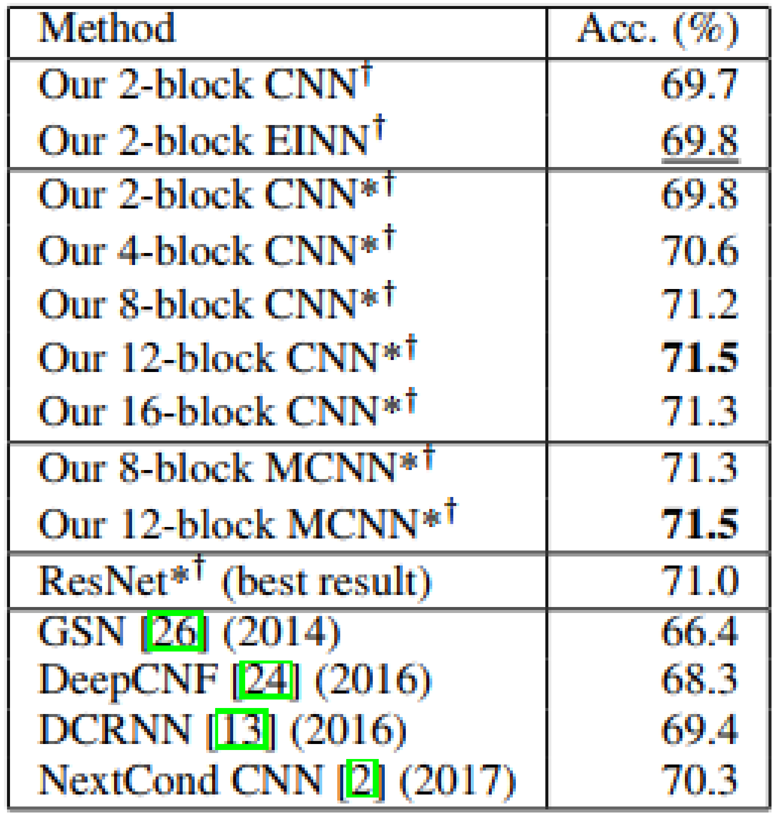
- Replaced ConvBlock with the modified ConvBlock
- Replaced the ConvBlocks with the residual blocks



(c) Modified-ConvBlock

Comparison

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.)



Method	Acc. (%)
Our 2-block CNN†	69.7
Our 2-block EINN†	69.8
Our 2-block CNN$*^{\dagger}$	69.8
Our 4-block CNN$*^{\dagger}$	70.6
Our 8-block CNN$*^{\dagger}$	71.2
Our 12-block CNN$*^{\dagger}$	**71.5**
Our 16-block CNN$*^{\dagger}$	71.3
Our 8-block MCNN$*^{\dagger}$	71.3
Our 12-block MCNN$*^{\dagger}$	**71.5**
ResNet$*^{\dagger}$ (best result)	71.0
GSN [26] (2014)	66.4
DeepCNF [24] (2016)	68.3
DCRNN [13] (2016)	69.4
NextCond CNN [2] (2017)	70.3

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