國立臺灣大學電機資訊學院生醫電子與資訊學研究所

碩士論文

Graduate Institute of Biomedical Electronics and Bioinformatics

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

利用深度學習來預測阿拉伯芥DNA序列中編碼基因的基因結構

Using deep learning to predict gene structures of the coding genes in DNA sequences of *Arabidopsis thaliana*

王擎天

Ching-Tien Wang

指導教授：趙坤茂 教授

Advisor: Kun-Mao Chao, Professor

共同指導教授：林仲彥 教授

Co-advisor: Chung-Yen Lin, Professor

中華民國109年7月

July 2020

doi:10.6342/NTU202002143

doi:10.6342/NTU202002143

國立臺灣大學碩士學位論文
口試委員會審定書

利用深度學習來預測阿拉伯芥 DNA 序列中編碼基因的基因結構

Using deep learning to predict gene structures of the coding genes in DNA sequences of *Arabidopsis thaliana*

本論文係王擎天君（學號 R06945055）在國立臺灣大學生醫電子與資訊學研究所完成之碩士學位論文，於民國 109 年 07 月 30 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

趙坤茂（指導教授）

林仲彥　　張育榮

所　　長：張瑞峰

doi:10.6342/NTU202002143

doi:10.6342/NTU202002143

謝辭

首先，我想感謝我的指導教授–趙坤茂教授。當初是在生物序列分析演算法的課程上認識趙坤茂教授，當時被趙教授的學識與教學風格所吸引，於是我決定要去報考生醫電資所並找趙教授來當我碩士論文的指導教授，事後證明這果然是個正確的選擇。這兩年多來我受到趙教授許多的協助與鼓勵，並從趙教授身上學到做人做事的態度和做研究的方法，這些都對我的研究有著重大的幫助。再來，我想感謝我的另一位指導教授–林仲彥教授。當初是去中研院實習時認識了林仲彥教授，也是在當時進行了基因註解這個題目。這兩年多來我跟林教授開過無數次的會議，這些會議使我的表達和口說能力有明顯進步，林教授的建議對我的研究有著很大的幫助。在這兩年多來也跟林教授有聊過無數個話題，這擴大我的視野，使我對生物資訊在學界和業界有著更多的認識。同時，我也要感謝張育榮教授能抽空來擔任口試委員。張教授在口試時對論文提出的批評與指教讓我重新去審視我的研究，張教授所提出了問題與建議也讓我的論文在應用層面上的改進有了全新的方向。另外，我也要感謝實驗室的陳淑華教授在這兩年多來對於我的研究報告和論文寫作上有非常多建議與指導，使我助益良多。我在這裡也要感謝國立臺灣大學和中央研究院在我研究期間提供我大量的資源使我可以順利的完成研究。我也要感謝生醫電資所的所辦在我就學期間幫助我無數次的忙也為我解決了不少問題。另外，我也要感謝社區的朋友們給我的鼓勵與祝福使我有信心來面對困難。我在這裡要特別感謝扶養我長大的父母–王忠祥與劉秀枝與我的妹妹們–王婷婷與王若蘋。他們一直在我身邊給我許多鼓勵與支持，讓我有信心來克服人生中遇到的種種困難。

doi:10.6342/NTU202002143

摘要

基因的結構可以使我們了解其功能，它可以透過如 Augustus 等模型的預測來獲得。這些模型為了註解 DNA 序列，需事先對其特徵組成進行分析並設計多個子模型來偵測。深度學習不需要事先分析其特徵組成並可以學習它所需要的特徵，使之容易應用在多個領域。本研究的目的為建立一個深度學習模型來對阿拉伯芥 DNA 序列上編碼基因的基因結構進行預測。本研究藉由 global run-on sequencing 和 Poly (A)-Test RNA-sequencing 的資料來清洗與重新註解現有的轉錄資料，並得到含有 977 編碼基因的註解。本研究提出一個全新的深度學習模型和新的損失函數。結果顯示深度學習在 macro F-score 的中位數為 0.969，而在 Augustus 的結果為 0.957，且統計結果顯示深度學習在 macro F-score 顯著優於 Augustus。本研究提出兩種後處理方法，一種名為邊界後處理方法（boundary post-processing method）來處理內含子的邊界，另一種名為長度過濾方法（length filtering method）來處理短片段。深度學習的預測結果經處理後在 16 個評分中有 9 個評分有顯著進步。深度學習的預測結果經後處理方法處理後顯示在 16 個評分中有 6 個顯著好於 Augustus 和 5 個顯著落後於 Augustus。這些結果顯示深度學習模型結合後處理方法可以和 Augustus 匹敵。另外，經後處理方法處理的深度學習預測結果可以在部分基因體上預測出平均為 18642 個含有已知蛋白質結構域的基因結構。整體來講，深度學習模型結合後處理方法可以成為在阿拉伯芥 DNA 序列上預測編碼基因的基因結構的替代方法。

關鍵字：阿拉伯芥、資料清洗、基因註解、深度學習、資料後處理

Abstract

The structure of the gene can help us to have a better understanding of its function, and it can be predicted by models such as Augustus. In order to annotate the DNA sequence by these models, the feature composition of annotation needed to be analyzed, and many submodels would be designed to detect these features. The deep learning does not need to analyze the feature composition and can learn the features it needs, and this makes it easily be applied in many fields. The purpose of the thesis is to build a deep-learning-based model to directly predict gene structures of coding genes in DNA sequences of *Arabidopsis thaliana*. Annotation with 977 coding gene structures was created by using data from global run-on sequencing and Poly (A)-Test RNA-sequencing to reannotate and filter the existed transcripts. A new deep learning model and loss were proposed. The median macro F-score of the deep learning model was 0.969, and the value of Augustus was 0.957. The statistical result showed that the result of the deep learning model in the macro F-score was significantly better than Augustus. Two post-processing methods were proposed, one named boundary post-processing method handled the boundary of the intron, and the other named length filtering method filtered out the region with short length. The revised result of the deep learning model showed that there were 9 out of 16 metrics performances were significantly improved. The revised result of the deep learning model showed that 6 out of 16 metrics were significantly better than Augustus, and 5 out of 16 metrics were significantly worse than Augustus. These results show that the deep learning model with the post-processing procedure is competitive to Augustus. Furthermore, the revised result of the deep learning model on the part of the genome showed that it could predict an average of 18642 gene structures that contained existed protein domains. Overall, the proposed

doi:10.6342/NTU202002143

deep learning model with the post-processing procedure can be an alternative method to predict gene structures of coding genes on DNA sequences of *Arabidopsis thaliana*.

Keywords: *Arabidopsis thaliana*, data cleaning, gene annotation, deep learning, post-processing

doi:10.6342/NTU202002143

Table of Contents

doi:10.6342/NTU202002143

doi:10.6342/NTU202002143

doi:10.6342/NTU202002143

Table of Figures

doi:10.6342/NTU202002143

doi:10.6342/NTU202002143

Table of Tables

doi:10.6342/NTU202002143

Table of Equations

Table of Algorithms

doi:10.6342/NTU202002143

List of Abbreviations

Abbreviation	Full description
AS	Acceptor site
BCE	Binary cross-entropy
BHR	Basic hierarchy relation
BN	Batch normalization
BR	Basic relation
CDS	Coding sequence
CE	Categorical cross-entropy
ChIP	Chromosome immunoprecipitation
CNN	Convolution neural network
CPSF	Cleavage and polyadenylation specificity factor
CRB	CNN-ReLU-BN
CS	Cleavage site
CstF	Cleavage stimulation factor
DHS	DNase I hypersensitive site
DL	Deep learning
DNA	Deoxyribonucleic acid
DRS	Direct RNA sequencing
DS	Donor site
DSE	Downstream element
GRO-seq	Global run-on sequencing
GRU	Gated recurrent unit
GTF	General transcription factor

doi:10.6342/NTU202002143

(Cont.) List of Abbreviations

Abbreviation	Full description
HMM	Hidden Markov model
HR	Hierarchy relation
nt	Nucleotide
PABP	Polyadenylation binding protein
PAP	Polyadenylation polymerase
PAT-seq	Poly (A)-Test RNA-sequencing
PEAT	Paired-end analysis of transcription start sites
ReLU	Rectified linear unit
RNA	Ribonucleic acid
RNAP	RNA polymerase
RNN	Recurrent neural network
RT-PCR	Reverse transcriptase-polymerase chain reaction
SNP	Single nucleotide polymorphism
snRNP	Small nuclear ribonucleoprotein
SVM	Support vector machine
TF	Transcription factor
TFBS	Transcription factor binding site
TSS	Transcription start site
UTR	Untranslated region

doi:10.6342/NTU202002143

doi:10.6342/NTU202002143

Chapter 1. Introduction

In order to understand the genes of a species, the most direct method is extracting its transcripts, using sequencing to get their sequences, and mapping them to its genome. Thanks to the growing number of sequencing data, the annotations of these transcripts were available at recent annotations, such as an annotation of *Arabidopsis thaliana* called Araport11 **[1]**. These annotations are hard to cover full transcriptome because it is hard to extract all the transcripts from different conditions. Besides, traditional RNA sequencing often gets fragments of RNA and cannot get the exact locations of the start sites and end sites of transcripts. There are two kinds of methods to improve the coverage of annotation. One is using the existed transcript structure on a similar DNA region from the same or related species to inference the transcript structure. The other one is to train a mathematical model with known transcript structures. Then, the model is used to predict the transcript annotation on the whole genome. Although it is almost impossible to gather all transcripts of one gene, we can indeed generate all "hypothetical transcripts" from "gene structure," which includes all splicing pairs in transcripts of the gene. If we can predict gene structure correctly, then we can generate all "hypothetical transcripts" by these splicing pairs. These "hypothetical transcripts" can be further studied their existence, their translation potential, and their potential function.

There was a hidden Markov model (HMM) called Augustus that could predict transcripts of eukaryotes **[2]**. The feature composition of annotation was first analyzed, and many submodels were carefully designed to detect these features. The model was used to predict transcript structure, not the gene structure. Recently, a technology called deep learning was widely used to the classification of the image **[3]**, prediction of gene function **[4]**, prediction of protein-coding potential of RNA **[5]**, and prediction of

doi:10.6342/NTU202002143

antimicrobial potential of peptide **[6]**. These results showed improvement from the traditional methods. Many studies tried to use deep learning model to annotate the genome of eukaryotes, like predicting the existence of cleavage sites **[7]** and splicing sites **[8, 9]**. Nevertheless, they could not predict the complete annotation directly. Recently, a deep learning model named DeepAnnotator **[10]** was proposed to predict gene structure on the genome of prokaryotes. However, DeepAnnotator was three separate models that predict part of gene structure, and then the result of these three models was merged into one gene structure. The models were trained separately, so they could not learn feature between the models.

Currently, there is no deep-learning-based model can directly predict gene structure of coding genes of eukaryotes. The purpose of this thesis is to build a deep-learning-based model to directly predict the gene structure of coding genes only by their DNA sequences, using *Arabidopsis thaliana* as an example. The overall workflow is shown in **Figure 1**. The first part is to use the existed data to create datasets of high confidence gene annotation, as shown in **Figure 1a** and **section 3.2. Section 3.3** describes the methods of inference methods, losses, and deep learning models. The second part is to use the part of training datasets to get the best hyperparameters by hyperparameter optimization, as shown in **Figure 1b** and **section 3.4**. The third part is to use models with the best hyperparameters to do cross-validation and testing, as shown in **Figure 1c, Figure 1d,** and **section 3.5**. The fourth part is to get the best reviser to revise the predicted result on the training dataset, as shown in **Figure 1e** and **section 3.6**. The fifth part is to predict gene annotation on potential transcript regions and revise it by the best reviser, as shown in **Figure 1f**, **section 3.5**, and **section 3.6**. **Section 3.7** describes the procedures of training and testing Augustus. All the names and versions of the main software and packages are shown in **Table S1**.

doi:10.6342/NTU202002143

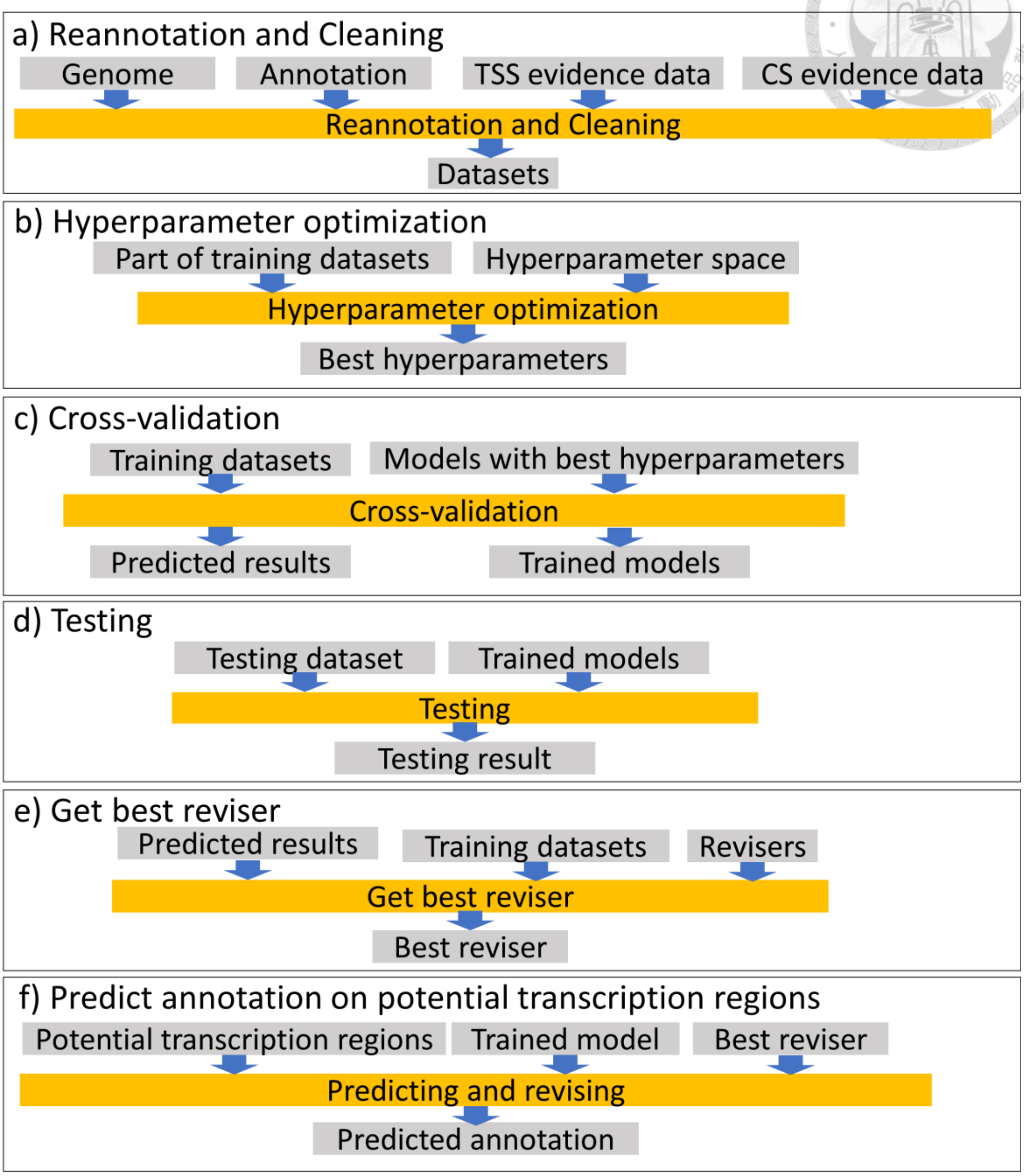


Figure 1. The workflow of the thesis

doi:10.6342/NTU202002143

Chapter 2. The Literature Review

2.1 Annotation identification on *Arabidopsis thaliana* ecotype Col-0

A plant called *Arabidopsis thaliana* is often used as a model organism to study plant genome. Human experts or machines annotate genes and transcripts of *Arabidopsis* with experimental datasets and annotations from the model prediction. TAIR10 **[11]** was the previous annotation of *Arabidopsis thaliana* ecotype Col-0. It used tools such as TopHat **[12]** with RNA-mapping results to identify transcripts. After filtering transcript annotation and adding peptide-mapping results, Augustus **[13]** was trained with these initial annotation. The trained model was used to predict transcripts on the genome. Then, human experts manually curated these results. Currently, the newest annotation of *Arabidopsis thaliana* ecotype Col-0 was Araport11 **[1]**. Araport11 used TopHat **[12],** Trinity **[14]**, and 113 RNA datasets from 11 tissues to construct transcript annotation. Then, peptide datasets with tools such as MAKER-P **[15]** were used to augment the TAIR10 dataset. Finally, the tool named PASA **[16]** was used to update the augmented TAIR10 dataset.

2.2 Transcription and splicing in eukaryotes

Transcription is a process to transcribe the information on DNA sequence to RNA sequence named transcript. First, general transcription factors (GTFs) and RNA polymerase (RNAP) bound to the promoter, then RNAP moved to the transcription start site (TSS) and starts to synthesized RNA **[17]**. The location of the promoter might be located in the region from -500 nucleotides (nts) from TSS to TSS based on the single nucleotide polymorphism (SNP) density profile in the previous study **[18]**. Based on the other research **[19]** in *Arabidopsis*, about 63% of transcription factor binding sites (TFBSs) were located in -400 nts to +200 nts from annotated TSSs, and these sites had

doi:10.6342/NTU202002143

passed conservation test in *Arabidopsis lyrata*, *Brassica oleracea*, and *Brassica rapa*. The RNA was then cleaved at the cleavage site (CS) and was added hundreds of adenines **[20]**. The cis-regulatory elements that participated in cleavage and polyadenylation were elements such as AAUAAA hexamer, CS, and downstream element (DSE) **[20]**. The cleavage and polyadenylation specificity factor (CPSF), which bound to AAUAAA hexamer, and cleavage stimulation factor (CstF), which was bound DSE, involved in cleavage **[20]**. Then, CPSF, polyadenylation polymerase (PAP), and polyadenylation binding protein (PABP) involved in polyadenylation **[20]**. Based on nucleotide composition profiles of around CS of *Arabidopsis* in the previous study **[21]**, the U-rich DSEs of most preferred CSs were located within the region downstream of CS to downstream 60 nts. The transcript needs to be spliced to become a mature transcript. Splicing had multiple elements involved like donor site (DS), acceptor site (AS), and branch point, and it also had factors like small nuclear ribonucleoproteins (snRNPs) that involved in splicing and bound to these elements **[22]**. Briefly, 2'OH of branch point attacked to DS of the RNA, and it caused RNA to be spliced at DS **[22]**. Then 3'OH of spliced RNA attacked to AS of RNA, and it caused RNA to be spliced at AS caused spliced RNAs to be joined **[22]**.

2.3 Alternative TSSs and alternative CSs

One gene can be transcribed into multiple isoforms based on different conditions. These conditions affect cell to choose different sites to start transcription, different sites to be cleavaged, different sites to act as splicing sites, and different sites to be spliced. Alternative TSSs can act as a regulatory mechanism or change its peptide product. It had been reported that alternative TSSs occurred in *Arabidopsis* when it was exposed by blue light **[23]** and that alternative TSSs occurred in mice during cerebellar

doi:10.6342/NTU202002143

development **[24]**. The paired-end analysis of transcription start sites protocol (PEAT protocol) could reveal the location distribution of TSSs **[25]**. The previous study **[26]** in *Arabidopsis* showed only a minority of TSS tag clusters had narrow and sharp distribution, and most TSS tag clusters had broad and flat shapes. Another technology called global run-on sequencing (GRO-seq) was also invented to show the location and strength of TSS **[27]**. Alternative CSs can also act as a regulatory mechanism or change their peptide products. Direct RNA sequencing (DRS) could provide locations and strengths of CSs **[28]**. The previous study **[21]** in *Arabidopsis* showed about 90% of DRS read were mapped on coding genes, and 8.2% of DRS reads were mapped on intergenic regions. Nearly half of these intergenic-DRS reads were located directly downstream of the annotated gene within 300 nts, and the reverse transcriptase-polymerase chain reaction (RT-PCR) experiment showed that DRS could reveal the position of the true cleavage site **[21]**. After extending the end of the gene to the location of DRS reads, about 94% of DRS reads were mapped on coding genes, and 74.9% of protein-coding genes had alternative CSs **[21]**. There was also a method called Poly (A)-Test RNA-sequencing (PAT-seq) that had been invented **[29]**. It could also provide locations and strengths of CSs. Both DRS and PAT-seq could avoid the internal priming problem and reveal locations of true CSs **[21, 29]**.

2.4 *Ab initio* transcript structure prediction

Ab initio transcript structure prediction is often predicted by the HMM. Annotation with the carefully cleaning procedure can be used to train the models and evaluate the performance of the models. The post-processing can be applied to the predicted result to correct the mistakes the models made. Many studies focused on the annotation of prokaryotes because the lengths of their genes are shorter than eukaryotes, and annotations of eukaryotes are much more complicated because of the existence of intron and alternative splicing. Most of the existed HMMs focused on annotation between the start codon and stop codon and treated untranslated regions (UTRs) and introns between UTRs as intergenic regions.

In the early study **[30]**, HMMs were proposed to predict coding genes in *E.coli*. The “gene” region in this study was defined as the sequence between the start codon and the stop codon. The model was composited by the start codon model, coding gene model, stop codon model, intergenic regions model, and long intergenic regions model. The model would be trained and predicted on each strand of the sequences independently. The post-processing would be applied to the prediction of trained models.

The model called Genie was proposed to predict coding genes in *Homo sapiens* **[31]**. The “gene” in this study was also defined as the sequence between the start codon and the stop codon. Genie was a generalized HMM-based model. The generalized HMM, unlike HMM, could generate sequence but not the character of each state, so it could generate sequences which their lengths were arbitrary distribution. Genie also integrated the intron model, exon model, and splicing site detectors. It also included frame constraints to make sure the length of the coding region was multiple of three.

Augustus **[2, 13, 32, 33]** was used to predict the annotation of the coding gene. The model was similar to Genie but with a complicated intron model. The intron model was composed of two submodels for long intron and short intron so that it could have better results. The model was further expanded to include hints. The expansion could improve its prediction and predict UTR and UTR related intron. By training multiple models and using a sampling algorithm, the model could also predict alternative transcripts.

doi:10.6342/NTU202002143

Some tools tried to combine many different models to get a better prediction. MAKER2 **[34]** was a pipeline that used results of SNAP **[35]**, GeneMark **[36]**, and Augustus to predict annotation. Seqping **[37]** was also a pipeline that used results of GlimmerHMM **[38]**, Augustus, and MAKER2 to predict the annotation of the plant.

2.5 Deep learning related techniques

Convolution neural network (CNN) had been widely used at object classification **[39]** and object detection **[40]**. The most often used function after the convolution layer was the rectified linear unit (ReLU), and its formula is shown in **Equation 1 [41]**. The other functions are standard logistic function, tanh, and softmax **[42]**, and their formulas are shown in **Equation 2**. The *x* indicates input value, x_i indicates the input value at dimension *i*, the s indicates the standard logistic function, the K indicates the number of output dimension, and *e* indicates Euler's number. The ReLU is often used because its largest gradient is one, so the gradient passes by will not be easily decreased. Recently, the stacked CNN architecture named ResNet **[43]** was proposed. Its main idea was to use a shortcut connection like $x_i = \mathrm{F}_i(x_{i-1}) + x_{i-1}$ to construct model. The F_i means any neural network at layer *i* and x_i means value after layer *i*. The shortcut connection made the layer could directly copy its input and add it to output of the layer. The shortcut connection made ResNet could train and backpropagate its gradient more efficiently while the model had many layers. Most of the experiment results showed that the model could achieve lower loss value when its layer number increased. A stacked CNN architecture named DenseNet **[3]** was proposed. It used shortcut connection like $x_i = \mathrm{Concat}(\mathrm{F}_i(x_{i-1}),\ x_{i-1})$ to construct model, so a layer could reuse all outputs of its previous layers. The F_i means any kind of neural network at layer *i*, x_i means value after layer *i*, and Concat means operator to concatenate all its inputs. Most of its experiment results also showed that the model could achieve lower loss value when its layer number increased. DenseNet could get lower loss than ResNet did while using fewer parameters.

Recurrent neural network (RNN) had been widely used in jobs with spatial or temporal data such as audio tagging and time series classification **[8, 10, 44, 45]**. **Equation 3** shows the primitive RNN formula **[46]**. The x_t means the input at timestep t, and the h_t means the hidden state at timestep t. The U and W are weights matrix, and b is a bias vector. The function σ is any activation function. The simple RNN has some severe issues during training. The weights U are shared by all timesteps. During backpropagation in the long sequence, if one of the values in weights U is larger than one, then the gradient will exponentially growth and causes a problem called gradient exploding problem, it makes the training procedure being unstable. During backpropagation in the long sequence, if one of its values is smaller than one, then the gradient will exponentially decay to zero, it causes a problem called vanishing gradient problem, it makes the weights hard to be updated. The more advanced RNN called gated recurrent unit (GRU) had been proposed **[47]**. It had reset gate r and update gate z. **Equation 4** shows the formula of the GRU. The x_t means the input at timestep t, and the h_t means the hidden state at timestep t. The U_i and W_i are weights matrixes for value i, and b_i is a bias vector for value i. The GRU could use its gate mechanism to relieve the vanishing gradient problem. If gate z_t is one, then the gradient will direct copy the previous gradient.

The range of gradient of each parameter may be large, and it is hard to set the learning rate for each parameter. An optimizer called Adam **[48]** was proposed to handle this issue. Adam would consider the square values of previous and current

doi:10.6342/NTU202002143

gradients of each parameter and use them to adjust the learning rate of each parameter. It also used the momentum to accelerate the training procedure.

The batch normalization layer (BN layer) was a kind of layer that normalized the input data, so the mean of each output feature was close to zero, and its variance was close to one **[49]**. During training over the mini-batch data, the mean and variance of each feature were used to normalization the input feature. During the testing phase, the mean and variance that were used to normalization the input feature were calculated by using the moving average of the means (μ_β) and variances (σ_β^2) of mini-batch data. The formulas are showed in **Equation 5**, **Equation 6**, and **Equation 7**. The M indicates the batch size, and ϵ indicates an arbitrarily small positive number. The main benefit of batch normalization was to accelerate the training speed by smoothing its optimization landscape **[50]**.

Dropout **[51]** was a simple method to prevent model overfitting the training data. It randomly dropped out some outputs of hidden units by probability $1 - p$ to generate submodels during training phase and used all the hidden units during the testing phase. The large model tends to overfit the data, so using the dropout could let the model be thinner and prevent overfitting. The hidden units in the model would work together, and they would highly dependent on each other. The co-adaption might decrease the ability of each hidden unit to produce useful information by itself. The dropout could break the co-adaption so that every hidden unit could generate useful information by itself. The experiments showed that the dropout in CNN could break the co-adaption between hidden units and archive better results when the data was large enough **[51]**. The experiments also showed the dropout rate around 0.5 in CNN could generate the best performance and had a similar result of the Monte-Carlo model average method **[51]**. The previous study **[52]** showed the dropout could be applied in a feed-forward

doi:10.6342/NTU202002143

connection in RNN to prevent overfitting and archive a better performance. The average outputs of hidden units during the training phase and the testing phase should be similar, but the dropout could lower the average of the outputs. So, the scaling must be applied to make them similar between the two phases. The dropout implementation of PyTorch **[53]** would scale the outputs of hidden units by $\frac{1}{1-p}$ during the training phase and would use the origin outputs during the testing phase.

Hyperparameter optimization is a process to find hyperparameters that may achieve the best result. There are three kinds of methods to find hyperparameters. One is called grid search, and it tries all combinations in hyperparameter space. The advantage of this method is that it can find the best hyperparameter set. The disadvantage of this method is that the number of combinations is too huge, so the time to explore all combinations is large. Another method is called random search. It randomly uses some hyperparameter sets. The advantage is that it reduces the time, and the disadvantage is that it cannot efficiently find a good hyperparameter set. The other was Bayesian optimization with the Gaussian process **[54]**. It used all the previous results to find the next hyperparameters to be used. The advantage was that it could efficiently find a good hyperparameter set.

$$\mathrm{ReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Equation 1. ReLU formula

doi:10.6342/NTU202002143

$$\mathrm{s}(x) = \frac{1}{1 + e^{-x}}$$

$$\tanh(x) = 2s(2x) - 1$$

$$\mathrm{softmax}(x)_{\mathrm{i}} = \frac{e^{x_i}}{\sum_{k=1}^{\mathrm{K}} e^{x_k}}$$

Equation 2. Sigmoid, tanh, and softmax

$$h_t = \sigma(Uh_{t-1} + Wx_t + b)$$

Equation 3. Simple RNN formula

$$r_t = \mathrm{s}(U_r h_{t-1} + W_r x_t + b_r)$$

$$z_t = \mathrm{s}(U_z h_{t-1} + W_z x_t + b_z)$$

$$\tilde{x}_t = \tanh(U_h(r_t \circ h_{t-1}) + W_h x_t + b_h)$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{x}_t$$

Equation 4. GRU formula

$$\mu_\beta = \frac{1}{\mathrm{M}} \sum_{i=1}^{\mathrm{M}} x_i$$

$$\sigma_\beta^2 = \frac{1}{\mathrm{M}} \sum_{i=1}^{\mathrm{M}} (x_i - \mu_\beta)^2$$

$$y_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}$$

Equation 5. Batch normalization formula without affine during the training phase

$$\mathrm{E}[x]_{\mathrm{new}} = (1 - \mathrm{momentum}) \times \mathrm{E}[x] + \mathrm{momentum} \times \mu_\beta$$

$$\mathrm{Var}[x]_{\mathrm{new}} = (1 - \mathrm{momentum}) \times \mathrm{Var}[x] + \mathrm{momentum} \times \frac{\mathrm{M}}{\mathrm{M} - 1} \sigma_\beta^2$$

Equation 6. Calculating mean and variance for the testing phase

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}}$$

Equation 7. Batch normalization formula without affine during the testing phase

2.6 Deep learning applications related to sequence annotation

The DeepPolyA was a CNN model that predicted whether the 161-nt RNA sequence included the cleavage site or not **[7]**. The metrics like the F-score and accuracy of DeepPolyA were better than the traditional approaches like support vector machine (SVM) and random forest.

The COSSMO was a CNN-RNN model that predicted the percentage selected indices of splicing sites **[8]**. Given an RNA sequence around the constitutive DS, multiple RNA sequences around alternative ASs, multiple spliced RNA sequences, and lengths between DS and ASs, COSSMO could predict the percentage selected indices of these ASs. The accuracy and coefficient of determination (R^2) of COSSMO were better than the traditional model, such as MaxEntScan **[55]**.

The SpliceAI was a CNN model that predicted whether the 10001-nt DNA sequence on the human genome was centered at DS or AS **[9]**. The result showed that the top-k accuracy of SpliceAI was better than the traditional method like MaxEntScan **[55]**. The predicted locations of splicing sites on the mutation sequences were agreed with the experimental data.

The model named DeepAnnotator was proposed to predict the gene structures of prokaryotes **[10]**. The DeepAnnotator was composited of three separate models. The first one predicted whether the center of RNA sequence was translation start site or not, and the second one predicted whether the RNA sequence was centered by the translation stop site or not, and the third one predicted whether the RNA sequence was centered by coding nucleotide or not. The integrated prediction of the three models was

doi:10.6342/NTU202002143

the predicted result of gene structures. The F-score of the DeepAnnotator was 94%, which was higher than the F-score-score of the Glimmer **[56]**.

doi:10.6342/NTU202002143

Chapter 3. Materials and Methods

3.1 The data preparation

The genome of *Arabidopsis thaliana* ecotype Col-0 named GCF_000001735.3 **[11]** was used as a reference genome. Transcriptome in nuclear chromosomes named Araport11 was used as annotation data **[1]**. 5’ GRO-seq dataset and GRO-seq datasets from the previous study **[57]** were first mapped on the genome by STAR **[58]**, then it was used to find locations of GRO-seq signals by HOMER **[59]**. The PAT-seq clusters in the dataset named SRP089899 were used as evidence of CSs **[60]**. The data source summary is showed in **Table S2**. The GRO-seq signals and PAT-seq clusters were treated as TSS evidence and CS evidence.

The preprocessing of the annotation is described below. The miRNAs in the transcriptome were removed. The left transcriptome was named as **background transcripts**. If any gene in background transcripts had inconsistent data, then it and all its transcripts would be removed. The left transcriptome was named as **consistent transcripts**.

3.2 The workflow of creating annotation datasets

There were three steps to generate the annotation datasets from the genome, transcriptome, TSS evidence, and CS evidence.

The first step used experimental data and consistent transcripts to get the reannotated transcripts. For every transcript, TSS evidence was considered related if it was located on the region from upstream *u* nts of its TSS to its CS. For every transcript, CS evidence was considered related if it was located on the region from the TSS to the downstream *d* nt of its CS. For every transcript, the strongest TSS evidence must locate on external 5’ UTR, and the strongest CS evidence must locate on external 3’ UTR;

otherwise, the transcript failed to be reannotated and was discarded. For every transcript, its TSS would be reannotated by its strongest TSS evidence, and its CS would be reannotated by its strongest CS evidence. The example of the reannotating transcript is shown in **Figure 2**. **Figure S1** shows examples of transcripts that fail to be reannotated. If all transcripts of a gene had the same boundary, then they would be preserved; otherwise, they were discarded. The left transcriptome was named as **reannotated transcripts**.

Reannotated transcripts must pass five filters. The **score filter** would remove the gene and all its transcripts if the gene had any transcript which its score was worse than T2. The **overlapped filter** would remove the gene and all its transcripts if one of its transcripts were overlapped with the transcript of other genes on the same strand. The **alternative-splicing-site filter** would remove the gene and all its transcripts if the gene had any alternative splicing site. The **coding filter** would remove the gene all its transcript if the gene had a non-coding transcript. The **hypothetical-protein filter** would remove the gene and all its transcripts if the protein of gene belonged to hypothetical protein. The left transcriptome was named as **filtered transcripts**. If the gene name of the transcript in background transcripts and reannotated transcripts did not exist in gene names of filtered transcripts, then these transcripts would be added to the dataset named **discarded transcripts**. The filtered transcripts would be treated as **preserved transcripts**. The recursive cleaning procedure would recursively remove preserved transcript and add it to discarded transcripts if the region around its gene was overlapped with the discarded transcripts. The procedure would be stopped until the number of preserved transcripts stopped decreasing. **Algorithm 1** shows the pseudocode of the recursive cleaning procedure. **Figure 3** shows the simple examples of discarding preserved transcript.

doi:10.6342/NTU202002143

The second step used the genome and the preserved transcripts to create gene annotation. The boundary was extended from upstream *u* nts to downstream *d* nts of the preserved transcript. The double-strand region in the boundary would be selected. The selected region was discarded if it had any nucleotide not belonging to nucleotide A, T, C, or G, or the number of genes that it covered was larger than one. The left selected regions were called clean selected regions. **Figure 4** shows examples of creating clean selected regions. The UTR and CDS were merged as exon, and then gene annotation was created by using locations of TSSs, CSs, and splicing sites of transcripts. All the regions that were not annotated were annotated as the intergenic region. The example in **Figure 5** shows the example of creating gene annotation of multiple-exon gene.

The third step split these regions, transcript annotation, and gene annotation into many datasets. The data located on all five chromosomes was named **Data**$_{\textbf{Whole}}$. The regions and their annotation were then be split according to their belonging chromosomes and strands. The data located on chromosomes 1, 2, 3, and 5 were named **Data**$_{\textbf{Train.}}$ **Train**$_{\textbf{1}}$ ~ **Train**$_{\textbf{8}}$ would be used to be training datasets for 8-fold cross-validation of the deep learning model. **Val**$_{\textbf{1}}$ ~ **Val**$_{\textbf{8}}$ would be used to be validation datasets for 8-fold cross-validation of the deep learning model. **Data**$_{\textbf{Test}}$ would be a testing dataset of the deep learning model. The region could be classified into three types. One was “region without exon,” another was “region with single-exon,” and the other was “region with multiple-exon.”. **Train**$_{\textbf{Small,}}$ which is a training dataset for the hyperparameter optimization procedure, was created by using regions that were the first half shortest lengths of each region type in **Train**$_{3}$. **Val**$_{\textbf{Small}}$, which is the validation dataset for hyperparameter optimization procedure, was created by using regions that were the first half shortest lengths of each region type in **Val**$_{\textbf{3}}$. The splitting result is shown in **Table S3**.

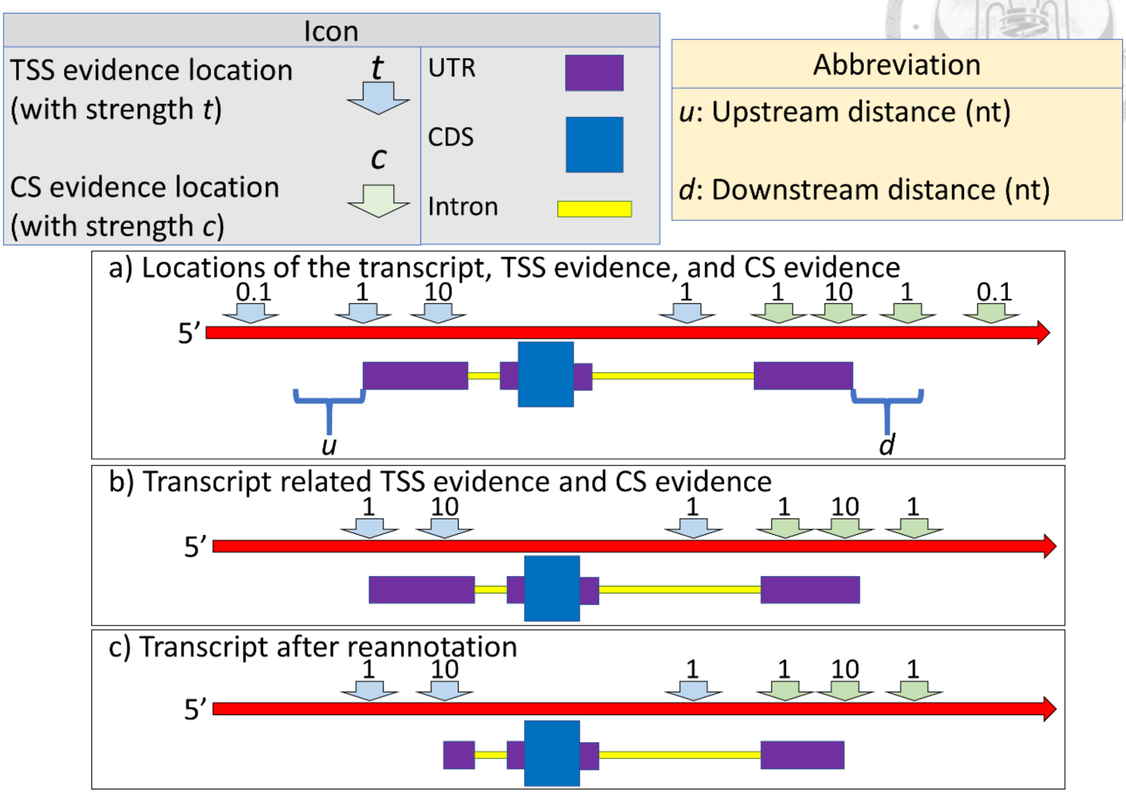

Icon
TSS evidence location
(with strength t)
t
CS evidence location
(with strength c)
c
UTR
CDS
Intron
Abbreviation
u: Upstream distance (nt)
d: Downstream distance (nt)
a) Locations of the transcript, TSS evidence, and CS evidence
0.1
1
10
1
1
10
1
0.1
5’
u
d
b) Transcript related TSS evidence and CS evidence
1
10
1
1
10
1
5’
c) Transcript after reannotation
1
10
1
1
10
1
5’

Figure 2. Find evidence related to each transcript and use the strongest evidence located on external UTR to redefine the boundary

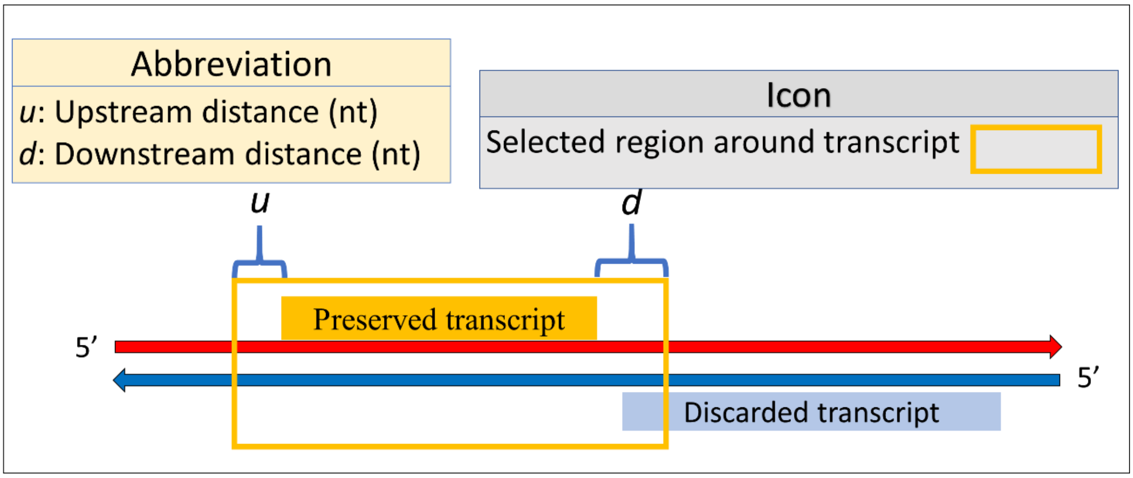

Abbreviation
u: Upstream distance (nt)
d: Downstream distance (nt)
Icon
Selected region around transcript
u
d
Preserved transcript
5’
5’
Discarded transcript

Figure 3. The example of getting discarded transcript and removing preserved transcript

doi:10.6342/NTU202002143

Algorithm 1. The recursive cleaning procedure

```
Required:
    P: The set of preserved transcripts
    D: The set of discarded transcripts
    u: The upstream distance from a transcript
    d: The downstream distance from a transcript

flag ← true
while flag do
    discaredNames ← empty set
    num ← GetGeneNumber(P)
    for each $p \in P$ do
        region ← CreateRegion(p,u,d)
        if region is overlapped with transcripts in D then
            name ← GetGeneName(p)
            discaredNames.add(name)
    for each $p \in P$ do
        if GetGeneName(p) $\in$ discaredNames then
            D.add(p)
            P.remove(p)
    if num = GetGeneNumber(P) then
        flag ← false
return P
```

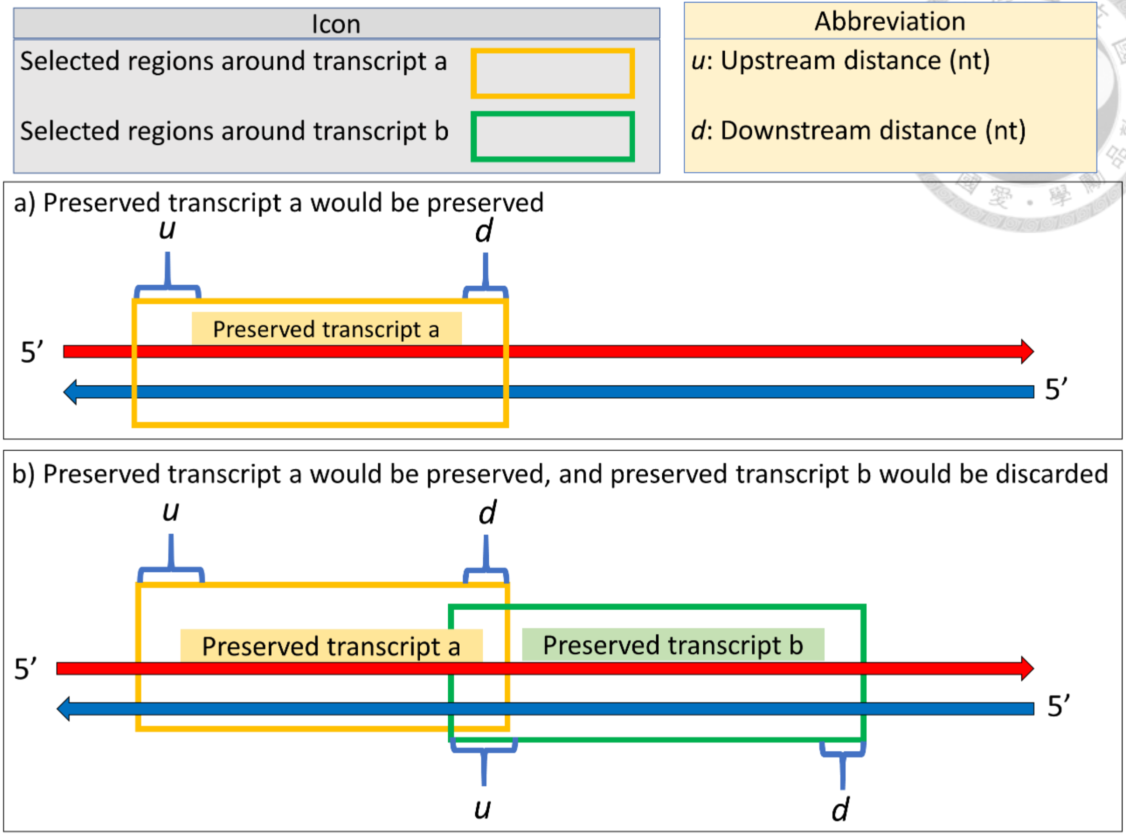

Icon
Selected regions around transcript a
Selected regions around transcript b
Abbreviation
u: Upstream distance (nt)
d: Downstream distance (nt)
a) Preserved transcript a would be preserved
u
d
Preserved transcript a
5′
5′
b) Preserved transcript a would be preserved, and preserved transcript b would be discarded
u
d
Preserved transcript a
Preserved transcript b
5′
5′
u
d

Figure 4. The examples of creating clean selected regions

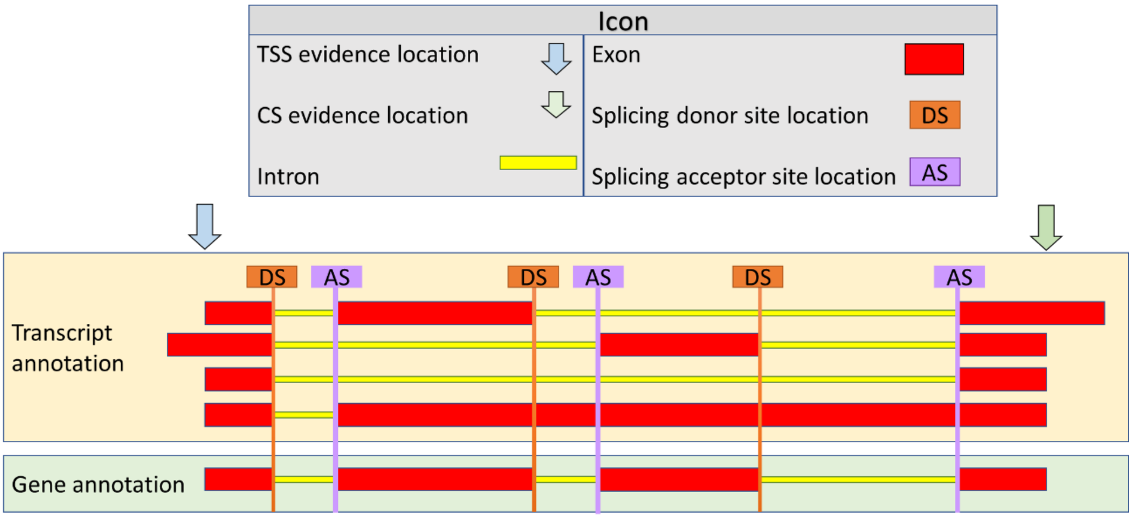


Figure 5. The demo of constructing gene annotation of a multiple-exon gene

(Notes: the intergenic region is not drawn in the figure.)

doi:10.6342/NTU202002143

3.3 Label inference methods, loss functions, and model architectures

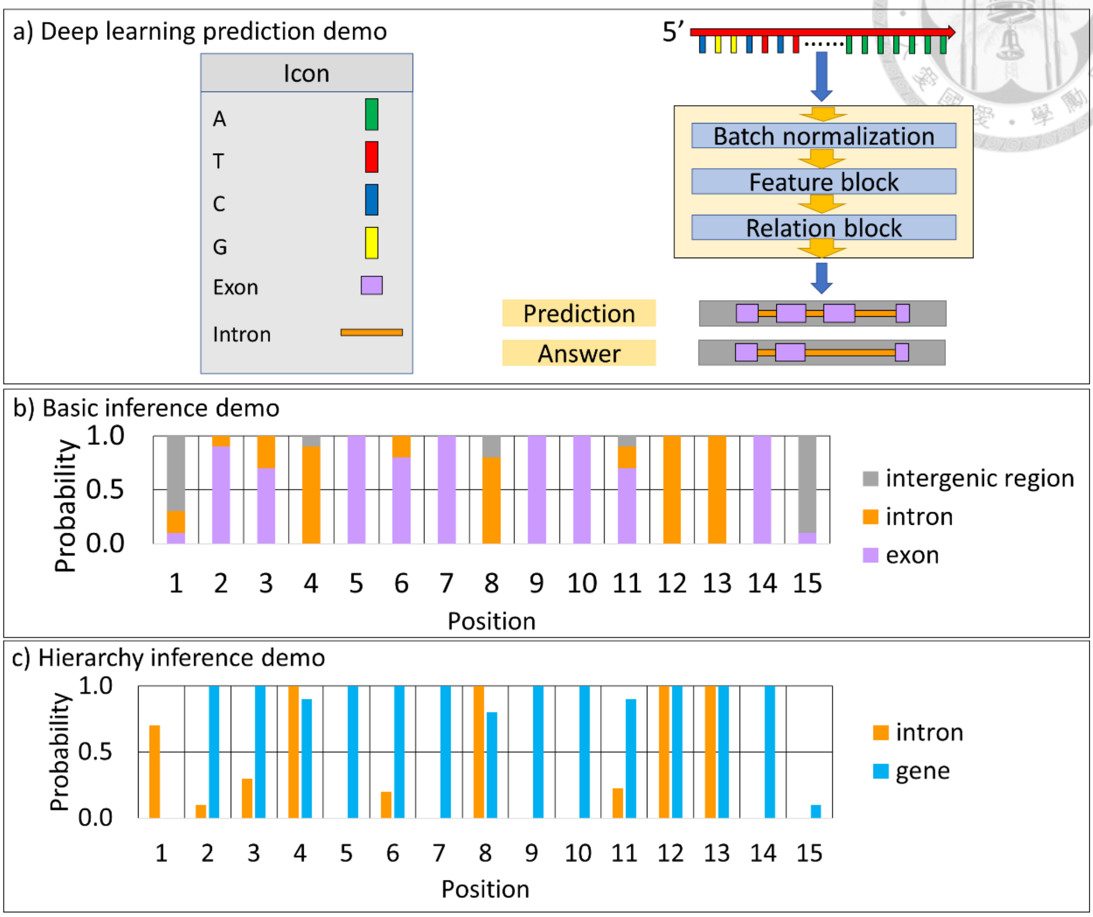


Figure 6. The schematic diagram of a) the model prediction demo, b) basic inference demo, and c) hierarchy inference demo

Figure 6a shows the schematic diagram of the model and its prediction example. The model would use a single-strand DNA sequence in a forward direction to predict gene annotation. The goal of the training process was to make its prediction as close to its answer as possible.

There were two types of inference methods. One method called the basic inference method predicted the probability of exon, intron, and the intergenic region at each position. The example is shown in **Figure 6b**. **Equation 8** shows the formula of the basic inference method. The other method that called the hierarchy inference

doi:10.6342/NTU202002143

method predicted the probability of gene and intron at each location. The example is shown in **Figure 6c**. **Equation 9** shows the formula of the hierarchy inference method. There were also two types of losses. The basic loss was the mean categorical cross-entropy, and the hierarchy loss was the mean of gene loss and intron loss. **Equation 11** shows the formula of the basic loss. **Equation 12** shows the formula of the hierarchy loss. **Equation 10** shows the formulas of the cross-entropy. The c_i indicated the predicted label type in position *i*. The $y'_{i,j}$ indicated the predicted value in position *i* and label j, and the $y_{i,j}$ indicated the true value in position *i* and label j. The C indicated the number of label types. The L indicated the number of labels.

The deep learning model was composed of three different kinds of layers or blocks, one is a batch normalization layer, another is the feature block, and the other is the relation block. **Figure 6a** shows the model architecture of the deep learning model. The basic building block of the feature block was CNN-ReLU-BN block (CRB block). The output and the input of the CRB block were concatenated as a concatenated CRB block. Many concatenated CRB blocks were stacked together as a feature block. **Figure 7** shows the architectures of the CRB block, concatenated CRB block, and feature block. There were three types of relation blocks. The first one is the basic relation block (BR block), and the kernel size and output dimension of its CNN were one and three. The second one is the basic hierarchy relation block (BHR block), and the kernel size and output dimension of its CNN were one and two. The third one is the hierarchy relation block (HR block), which is composed of two BHR blocks in which the output dimension of each block is one. The first BHR block predicted gene probability, and the second BHR block predicted intron probability. **Figure 8** shows the architectures of the three types of relation blocks. The model which used the BR block used the basic loss and basic inference method. The model which used the BHR block and HR block used

doi:10.6342/NTU202002143

the hierarchy loss and hierarchy inference method. The weights and bias initialization are described in **Table S4**. Models were built and trained by PyTorch, and all the other parameter initialization was set to the default value of PyTorch.

$$c_i = \begin{cases} \text{exon} & \arg\max_j y'_{i,j} = 0 \\ \text{intron} & \arg\max_j y'_{i,j} = 1 \\ \text{intergenic region} & \arg\max_j y'_{i,j} = 2 \end{cases}$$

Equation 8. Basic inference method

$$c_i = \begin{cases} \text{intergenic region} & {y'}_{i,0} < \text{threshold}_{\text{gene}} \\ \text{exon} & {y'}_{i,0} \geq \text{threshold}_{\text{gene}} \text{ and } {y'}_{i,1} < \text{threshold}_{\text{intron}} \\ \text{intron} & {y'}_{i,0} \geq \text{threshold}_{\text{gene}} \text{ and } {y'}_{i,1} \geq \text{threshold}_{\text{intron}} \end{cases}$$

Equation 9. Hierarchy inference method

$$\text{BCE}({y'}_i, y_i) = -(y_i \times \ln {y'}_i + (1 - y_i) \times \ln(1 - {y'}_i))$$
$$\text{CE}({y'}_i, y_i) = -\sum_{j=0}^{\text{C}-1} y_{i,j} \times \ln {y'}_{i,j}$$

Equation 10. Binary cross-entropy (BCE) and categorical cross-entropy (CE)

$$\text{Loss}_{\text{basic}}(y', y) = \frac{1}{\text{L}} \sum_{i=0}^{\text{L}-1} \text{CE}({y'}_i, y_i)$$

Equation 11. Basic loss

$$\mathrm{Loss_{gene}}\left(y',y\right)=\frac{1}{\mathrm{L}}\sum_{i=0}^{\mathrm{L}-1}\mathrm{BCE}\left(y'_{i,0},y_{i,0}\right)$$

$$\mathrm{Loss_{intron}}\left(y',y\right)=\frac{1}{\sum_{i=0}^{\mathrm{L}-1}y_{i,0}}\sum_{i=0}^{\mathrm{L}-1}y_{i,0}\times\mathrm{BCE}\left(y'_{i,1},y_{i,1}\right)$$

$$\mathrm{Loss_{hierarchy}}\left(y',y\right)=\frac{\mathrm{Loss_{gene}}\left(y',y\right)+\mathrm{Loss_{intron}}\left(y',y\right)}{2}$$

Equation 12. Gene loss, intron loss, and hierarchy loss

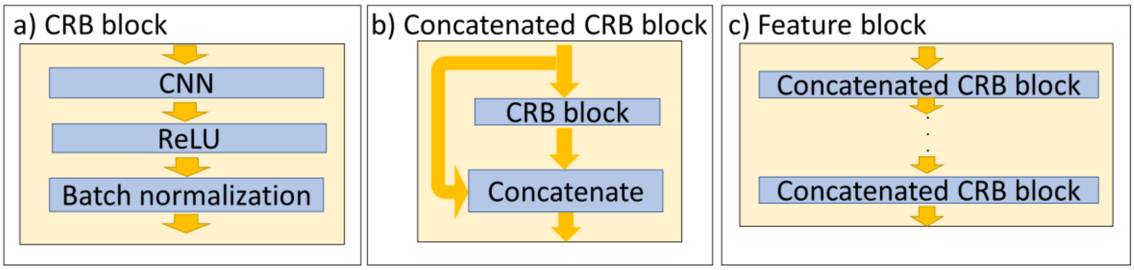


Figure 7. The architectures of the a) CRB block, b) concatenated CRB block, and c) feature block

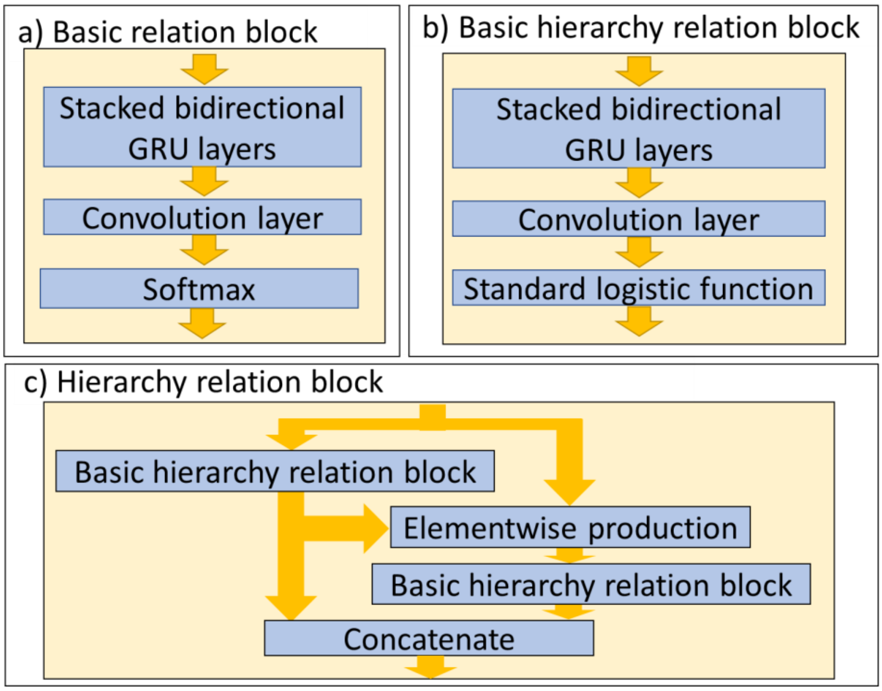

a) Basic relation block
Stacked bidirectional GRU layers
Convolution layer
Softmax
b) Basic hierarchy relation block
Stacked bidirectional GRU layers
Convolution layer
Standard logistic function
c) Hierarchy relation block
Basic hierarchy relation block
Elementwise production
Basic hierarchy relation block
Concatenate

Figure 8. The schematic diagrams of the relation blocks

3.4 Hyperparameter optimization procedure, cross-validation, testing, and augmentation

Hyperparameter optimization procedure first used a random search to generate 40 hyperparameter sets and using them to create 40 models. These models were trained, and the macro F-scores of the lowest validation losses were recorded. Then, Bayesian optimization with the Gaussian process used all its previous results to generated a hyperparameter set that may achieve the highest validation macro F-score. If the Bayesian optimization found there was no improvement in five trials, then the optimization procedure would stop. The hyperparameter set that achieved the highest validation macro F-score was used for cross-validation. The hyperparameter space is described in **Table 1**, and the total space size is 3072. The hyperparameter optimization procedure was implemented with a python package called Optuna **[61]**. Adam optimizer was used due to its ability to accelerate the training procedure. During training, if the largest gradient was larger than one, then all the gradients were rescaled so that the largest gradient was one. The batch size was 16, so the model could fit the memory limitation of the graphics processing unit. The learning rate started at 0.001, as suggested by Kingma's study **[48]**. The epoch of the training procedure was 50, and the training procedure would stop earlier if the validation loss were not decreasing for ten epochs. The DNA sequence and gene annotation would be converted to numeric vectors by one-hot-encoding.

The settings of the cross-validation of the deep learning model were similar to the hyperparameter optimization procedure. There were only four differences. The first one was that the epochs were extended to 100. The second one was that the training procedure was stopped when validation loss stopped decreasing for 20 epochs. The third one was that the batch size would decrease until it could fit the GPU memory. The fourth one was that the dropout was used with a dropout rate of 0.5, and the dropout was applied between layers of the relation block and after batch normalization layers in the feature block. The deep learning models were created based on the best hyperparameter set. For each model, the weights of the lowest validation loss were saved. The trained deep learning models were then tested on **Data$_{\textbf{Test}}$.**

In order to help deep learning model be useful in real-world scenarios, data must be augmented during training. During training, the regions would first be randomly truncated. If the region included gene had the upstream region with *u* nts, then the region would be truncated randomly from zero to *u*/10 nts from the 5’ of the region. If the region included gene had the downstream region with *d* nts, then the region would be truncated randomly from zero to *d*/10 nts from the 3’ of the region. If the origin region with length *l* did not include any gene, then the region was first truncated randomly from zero to *l*/2 from the 5’ of the region. Then the truncated region with length *r* was truncated randomly from zero to *r*/2 from the 3’ of the region. After the regions were truncated randomly, three regions would be randomly merged into one region. If there were two regions left, then they would stay the same or be randomly merged into one region. If there were one region left, then it would stay the same. **Figure 9** shows an augmentation example. During validating or testing, the origin dataset would be used.

doi:10.6342/NTU202002143

Table 1. The hyperparameter space of the model

	Hyperparameter	**Values or type**
Feature block	**Layer number of concatenated CRB blocks**	4, 8, 12, and 16
	The output channel number of the CRB block	4, 8, 12, and 16
	Kernel size of CNN	513, 1025, 1537, and 2049
Relation block	**Relation block type**	BR, BHR, HR
	Hidden size of each layer in each direction	64, 96, 128, and 160
	Layer number	1, 2, 3, and 4

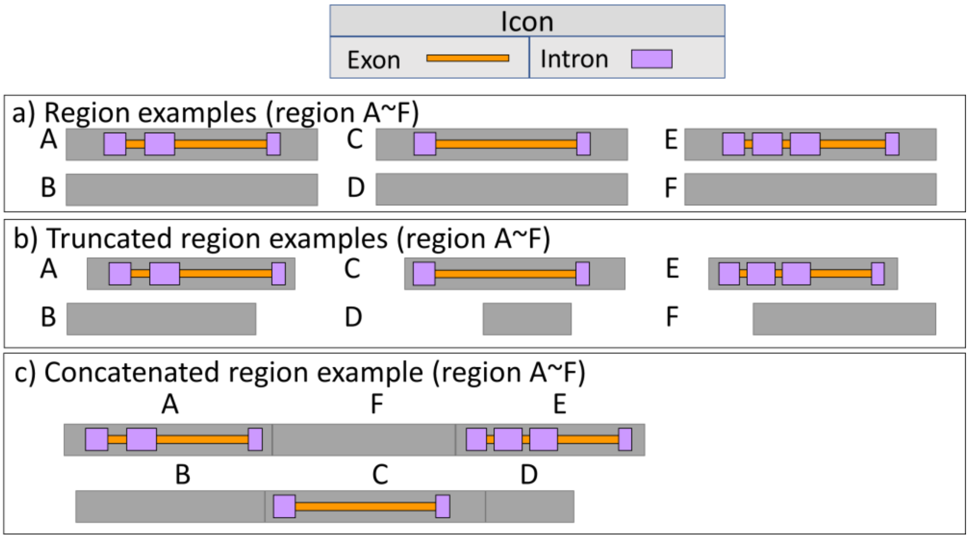

Icon
Exon
Intron
a) Region examples (region A~F)
A
C
E
B
D
F
b) Truncated region examples (region A~F)
A
C
E
B
D
F
c) Concatenated region example (region A~F)
A
F
E
B
C
D

Figure 9. Augmentation example (Notes: The direction of every region is 5’ to 3’.)

3.5 Comparison of results on the testing dataset and potential transcript regions

There were several metrics to evaluate the performance of prediction. The base-level metrics simply considered the performance on the base level. There was an F-score for each kind of label. There was a macro F-score for overall performance. These formulas are shown in **Equation 13** and **Equation 14**. The TP indicates the number of true-positive, the FP indicates the number of false-positive, the FN indicates the number of false-negative, and the N indicates the number of classes.

There also had other metrics that consider different levels like block-level and chain-block-level. The block-level F-scores considered the performance on the block level like exon block and intron block. The predicted block was considered accurate if the boundary of the predicted block was the same as the boundary of the annotated block. The chain-block-level F-score considered the performance on the chain-block levels like gene and chained introns. The building blocks of the gene were all the exons in the same gene. The building blocks of the chained introns were all the introns in the same gene. The chained blocks were considered as correct if all the blocks were correct. **Figure S2** shows examples of annotation and its performance.

There were also metrics to evaluate the performance of site prediction. **Equation 15** shows formulas about the distance of sites. The *t* indicates the target site, *s* indicates the source site, *n* indicates the number of source sites, *m* indicates the number of target sites, *p* indicates predicted site, and *a* indicates the answer site. The $\mathbf{distance}_{\boldsymbol{s,t,j}}$ is the distance between the closest source site to target sites j. If there were no source sites around the target site *j*, the distance would be assigned as NaN. The $\mathbf{distance}_{\boldsymbol{s,t}}$ is mean distance between the closest source site to target sites, and the mean method named nanmean would ignore the NaN. The $\mathbf{distance}_{\mathbf{mean}}$ is the mean distance between predicted sites and answer sites. The F-score of each kind of site could also be calculated, and it indicated how well the locations of prediction and answer were matched. **Figure S3** shows the examples of annotation at the block level, their gene boundaries, and metrics of distance and site prediction. **Table 2** shows a summary of metrics to be compared.

The number of data to be compared is too small to use parametric tests like Students' t-test **[62]**, so the Wilcoxon rank-sum test **[63]** was used. It assigned each observation a rank and got the rank sum of each group, and then it calculated the probability of a statistic called *U* to get the p-value. **Equation 16** shows the equation of the Wilcoxon rank-sum test. The r_i is the rank of the data *i*, and the *R* is the sum of r_i. The n_i is the number of data in dataset *i*. The U_i is the statistic value *U* of the dataset *i*. The smaller *U* was used to calculate the probability of observation. If the medians of the two datasets were not equal, the one-tailed test was used. Otherwise, the two-tailed was used. The test was calculated by a module named exactRankTests **[64]**.

In order to get the gene annotation on the whole genome and to reduce time consumption, the **potential transcript regions** were selected from the genome. First of all, the regions around the existed transcriptome of Araport11 on both strands were selected. Secondly, the region which has nucleotide other than A, T, C, and G, was discarded. Thirdly, the regions were merged if they were overlapped to each other. The merged region was called the **potential transcript region**. **Figure 10** shows the example of creating potential transcript regions. The models were used to predict gene annotation on potential transcript regions. The three peptide sequences of each gene were generated by transeq from EMBOSS **[65]**. The HMMER3 **[66]** and pfam_scan.pl **[67]** were used to scanning the peptide sequence to domains in Pfam-A (version 32.0) **[68]**. If any of the peptide sequences of the gene had at least one domain existed in Pfam-A, then the gene was viewed as the domain-including gene.

doi:10.6342/NTU202002143

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$
$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$
$$\mathrm{F}_1 = \frac{2 \cdot \mathrm{recall} \cdot \mathrm{precision}}{\mathrm{recall} + \mathrm{precision}}$$

Equation 13. Recall formula, precision formula, and F-score formula

$$\text{macro recall} = \frac{1}{\mathrm{N}} \sum_{i}^{\mathrm{N}} \mathrm{recall}_i$$
$$\text{macro precision} = \frac{1}{\mathrm{N}} \sum_{i}^{\mathrm{N}} \text{precision }_i$$
$$\text{macro } \mathrm{F}_1 = \frac{2 \cdot \text{macro recall} \cdot \text{macro precision}}{\text{macro recall} + \text{macro precision}}$$

Equation 14. Formulas of macro recall, macro precision, and macro F-score

$$\mathrm{distance}_{s,t,j} = \begin{cases} \min\limits_{i \in [1,n]} \left| d_{s,i} - d_{t,j} \right| & \text{if } n > 0 \\ \mathrm{NaN} & \text{if } n = 0 \end{cases}$$
$$\mathrm{distance}_{s,t} = \underset{j \in [1,m]}{\mathrm{nanmean}} (\mathrm{distance}_{s,t,j})$$
$$\mathrm{distance}_{\mathrm{mean}} = \frac{\mathrm{distance}_{p,a} + \mathrm{distance}_{a,p}}{2}$$

Equation 15. The mean distance between predicted sites and answer sites

$$R = \sum_i r_i$$
$$U_1 = n_1 n_2 + \frac{n_1(n_1 + 1)}{2} - R_1$$
$$U_2 = n_1 n_2 + \frac{n_2(n_2 + 1)}{2} - R_2$$
$$U = \min(U_1, U_2)$$

Equation 16. The Wilcoxon rank-sum test

doi:10.6342/NTU202002143

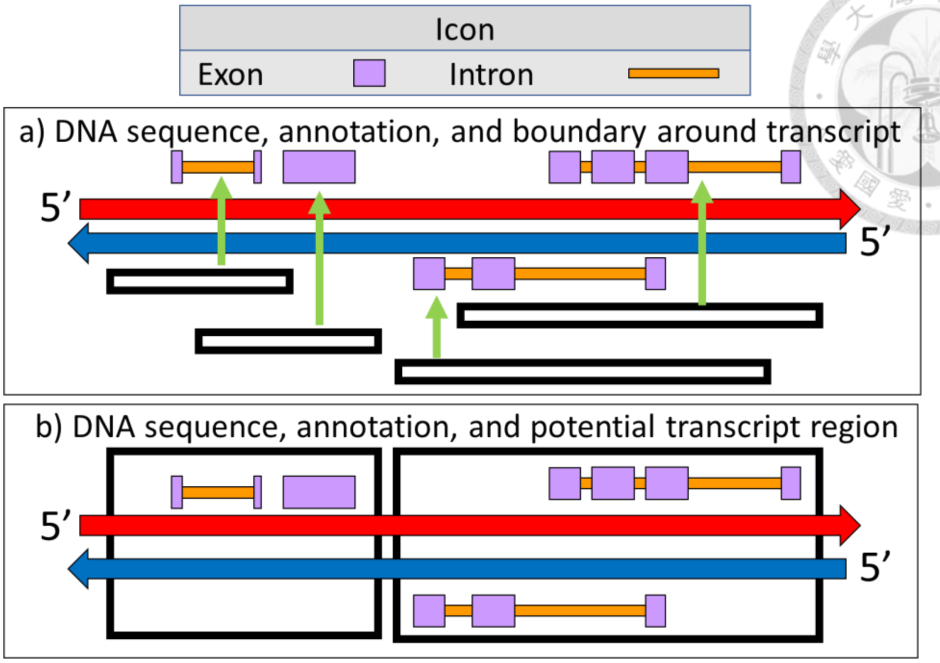

Icon
Exon
Intron
a) DNA sequence, annotation, and boundary around transcript
5'
5'
b) DNA sequence, annotation, and potential transcript region
5'
5'

Figure 10. The examples of data related to potential transcript region

Table 2. The metrics to evaluate the performance

Metric type	**Details**
F-score	Exon, intron, and intergenic region
Macro F-score	Base level
F-score of block-level	Exon block and intron block
F-score of chain-block-level	Gene and chained introns
Mean distance	TSS, CS, DS, and AS
F-score of site prediction	TSS, CS, DS, and AS

3.6 The post-processing procedure

There were two kinds of problems in the predicted annotation. The one called boundary problem was a problem that the boundary of the predicted intron could not perfectly match the true intron. The other called fragment problem was a problem that there were regions that their lengths were too short. The boundary post-processing

doi:10.6342/NTU202002143

method would relieve the boundary problem. The length filtering method would relieve the fragment problem. The post-processing procedure using these two methods was applied to the predicted result of deep learning to relieve these two problems. **Figure 11** shows the schematic diagram of revising the predicted result of the deep learning model.

Before deciding the length threshold of fragments, the distribution of regions has to be measured and fitted by the model. The log-length of the exon, intron, and gene was assumed could be fit by the Gaussian mixture model with two components. The outlier boundary for each component of the Gaussian model was set to be three standard deviations lower to the mean value. The smallest outlier boundary of the components would be the threshold to decide whether the region was fragment or not. The log-length distribution of the predicted intergenic regions was used to be fit by the Gaussian mixture model with four components. The smallest mean of four components was used to be the threshold. **Figure 12** shows the example of the log-length distribution of the intergenic region and the example of the Gaussian model and the log-length distribution of the predicted intergenic region. The blue line in **Figure 12b** indicates the summation of all models, and the other lines indicate each component of the Gaussian model.

The following describes the length filtering method. If the lengths of the exon, intron, or intergenic region were smaller than the threshold, then they would be tagged as fragments. Then, the shortest fragment was chosen to be processed. If there were multiple shortest fragments, the most upstream one was chosen to be processed. If the neighbors of the shortest fragment had the same kind of label, then the label of the shortest fragment was assigned as the label of neighbors, and they were merged into one block. Otherwise, the label of the shortest fragment was assigned as the label of longer neighbor, and they were merged into one block. The block was retagged as fragment if it were smaller than its threshold. The whole procedure was repeated until there was no

doi:10.6342/NTU202002143

fragment of exon, intron, or intergenic region left. After that, the neighboring exons and introns were grouped into the gene. If the gene were smaller than its length threshold, then it would be retagged as the intergenic region. The exons and introns on it would be retagged as part of the intergenic region.

The following described the boundary post-processing method. First of all, the locations of canonical motifs of the splicing sites must be found. For every predicted splicing site, the region around *r* nts was selected, and the location of its nearest canonical motif was recorded if it existed**.** The radius *r* nts were decided by the scaled **Distance**$_{\mathbf{a,\ p}}$. If there were two nearest canonical motifs, the most upstream one was chosen. If the canonical motif was indeed around the predicted splicing site and was inside the gene, then the splicing site was tagged as valid. If the splicing donor site and splicing acceptor site of the intron were valid, then the boundary of the intron was modified by the location of their nearest motifs. Otherwise, the intron would be discarded. If there were introns overlapped, then they were merged into one intron. **Figure 13** shows examples of boundary post-processing procedure.

The prediction results on **Val**$_{\mathbf{1}}$ to **Val**$_{\mathbf{8}}$ were merged into one dataset named **Predicted**$_{\mathbf{Val}}$. The grid search was used to find the best hyperparameters of the post-processing procedure to create the best reviser to revise **Predicted**$_{\mathbf{Val}}$. The best reviser would revise the predicted results of deep learning. **Table 3** shows the hyperparameter settings in the hyperparameter space of the post-processing procedure. **Equation 17** shows the Loss$_{revision}$ to evaluate the performances of the hyperparameter sets. The eb, ib, g, and ci in **Equation 17** indicate exon block, intron block, gene, chained introns, and. The t, c, d, and a in **Equation 17** indicate TSS, CS, DS, and AS.

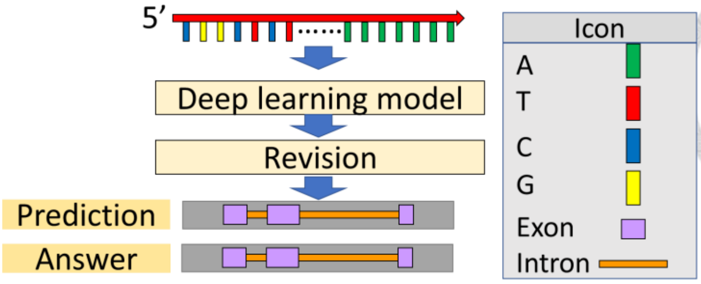


Figure 11. Prediction and revision of deep learning model

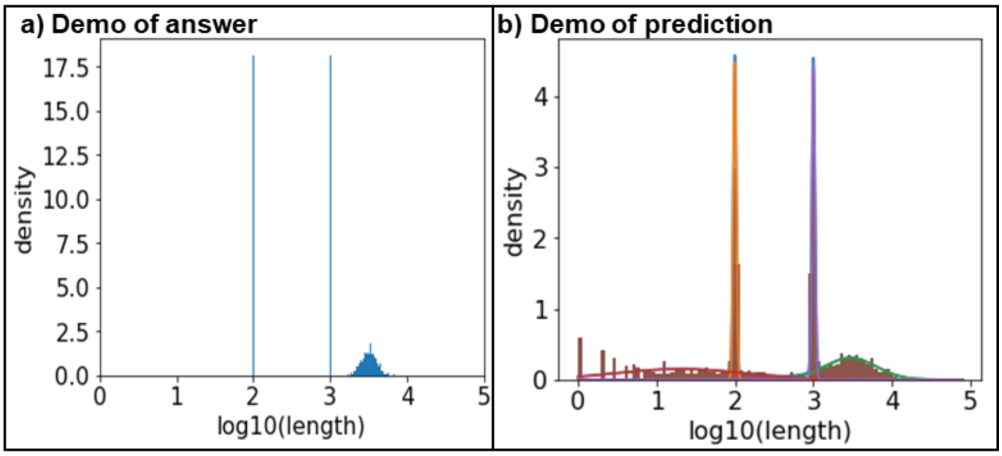


Figure 12. Examples of the log-length distribution of the intergenic region and the Gaussian model and the log-length distribution of the predicted intergenic region

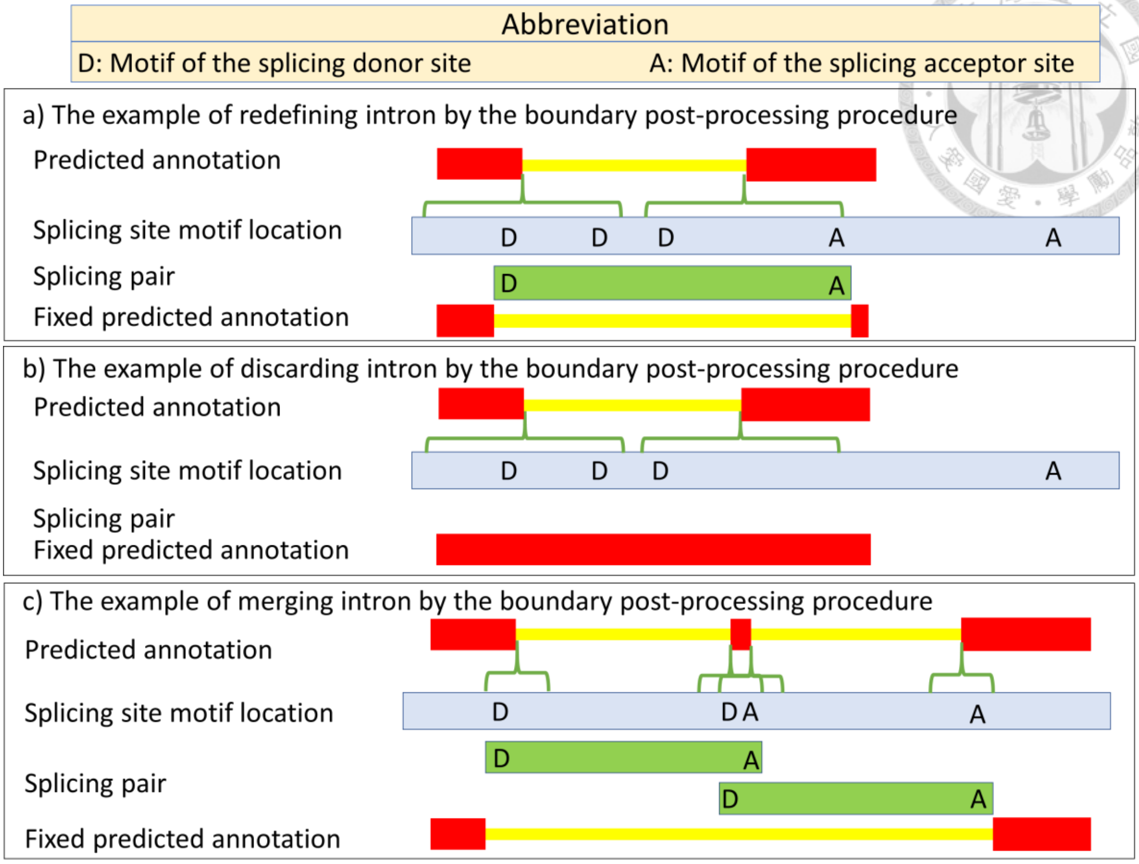

Abbreviation
D: Motif of the splicing donor site
A: Motif of the splicing acceptor site
a) The example of redefining intron by the boundary post-processing procedure
Predicted annotation
Splicing site motif location
D D D A A
Splicing pair
D A
Fixed predicted annotation
b) The example of discarding intron by the boundary post-processing procedure
Predicted annotation
Splicing site motif location
D D D A
Splicing pair
Fixed predicted annotation
c) The example of merging intron by the boundary post-processing procedure
Predicted annotation
Splicing site motif location
D DA A
Splicing pair
D A
D A
Fixed predicted annotation

Figure 13. The examples of the boundary post-processing procedure (The red block indicates exon, and yellow block indicates intron)

Table 3. Hyperparameter settings of the post-processing procedure

ID	**Procedure**	**The scale of Distance$_{a,\ p}$**
1	Length filtering method	NaN
2, 3, 4, and 5	Boundary post-processing method	0, 1, 2, and 3
6, 7, 8, and 9	Length filtering method Boundary post-processing method	0, 1, 2, and 3
10, 11, 12, and 13	Boundary post-processing method Length filtering method	0, 1, 2, and 3

doi:10.6342/NTU202002143

$$\mathrm{Score_{block}} = \sum_{k\in\{\mathrm{eb,ib,g,ci}\}} \mathrm{F}_{1,k}$$
$$\mathrm{Score_{site}} = \sum_{k\in\{\mathrm{t,c,d,a}\}} \mathrm{F}_{1,k}$$
$$\mathrm{Score_{distance}} = \sum_{k\in\{\mathrm{t,c,d,a}\}} \frac{1}{\mathrm{distance}_{\mathrm{mean},k} + 1}$$
$$\mathrm{Loss_{revision}} = 13 - \frac{\mathrm{Score_{block}} + \mathrm{Score_{site}} + \mathrm{Score_{distance}} + \mathrm{macro\ F_{1,base}}}{13}$$

Equation 17. The formula of the Loss$_{\text{revision}}$ (Notes: The macro F$_{1,\text{ base}}$ indicates the macro F-score of the base level.)

3.7 Training and testing procedure of Augustus

Augustus was used to comparing the deep learning model. **Figure 14** shows the schematic diagram of gene annotation prediction by Augustus**.** The Augustus directly used double-strand DNA to predict transcript annotation. Then the predicted transcript annotation was converted to gene annotation. The following describes the procedure of training and testing Augustus, which took the previous study **[69]** as a reference. First of all, the configuration file was created by “new_species.pl.” The parameters were updated by using the “etraining” to train on **Data$_{\text{Train}}$**. After that, the hyperparameters were updated by “optimize_augustus.pl.” The parameters were updated again by using the “etraining.” Finally, the updated model was tested on the **Data$_{\text{Test}}$**. The whole procedure was repeated eight times.

The predicted transcripts of Augustus have multiple TSSs, multiple CSs, and alternative splicing sites. The transcripts with most often TSS were preserved. If there were many TSSs, then the transcripts with most upstream TSS were preserved. Then, the transcripts with most often CS were preserved. If there still were many CSs, then the transcripts with most upstream CS were preserved. For each kind of splicing site, the

doi:10.6342/NTU202002143

most often site was selected. If there were multiple sites, then the most upstream site was selected. Then the gene annotation was derived by these transcripts.

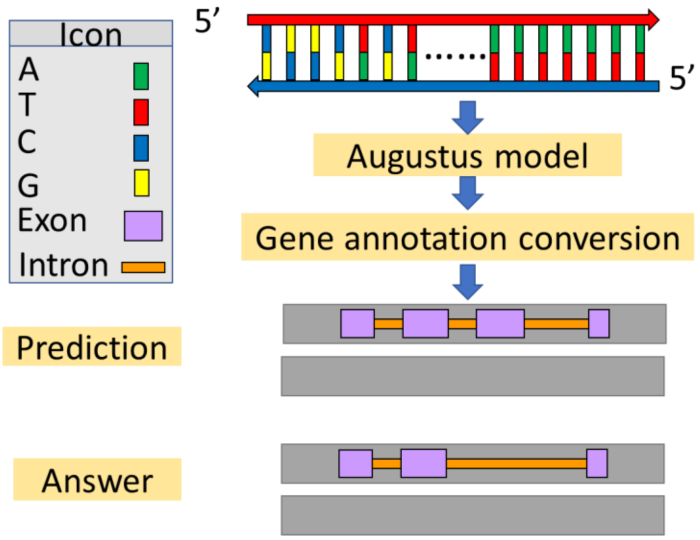


Figure 14. The schematic diagram of gene annotation prediction by Augustus

Chapter 4. Results

4.1 The different settings of the boundary around the gene

The nucleotide composition profiles of the previous study **[21]** showed that the DSEs were located within the region downstream of CS to downstream 100 nts. So, the downstream distance from CS was set to be 100 nts. The locations of TFBS, chromosome immunoprecipitation sites (ChIP sites), and DNase I hypersensitive sites (DHSs) from Yu's study were used as experimental data of the transcription **[19]**. The numbers of genes that have transcription evidence located around it are showed in **Table 4**. By analyzing Yu's data, the distributions of the most positive location difference between specific sites to each TSS of the gene are shown in **Figure 15**. **Figure 16** shows the percentage of genes supported by evidence and the number of data of different settings of upstream distance. The result with 1000-nt upstream distance had about 98% of genes that had evidence supported, and had a suitable number of regions to be used. So, the upstream distance from TSS was set to be 1000 nts.

Table 4. The number of genes which have at least one data located around it

Type	**Number**	**Type**	**Number**
TFBS	15657	**DHS**	26224
ChIP site	15021	**Any**	28412

doi:10.6342/NTU202002143

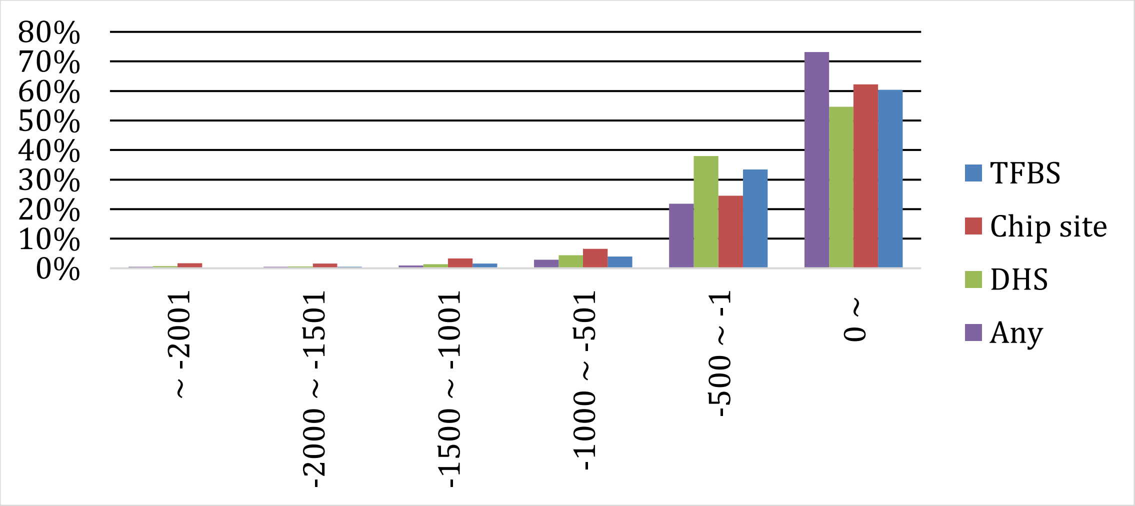


Figure 15. The percentage plot of genes which have evidence within the range

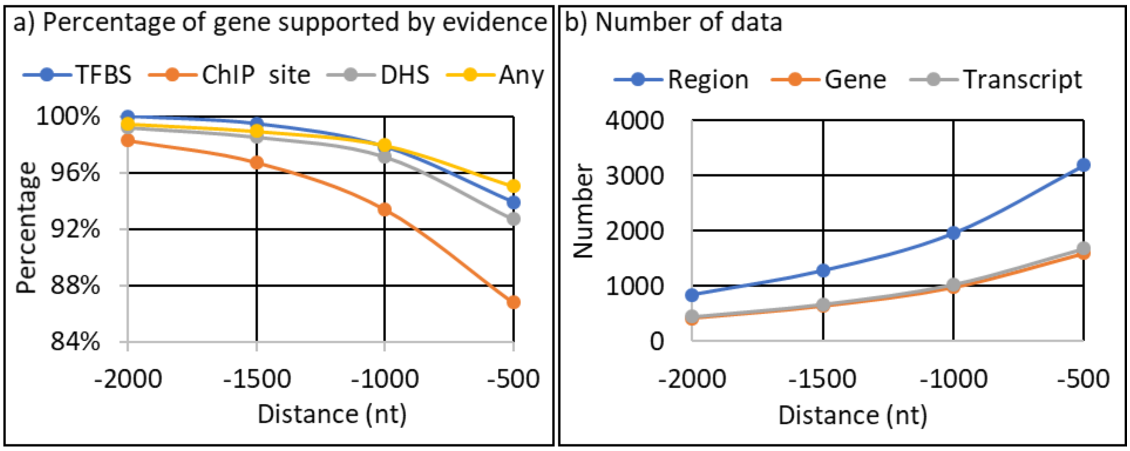


Figure 16. The plots of the percentages of genes supported by evidence and the numbers of data of different upstream distance settings

4.2 The statistic result of the experimental data and transcripts

The high-quality data was created by using the procedures described in **section 3.2** and the distance setting in **section 4.1.** The numbers of the GRO-seq signals and the PAT-seq clusters are shown in **Table 5** and **Table 6**. The numbers and ratios of the locations of the strongest evidence are shown in **Table 7** and **Table 8.** The ratios of the GRO-seq signals and the PAT-seq clusters are shown in **Table 9.** The mode location

doi:10.6342/NTU202002143

differences between locations of the reference site and the experimental site are shown in **Table 10.** The statistic result of distances and the location difference between TSSs and CSs before and after reannotated is showed in **Table 11.** The numbers about background transcripts, consistent transcripts, reannotated transcripts are shown in **Figure 17**.

Table 5. The number of GRO-seq signal

Data	Number
Raw data	14863
Data located on external 5' UTR of transcript	8870
The strongest data located on external 5' UTR of transcript	8362

Table 6. The number of PAT-seq cluster

Data	Number
Raw data	112226
Data located on external 3' UTR of transcript	45927
The strongest data located on external 3' UTR of transcript	21260

Table 7. The number and ratio of the locations of the strongest GRO-seq signal

Location	Number	Ratio
External 5' UTR of transcript	8362	68%
The region on the transcript but not on external 5' UTR	2624	21%
The region upstream of transcript	1335	11%

doi:10.6342/NTU202002143

Table 8. The number and ratio of locations of the strongest PAT-seq cluster

Location	**Number**	**Ratio**
External 3' UTR of transcript	21260	78%
The region on the transcript but not on external 3' UTR	3396	13%
The region downstream of the transcript	2457	9%

Table 9. The ratios of the GRO-seq signals and the PAT-seq clusters

	TSS evidence	**CS evidence**
The percentages of evidence located in corresponding external UTR among all evidence	59.7%	40.9%
The percentage of evidence located in corresponding external UTR is the strongest	94.3%	46.3%

Table 10. The mode location difference between reference and experimental site

	TSS	**CS**
The closest reference location to the experimental location	0	1
The closest experimental location to the reference location	0	-1

Table 11. The statistic result of distances between the reference and redefined site

	Min	**Median**	**Mode**
TSS	0	81	0
CS	0	83	0

doi:10.6342/NTU202002143

4.3 The statistic result of transcripts and regions after filtering and cleaning

The summary of the numbers of the transcripts is shown in **Figure 17**. The Venn diagram of the transcripts passed filters is shown in **Figure S4**. **Table S6** shows the statistic result of each kind of region. The percentages of each label are displayed in **Figure 18**. The number of regions on each dataset is shown in **Table S5**. **Figure 19** shows the nucleotide composition and motif around each kind of site. The zero in **Figure 19** indicates the location after TSS, location before the CS, location after DS, and location before the AS. **Table 12** shows the numbers of the splicing site motifs and their percentage in the whole dataset. **Table S7** shows the motifs and their percentages of the splicing site in Data$_{\text{Train}}$.

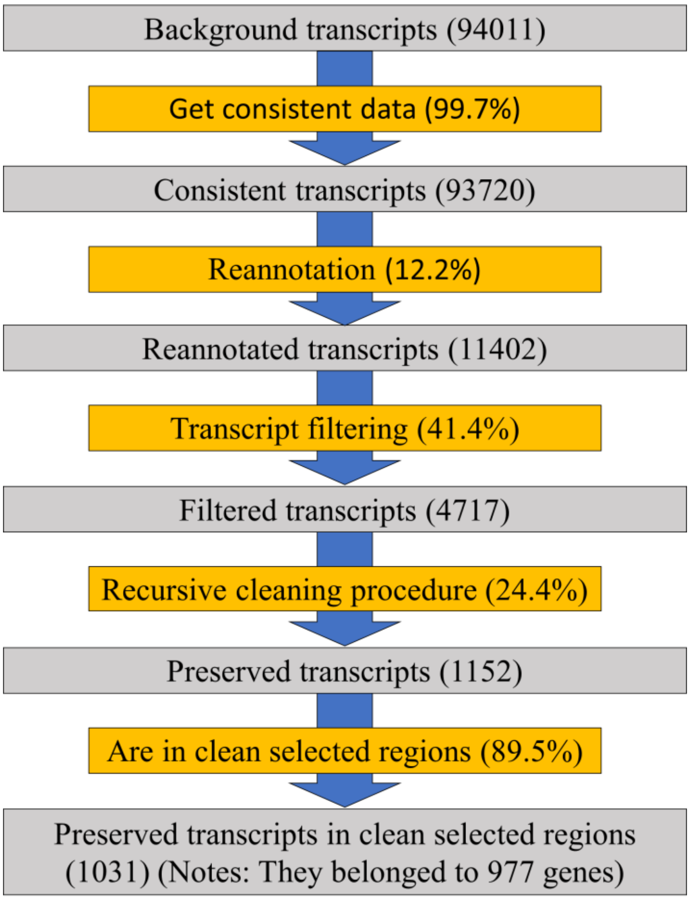

Background transcripts (94011)
Get consistent data (99.7%)
Consistent transcripts (93720)
Reannotation (12.2%)
Reannotated transcripts (11402)
Transcript filtering (41.4%)
Filtered transcripts (4717)
Recursive cleaning procedure (24.4%)
Preserved transcripts (1152)
Are in clean selected regions (89.5%)
Preserved transcripts in clean selected regions (1031) (Notes: They belonged to 977 genes)

Figure 17. The numbers and percentages of transcripts after each preprocessing

doi:10.6342/NTU202002143

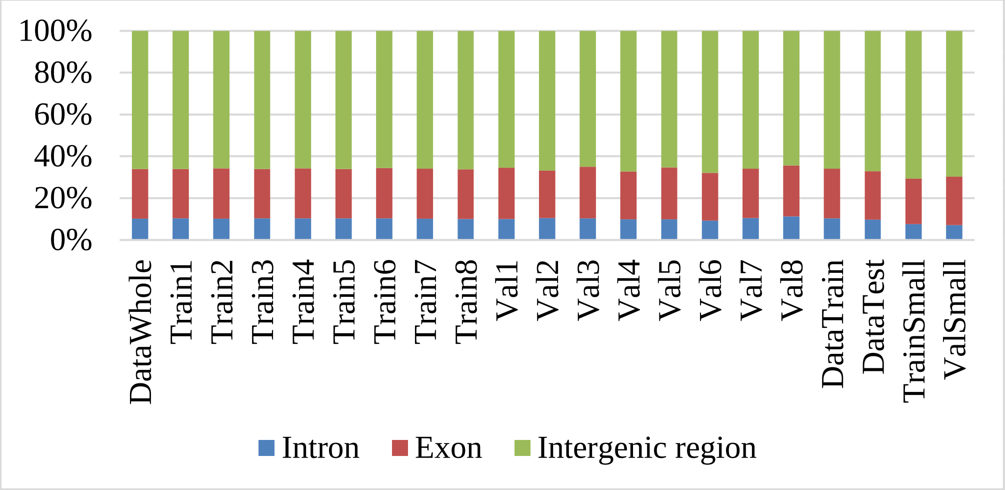


Figure 18. The ratios of each kind of label in different datasets

Table 12. The data of the splicing site motifs in gene annotation

Splicing donor site			**Splicing acceptor site**		
Motif	**Count**	**Percentage (%)**	**Motif**	**Count**	**Percentage (%)**
GT	4019	99.235	**AG**	4050	100
GC	30	0.741			
TT	1	0.025			

doi:10.6342/NTU202002143

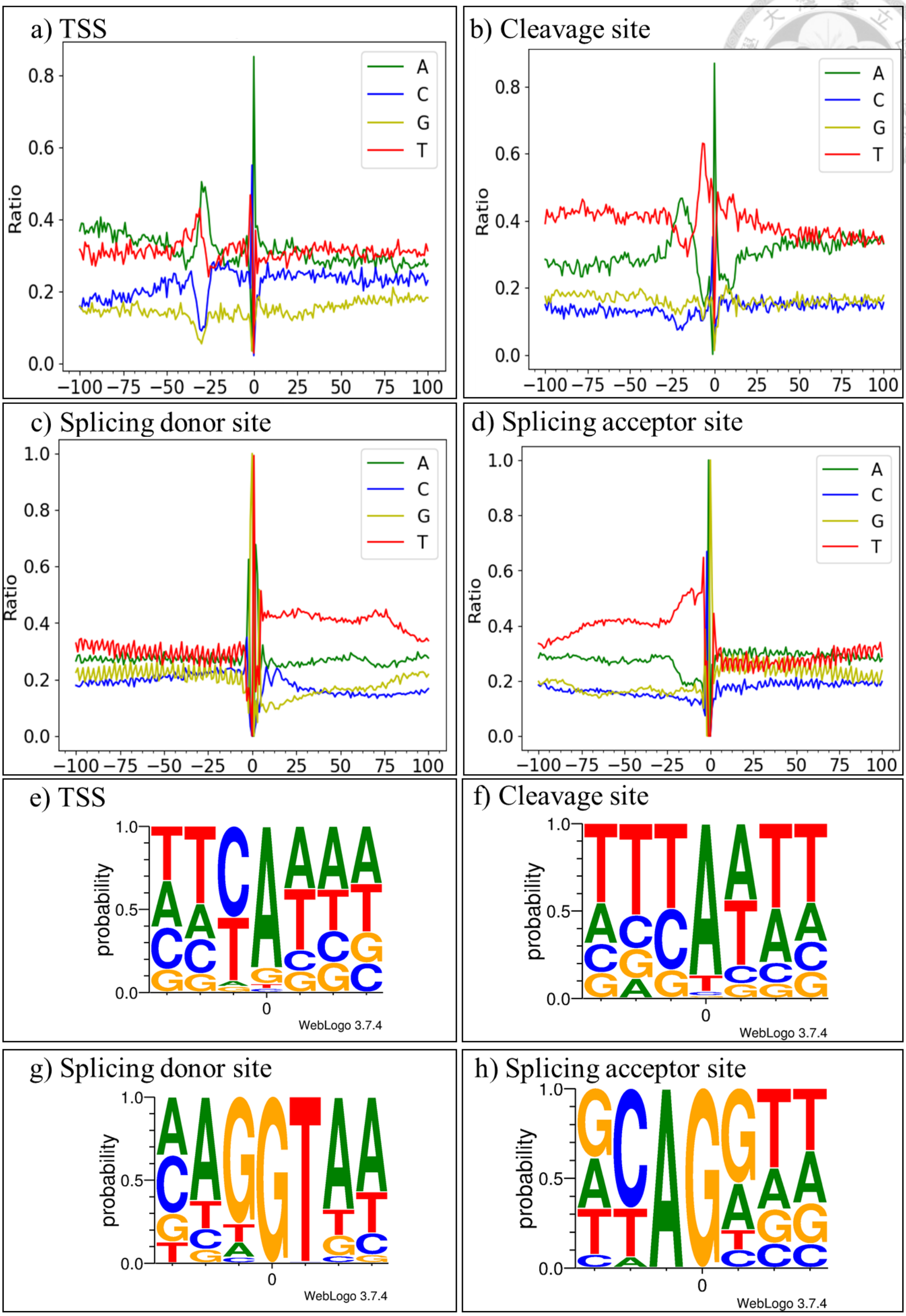


Figure 19. The nucleotide composition (a~d) and motif (e~h) of each site in gene annotation (Notes: The motifs are generated by WebLogo3 [70].)

doi:10.6342/NTU202002143

4.4 Hyperparameter searching result on the small dataset

The hyperparameter sets were created based on the method described in **section 3.4**. The first 40 trials were created based on the random search, and the left trials were created based on the Bayesian optimization. The total number of trials was 53. The Bayesian optimization had executed 13 times. The optimization was early stopped because there was no improvement in these five trials after the **trial$_{47}$**. **Trial$_{47}$** had the best macro F-score, which was 0.95. **Table 13** shows the base performance of **trial$_{47}$**. **Table 14** shows the settings with the highest count in the top ten trials. **Table 15** shows the best top ten trials of 53 trials.

Table 13. The base performance of the trial$_{47}$

Validation metric	**Exon F-score**	**Intron F-score**	**Intergenic region F-score**	**Macro F-score**
Value	0.94192	0.9338	0.98299	0.95343

Table 14. The settings with the highest count in the top ten trials

Block type	**Hyperparameter**	**Setting with the highest count**
Feature block	**Layer number of concatenated CRB blocks**	12
	The output channel number of the CRB block	4
	Kernel size of CNN	2049
Relation block	**Relation block type**	HR
	Hidden size of each layer in each direction	160
	Layer number	4

doi:10.6342/NTU202002143

Table 15. Top 10 Macro F-score and their hyperparameters in decreasing order

	Feature block			Relation block			
Id	**Layer number**	**Output number of each layer**	**Kernel size**	**Relation type**	**Hidden number**	**Layer number**	**Macro F-score**
47	16	4	2049	HR	160	4	0.95343
43	12	4	2049	HR	64	4	0.95014
1	12	4	2049	HR	160	3	0.94427
18	8	4	1025	BHR	160	4	0.94379
15	12	4	1025	HR	96	2	0.94210
24	12	4	2049	BR	64	3	0.93873
45	16	4	2049	BHR	160	4	0.93653
46	16	4	2049	BHR	160	4	0.93312
52	4	4	2049	HR	64	4	0.93191
49	4	4	2049	BHR	160	4	0.93105

4.5 Result comparison of deep learning model and Augustus on the testing dataset

Based on the setting of **trial$_{47}$**, the eight-fold cross-validation was executed, as depicted in **section 3.4**. The batch size was slightly decreased from 16 to 14 to fit the memory of the GPU. The eight Augustus models were trained based on the method in **section 3.7**. **Figure 20** shows the comparison plots of performances between Augustus and the deep learning model on the **Data$_{\text{Test}}$**. Wilcoxon rank-sum test was used to test their significance. If its p-value is less than 0.05, then it was considered as statistically significant, and it has a superscript with one star (*). The performances between Augustus and the deep learning model are shown in **Table 16**. The tick ✓ in the table,

doi:10.6342/NTU202002143

means the result of the deep learning model was significantly better than the result of Augustus. The triangle ▲ in the table, means the results have no statistical significance. The cross ✕ in the table, means the result of the deep learning model was significantly worse than the result of Augustus. **Table 17** shows the statistic result of Distance$_{a, p}$ on **Predicted$_{\text{Val}}$.** **Figure 21** and **Figure 22** show the distributions and the Gaussian models of log-lengths of exons, introns, genes, and intergenic regions in **Data$_{\text{Train}}$** and **Predicted$_{\text{Val}}$.**

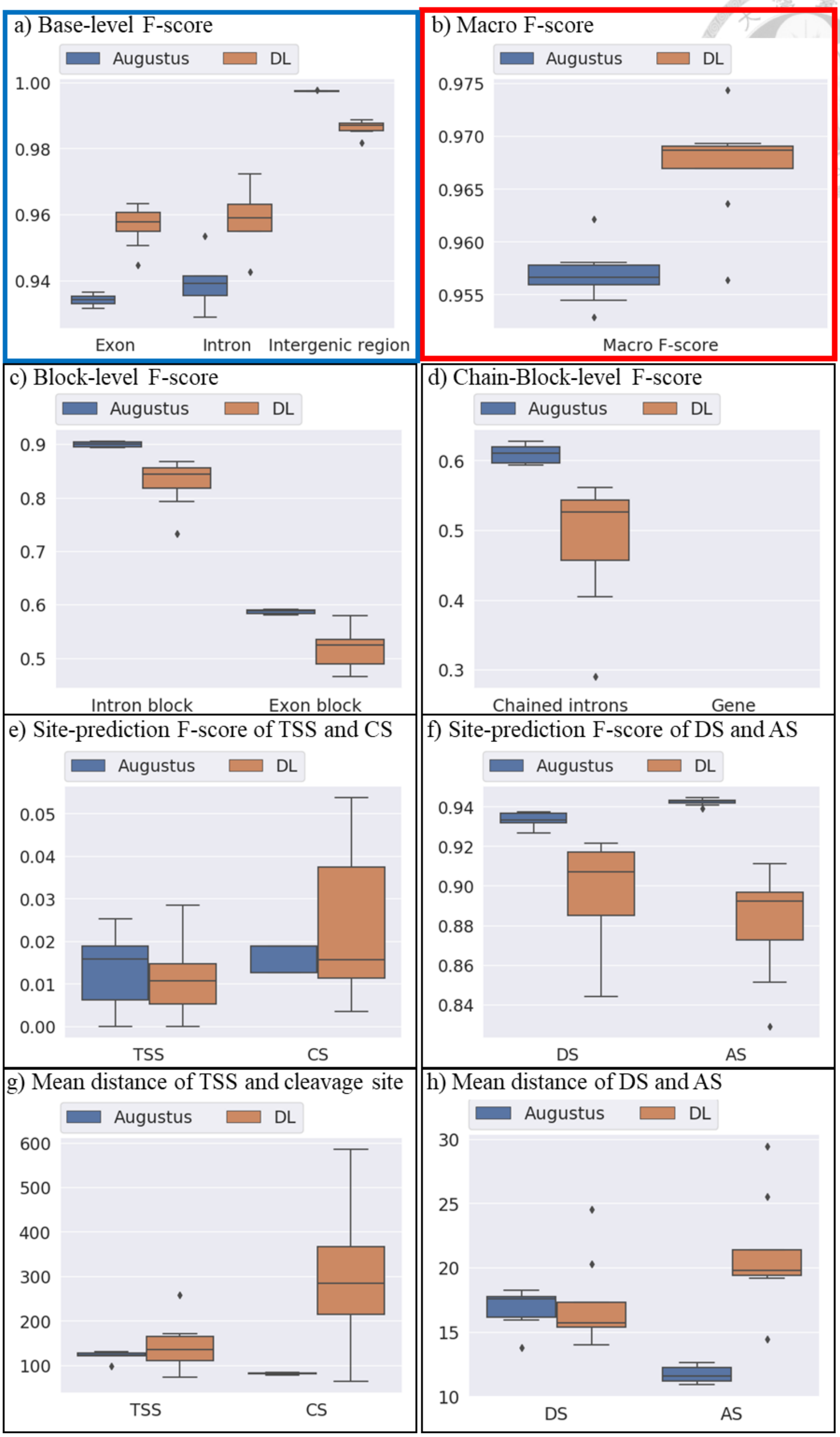


Figure 20. The comparison plots between Augustus and deep learning (DL)

Table 16. The performances between Augustus and the DL on the testing dataset

Median	Augustus	DL	Compare	P-value	Status
Exon F-score	0.934	0.958	Greater	0.00008*	✓
Intron F-score	0.939	0.959	Greater	0.00016*	✓
Intergenic region F-score	0.997	0.987	Less	0.00008*	✗
Macro F-score	0.957	0.969	Greater	0.00233*	✓
Intron block F-score	0.900	0.844	Less	0.00008*	✗
Exon block F-score	0.588	0.524	Less	0.00008*	✗
Chained introns F-score	0.611	0.526	Less	0.00008*	✗
Gene F-score	0.000	0.000	Equal	1.00000	▲
TSS F-score	0.016	0.011	Less	0.29417	▲
CS F-score	0.019	0.016	Less	0.42782	▲
DS F-score	0.933	0.907	Less	0.00008*	✗
AS F-score	0.943	0.892	Less	0.00008*	✗
Mean distance of TSS	123.4	135.3	Greater	0.25268	▲
Mean distance of CS	81.9	283.7	Greater	0.00521*	✗
Mean distance of DS	17.6	15.7	Less	0.22090	▲
Mean distance of AS	11.6	19.8	Greater	0.00008*	✗

Table 17. The Distance$_{a, p}$ about the splicing sites on the Predicted$_{Val}$

Distance$_{a, p}$	Splicing donor site	Splicing acceptor site
Value	21.3	23.4

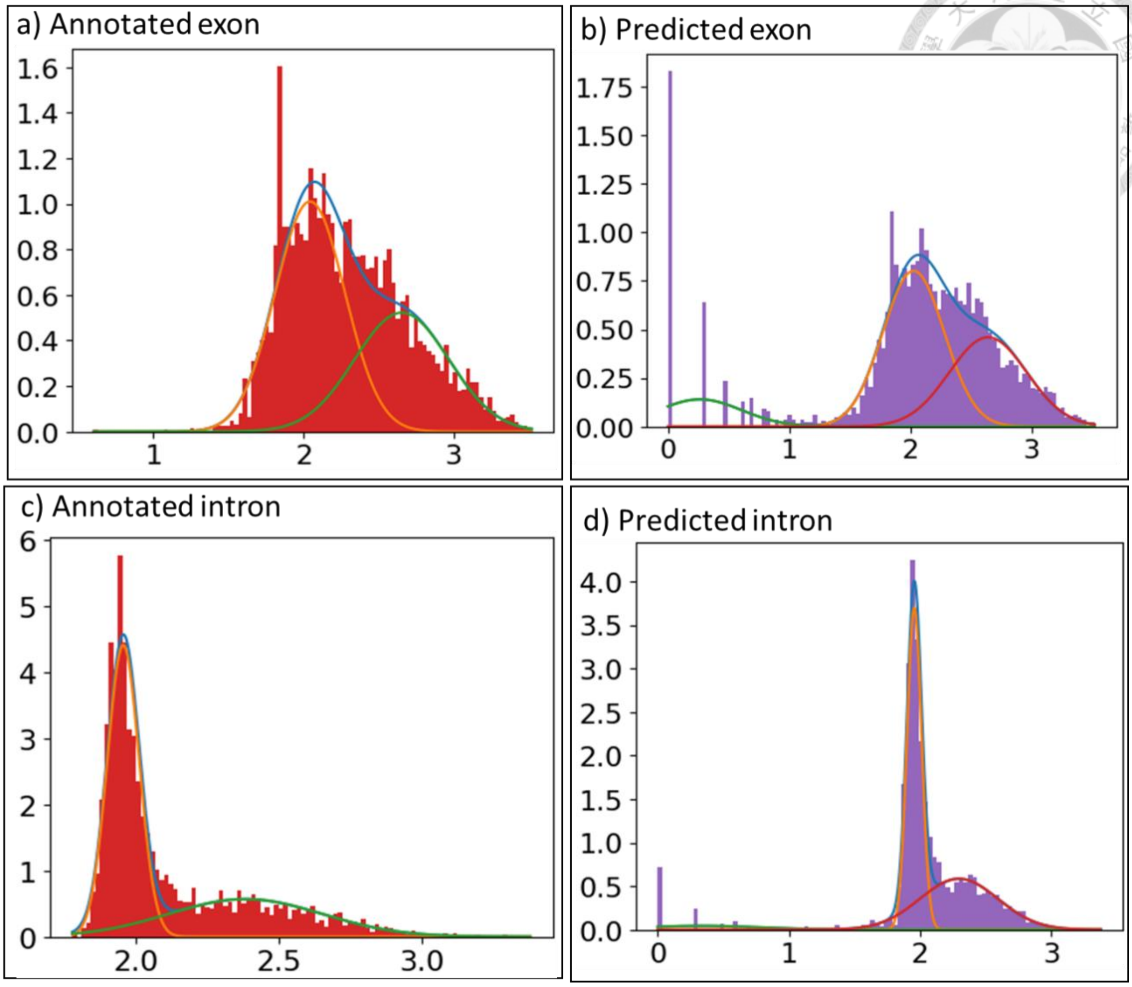


Figure 21.The distribution and the Gaussian model of the log-length (nt) of the exon and intron on the Data$_{\text{Train}}$ and Predicted$_{\text{Val}}$ (x-axis: log 10 of length, y-axis: density) (Notes: The blue line in the figures means the summation of all models, and the others mean components of the Gaussian model.)

doi:10.6342/NTU202002143

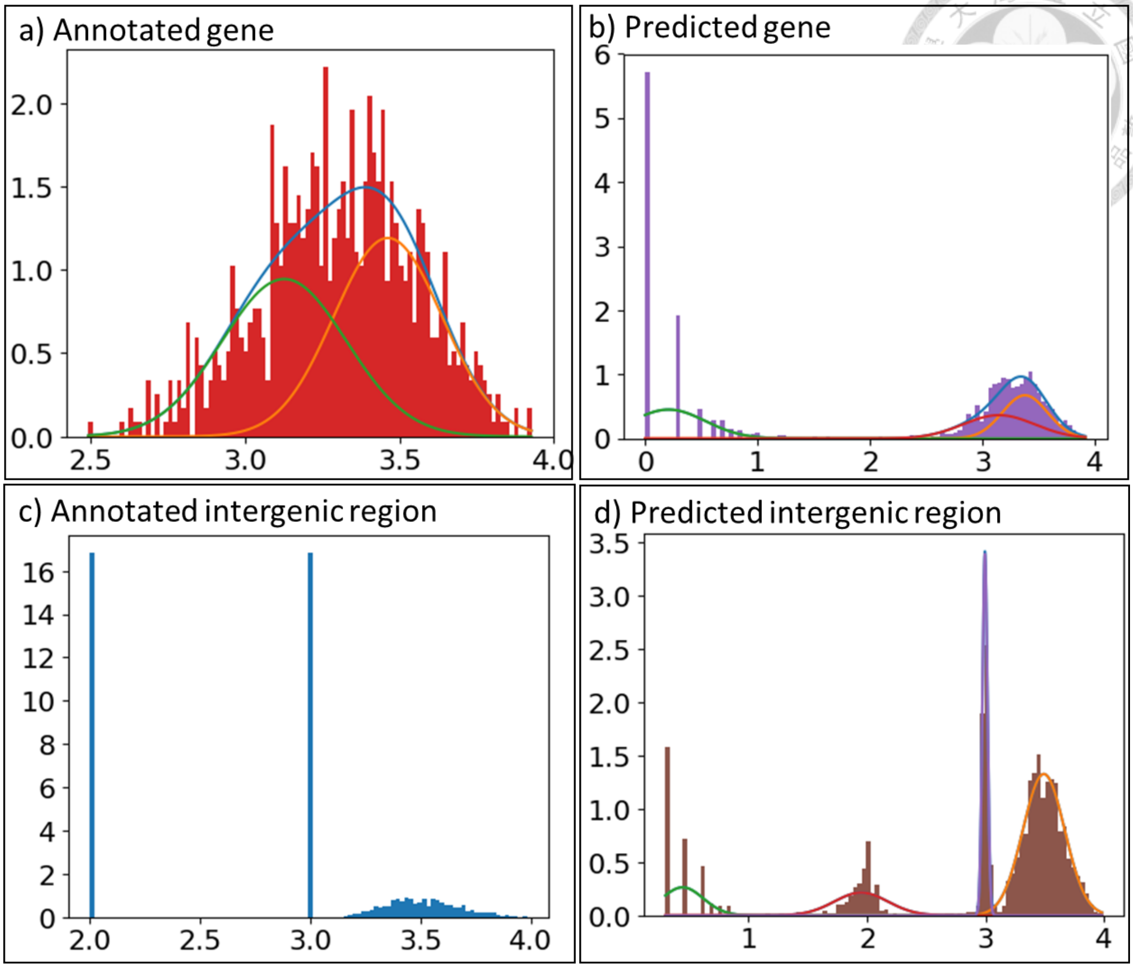


Figure 22. The distribution and the Gaussian model of the log-length (nt) of the gene and intergenic region on the Data$_{\text{Train}}$ and Predicted$_{\text{Val}}$ (x-axis: log 10 of length, y-axis: density) (Notes: The blue line in the figures means the summation of all models, and the others mean components of the Gaussian model.)

4.6 The revised result of deep learning model on the testing dataset

The post-processing procedure introduced in **section 3.6** was applied to Predicted$_{\text{Val}}$. **Table S8** shows the hyperparameters of revision settings and their Loss$_{\text{revision}}$ on Predicted$_{\text{Val}}$. The Reviser$_8$ and Reviser$_9$ had the lowest values, and the values are 0.502. The Reviser$_8$ was chosen to be the best reviser to revise the predicted result of deep learning. **Table 18** shows the details of the Reviser$_8$. The Reviser$_8$ was applied to the predicted results on $\text{Predicted}_{\text{Val}}$. **Figure 23** shows the comparison plots of performances between the origin result and the revised result of the deep learning model. The performances between the origin result and the revised result of the deep learning model are shown in **Table 19**. The intergenic region F-score in origin and revised, which estimated at the fifth decimal places, are 0.98705 and 0.98714. The F-score of the CS prediction in origin and revised, which estimated at the fifth decimal places, are 0.01566 and 0.01563.

Table 18. The detail of the Reviser_{8}

Method (in order)	**Type**	**Value**
Length filtering method	**Exon length**	22.2
	Intron length	37.9
	Intergenic region length	2.8
	Gene length	333.1
Boundary post-processing method	**DS distance**	42.7
	AS distance	46.8

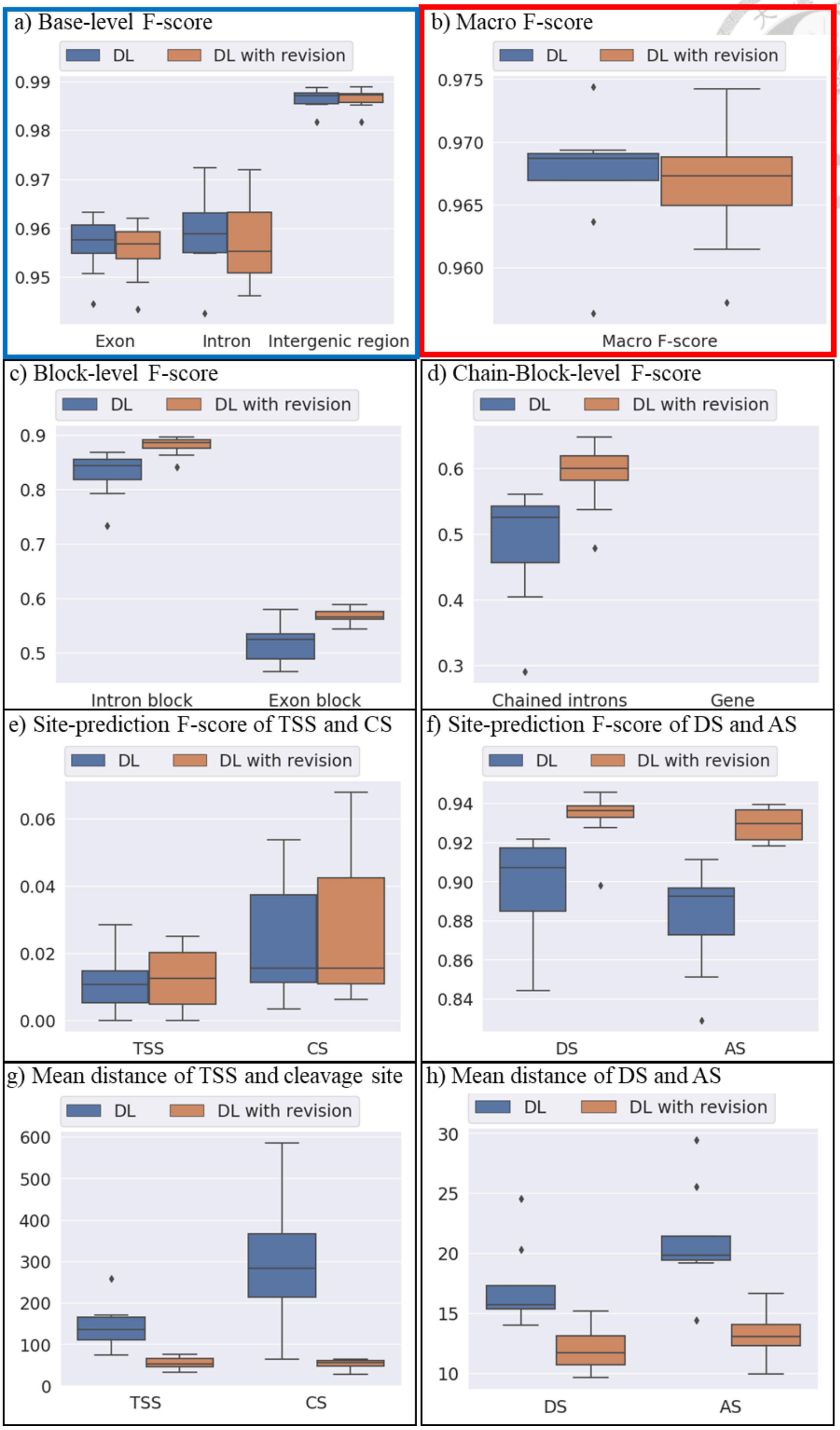


Figure 23. The comparison plots between origin result and the revised result of the deep learning model

doi:10.6342/NTU202002143

Table 19. The performances between origin result and the revised result of the deep learning model

Median	Origin	Revised	Status	P-value	Status
Exon F-score	0.958	0.957	Less	0.32269	▲
Intron F-score	0.959	0.955	Less	0.32269	▲
Intergenic region F-score	0.987	0.987	Greater	0.56076	▲
Macro F-score	0.969	0.967	Less	0.25268	▲
Intron block F-score	0.844	0.886	Greater	0.00148*	✓
Exon block F-score	0.524	0.566	Greater	0.00521*	✓
Chained introns F-score	0.526	0.600	Greater	0.00521*	✓
Gene F-score	0.000	0.000	Equal	1.00000	▲
TSS F-score	0.011	0.013	Greater	0.34957	▲
CS F-score	0.016	0.016	Less	0.62922	▲
DS F-score	0.907	0.936	Greater	0.00148*	✓
AS F-score	0.892	0.930	Greater	0.00008*	✓
Mean distance of TSS	135.3	52.6	Less	0.00016*	✓
Mean distance of CS	283.7	55.8	Less	0.00016*	✓
Mean distance of DS	15.7	11.7	Less	0.00031*	✓
Mean distance of AS	19.8	13.1	Less	0.00016*	✓

doi:10.6342/NTU202002143

4.7 Comparison of the revised result of deep learning model and the result of Augustus on the testing dataset and potential transcript regions

Figure 24 shows the comparison plot of performances between the result of Augustus and the revised result of the deep learning model on **Data$_{\mathbf{Test}}$**. The performances between the result of Augustus and the revised result of the deep learning model on **Data$_{\mathbf{Test}}$** are shown in **Table 20**.

For each transcript in Araport11, a DNA region was selected from upstream 1000 nts of its TSS to downstream 100 nts of its CS. Only the DNA region with nucleotide A, T, C, and G was preserved, and the preserved regions were merged as potential transcript regions, as describes in **section 3.5**. The gene structure was created from transcripts in potential transcript regions. For each gene, three peptides derived from it were search against domains in Pfam-A, and the gene was viewed as a domain-support gene if any peptide included any domain. The genes derived from transcripts of Araport11, Augustus, and revised prediction of deep learning model on these regions were extracted and were evaluated by the above methods. **Figure 25, Figure 26**, and **Figure 27** show the boxplot of the results. **Table 21** shows the average number of genes that match any domain in Pfam-A.

doi:10.6342/NTU202002143

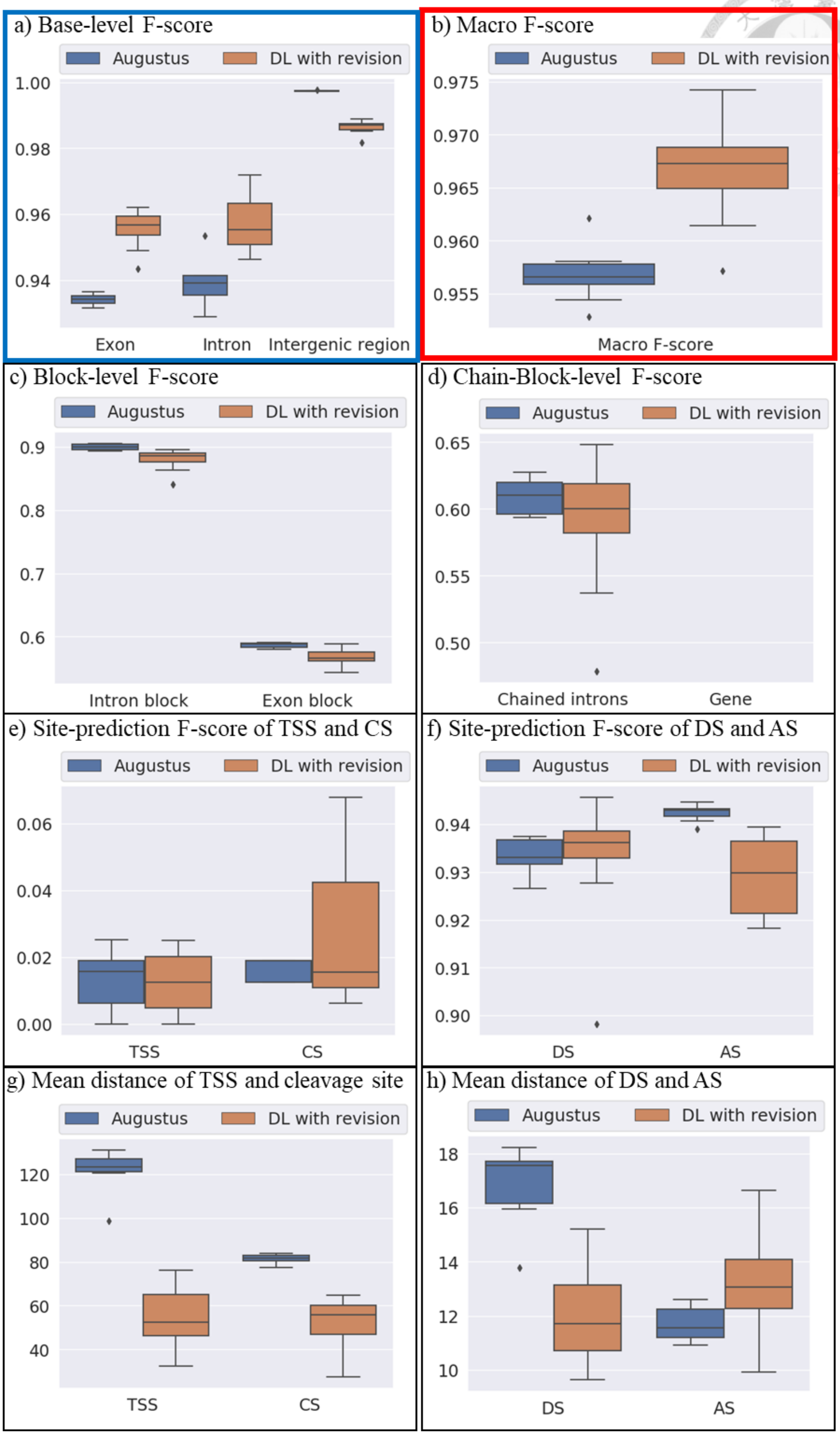


Figure 24. The comparison plots between Augustus and DL with revision

doi:10.6342/NTU202002143

Table 20. The performances between the result of Augustus and the revised result of deep learning model on the testing dataset

Median	Augustus	DL with revision	Status	P-value	Status
Exon F-score	0.934	0.957	Greater	0.00008*	✓
Intron F-score	0.939	0.955	Greater	0.00093*	✓
Intergenic region F-score	0.997	0.987	Less	0.00008*	✗
Macro F-score	0.957	0.967	Greater	0.00093*	✓
Intron block F-score	0.900	0.886	Less	0.00093*	✗
Exon block F-score	0.588	0.566	Less	0.00140*	✗
Chained introns F-score	0.611	0.600	Less	0.36962	▲
Gene F-score	0.000	0.000	Equal	1.00000	▲
TSS F-score	0.016	0.013	Less	0.34934	▲
CS F-score	0.019	0.016	Less	0.31057	▲
DS F-score	0.933	0.936	Greater	0.24382	▲
AS F-score	0.943	0.930	Less	0.00016*	✗
Mean distance of TSS	123.4	52.6	Less	0.00008*	✓
Mean distance of CS	81.9	55.8	Less	0.00008*	✓
Mean distance of DS	17.6	11.7	Less	0.00016*	✓
Mean distance of AS	11.6	13.1	Greater	0.02494*	✗

doi:10.6342/NTU202002143

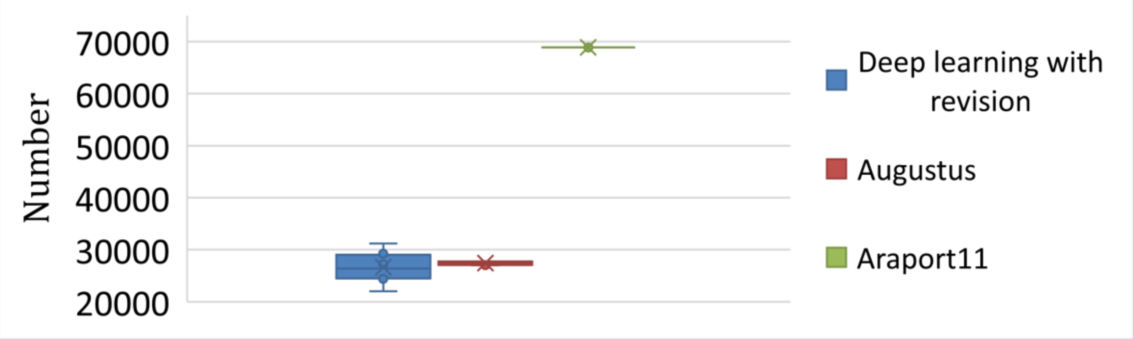


Figure 25. The boxplot of the number of genes

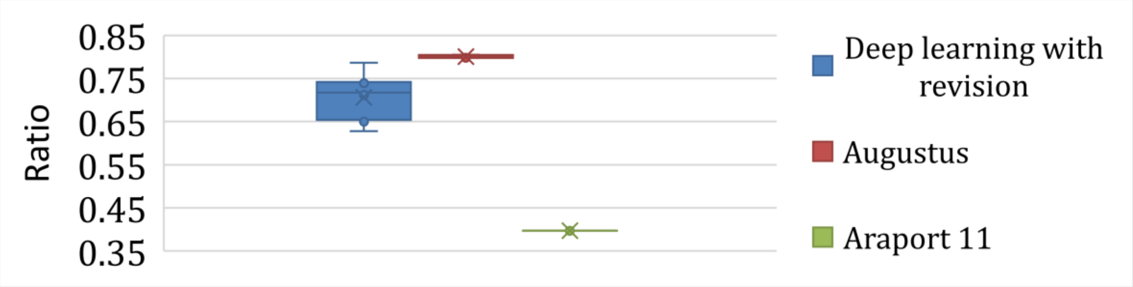


Figure 26. The boxplot of the ratio of genes that match any domain in Pfam-A

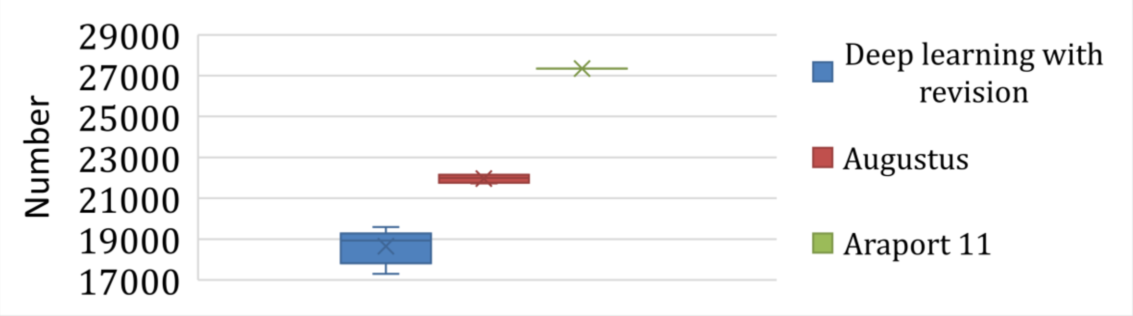


Figure 27. The boxplot of the number of genes that match any domain in Pfam-A

Table 21. The average number of genes that match any domain in Pfam-A

The deep learning model with revision	Augustus	Araport 11
18642.1	21961.3	27343

doi:10.6342/NTU202002143

Chapter 5. Discussion

5.1 Different kinds of evidence can affect the percentage of the genes that have evidence supported

Figure 16 shows that the percentages of the TFBS are higher than the percentages of the DHS and the ChIP site. The reason behind this might be that the TFBS was predicted by searching its similarity to existed motifs, and the DHS and the ChIP site were from the experimental data. The experiment might lose some locations related to transcription. The percentages of DHSs are higher than the percentages of the ChIP sites, and this may be caused by the experimental property they had. DHSs are the location that lacks chromatin on DNA sequence, so the DNase I can hydrolyze that parts of DNA. It is less specific than the result of the ChIP site because the ChIP site is from the ChIP-seq, which captures transcription factors (TFs) by using the TF-specific antibody. So, the DHS is expected to be less specific than the ChIP site. The percentage of the gene which has any evidence in it is slightly lower than the percentage of TFBS at settings of 2000-nucleotide and 1500-nucleotide upstream distance, as depicted in **Figure 16**. It is because only the DNA region from -2000 nts to +200 nts was used to search TFBSs, but the experimental data did not have this restrain **[19]**. The same reason causes the problem in the setting of 1500-nucleotide upstream distance.

5.2 The transcripts have transcription-related evidence around their TSSs

The previous study showed that 63% of predicted TFBSs were located -400 nts to +200 nts and that 86% of predicted TFBSs were located -1000 nts to +200 nts **[19]**. It showed most of the TFBSs located within the region around TSS. **Figure 15** shows that most of the genes have at least one evidence of transcription located directly on the gene body. **Figure 16** shows the percentage of genes that have at least one evidence located

doi:10.6342/NTU202002143

within the region with specific upstream distances. The percentages in **Figure 16** are pretty high because the gene had averagely 5.4 TFBSs **[19]**, so one gene likely has at least one TFBS in the setting of 1000-nt upstream distance. The previous study **[71]** showed that some elements located on the gene (especially on introns) could indeed affect transcription initialization. The result in **Figure 15** agrees with the previous study that elements downstream of TSS might relate to transcription.

5.3 Different upstream distances can have a massive impact on the number of data and the percentage of genes supported by transcription evidence

As shown in **Figure 16,** the settings of the upstream distance have a massive impact on the number of data. The result with 500-nt upstream distance has the most significant number of data, but the percentage of genes that have transcription evidence supported is the lowest. The result with 2000-nt upstream distance has the most significant percentage of genes that have transcription evidence supported, but the number of data is the lowest. The larger upstream distance is used, the fewer data are included. This situation is the opposite of the percentage of genes that have transcription evidence supported. The larger upstream distance is used, the larger percentage of genes have transcription related evidence supported. Because the larger upstream distance is, there are more chances of covering the low confidence transcript, and the region covers the low confidence transcript will be discarded.

5.4 Most locations of evidence are near external UTRs

About 60% of GRO-seq signal locations are located in external 5' UTR of the consistent transcripts, and 41% of PAT-seq cluster locations are located in external 3' UTR of the consistent transcripts, as shown in **Table 9**. There are many GRO-seq signals and PAT-seq clusters located outside the existed transcript. There may be three reasons behind this. The first possible reason is that these signals are not filtered by the threshold so that they might be noise generated by background. The second possible reason is that many transcripts have not been annotated. The low expression or specific expression situation of these transcripts might be the reason that they are not annotated. The third possible reason is that some transcript annotation lacks UTR or its UTRs are not annotated correctly, so the evidence cannot locate on their external UTRs correctly. **Table 7** shows that most of the strongest GRO-seq signals are located on external 5’ UTR, it indicates that the GRO-seq signals can indeed capture location around the TSS. **Table 8** shows that most of the strongest PAT-seq clusters are located on external 3’ UTR, it shows that the PAT-seq clusters can indeed capture location around the CS. **Table 9** shows that 94% of GRO-seq signals located on external 5’ UTR of the consistent transcripts are the strongest signal and that 46% of PAT-seq signals located on external 3’ UTR of the consistent transcripts are the strongest signal. The percentage of GRO-seq signals is twice as large as the percentage of PAT-seq signals. It seems that the 5’ UTR that has GRO-seq signals will likely have about one GRO-seq signals on it and that 3’ UTR that has PAT-seq clusters will likely have about two PAT-seq clusters located on it.

5.5 The boundary of the reannotated transcript is close to the existed boundary

Table 10 shows that the mode location differences between the reference TSSs and experimental TSSs are zero nts, and this indicates that the TSS evidence can match the locations of existed TSSs. **Table 11** shows that the mode location difference of TSSs before and after redefined is zero nts, and this indicates that the location of the strongest experimental TSSs has no bias to reference. **Table 10** shows the most often locations of the experimental CSs are 1-nt upstream from reference CSs, and this indicates that the location of experimental CSs has a bias to reference CSs. **Table 11** shows that the mode location difference of CSs before redefined and after redefined is zero nucleotides. There might be a reason behind the difference. The experimental CSs are locations defined by PAT-seq clusters, but the redefined CSs are locations defined by the strongest PAT-seq cluster in the transcript. So, the PAT-seq clusters are likely to be 1-nt upstream from reference CSs, but the strongest PAT-seq clusters have no bias to reference CSs and that locations of the experimental CSs agree with locations of the reference CSs.

5.6 The nucleotide compositions around different kinds of sites agree with the previous studies

The previous study that used full-length cDNA to get nucleotide compositions on *Arabidopsis* showed that 60% of nucleotides at TSS were nucleotide A and that there were peaks of nucleotide T and a peak of nucleotide C before the TSS, and it also showed AT-rich region was upstream 30 nts from TSS **[72]**. The previous study which used the GRO-seq signal to define TSS showed that the nucleotide A was the most dominant nucleotide on the TSS, the nucleotide T or nucleotide C was upstream from TSS, and AT-rich region was upstream 30 nts from TSS **[57]**. **Figure 19a** and **Figure 19e** also shows similar results to these previous studies **[57, 72]**. The previous studies also showed that there was a peak of nucleotide A, followed by a peak of nucleotide T, and then followed by a peak of nucleotide A around the CS, and they also showed the T-rich region was downstream 20~100 nts of the CS **[21, 72, 73]**. The previous result studied element around the CS showed the last nucleotide before the CS is dominated by nucleotide A **[73]**. **Figure 19b** and **Figure 19f** also shows similar results to the previous studies **[21, 72, 73]**. As shown in **Table 12**, 99% of DS motifs in gene annotation are GT, and 100% of AS motifs in gene annotation are AG. The previous result also showed that the motif of canonical DS was GT, and the motif of the canonical AS was AG **[72]**, So the canonical motif in **Table 12** agrees with the previous result **[72]**.

5.7 There is a tradeoff between the quality of annotation and the number of transcripts, and the number of high-quality data is rare

The numbers of transcripts are shown in **Figure 17**. It shows that 99.7% of background transcripts are consistent, and it means the data is reliable. Only 12.2% of consistent transcripts can be reannotated by the evidence. It is the most significant bottleneck of data cleaning, and it shows only little data have evidence supported on their boundary. About 41.4% of reannotated transcripts can pass five transcript filters. Only 24.4% of filtered transcripts have no discarded transcripts nearby and can be preserved, and it is the second significant bottleneck of data cleaning. 89.5% of transcripts are in clean selected regions. The number of the consistent transcripts is 93720, and the number of transcripts in the clean selected regions is 1031. It shows that there are only about 1% of the consistent transcript left in the data. There is a tradeoff between the quality of annotation and the number of transcripts. Although the data is clean, the number of transcripts is too small, and it makes models hard to recover the whole transcriptome.

5.8 The hyperparameter optimization can find well hyperparameters in a few trials

The best trial is **trial$_{47}$**, and its macro F-score is 0.953, as shown in **Table 15.** The settings with the highest count in **Table 14** similar to the best trial, **trial$_{47}$**, except the number of layers in feature block is 12, not 16. The F-scores of the exon, intron, and the intergenic region are 0.942, 0.934, and 0.983, as shown in **Table 13**. The label order of the biggest F-score to the smallest F-score is the intergenic region, exon, and intron. The label order of the largest portion to the smallest portion is the intergenic region, exon, and intron, as shown in **Figure 18.** These two orders are the same, and it shows that the higher portion the label number has, the higher its F-score is. The layer number of the concatenated CRB block in the feature block tends to be as large as 12. It is because as the number of layers increase, it can learn more abstract features. The phenomenon is similar to ResNet, Wide Residual Networks, DenseNet, and EfficientNet, which showed a deeper layer could have better performance **[3, 43, 74, 75]**. The output number of concatenated CRN blocks in the feature block tends to be as small as 4. The phenomenon is a contradiction with the observation in the previous researches that showed that the larger output number of each block could increase performance **[3, 27, 75]**. Although the DenseNet can have very thin layers due to its ability to reuse features of all previous layers, the research showed that the larger output number of each block could increase performance **[3]**. The phenomenon might be caused by the overfitting when the model parameters are too many, and the data is not sufficient to make a model generalize well on unseen data. So, the dropout would be used during cross-validation to prevent overfitting. The kernel size in the feature block tends to be as large as 2049. The phenomenon is expected because the larger kernel size can increase the receptive field. So, it can capture a larger feature. The number of hidden units in each RNN layer tends to be as large as 160. The phenomenon is expected because the more hidden unit can increase the memory and their parameters to learn more features. The previous research showed that RNN with more hidden units could achieve better results **[76]**. The number of RNN layers tends to be as large as 4. As mentioned before, it is because as the number of layers increase, it can learn more abstract features. The models that used the hierarchy inference method have better macro F-score than the models that used the basic inference method. Among the model used the hierarchy inference method, the five out of ten of the relation block type are hierarchy relation block; the four out of ten are basic hierarchy relation block. It seems the hierarchy relation block is slightly better than the basic hierarchy relation block.

5.9 The result of deep learning and result of Augustus have their strength and weakness

Augustus and deep learning have their strength and weakness, as shown in **Table 16**. The exon F-score, intron F-score, and macro F-score of the deep learning model are significantly better than Augustus. Intergenic region F-score of deep learning model is significantly worse than Augustus. Besides the base-level metrics, the additional metrics were also be used to test these two kinds of models. Both block-level performances of the deep learning model are significantly worse than Augustus. One of the two chain-block-level performances, two of the four distance performances, and two of the four site prediction of the deep learning model are significantly worse than Augustus. Although the overall base-level performances of the deep learning model are significantly better than Augustus, the additional performances show the opposite situation. The design of the loss function and hyperparameter optimization procedure might cause this problem. The loss function is designed to minimize the difference between answer and prediction on the base level, and the optimized target of hyperparameter optimization is macro F-score of the base level. It is hard to design the loss function on these additional metrics. So, the post-processing procedure would be applied to the predicted result of deep learning to improve the overall performance.

doi:10.6342/NTU202002143

5.10 The result of deep learning has fragment problem and boundary problem, and the data in $\mathbf{Data}_{\mathbf{Train}}$ and $\mathbf{Predicted}_{\mathbf{Val}}$ can provide information for post-processing procedure

The log-length distributions of the exon, intron, and gene on $\mathbf{Data}_{\mathbf{Train}}$ and $\mathbf{Predicted}_{\mathbf{Val}}$ are similar, and the only difference is there are some predicted regions shorter than most regions of each type, as shown in from **Figure 21**, **Figure 22a**, and **Figure 22b**. The distributions of the log-length of the intergenic region on $\mathbf{Data}_{\mathbf{Train}}$ and $\mathbf{Predicted}_{\mathbf{Val}}$ are similar, as shown in **Figure 22c** and **Figure 22d**. The log-length distributions around 2 and 3 are the lengths of downstream intergenic regions and upstream intergenic regions, and the log-length distribution around 3.5 is the lengths of regions with no exon. The only difference is the log-length distribution around 0.5, which is the length distribution of fragments of the predicted intergenic region. The values of Gaussian models will be used in the length filtering method, and the fragment problem can be relieved, as described in **section 3.6**.

Table 17 shows the $\text{Distance}_{a,\ p}$ of DS is 21 nts, and the $\text{Distance}_{a,\ p}$ of AS is also 23 nts. These indicate the answer site is nearby the predicted site. The problem can be relieved by the boundary post-processing method, as described in **section 3.6**. The boundary post-processing method needs canonical motifs on the $\mathbf{Data}_{\mathbf{Train}}$. **Table S7** shows that the canonical motifs of the DS on the $\mathbf{Data}_{\mathbf{Train}}$ are GT, and it also shows that the canonical motif of the AS on the $\text{Data}_{\text{Train}}$ is AG. As expected, these canonical motifs on $\mathbf{Data}_{\mathbf{Train}}$ are the same as canonical motifs on $\mathbf{Data}_{\mathbf{whole}}$, as shown in **Table 12**. The canonical motifs GT and AT are used for the boundary post-processing method to relieved the boundary problem, as described in **section 3.6**.

doi:10.6342/NTU202002143

5.11 The post-processing procedure can improve the result of the deep learning model

As shown in **Table 18**, **reviser$_8$**, which is the best reviser, treats region which its length is small than its threshold as a fragment and removes it by the length filtering method. Then it uses the boundary post-processing method to fix the intron boundary. The exon F-score, intron F-score, macro F-score, and F-score of CS prediction are worse after **reviser$_8$** is used, as shown in **Figure 23a**, **Figure 23b**, and **Figure 23e**. Fortunately, as shown in **Table 19,** the statistic result shows no significant difference in these metrics after **reviser$_8$** is used. As shown in **Figure 23d** and **Table 19**, the gene F-score has not changed after the **reviser$_8$** is used. The F-score of TSS prediction and intergenic region F-score are better after **reviser$_8$** is used, as shown in **Figure 23a** and **Figure 23e**. Unfortunately, as shown in **Table 19,** the statistic result shows no significant difference after **reviser$_8$** is used. All the other metrics are better after the post-processing procedure, as shown in **Figure 23**. They are all significantly better after the post-processing procedure, as shown in **Table 19**. It shows the post-processing procedure can indeed improve almost all the metrics and without any significant loss of performances.

5.12 The deep learning model with the post-processing procedure is competitive to Augustus in many places

The deep learning model with the post-processing procedure has been compared again to Augustus. The result is shown in **section 4.7**. As shown in **Table 20**, two of the three base-level F-scores, and macro F-score on the base level, and three of the four distance performances of the deep learning model with the post-processing procedure are significantly better than Augustus. The p-values of the tests in chain-block-level performances do not reach the statistical significance, as shown in **Table 20.** As shown in **Table 20**, one of the three base-level F-scores, both block-level performances, one of the four distance performances, and one of the four site prediction of the deep learning model with the post-processing procedure are significantly worse than Augustus. Overall, the base-level performances and distances of the deep learning model with the post-processing procedure are better than Augustus. The block-level performances of the deep learning model with the post-processing procedure are worse than Augustus. The chain-base-level performances and site prediction of the deep learning model with the post-processing procedure are similar to Augustus. As shown in **Table 20**, the 6 out of 16 metrics in the revised result of deep learning are significantly better than Augustus, the 5 out of 16 metrics in the revised result of deep learning are significantly worse than Augustus, and 5 out of 16 metrics have no statistical significance. These results show that the deep learning model with the post-processing procedure is competitive to Augustus.

5.13 The difficulty of getting a good result in each metric

As shown in **Figure 24**, the mean distances in the DS and AS are smaller than mean distances in TSS and CS, and the site-prediction F-scores of DS and AS are far higher than site-prediction F-scores of TSS and CS. This phenomenon might be because the motifs of the DS and AS are more conserved than motifs of the TSS and CS, as shown in **Figure 19.** There are 3415 DSs and 3415 ASs in **Data**$_{\textbf{Train}}$, as shown in **Table S7**. As shown in **Table S5**, there are 1636 regions in **Data**$_{\textbf{Train}}$, so there are 818 transcripts in **Data**$_{\textbf{Train}}$. It indicates that there are 818 TSSs and 818 CSs in **Data**$_{\textbf{Train}}$. So, the other possible reason is that there are more DSs and ASs than TSSs and cleavages sites, so it makes the performance s of the splicing site could be better.

As shown in **Figure 24**, different types of metrics have some interesting trends. The performances of base-level metrics are higher than performances of block-level metrics because a correct block needs all its bases to be correct, and all its bases to be correct are hard. The block-level metrics are higher than chain-block-level metrics because correct chained blocks need all their blocks to be correct, and all their blocks to be correct are very hard. The intron block F-scores are higher than the exon block F-score because the boundary sites of the intron, which are DS and AS, have higher performances than TSS and CS, which are the two boundary sites of the exon. The chain-intron F-scores are higher than the gene F-score because the performance of the intron block is better than the performance of exon block. It seems the site prediction performance of the TSS and CS is a key point in the overall performance. If the site prediction performance of the TSS and CS can be improved, then the mean distance of them, exon block F-score, and gene F-score can all be improved. As mentioned above, the number of TSS and CS is very low. So, increasing the number of them or designing some loss for them would be the possible solutions.

5.14 The deep learning model with the post-processing procedure can predict domain-including genes in potential transcript regions

As shown in **Figure 26**, the percentage of genes derived from Araport11 has at least one domain in Pfam-A is very low. It is because genes in Araport11 were derived from all kinds of transcripts. It included transcripts like non-coding transcripts. The number of Araport11 protein-coding genes was 27655 **[1]**. It is similar to the number of domain-including genes in potential transcript regions, which is 27343, as shown in **Figure 27** and **Table 21**. As shown in **Figure 25** and **Figure 26,** the numbers of genes from the result of Augustus and the revised result of deep learning model are similar,

doi:10.6342/NTU202002143

but the ratio of domain-including genes from the deep learning model with revision is less than the ratio from Augustus. The poor performance of the revised result of deep learning on the exon block and the intron block might be the reason. The performance on the exon block and the intron block should be improved, so the predicted result on the potential transcript regions would be better. The numbers of domain-supported genes from the deep learning model with revision and Augustus are lower than the number of domain-supported genes from Araport11, as shown in **Figure 27.** There are two possible reasons. One is that only a few genes are used to train both models, so these two kinds of models cannot predict all the domain-supported genes in the potential transcription regions. The other reason is that the number of domain-supported genes of Araport11 may be overestimated, and the actual number of domain-supported genes is not that much.

5.15 The comparison of other annotation applications

The proposed deep learning model has some advantages over some existed methods. Unlike DeepAnnotator **[10]** that trained multiple submodels independently to accomplish the task, the submodels of the proposed deep learning model can be trained together. The transcript-related basic hierarchy block in the hierarchy block can learn features from the feature block. The intron-related basic hierarchy block in the hierarchy block can learn features from the feature block and transcript-related basic hierarchy block. Unlike SpliceAI **[9]** could only predict the splicing site and DeepPolyA **[7]** could only predict the CS, the proposed model can predict full annotation. The proposed architecture of the deep learning model does not need to analyze the feature composition and can learn the features it needs, unlike Augustus **[2]**, which needed to

doi:10.6342/NTU202002143

analyze the feature composition of annotation, and needed to design many submodels to detect these features.

5.16 Future work on improving model

Currently, the proposed deep learning model cannot be significantly better than Augustus in all the metrics. Although the overall base-level metrics and overall distance metrics have better results than the results of Augustus, the other metrics still need to be improved. One possible solution is to change the target of hyperparameter optimization from macro F-score to $\mathrm{Loss}_{\mathrm{revision.}}$ The other possible solution is to incorporate the 8362 experimental TSSs and 21260 experimental CSs to existed data, as shown in **Table 5** and **Table 6**. So, the model can have more data to use. The deep learning model needs a large number of data to achieve a better result due to its high number of parameters. Currently, the number of high-quality data is far smaller than the number of coding genes. There is indeed a tradeoff between the quality of data and the number of data, so one way to get more data is to increase the tolerance of the preprocessing procedure to the low-quality data. As discussed in **section 5.3**, the upstream distance can affect the number of data. The larger the upstream distance is, the lower the percentage of genes that have evidence supported is. The result with the 1000-nt upstream distance has 98% of genes with transcription evidence supported, 977 genes, 1031 transcripts, and 1954 regions. In order to get more data, the upstream distance can be reduced to 500 nts, so the result with 500-nt upstream distance has 95% of genes with transcription evidence supported, 1593 genes, 1678 transcripts, and 3186 regions. The number of data will have a significant increment, while the percentage of genes with transcription evidence supported will slightly decrease. As discussed in **section 5.7**, only about 12.2% of transcripts can be reannotated, and 24.4% of transcripts can pass the recursive cleaning

doi:10.6342/NTU202002143

procedure. In order to increase the data, more evidence should be collected, so the number of transcripts can be reannotated can increase. The recursive cleaning procedure will filter out transcripts that have any discarded transcripts nearby so that the annotation can be clean. The increased number of data by collecting more evidence can decrease the number of discarded transcripts, so the number of transcripts passed the recursive cleaning procedure will be increased. There is another method to increase the number of transcripts. In this method, the low-quality transcript will be used, but the loss of deep learning model will be redesign, so the high-quality transcript has higher weights than the low-quality transcripts. So, the deep learning model can focus on predicting the high-quality transcript and also learn some useful features from low-quality transcripts.

The transcript structure is mostly derived from short reads. Lacking sufficient reads might cause problems like merging multiple kinds of transcripts of the same gene into one transcript, and this might mislead us the combination of splicing sites. Gene structure, a reduced structure, is created from sites like canonical TSS, canonical CS, and all canonical splicing sites of transcripts. Although the gene structure can derive the transcript structures, it cannot provide a real combination of sites and cannot derive transcripts with multiple TSSs, multiple CSs, and alternative splicing sites. Currently, the deep learning model can only predict gene structure. In order to predict transcript data when there is sufficient high-quality transcriptome, some modification of the model needs to be done.

The deep learning model shows it can predict genes of the *Arabidopsis*, but it has not shown its ability to be applied to other species. There are two problems when applied to other species. The first problem is that many species do not have transcriptome data. The second problem is that only a few species that have boundary

doi:10.6342/NTU202002143

evidence. The solution to the first problem is to use the existed transcriptome of closely related species to predict the transcriptome of species that have not to be annotated. A solution to the second problem is to treat the regions that start with its first translation start site to its first translation stop site as genes and treat other regions as intergenic regions. This solution will make the model focus on coding regions and introns that surrounded by coding regions and treat any noncoding part as intergenic regions, so the boundary evidence will not be needed anymore. Another solution to the second problem is to train the model by transcripts supported by boundary evidence of closely related species. So, the model could be applied to species that lack boundary evidence.

Chapter 6. Conclusion

The thesis showed the procedures to clean and reannotate coding transcripts of *Arabidopsis thaliana* by using multiple filters and data from Araport11 **[1]**, genome, GRO-seq, and PAT-seq. The results showed there was a tradeoff between the quality of annotation and the number of data, and the high-quality coding transcripts were very rare. The methods to create gene annotation, to build the deep-learning-based model, and to revise the result were proposed. The median macro F-score of the deep learning was 0.969, and the median macro F-score of Augustus was 0.957. The Wilcoxon rank-sum test showed that the macro F-score of the deep learning model was significantly better than Augustus. The post-processing procedure could significantly improve the performance of the deep learning model. The revised results showed that the overall base-level metrics and the overall distance metrics of the deep learning model were significantly better than Augustus **[2]**. The model could directly predict gene structures of coding genes on potential transcript regions only by the DNA sequences of *Arabidopsis thaliana*. The result showed that these genes included domain sequences, which made them more reliable. The architecture of the proposed model was more straightforward than the architecture of Augustus. Unlike DeepPolyA **[7]** predicted the CS and SpliceAI **[9]** predicted the splicing site, the proposed model could predict the full gene structure. Unlike DeepAnnotator **[10]**, which used separate models to accomplish the task, the proposed model could accomplish the task by one model. Overall, the proposed deep learning model with a post-processing procedure could be an alternative method to annotate the gene structure of the coding gene on the DNA sequence of *Arabidopsis thaliana*. The performance of the proposed model still needs to be improved, and the problems of predicting annotations of other species that lack enough data are still a challenge that needs to be overcome.

doi:10.6342/NTU202002143

References

1. Cheng C-Y, Krishnakumar V, Chan AP, Thibaud-Nissen F, Schobel S, Town CD: **Araport11: a complete reannotation of the Arabidopsis thaliana reference genome**. *The Plant Journal* 2016, **89**(4):789-804.

2. Stanke M, Waack S: **Gene prediction with a hidden Markov model and a new intron submodel**. *Bioinformatics* 2003, **19**(suppl_2):ii215-ii225.

3. Huang G, Liu Z, van der Maaten L, Weinberger KQ: **Densely Connected Convolutional Networks**. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR): 2017*. IEEE: 2261-2269.

4. Quang D, Xie X: **DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences**. *Nucleic Acids Research* 2016, **44**(11):e107-e107.

5. Hill ST, Kuintzle R, Teegarden A, Merrill IIIE, Danaee P, Hendrix DA: **A deep recurrent neural network discovers complex biological rules to decipher RNA protein-coding potential**. *Nucleic Acids Research* 2018, **46**(16):8105-8113.

6. Veltri D, Kamath U, Shehu A: **Deep learning improves antimicrobial peptide recognition**. *Bioinformatics* 2018, **34**(16):2740-2747.

7. Gao X, Zhang J, Wei Z, Hakonarson H: **DeepPolyA: A Convolutional Neural Network Approach for Polyadenylation Site Prediction**. *IEEE Access* 2018, **6**:24340-24349.

8. Bretschneider H, Gandhi S, Deshwar AG, Zuberi K, Frey BJ: **COSSMO: predicting competitive alternative splice site selection using deep learning**. *Bioinformatics* 2018, **34**(13):i429-i437.

9. Jaganathan K, Panagiotopoulou SK, McRae JF, Darbandi SF, Knowles D, Li YI, Kosmicki JA, Arbelaez J, Cui W, Schwartz GB: **Predicting splicing from primary sequence with deep learning**. *Cell* 2019, **176**(3):535-548.

10. Amin MR, Yurovsky A, Tian Y, Skiena S: **DeepAnnotator: Genome Annotation with Deep Learning**. In: *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics; Washington, DC, USA*. Association for Computing Machinery 2018: 254–259.

11. Lamesch P, Berardini TZ, Li D, Swarbreck D, Wilks C, Sasidharan R, Muller R, Dreher K, Alexander DL, Garcia-Hernandez M *et al*: **The Arabidopsis Information Resource (TAIR): improved gene annotation and new tools**. *Nucleic Acids Research* 2012, **40**(D1):D1202-D1210.

12. Trapnell C, Pachter L, Salzberg SL: **TopHat: discovering splice junctions with RNA-Seq**. *Bioinformatics* 2009, **25**(9):1105-1111.

13. Stanke M, Keller O, Gunduz I, Hayes A, Waack S, Morgenstern B: **AUGUSTUS: ab initio prediction of alternative transcripts**. *Nucleic Acids Research* 2006, **34**(suppl_2):W435-W439.

14. Grabherr MG, Haas BJ, Yassour M, Levin JZ, Thompson DA, Amit I, Adiconis X, Fan L, Raychowdhury R, Zeng Q: **Full-length transcriptome assembly from RNA-Seq data without a reference genome**. *Nature biotechnology* 2011, **29**(7):644.

15. Campbell MS, Law M, Holt C, Stein JC, Moghe GD, Hufnagel DE, Lei J, Achawanantakun R, Jiao D, Lawrence CJ: **MAKER-P: a tool kit for the rapid creation, management, and quality control of plant genome annotations**. *Plant physiology* 2014, **164**(2):513-524.

doi:10.6342/NTU202002143

16. Haas BJ, Delcher AL, Mount SM, Wortman JR, Smith Jr RK, Hannick LI, Maiti R, Ronning CM, Rusch DB, Town CD *et al*: **Improving the Arabidopsis genome annotation using maximal transcript alignment assemblies**. *Nucleic Acids Research* 2003, **31**(19):5654-5666.

17. Haberle V, Stark A: **Eukaryotic core promoters and the functional basis of transcription initiation**. *Nature Reviews Molecular Cell Biology* 2018, **19**(10):621-637.

18. Korkuc P, Schippers JHM, Walther D: **Characterization and identification of cis-regulatory elements in Arabidopsis based on single-nucleotide polymorphism information**. *Plant physiology* 2014, **164**(1):181-200.

19. Yu C-P, Lin J-J, Li W-H: **Positional distribution of transcription factor binding sites in Arabidopsis thaliana**. *Scientific Reports* 2016, **6**:25164.

20. Neve J, Patel R, Wang Z, Louey A, Furger AM: **Cleavage and polyadenylation: Ending the message expands gene regulation**. *RNA Biol* 2017, **14**(7):865-890.

21. Sherstnev A, Duc C, Cole C, Zacharaki V, Hornyik C, Ozsolak F, Milos PM, Barton GJ, Simpson GG: **Direct sequencing of Arabidopsis thaliana RNA reveals patterns of cleavage and polyadenylation**. *Nature structural & molecular biology* 2012, **19**(8):845.

22. Kornblihtt AR, Schor IE, Alló M, Dujardin G, Petrillo E, Muñoz MJ: **Alternative splicing: a pivotal step between eukaryotic transcription and translation**. *Nature Reviews Molecular Cell Biology* 2013, **14**(3):153-165.

23. Kurihara Y, Makita Y, Kawashima M, Fujita T, Iwasaki S, Matsui M: **Transcripts from downstream alternative transcription start sites evade**

uORF-mediated inhibition of gene expression in Arabidopsis. *Proceedings of the National Academy of Sciences* 2018, **115**(30):7831-7836.

24. Zhang P, Dimont E, Ha T, Swanson DJ, Hide W, Goldowitz D: **Relatively frequent switching of transcription start sites during cerebellar development**. *BMC Genomics* 2017, **18**(1):461.

25. Ni T, Corcoran DL, Rach EA, Song S, Spana EP, Gao Y, Ohler U, Zhu J: **A paired-end sequencing strategy to map the complex landscape of transcription initiation**. *Nature Methods* 2010, **7**(7):521-527.

26. Morton T, Petricka J, Corcoran DL, Li S, Winter CM, Carda A, Benfey PN, Ohler U, Megraw M: **Paired-end analysis of transcription start sites in Arabidopsis reveals plant-specific promoter signatures**. *The Plant Cell* 2014:tpc-114.

27. Core LJ, Waterfall JJ, Lis JT: **Nascent RNA Sequencing Reveals Widespread Pausing and Divergent Initiation at Human Promoters**. *Science* 2008, **322**(5909):1845.

28. Ozsolak F, Platt AR, Jones DR, Reifenberger JG, Sass LE, McInerney P, Thompson JF, Bowers J, Jarosz M, Milos PM: **Direct RNA sequencing**. *Nature* 2009, **461**:814.

29. Harrison PF, Powell DR, Clancy JL, Preiss T, Boag PR, Traven A, Seemann T, Beilharz TH: **PAT-seq: a method to study the integration of 3'-UTR dynamics with gene expression in the eukaryotic transcriptome**. *RNA* 2015, **21**(8):1502-1510.

30. Krogh A, Mian IS, Haussler D: **A hidden Markov model that finds genes in E. coli DNA**. *Nucleic Acids Research* 1994, **22**(22):4768-4778.

doi:10.6342/NTU202002143

31. Haussler, David DK, Eeckman, H MGRF: **A generalized hidden Markov model for the recognition of human genes in DNA**. In: *Proceedings of the International Conference on Intelligent Systems for Molecular Biology, St Louis: 1996*. 134-142.

32. Stanke M, Diekhans M, Baertsch R, Haussler D: **Using native and syntenically mapped cDNA alignments to improve de novo gene finding**. *Bioinformatics* 2008, **24**(5):637-644.

33. Stanke M, Morgenstern B: **AUGUSTUS: a web server for gene prediction in eukaryotes that allows user-defined constraints**. *Nucleic Acids Research* 2005, **33**(suppl_2):W465-W467.

34. Holt C, Yandell M: **MAKER2: an annotation pipeline and genome-database management tool for second-generation genome projects**. *BMC Bioinformatics* 2011, **12**(1):491.

35. Korf I: **Gene finding in novel genomes**. *BMC Bioinformatics* 2004, **5**(1):59.

36. Lomsadze A, Ter-Hovhannisyan V, Chernoff YO, Borodovsky M: **Gene identification in novel eukaryotic genomes by self-training algorithm**. *Nucleic Acids Research* 2005, **33**(20):6494-6506.

37. Chan K-L, Rosli R, Tatarinova TV, Hogan M, Firdaus-Raih M, Low E-TL: **Seqping: gene prediction pipeline for plant genomes using self-training gene models and transcriptomic data**. *BMC Bioinformatics* 2017, **18**(1):1-7.

38. Majoros WH, Pertea M, Salzberg SL: **TigrScan and GlimmerHMM: two open source ab initio eukaryotic gene-finders**. *Bioinformatics* 2004, **20**(16):2878-2879.

doi:10.6342/NTU202002143

39. Krizhevsky A, Sutskever I, Hinton GE: **Imagenet classification with deep convolutional neural networks**. In: *Advances in neural information processing systems: 2012*. 1097-1105.

40. Redmon J, Divvala S, Girshick R, Farhadi A: **You only look once: Unified, real-time object detection**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition: 2016*. 779-788.

41. Glorot X, Bordes A, Bengio Y: **Deep sparse rectifier neural networks**. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics: 2011*. 315-323.

42. Bridle JS: **Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters**. In: *Advances in neural information processing systems: 1990*. 211-217.

43. He K, Zhang X, Ren S, Sun J: **Deep residual learning for image recognition**. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 2016*. 770-778.

44. Xu Y, Kong Q, Huang Q, Wang W, Plumbley MD: **Attention and Localization based on a Deep Convolutional Recurrent Model for Weakly Supervised Audio Tagging**. In: *Conference of the International Speech Communication Association: 8/20/2017*. 2017: 3083-3087.

45. Ismail Fawaz H, Forestier G, Weber J, Idoumghar L, Muller P-A: **Deep learning for time series classification: a review**. *Data Mining and Knowledge Discovery* 2019, **33**(4):917-963.

46. Siegelmann HT, Sontag ED: **On the computational power of neural nets**. *Journal of computer and system sciences* 1995, **50**(1):132-150.

doi:10.6342/NTU202002143

47. Cho K, van Merrienboer B, Gulcehre C, Bougares F, Schwenk H, Bengio Y: **Learning phrase representations using RNN encoder-decoder for statistical machine translation**. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014): 2014*.

48. Kingma DP, Ba JL: **Adam: A Method for Stochastic Optimization**. In: *International Conference on Learning Representations: 1/1/2015*. 2015.

49. Ioffe S, Szegedy C: **Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift**. In: *Proceedings of the 32nd International Conference on Machine Learning; Proceedings of Machine Learning Research*: Edited by Francis B, David B. PMLR 2015: 448--456.

50. Santurkar S, Tsipras D, Ilyas A, Madry A: **How does batch normalization help optimization?** In: *Advances in Neural Information Processing Systems: 2018*. 2483-2493.

51. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R: **Dropout: a simple way to prevent neural networks from overfitting**. *The Journal of Machine Learning Research* 2014, **15**(1):1929-1958.

52. Pham V, Bluche T, Kermorvant C, Louradour J: **Dropout improves recurrent neural networks for handwriting recognition**. In: *2014 14th International Conference on Frontiers in Handwriting Recognition: 2014*. IEEE: 285-290.

53. Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A: **Automatic differentiation in PyTorch**. In.; 2017.

54. Snoek J, Larochelle H, Adams RP: **Practical bayesian optimization of machine learning algorithms**. In: *Advances in neural information processing systems: 2012*. 2951-2959.

doi:10.6342/NTU202002143

55. Yeo G, Burge CB: **Maximum entropy modeling of short sequence motifs with applications to RNA splicing signals**. *Journal of Computational Biology* 2004, **11**(2-3):377-394.

56. Salzberg SL, Delcher AL, Kasif S, White O: **Microbial gene identification using interpolated Markov models**. *Nucleic acids research* 1998, **26**(2):544-548.

57. Hetzel J, Duttke SH, Benner C, Chory J: **Nascent RNA sequencing reveals distinct features in plant transcription**. *Proceedings of the National Academy of Sciences* 2016, **113**(43):12316.

58. Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, Gingeras TR: **STAR: ultrafast universal RNA-seq aligner**. *Bioinformatics* 2013, **29**(1):15-21.

59. Heinz S, Benner C, Spann N, Bertolino E, Lin YC, Laslo P, Cheng JX, Murre C, Singh H, Glass CK: **Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and B cell identities**. *Molecular cell* 2010, **38**(4):576-589.

60. Zhu S, Ye W, Ye L, Fu H, Ye C, Xiao X, Ji Y, Lin W, Ji G, Wu X: **PlantAPAdb: A Comprehensive Database for Alternative Polyadenylation Sites in Plants**. *Plant Physiology* 2020, **182**(1):228.

61. Akiba T, Sano S, Yanase T, Ohta T, Koyama M: **Optuna: A next-generation hyperparameter optimization framework**. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining: 2019*. 2623-2631.

62. Student: **The probable error of a mean**. *Biometrika* 1908:1-25.

doi:10.6342/NTU202002143

63. Mann HB, Whitney DR: **On a test of whether one of two random variables is stochastically larger than the other**. *The annals of mathematical statistics* 1947:50-60.

64. Hothorn T, Hornik K, Hothorn MT: **Package 'exactRankTests'**. 2019.

65. Rice P, Longden I, Bleasby A: **EMBOSS: The European Molecular Biology Open Software Suite**. *Trends in Genetics* 2000, **16**(6):276-277.

66. Potter SC, Luciani A, Eddy SR, Park Y, Lopez R, Finn RD: **HMMER web server: 2018 update**. *Nucleic acids research* 2018, **46**(W1):W200-W204.

67. Finn RD, Bateman A, Clements J, Coggill P, Eberhardt RY, Eddy SR, Heger A, Hetherington K, Holm L, Mistry J: **Pfam: the protein families database**. *Nucleic acids research* 2014, **42**(D1):D222-D230.

68. El-Gebali S, Mistry J, Bateman A, Eddy SR, Luciani A, Potter SC, Qureshi M, Richardson LJ, Salazar GA, Smart A *et al*: **The Pfam protein families database in 2019**. *Nucleic Acids Research* 2018, **47**(D1):D427-D432.

69. Hoff KJ, Stanke M: **Predicting genes in single genomes with augustus**. *Current protocols in bioinformatics* 2019, **65**(1):e57.

70. Crooks GE, Hon G, Chandonia J-M, Brenner SE: **WebLogo: a sequence logo generator**. *Genome Research* 2004, **14**(6):1188-1190.

71. Gallegos JE, Rose AB: **Intron DNA sequences can be more important than the proximal promoter in determining the site of transcript initiation**. *The Plant Cell* 2017:tpc-00020.

72. Alexandrov NN, Troukhan ME, Brover VV, Tatarinova T, Flavell RB, Feldmann KA: **Features of Arabidopsis genes and genome discovered using full-length cDNAs**. *Plant molecular biology* 2006, **60**(1):69-85.

doi:10.6342/NTU202002143

73. Loke JC, Stahlberg EA, Strenski DG, Haas BJ, Wood PC, Li QQ: **Compilation of mRNA polyadenylation signals in Arabidopsis revealed a new signal element and potential secondary structures**. *Plant physiology* 2005, **138**(3):1457-1468.

74. Tan M, Le Q: **EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks**. In: *International Conference on Machine Learning: 2019*. 6105-6114.

75. Zagoruyko S, Komodakis N: **Wide Residual Networks**. In: *British Machine Vision Conference: 1/1/2016*. 2016.

76. Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J: **LSTM: A Search Space Odyssey**. *IEEE Trans Neural Networks Learn Syst* 2017, **28**(10):2222-2232.

77. Quinlan AR, Hall IM: **BEDTools: a flexible suite of utilities for comparing genomic features**. *Bioinformatics* 2010, **26**(6):841-842.

78. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, Subgroup GPDP: **The Sequence Alignment/Map format and SAMtools**. *Bioinformatics* 2009, **25**(16):2078-2079.

79. Grisel O, Mueller A, Lars, Gramfort A, Louppe G, Prettenhofer P, Blondel M, Niculae V, Nothman J, Joly A *et al*: **scikit-learn/scikit-learn: Scikit-learn 0.22.2.post1**. In., 0.22.2.post1 edn: Zenodo; 2020.

80. Head T, MechCoder, Louppe G, Shcherbatyi I, fcharras, Vinícius Z, cmmalone, Schröder C, nel, Campos N *et al*: **scikit-optimize/scikit-optimize: v0.5.2**. In., v0.5.2 edn: Zenodo; 2018.

81. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J: **SciPy 1.0: fundamental**

algorithms for scientific computing in Python. *Nature methods* 2020, **17**(3):261-272.

82. Hunter JD: **Matplotlib: A 2D Graphics Environment**. *Computing in Science & Engineering* 2007, **9**(3):90-95.

83. Waskom M, Botvinnik O, Ostblom J, Lukauskas S, Hobson P, MaozGelbart, Gemperline DC, Augspurger T, Halchenko Y, Cole JB *et al*: **mwaskom/seaborn: v0.9.1 (January 2020)**. In., v0.9.1 edn: Zenodo; 2020.

84. Walt Svd, Colbert SC, Varoquaux G: **The NumPy Array: A Structure for Efficient Numerical Computation**. *Computing in Science & Engineering* 2011, **13**(2):22-30.

85. McKinney W: **Data Structures for Statistical Computing in Python**. In*: 2010 2010*. 56-61.

86. **venn 0.1.3** [https://pypi.org/project/venn/]

87. He K, Zhang X, Ren S, Sun J: **Delving deep into rectifiers: Surpassing human-level performance on imagenet classification**. In: *Proceedings of the IEEE International Conference on Computer Vision: 2015*. 1026-1034.

88. Glorot X, Bengio Y: **Understanding the difficulty of training deep feedforward neural networks**. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics: 2010*. 249-256.

doi:10.6342/NTU202002143

Supplementary Figures

doi:10.6342/NTU202002143

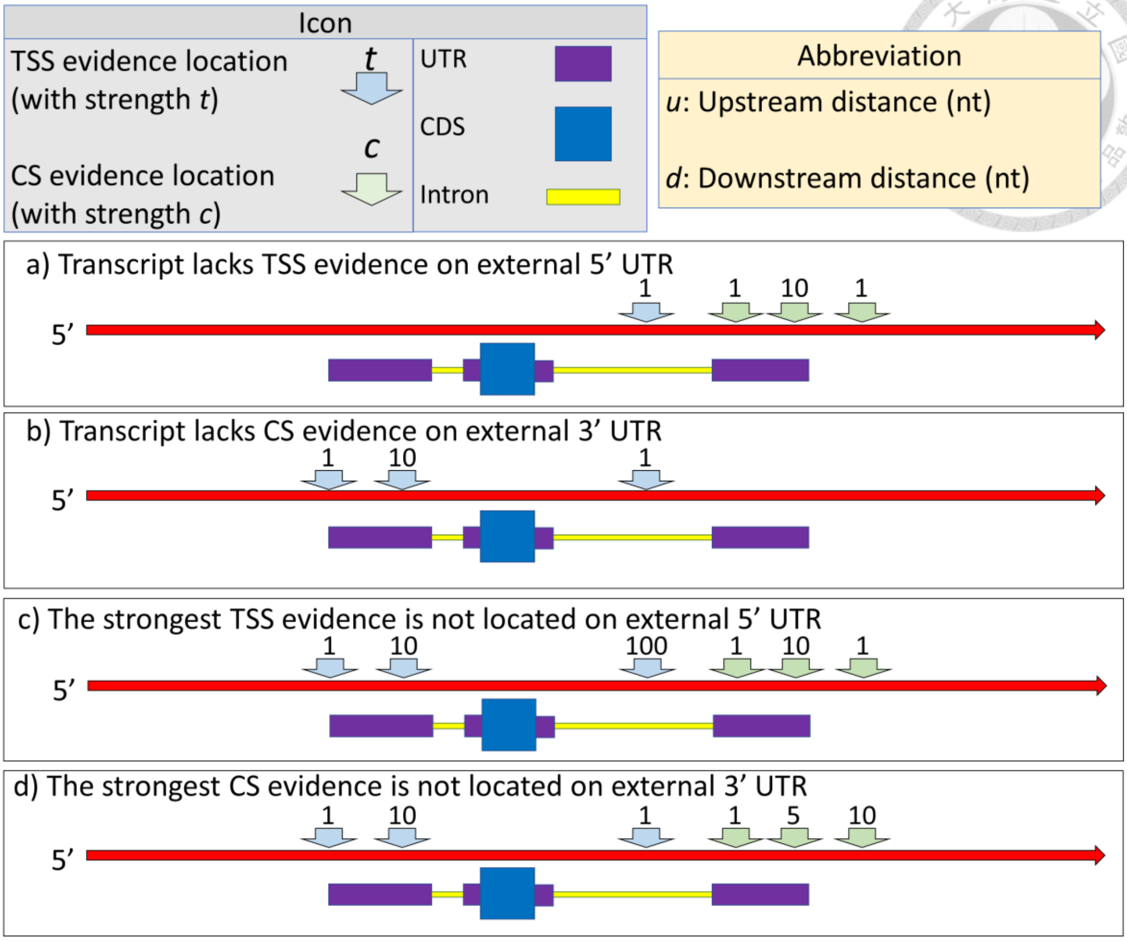
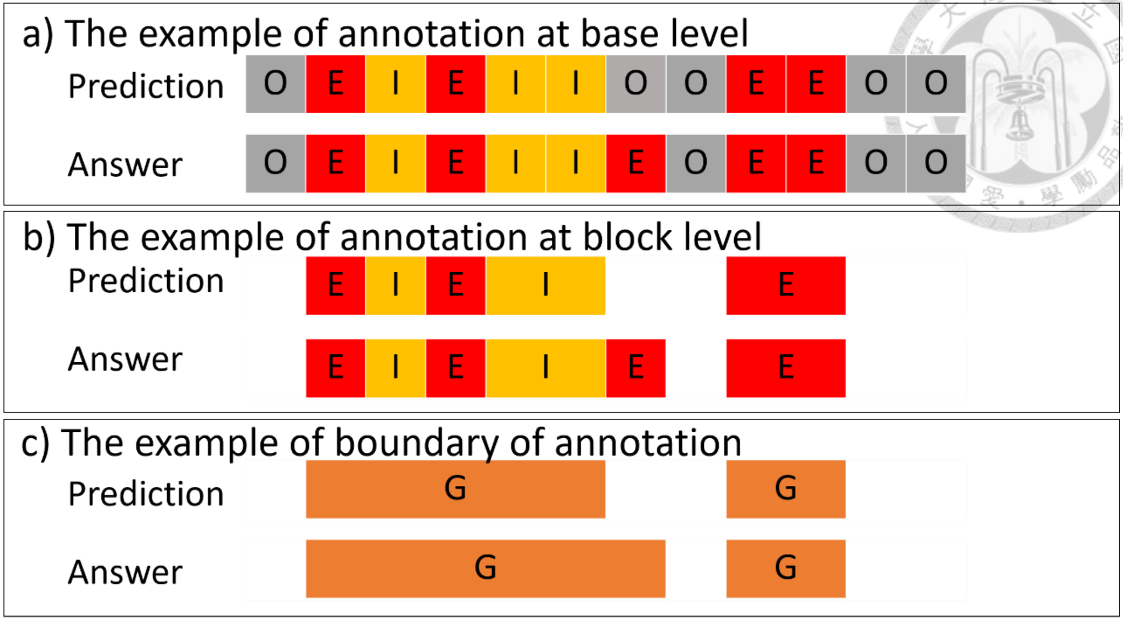

Icon
TSS evidence location (with strength *t*)
t
CS evidence location (with strength *c*)
c
UTR
CDS
Intron
Abbreviation
u: Upstream distance (nt)
d: Downstream distance (nt)
a) Transcript lacks TSS evidence on external 5′ UTR
5′
1
1
10
1
b) Transcript lacks CS evidence on external 3′ UTR
5′
1
10
1
c) The strongest TSS evidence is not located on external 5′ UTR
5′
1
10
100
1
10
1
d) The strongest CS evidence is not located on external 3′ UTR
5′
1
10
1
1
5
10

Figure S1. **Examples of transcripts failed to be reannotated (Assumed all the evidence related to the transcript)**


a) The example of annotation at base level
Prediction
O E I E I I O O E E O O
Answer
O E I E I I E O E E O O
b) The example of annotation at block level
Prediction
E I E I E
Answer
E I E I E E
c) The example of boundary of annotation
Prediction
G G
Answer
G G

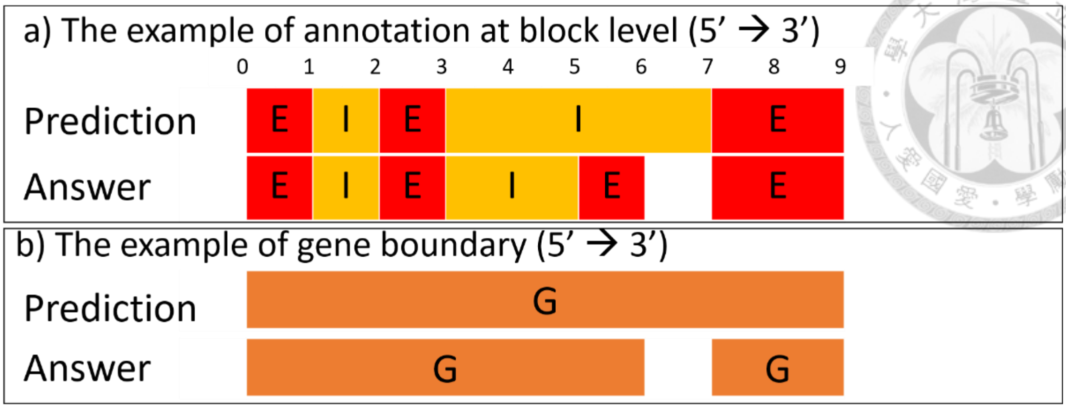
d) Recalls, precisions, and F-scores of the examples

Type\Metric	Recall	Precision	F-score
Exon	0.80	1.00	0.89
Intron	1.00	1.00	1.00
Other	1.00	0.80	0.89
Intron block	1.00	1.00	1.00
Exon block	0.75	1.00	0.86
Chained introns	1.00	1.00	1.00
Gene	0.50	0.50	0.50

e) Macro recalls, macro precisions, and macro F-scores of the examples

Metric	Macro recall	Macro precision	Macro F-score
Value	0.93	0.93	0.93

Figure S2. The examples of annotation at every level, their gene boundaries, and metric results on the base level, the block level, and chain-block level



a) The example of annotation at block level (5' → 3')

b) The example of gene boundary (5' → 3')

c) The distances of the closest source site to target site i

Type\Metric	Distance$_{p,a,i}$	Distance$_{a,p,i}$
TSS	0,7	0
Cleavage site	3,0	0
Splicing donor site	0,0	0,0
Splicing acceptor site	0,2	0,2

d) The mean distances

Type\Metric	Distance$_{p, a}$	Distance$_{a, p}$	Distance$_{mean}$
TSS	3.5	0	1.75
Cleavage site	1.5	0	0.75
Splicing donor site	0	0	0
Splicing acceptor site	1	1	1

e) The recalls, precisions, and F-scores of site predictions

Type\Metric	Recall	Precision	F-score
TSS	0.50	1.00	0.67
Cleavage site	0.50	1.00	0.67
Splicing donor site	1.00	1.00	1.00
Splicing acceptor site	0.50	0.50	0.50

Figure S3. The examples of annotation and metrics of distances and site predictions

doi:10.6342/NTU202002143

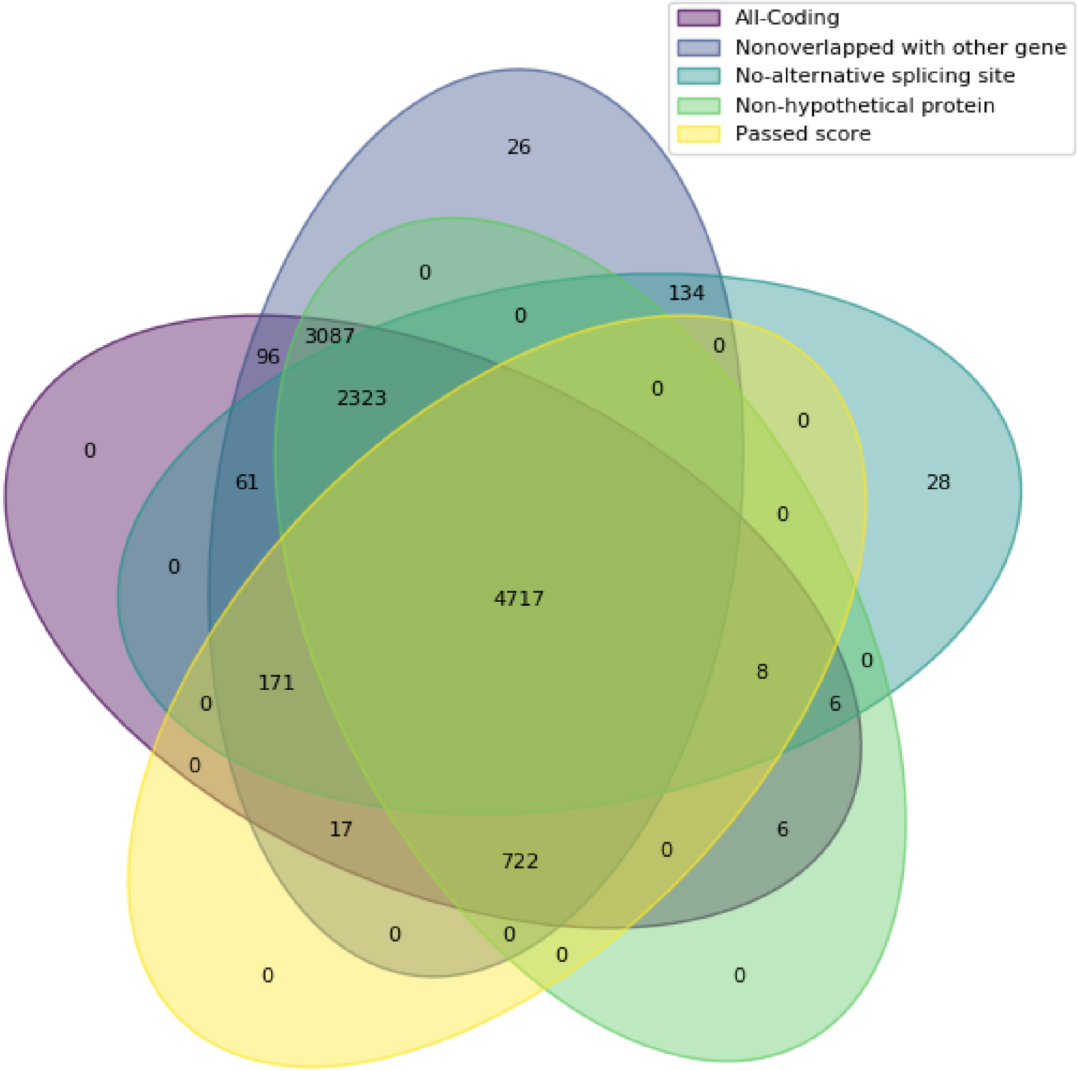


Figure S4. **The Venn diagram of the transcripts passed filters**

Supplementary Tables

Table S1. The version of tools

Name	Version	Name	Version
AUGUSTUS [2, 13, 32, 33]	3.3.2	**PyTorch [53]**	1.4.0
bedtools [77]	2.25.0	**SAMtools [78]**	0.1.19-96b5f2294a
exactRankTests [64]	0.8-31	**Scikit-Learn [79]**	0.22.2.post1
HMMER3 [66]	3.3	**Scikit-Optimize [80]**	0.5.2
HOMER [59]	4.10	**Scipy [81]**	1.4.1
Matplotlib [82]	3.0.3	**Seaborn [83]**	0.9.1
NumPy[84]	1.18.5	**STAR [58]**	2.6.1a
Optuna [61]	1.2.0	**transeq [65]**	6.6.0
Pandas[85]	0.24.0	**venn [86]**	0.1.3
pfam_scan.pl [67]	1.6		

Table S2. Data source summary

Type	Name or Id
Genome	GCF_000001735.3 **[11]**
Transcriptome	Araport11 **[1].**
5’ GRO-seq	SRR3647033 **[57]**
GRO-seq datasets	SRR3647034 **[57]** and SRR3647035 **[57]**
PAT-seq processed dataset	SRP089899 **[60]**

doi:10.6342/NTU202002143

Table S3. The names and sources of the datasets (The number mean chromosome)

Name	Source	Name	Source	Name	Source
Train$_1$	1+, 2, 3, and 5	**Val$_1$**	1-	**Data$_{\text{Whole}}$**	1, 2, 3, 4, and 5
Train$_2$	1-, 2, 3, and 5	**Val$_2$**	1+	**Data$_{\text{Train}}$**	1, 2, 3, and 5
Train$_3$	1, 2 +, 3, and 5	**Val$_3$**	2-	**Data$_{\text{Test}}$**	4
Train$_4$	1, 2-, 3, and 5	**Val$_4$**	2+	**Train$_{\text{Small}}$**	Train$_3$
Train$_5$	1, 2, 3+, and 5	**Val$_5$**	3-	**Val$_{\text{Small}}$**	Val$_3$
Train$_6$	1, 2, 3-, and 5	**Val$_6$**	3+		
Train$_7$	1, 2, 3, and 5+	**Val$_7$**	5-		
Train$_8$	1, 2, 3, and 5-	**Val$_8$**	5+		

Table S4. Weights and bias initialization (Notes: The fan_in means the number of input channel)

Block	Layer	Method	Method Formula
Feature block	CNN	Kaiming initialization **[87]**	$\mathbf{Weights} \in \mathbf{Uniform}(-\boldsymbol{x}, \boldsymbol{x}), \mathbf{where}\ \boldsymbol{x} = \sqrt{\frac{\mathbf{6}}{\mathbf{fan\_in}}}$
			$\mathbf{Bias} = \mathbf{0}$
Relation block	RNN	Xavier initialization **[88]**	$\mathbf{Weights} \in \mathbf{Uniform}(-\boldsymbol{x}, \boldsymbol{x}), \mathbf{where}\ \boldsymbol{x} = \sqrt{\frac{\mathbf{3}}{\mathbf{fan\_in}}}$
	CNN		$\mathbf{Bias} = \mathbf{0}$

Table S5. The number of regions on each dataset

Dataset	**Number**	**Dataset**	**Number**	**Dataset**	**Number**	**Dataset**	**Number**
Train$_1$	1382	**Train$_5$**	1450	**Val$_1$**	254	**Val$_5$**	186
Train$_2$	1382	**Train$_6$**	1450	**Val$_2$**	254	**Val$_6$**	186
Train$_3$	1498	**Train$_7$**	1396	**Val$_3$**	138	**Val$_7$**	240
Train$_4$	1498	**Train$_8$**	1396	**Val$_4$**	138	**Val$_8$**	240
Data$_{\text{Train}}$	1636	**Data$_{\text{Test}}$**	318				

Table S6. The summary of regions with single exon, regions with multiple exons, regions with no exon (no gene), and all regions

Region type	**Count**	**Length**		
		Min	**Median**	**Max**
Regions with single exon	220	1411	2276	5005
Regions with multiple exons	757	1723	3466	9580
Regions with no exon (no gene)	977	1411	3129	9580
All	1954	1411	3129	9580

Table S7. The statistical result of DS and AS in gene annotation on Data$_{\text{Train}}$

Splicing donor site			**Splicing acceptor site**		
Motif	**Count**	**Percentage (%)**	**Motif**	**Count**	**Percentage (%)**
GT	3391	99.297%	**AG**	3415	100
GC	23	0.673%			
TT	1	0.029%			

Table S8. Hyperparameter setting and $\text{Loss}_{\text{revision}}$ of the post-processing procedures (L indicates Length filtering and B indicates Boundary post-processing)

Id	**Methods**	**Distance scale**	**$\text{Loss}_{\text{revision}}$**
Origin	NaN	NaN	0.52156
Reviser_{1}	L method	NaN	0.50812
Reviser_{2}	B method	0	0.51764
Reviser_{3}	B method	1	0.51008
Reviser_{4}	B method	2	0.50984
Reviser_{5}	B method	3	0.50999
Reviser_{6}	L method and then the B method	0	0.51292
Reviser_{7}	L method and then the B method	1	0.50220
Reviser_{8}	L method and then the B method	2	0.50196
Reviser_{9}	L method and then the B method	3	0.50196
Reviser_{10}	B method and then the L method	0	0.51374
Reviser_{11}	B method and then the L method	1	0.50247
Reviser_{12}	B method and then the L method	2	0.50225
Reviser_{13}	B method and then the L method	3	0.50240