PLOS ONE

RESEARCH ARTICLE

Identification of RNA pseudouridine sites using deep learning approaches

Abu Zahid Bin Aziz[1]*, **Md. Al Mehedi Hasan**[1], **Jungpil Shin**[2]

1 Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Rajshahi, Bangladesh, **2** School of Computer Science and Engineering, University of Aizu, Aizuwakamatsu, Japan

* abuzahid.cse@gmail.com

OPEN ACCESS

Citation: Aziz AZB, Hasan M.AM, Shin J (2021) Identification of RNA pseudouridine sites using deep learning approaches. PLoS ONE 16(2): e0247511. https://doi.org/10.1371/journal.pone.0247511

Editor: Y-h. Taguchi, Chuo University, JAPAN

Received: September 30, 2020

Accepted: February 8, 2021

Published: February 23, 2021

Data Availability Statement: All relevant data are within the manuscript and its Supporting information files. You can also access the datasets here: http://103.99.176.239/ipseumulticnn/datasets.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Abstract

Pseudouridine(Ψ) is widely popular among various RNA modifications which have been confirmed to occur in rRNA, mRNA, tRNA, and nuclear/nucleolar RNA. Hence, identifying them has vital significance in academic research, drug development and gene therapies. Several laboratory techniques for Ψ identification have been introduced over the years. Although these techniques produce satisfactory results, they are costly, time-consuming and requires skilled experience. As the lengths of RNA sequences are getting longer day by day, an efficient method for identifying pseudouridine sites using computational approaches is very important. In this paper, we proposed a multi-channel convolution neural network using binary encoding. We employed k-fold cross-validation and grid search to tune the hyperparameters. We evaluated its performance in the independent datasets and found promising results. The results proved that our method can be used to identify pseudouridine sites for associated purposes. We have also implemented an easily accessible web server at http://103.99.176.239/ipseumulticnn/.

Introduction

Pseudouridine is the most common RNA modification observed in both prokaryotes and eukaryotes [1]. It is formed by the Ψ synthase enzyme which leads to the proof of its occurrence in various kinds of RNAs [2]. This enzyme separates the uridine residue's base from its sugar and rotates it 180˚ along the N3-C6 axis. The separation is completed by the subsequent reattachment of the base's 5'-carbon to the 1'-carbon of the sugar which results in the formation of an isomer of uridine, Pseudouridine [3]. Psudouridines play a vital role in both biological and genetic aspects of RNAs, especially for tRNA and rRNA. In case of rRNA, ribonucleoproteins are proved to be needed for pseudouridylation [4]. Psudouridines also work as a powerful mechanism for stabilizing tRNAs in both single and double-stranded regions [5]. Besides, different species present different prospects due to pseudouridines such as U6 snRNA mutants pseudouridylate at Ψ28 contributing to the filamentation growth program [6]. Furthermore, mRNAs incorporated with Ψ increase translation efficiency and

restrict innate immune response [7]. Therefore, an effective method for identifying Ψ sites has a vital significance.

Some laboratory techniques have been introduced over the years producing promising results. Carlile et al. introduced a transcriptome-wide pseudouridine-seq approach where Lovejoy et al. used induced termination of reverse transcription in their work [8, 9]. Furthermore, Schwartz et al. developed a transcriptome-wide quantitative mapping system to identify pseudouridine [10]. All of these systems are not only expensive but also time consuming. Moreover, skilled and experienced people are required to maintain these systems. That is why a more user-friendly method is required for identifying pseudouridine sites.

Despite the necessity, there are not many in silico methods to identify Ψ sites from nucleotide sequences. Li et al. introduced an SVM based web server which is, as far as we know, the first computational method to identify pseudouridine synthase (PUS) specific Ψ sites [11]. They extracted features from the nucleotides around the Ψ sites which provided good results for human and yeast samples. Later, their performance was improved by taking account of the chemical properties and the occurrence frequency density distributions of nucleotides by iRNA-Pseu, proposed by Chen et al. Their work also covered another species (*M. musculus*) [12]. He et al. proposed another web server named PseUI by using SVM [13]. First, they generated five different types of features and selected one by using the sequential forward feature selection approach.

Among the recent works, Tahir et al. implemented both machine learning and deep learning methods in their work [14]. They extracted features using n-gram and MMI in their SVM classifier and adopted a convolutional neural network (CNN) in their deep learning method, where the CNN classifier produced better performance. To the best of our knowledge, this is the only method that applied deep learning methodologies for this task so far. Using the best features from forward and incremental features, Liu et al. proposed a gradient boosting based method named XG-Pseu [15]. Furthermore, Mu et al. proposed an ensemble model named iPseu-Layer consisted of three machine learning techniques [16]. They employed random forest for the final prediction.

Many of the recent works used PseKNC for feature extraction [17–19]. That is why we wanted to adopt a CNN model which does not require any additional feature extraction technique. CNN has already proven to be useful in computer vision problems. Recently CNN has been producing satisfactory results in nucleotide-based datasets [14, 20–23]. In this work, we employed a CNN model where multiple channels of convolution layers with different sized filters are applied separately. Each of these convolution layers is then added to a max-pooling layer and concatenated. Our model yielded satisfactory results in the benchmark and independent datasets.

Materials and methods

Dataset collection

In this work, data were collected for three different species which are *H. sapiens*, *S. cerevis*iae and *M. musculus* represented by HS, SC and MM respectively. There were three benchmark datasets, HS_990, SC_628, and MM_944, one for each species for training purposes. Each of these datasets was balanced in terms of the number of samples. These are the same datasets used in Chen et al's work where they downloaded the RNA sequences from RMBase [12, 24]. In addition to these benchmark datasets, Chen et al. also gave two independent datasets, HS_200 and SC_200 for testing purposes which were for *H. sapiens* and *S. cerevisiae* but not for *M. musculus*. In both HS_200 and SC_200, the number of positive and negative samples

was equal. In the datasets, RNA sequences were formulated as shown:

$$R_{\xi}(U) = N_{-\xi}N_{-(\xi-1)} \cdots N_{-1}UN_{1} \cdots N_{+(\xi-1)}N_{\xi} \quad (1)$$

Here, U indicates "uridine", $N_{-\xi}$ denotes the ξ-th upstream nucleotide towards the 5' end and $N_{+\xi}$ denotes the ξ-th downstream nucleotide towards the 3' end from the central uridine. The value of ξ in HS_990 and MM_944 was 10 and 15 in SC_628.

Data preprocessing

Before applying the RNA sequences to our model, we needed to preprocess it first. There was only one step involved in the preprocessing step, which was binary "one-hot" encoding to convert our inputs into a 2-dimensional matrix. Each of the nucleotides of an input sequence was represented as a row vector where all the values are zero except for one value. We applied two separate techniques for this task.

General "one-hot" encoding. In this technique, the length of these row vectors was four which is the number of nucleotides found in RNA. Therefore, a sequence with N nucleotides would be a (N x 4) matrix. The 1D vectors we chose for the nucleotides were: ("A" = [1, 0, 0, 0], "U" = [0, 1, 0, 0],"C" = [0, 0, 1, 0], "G" = [0, 0, 0, 1]).

Merged-seq "one-hot" encoding. We also applied another technique by predicting secondary structures using RNAfold. Studies showed that secondary structure revealed critical structural features to detect Ψ sites [25]. We wanted to simulate this mechanism in computational methodologies. That's why we predicted the secondary structure and merged it with the original sequence. We called it "merged-seq". The secondary structure provided a new set of features and by merging with the original sequence we generated some more features. This technique provided good predictive performance in Zheng et al.'s pre-miRNA detection [23]. The encoding process is shown in Fig 1. The predicted secondary structure and merged sequence for each RNA sequence can be found in the supporting information or in this link: http://103.99.176.239/ipseumulticnn/datasets. The following steps were followed for this technique:

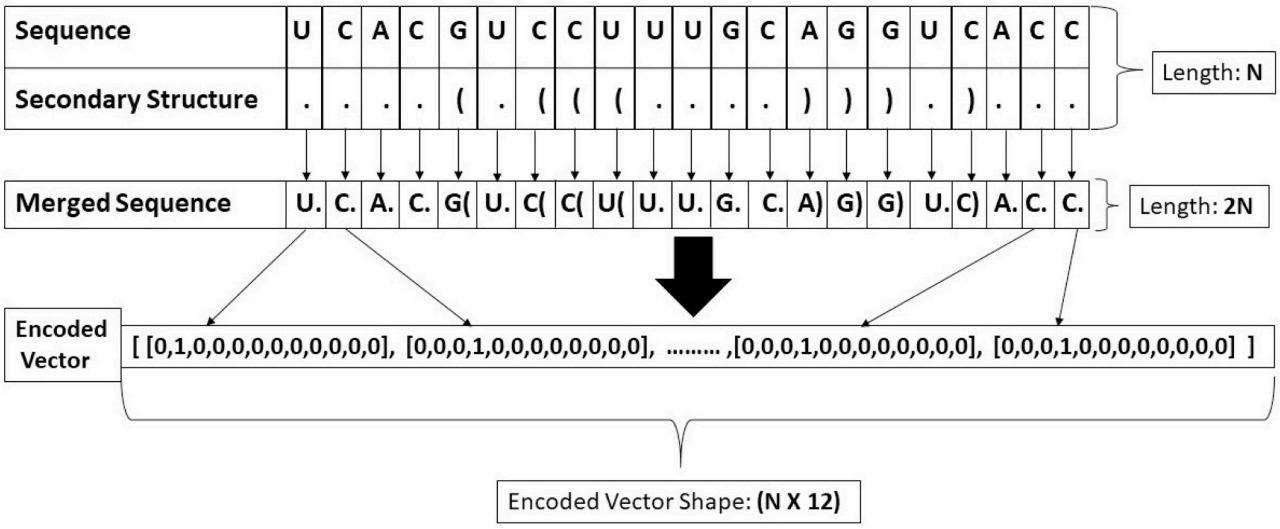

Fig 1. Vectorization process of the RNA sequences. Here, secondary structure was the predicted result from rnafold. Merged sequence was the pair of the original sequence and secondary structures. This merged sequence was then encoded using "one-hot" technique.

https://doi.org/10.1371/journal.pone.0247511.g001

- First, we predicted the secondary structure of the original sequence using RNAfold [26]. This structure had three types of symbols: ".", "(" and ")". The "(" and ")" indicated that a nucleotide at 5'-end and it's complimentary nucleotide at 3'-end is paired and the "." indicated that the nucleotide is not paired with any other nucleotide.
- Then, we formed a merged sequence that consisted of the original sequence and the secondary structure. This merged sequence had N pairs, N being the length of the sequence. The pairs were formed by taking one nucleotide from the original sequence and one symbol from the secondary structure.
- As there were four types of bases in RNA and three types of indicators in the secondary structure, we had 12 types of pairs in the merged sequences. After that, we encoded the pairs of the merged sequences using "one-hot" technique. So after encoding, an RNA sequence of length, N became a two-dimensional matrix of (N x 12). So for both the HS and MM datasets, the preprocessed inputs turned into a (21 x 12) matrix and for the SC dataset, the inputs turned into a (31 x 12) matrix.

CNN architecture

After preprocessing ("one-hot" encoding), the converted 2D inputs were fed to a convolutional neural network. Generally, in a CNN model, the inputs are connected to some convolution and max-pooling layers, followed by a couple of fully connected layers that are connected to the output layer. But in our case, the preprocessed inputs are fed to a multi-channel CNN model which has been very effective in various text classification tasks [27, 28]. The motivation behind this approach was to make sure a sequence is processed at different lengths at a time. In a sequential model, we can use only one size of filter for each convolution which may not extract the best features all the time. That's why we applied multiple channels of feature extraction operations (convolution and max-pooling) to the input sequence and integrated the features for better Ψ identification. A general architecture of our multi-channel model is shown in Fig 2.

Each channel of our model started with a convolution layer. We tuned the number of channels and the height of the filters of the convolution layer. The width of the filters remained unchanged. Each of these convolution layers was then connected to a max-pooling layer. Then, the max-pooling layers were concatenated together to combine the features extracted by the convolution and max-pooling layers. Next, the max-pooling layers are connected to the first fully connected layer which had 1024 nodes. After that, we employed dropout regularization to reduce the number of parameters. Then, the final layer was connected which gave a probability distribution of the classes. From the probability distribution, the final output was predicted.

The number of convolution layers was selected by applying k-fold cross-validation and grid search. Cross-validation also helped us to select the learning rate, dropout probability and height of the filters. Relu activation function was employed in every layer except for the last layer where the softmax activation function was used. This was the general structure of our model. Only the height of filters and the number of convolution layers varied for different datasets. We used categorical cross-entropy as the loss function. We also examined some well-known optimizers like Adam, Gradient descent, RMSprop etc. to minimize the loss function. Among these optimizers, Adam produced the best optimization.

Method evaluation metrics

Four evaluation metrics have been frequently used to evaluate the quality of a method in recent studies [29–31]. To calculate them, we required four parameters: true positive (TP), true

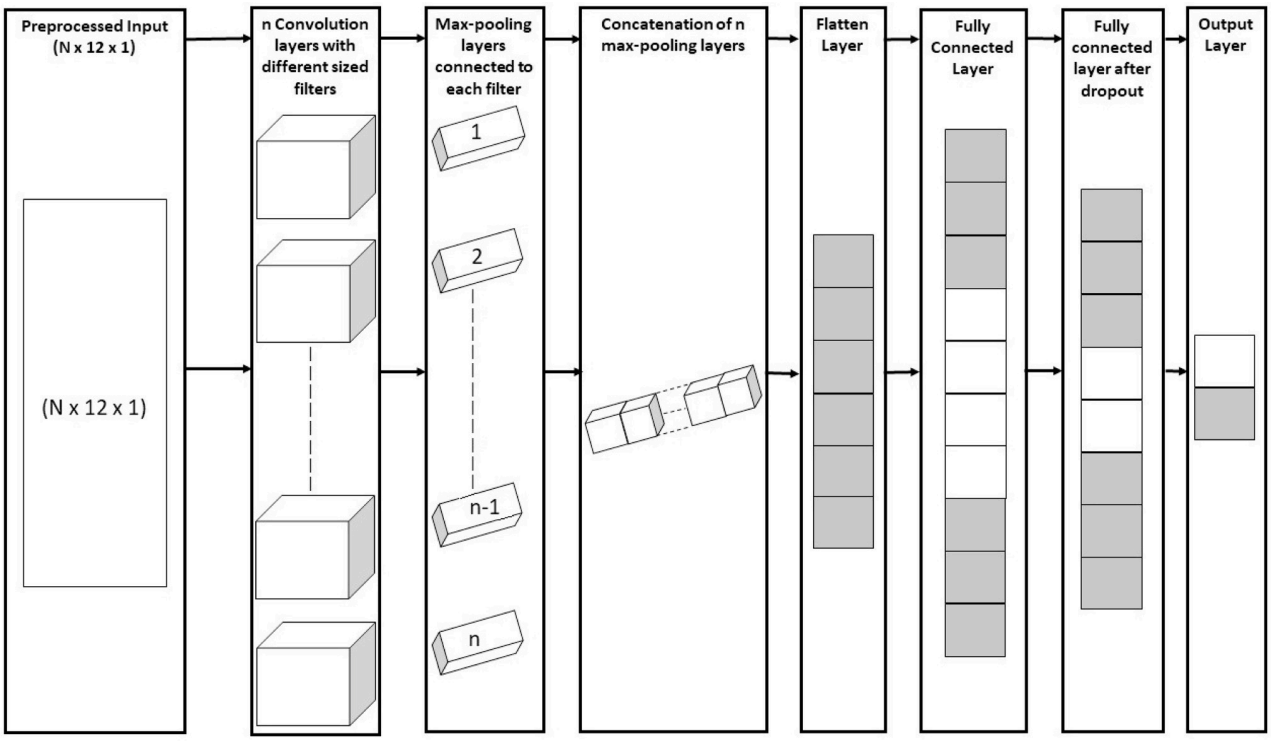

Fig 2. The architecture of our multi-stage CNN model.

https://doi.org/10.1371/journal.pone.0247511.g002

negative (TN), false positive (FT) and false negative (FN). The equations for the evaluation metrics are given below:

- Sensitivity (SN):

$$\text{Sensitivity} = \text{TP}/(\text{TP} + \text{FN}) \tag{2}$$

- Specificity(SP):

$$\text{Specificity} = \text{TN}/(\text{TN} + \text{FP}) \tag{3}$$

- Mathews Correlation Coefficient (MCC):

$$\text{MCC} = (\text{TP} * \text{TP} - \text{TP} * \text{FN})/[(\text{TP} + \text{FP}) * (\text{TP} + \text{FN}) * (\text{TN} + \text{FP}) * (\text{TN} + \text{FN})]^{1/2} \tag{4}$$

- Accuracy (AC):

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN}) \tag{5}$$

Results and discussions

Hyperparameter tuning

Hyperparameter tuning is vital to maximize a model's predictive performance. On the benchmark datasets, we tuned a number of hyperparameters to fine-tune our model. We did it in three separate steps using k-fold cross-validation and grid search. We used k = 5 to compare

Table 1. The ranges of values of the hyperparameters of the benchmark datasets.

Hyperparameters	Ranges of values	Selected values	
		SC_628	HS_990, MM_944
Batch size	[16, 32, 64, 128]	16	16
No. of epochs	[10, 50, 100, 200]	50	50
No. of channels	[5, 7, 9, 11]	7	9
Height of filters	[3, 5, 7, 9]	7	5
Learning rate	[0.0001, 0.0003, 0.0005, 0.0007, 0.001]	0.0005	0.0005
Dropout probabillity	[0.4, 0.45, 0.5, 0.55, 0.6]	0.4	0.5

https://doi.org/10.1371/journal.pone.0247511.t001

our results with the existing works as they also applied cross-validation using the same value. This implied that we divided the benchmark datasets into 5 folds. Among them, 4 folds were used for training and the remaining fold was used for testing that particular model.

First, we tuned the number of epochs and batch size. Then, we tuned the number of channels and the height of convolution filters using the values from the first step. The number of channels was tuned to investigate how many of them can be separately connected to the input layer to produce the best accuracy. Finally, using the values from the previous steps we tuned the learning rate and dropout probability. Grid search was adopted to select the values that produced the best result.

The considered and selected values for the hyperparameters are given in Table 1. We calculated accuracies for every possible combination of values of these hyperparameters and selected the ones that provided the highest accuracy. Merged-seq "one-hot" encoding was used when we tuned the hyperparameters. Then we trained our model by applying general and merged-seq "one-hot" encoding separately using the tuned values. As the shape of the inputs were different in the datasets, the selected values were not the same. They were used to train our model in the benchmark datasets and were evaluated by the independent data.

Training

Since the performance of CNN in computer vision and NLP tasks is well established, we wanted to use its classification success for biological sequence inputs. After the concatenation of the multiple convolution and max-pooling layers of our multi-stage CNN model, the number of parameters increased significantly. That is why to reduce the number of parameters, we employed dropout regularization after the first fully-connected layer. We also applied early stopping to make sure there was no overfitting in our model which means we stopped the training process if the validation loss did not improve after a certain consecutive epochs. After tuning the hyperparameters, we used the selected values to train our model in the benchmark datasets. The validation and training process were done in a core i5 laptop having NVIDIA 940m as GPU. Because of the grid search, the validation process took almost an hour to complete and the training process took about 2-3 minutes. We implemented our model using Keras Framework (2.2) with TensorFlow as backend.

We trained our model on the benchmark datasets using both general "one-hot" encoding and merged-seq "one-hot" encoding separately. Among these techniques, merged-seq "one-hot" encoding produced better performance. We employed the same model architecture in both cases using the tuned hyperparameters. We compared the performance of our models with the existing predictors (iRNA-PseU [12], PseUI [13], iPseU-CNN [14], XGboost [15], iPseU-Layer [16]) on the benchmark datasets which is shown in Table 2. From the table, we can see that our models produced satisfactory results. The training accuracy of our model was

Table 2. Comparison of the evaluation metrics with the existing predictors on the benchmark datasets.

Predictors	Benchmark Datasets											
	SC_628				HS_990				MM_944			
	AC(%)	SN(%)	SP(%)	MCC	AC(%)	SN(%)	SP(%)	MCC	AC(%)	SN(%)	SP(%)	MCC
iRNA-PseU [12]	64.49	64.65	64.33	0.29	60.40	61.01	59.80	0.21	69.07	73.31	64.83	0.38
PseUI [13]	65.13	62.74	67.52	0.30	64.24	64.85	63.64	0.28	70.44	74.58	66.31	0.41
iPseU-CNN [14]	68.15	66.84	69.45	0.37	66.68	65.0	68.78	0.34	71.81	74.79	69.11	0.44
XGboost [15]	68.15	66.84	69.45	0.37	65.44	63.64	67.24	0.31	72.03	76.48	67.57	0.45
iPseU-Layer [16]	89.34	84.68	93.76	0.79	79.70	71.18	88.22	0.60	80.08	77.92	81.82	0.60
ours(General)	81.50	76.0	87.0	0.61	77.0	79.5	74.5	0.53	80.50	86.0	75.0	0.55
ours(merged-seq)	85.85	88.29	83.37	0.72	78.83	85.61	72.07	0.59	77.23	76.62	77.60	0.54

https://doi.org/10.1371/journal.pone.0247511.t002

less than that of the iPseu-Layer because of their model's overfitting which is stated by Mu et al. That's why our model had better accuracy in the independent dataset despite having less accuracy in the benchmark datasets. Even though our model didn't achieve the most accuracy it had increased sensitivity by 4.26% and 20.27% in SC_628 and HS_990 datasets respectively. In our case, sensitivity represents the ratio of correctly identified Ψ sites to all sequences which had Ψ sites in reality. That means our models were able to predict actual Ψ sites quite well.

Comparative analysis

After training our models in the benchmark datasets, we examined its performance in the independent datasets by comparing the evaluation metrics with the existing predictors (iRNA-PseU [12], PseUI [13], iPseU-CNN [14], iPseU-Layer [16]). The findings are shown in Table 3. Similar to our training process, we tested for both general and merged-seq encoded models. Although both models produced better results than the existing predictors, the merged-seq encoded model outperformed them all.

Among the existing methods, iPseU-CNN produced the best performance in the SC_200 dataset. So, we calculated the amount of increased performance with respect to this classifier. In the SC_200 dataset, the specificity, accuracy and MCC was increased by 6.65%, 2% and 6.38% respectively for our general "one-hot" encoded model. But for our merged-seq "one-hot" encoded model, accuracy increased by 4.08%, sensitivity increased by 16.34% and MCC increased by 12.76%. Here, our merged-seq "one-hot" encoded model produced better performance.

In the HS_200 dataset, iPseU-Layer produced the best performance among the existing methods. In this dataset, our general "one-hot" encoded model had improved performance

Table 3. Comparison of the performance of our model with the existing predictors on the independent datasets.

Predictors	Independent Datasets							
	SC_200				HS_200			
	AC(%)	SN(%)	SP(%)	MCC	AC(%)	SN(%)	SP(%)	MCC
iRNA-PseU [12]	60.00	63.00	57.00	0.20	61.50	58.00	65.00	0.23
PseUI [13]	68.50	65.00	72.00	0.37	65.50	63.00	68.00	0.31
iPseU-CNN [14]	73.50	68.76	77.42	0.47	69.00	77.72	60.81	0.40
iPseU-Layer [16]	72.50	68.00	77.00	0.45	71.00	63.00	79.00	0.43
ours(General)	75.00	67.00	83.00	0.50	72.5	80.00	65.0	0.44
ours(merged-seq)	76.5	80.00	73.00	0.53	74.00	73.00	75.00	0.48

https://doi.org/10.1371/journal.pone.0247511.t003

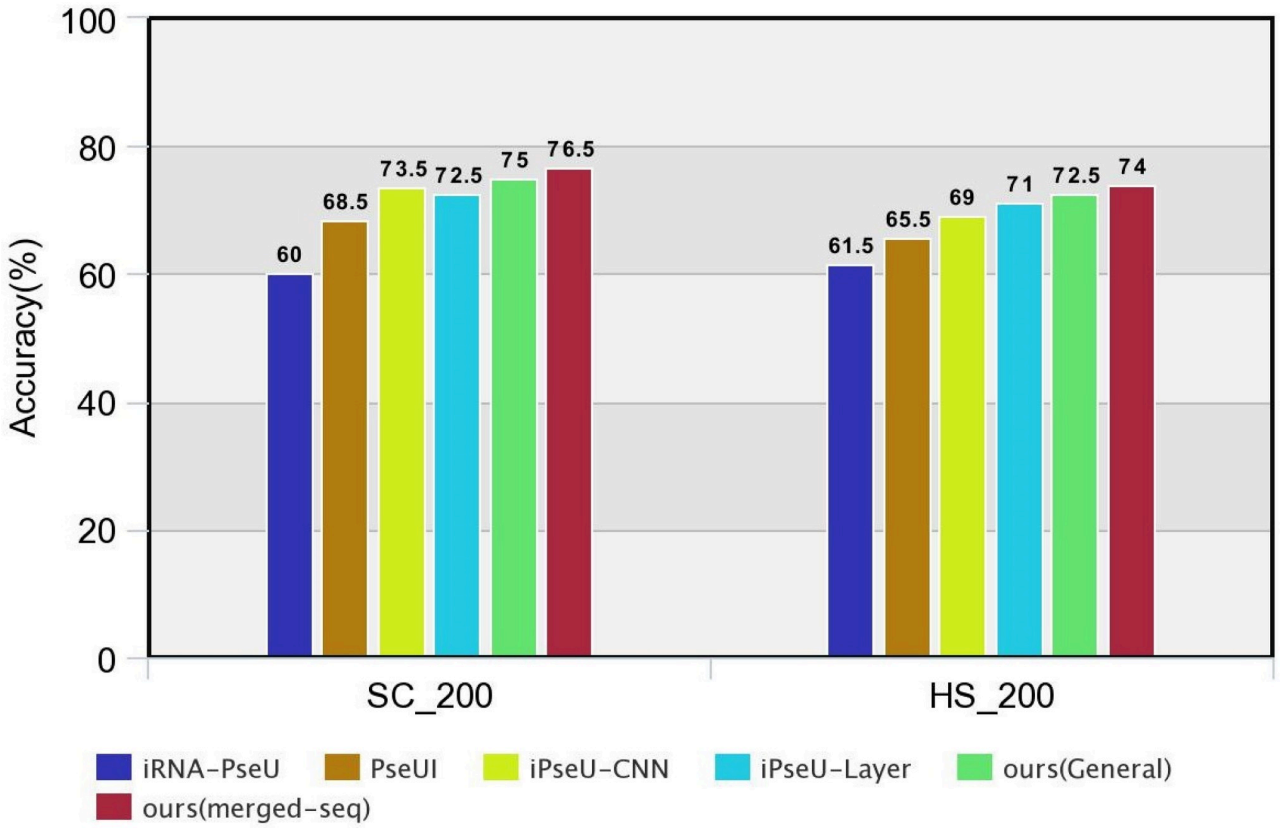

Fig 3. Graphical comparison of our models with the existing works in the independent datasets.

https://doi.org/10.1371/journal.pone.0247511.g003

in accuracy by 2.11%, sensitivity by 26.98% and MCC by 2.32%. On the other hand, our "merged-seq" encoded model outperformed iPseU-Layer in accuracy, sensitivity and MCC by 4.22%, 15.87% and 11.62% respectively. Similar to the SC_200 dataset, our merged-seq "one-hot" encoded model produced better evaluation metrics in this dataset.

Since we applied deep learning methodologies in our work, we wanted to produce better results than other deep learning methodologies. As far as we know, iPseU-CNN is the only available deep learning methodology that used the same datasets as us. Although their encoding is similar to our general encoding technique, they adopted a single-stage sequential model where our model had multi-stage architecture. Our both general and merged-seq "one-hot" encoded model had better accuracy, sensitivity and MCC than iPseU-CNN. So we can say that our models outperform the existing deep learning methodologies in every evaluation metric. To enhance the comparison, we provided a graphical comparison of our models with the state of the art methods in the independent datasets which is depicted in Fig 3.

We also plotted the receiver operating characteristic (ROC) curve on the benchmark and independent datasets to have a better understanding of our merged-seq "one-hot" encoded model. The plot is illustrated in Fig 4. ROC curve tells us how well a model can differentiate between classes. Our model achieved 0.88, 0.94 and 0.83 AUC (Area Under Curve) score on the HS_990, SC_628 and MM_944 benchmark datasets respectively. In case of the independent datasets, our model produced 0.77 and 0.78 on the HS_200 and SC_200 datasets respectively.

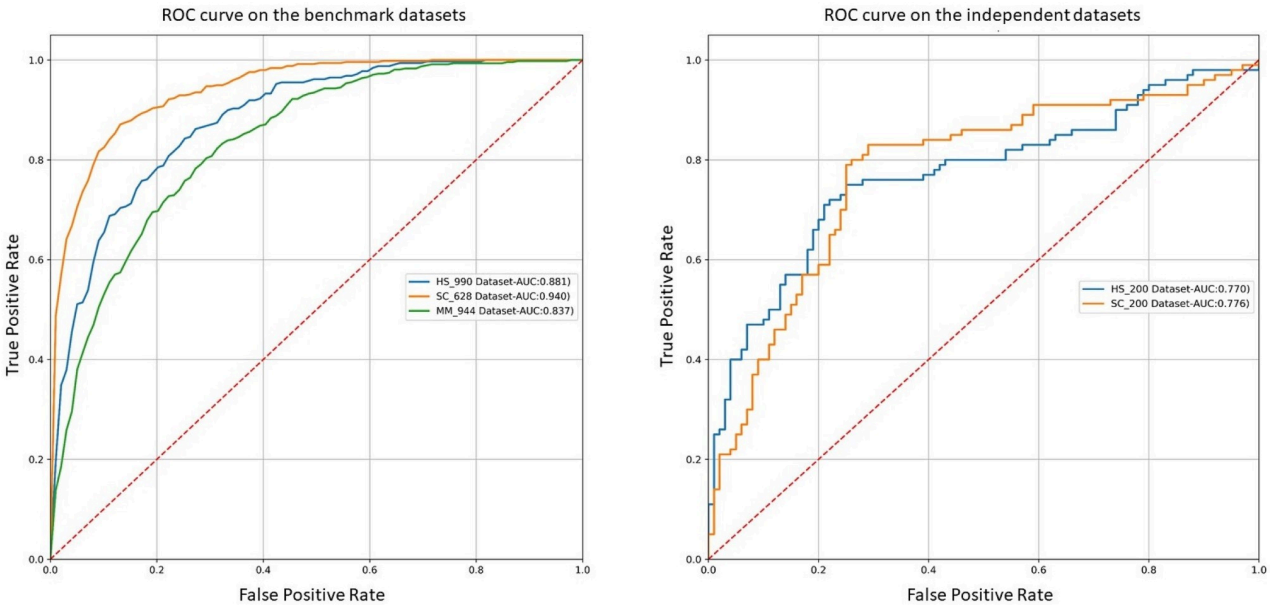

Fig 4. Illustration of the performance of our model in the benchmark and independent datasets using ROC curve.

https://doi.org/10.1371/journal.pone.0247511.g004

Visualization of the learned features

We visualized the outputs after the concatenation of the multi-stage convolution and max-pooling layers to gain further insights into the learned features for both general and merged-seq "one-hot" encoded models. We employed similar approaches used in recent CNN based works [32–34] to convert the kernel outputs into motifs. Then we used sequence logos to visualize and compare them with the logos generated from the independent datasets (Fig 5). The logos were generated in terms of probabilities (first three rows) and information contents (last three rows). From the sequence logos we can see that despite having some differences with the ground truth for the general "one-hot" encoding, the logos of the merged-seq "one-hot" encoding based models are quite similar to the ground truths. We can also observe from the information content logos that our models were able to capture the motifs around the central uracil(U) quite well for both datasets.

Discussion

Our merged-seq "one-hot" encoded classifier is already implemented and taken to the next stage by providing a user-friendly web server. In this work, we tried to tune only those hyper-parameters that can impact the performance of our classifier positively. Nevertheless, tuning other hyperparameters may result in improved performance. In our merged-seq "one-hot" encoding, the secondary structure of RNA played a vital role in improving the overall performance. We can further investigate how these new features are helping to improve the predictive performance. We also noticed some false positives for our merged-seq "one-hot" encoded model because of the secondary structure provided by RNAfold. We can investigate other secondary structure predictors in future for further improvements. We can also look for other encoding techniques of RNA sequences like Word2Vec other than "one-hot" encoding in the future. Furthermore, we can extend our work by applying our model to other species for Ψ site identification. Besides, there are other RNA modifications such as inosine (I), m3c, m5c etc. We can investigate whether our classifier can identify those sites from RNA sequences as

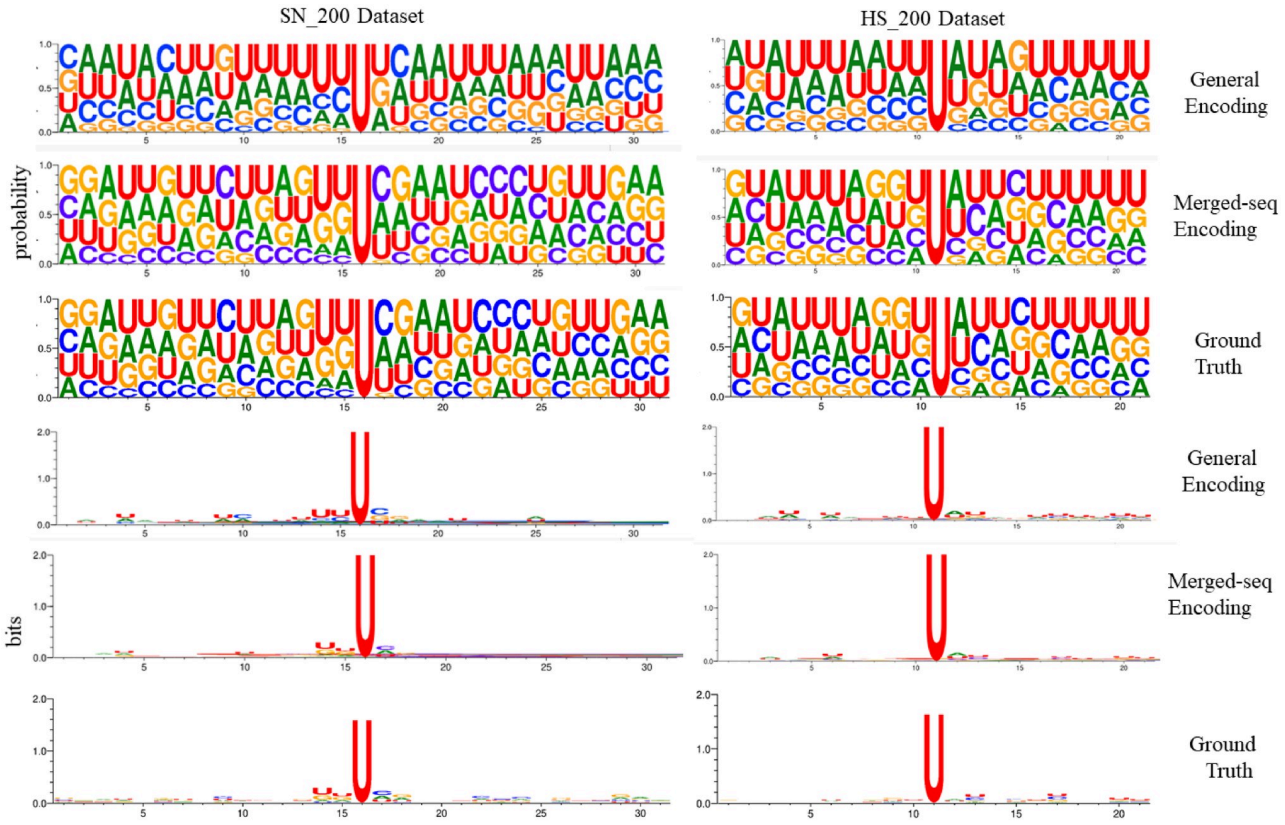

Fig 5. A comparison between the learned motifs of the general and merged-seq "one-hot" encoding. The ground truths were generated using Weblogo [35].

https://doi.org/10.1371/journal.pone.0247511.g005

well. Moreover, compared to the existing methods, our model produced the most accuracy in both HS_200 and SC_200 dataset.

Conclusion

The purpose of our work was to identify pseudouridine sites from RNA sequences using computational methods, in our case, a multi-stage convolutional neural network. After preprocessing our data using "one-hot" encoding, we adopted a CNN model having multiple convolution and max-pooling layers connected to the input layer individually, which was followed by a couple of fully-connected layers and an output layer. We applied k-fold cross-validation and grid search for hyperparameter tuning. We trained our model by using the selected values from tuning. Then we tested the performance of our model using the independent datasets and found 74% accuracy in the HS_200 dataset and 76.5% accuracy in the SC_200 dataset. It is projected that our classifier can become a helpful tool for identifying Ψ sites. We can also say that CNN can be used as an important method for classifying biological data.

Supporting information

S1 File. The benchmark and independent datasets with the secondary structure by RNAfold and merged sequence that we applied in this work.
(ZIP)

S2 File. Probabilities of each sequence of the independent datasets.
(ZIP)

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable and constructive comments which were very helpful for improving the contents of the manuscript.

Author Contributions

Methodology: Abu Zahid Bin Aziz.

Software: Abu Zahid Bin Aziz.

Supervision: Md. Al Mehedi Hasan, Jungpil Shin.

Visualization: Abu Zahid Bin Aziz.

Writing – original draft: Abu Zahid Bin Aziz.

Writing – review & editing: Md. Al Mehedi Hasan.

References

1. Hudson GA, Bloomingdale RJ, Znosko BM. Thermodynamic contribution and nearest-neighbor parameters of pseudouridine-adenosine base pairs in oligoribonucleotides. Rna. 2013; 19(11):1474–1482. https://doi.org/10.1261/rna.039610.113 PMID: 24062573
2. Ge J, Yu YT. RNA pseudouridylation: new insights into an old modification. Trends in biochemical sciences. 2013; 38(4):210–218. https://doi.org/10.1016/j.tibs.2013.01.002 PMID: 23391857
3. Charette M, Gray MW. Pseudouridine in RNA: what, where, how, and why. IUBMB life. 2000; 49 (5):341–351. https://doi.org/10.1080/152165400410182 PMID: 10902565
4. Bousquet-Antonelli C, Henry Y, Gélugne JP, Caizergues-Ferrer M, Kiss T. A small nucleolar RNP protein is required for pseudouridylation of eukaryotic ribosomal RNAs. The EMBO journal. 1997; 16 (15):4770–4776. https://doi.org/10.1093/emboj/16.15.4770 PMID: 9303321
5. Davis DR, Veltri CA, Nielsen L. An RNA model system for investigation of pseudouridine stabilization of the codon-anticodon interaction in tRNALys, tRNAHis and tRNATyr. Journal of Biomolecular Structure and Dynamics. 1998; 15(6):1121–1132. https://doi.org/10.1080/07391102.1998.10509006 PMID: 9669557
6. Basak A, Query CC. A pseudouridine residue in the spliceosome core is part of the filamentous growth program in yeast. Cell reports. 2014; 8(4):966–973. https://doi.org/10.1016/j.celrep.2014.07.004 PMID: 25127136
7. Karijolich J, Yu YT. The new era of RNA modification. RNA. 2015; 21(4):659–660. https://doi.org/10.1261/rna.049650.115 PMID: 25780180
8. Carlile TM, Rojas-Duran MF, Zinshteyn B, Shin H, Bartoli KM, Gilbert WV. Pseudouridine profiling reveals regulated mRNA pseudouridylation in yeast and human cells. Nature. 2014; 515(7525):143–146. https://doi.org/10.1038/nature13802 PMID: 25192136
9. Lovejoy AF, Riordan DP, Brown PO. Transcriptome-wide mapping of pseudouridines: pseudouridine synthases modify specific mRNAs in S. cerevisiae. PLoS One. 2014; 9(10). https://doi.org/10.1371/journal.pone.0110799 PMID: 25353621
10. Schwartz S, Bernstein DA, Mumbach MR, Jovanovic M, Herbst RH, León-Ricardo BX, et al. Transcriptome-wide mapping reveals widespread dynamic-regulated pseudouridylation of ncRNA and mRNA. Cell. 2014; 159(1):148–162. https://doi.org/10.1016/j.cell.2014.08.028 PMID: 25219674
11. Li YH, Zhang G, Cui Q. PPUS: a web server to predict PUS-specific pseudouridine sites. Bioinformatics. 2015; 31(20):3362–3364. https://doi.org/10.1093/bioinformatics/btv366 PMID: 26076723
12. Chen W, Tang H, Ye J, Lin H, Chou KC. iRNA-PseU: Identifying RNA pseudouridine sites. Molecular Therapy-Nucleic Acids. 2016; 5:e332. PMID: 28427142
13. He J, Fang T, Zhang Z, Huang B, Zhu X, Xiong Y. PseUI: pseudouridine sites identification based on RNA sequence information. BMC bioinformatics. 2018; 19(1):306. https://doi.org/10.1186/s12859-018-2321-0 PMID: 30157750

14. Tahir M, Tayara H, Chong KT. iPseU-CNN: Identifying RNA pseudouridine sites using convolutional neural networks. Molecular Therapy-Nucleic Acids. 2019; 16:463–470. https://doi.org/10.1016/j.omtn.2019.03.010 PMID: 31048185

15. Liu K, Chen W, Lin H. XG-PseU: an eXtreme Gradient Boosting based method for identifying pseudouridine sites. Molecular Genetics and Genomics. 2020; 295(1):13–21. https://doi.org/10.1007/s00438-019-01600-9 PMID: 31392406

16. Mu Y, Zhang R, Wang L, Liu X. iPseU-Layer: Identifying RNA Pseudouridine Sites Using Layered Ensemble Model. Interdisciplinary Sciences: Computational Life Sciences. 2020; p. 1–11. PMID: 32170573

17. Guo SH, Deng EZ, Xu LQ, Ding H, Lin H, Chen W, et al. iNuc-PseKNC: a sequence-based predictor for predicting nucleosome positioning in genomes with pseudo k-tuple nucleotide composition. Bioinformatics. 2014; 30(11):1522–1529. https://doi.org/10.1093/bioinformatics/btu083 PMID: 24504871

18. Feng P, Yang H, Ding H, Lin H, Chen W, Chou KC. iDNA6mA-PseKNC: Identifying DNA N6-methyladenosine sites by incorporating nucleotide physicochemical properties into PseKNC. Genomics. 2019; 111(1):96–102. https://doi.org/10.1016/j.ygeno.2018.01.005 PMID: 29360500

19. Yang H, Qiu WR, Liu G, Guo FB, Chen W, Chou KC, et al. iRSpot-Pse6NC: Identifying recombination spots in Saccharomyces cerevisiae by incorporating hexamer composition into general PseKNC. International journal of biological sciences. 2018; 14(8):883. https://doi.org/10.7150/ijbs.24616 PMID: 29989083

20. Yang B, Liu F, Ren C, Ouyang Z, Xie Z, Bo X, et al. BiRen: predicting enhancers with a deep-learning-based model using the DNA sequence alone. Bioinformatics. 2017; 33(13):1930–1936. https://doi.org/10.1093/bioinformatics/btx105 PMID: 28334114

21. Aoki G, Sakakibara Y. Convolutional neural networks for classification of alignments of non-coding RNA sequences. Bioinformatics. 2018; 34(13):i237–i244. https://doi.org/10.1093/bioinformatics/bty228 PMID: 29949978

22. Zheng X, Xu S, Zhang Y, Huang X. Nucleotide-level Convolutional Neural Networks for Pre-miRNA Classification. Scientific reports. 2019; 9(1):1–6. https://doi.org/10.1038/s41598-018-36946-4 PMID: 30679648

23. Zheng X, Fu X, Wang K, Wang M. Deep neural networks for human microRNA precursor detection. BMC bioinformatics. 2020; 21(1):1–7. https://doi.org/10.1186/s12859-020-3339-7 PMID: 31931701

24. Sun WJ, Li JH, Liu S, Wu J, Zhou H, Qu LH, et al. RMBase: a resource for decoding the landscape of RNA modifications from high-throughput sequencing data. Nucleic acids research. 2016; 44(D1):D259–D265. https://doi.org/10.1093/nar/gkv1036 PMID: 26464443

25. Carlile TM, Martinez NM, Schaening C, Su A, Bell TA, Zinshteyn B, et al. mRNA structure determines modification by pseudouridine synthase 1. Nature chemical biology. 2019; 15(10):966–974. https://doi.org/10.1038/s41589-019-0353-z PMID: 31477916

26. Gruber AR, Lorenz R, Bernhart SH, Neuböck R, Hofacker IL. The vienna RNA websuite. Nucleic acids research. 2008; 36(suppl_2):W70–W74. https://doi.org/10.1093/nar/gkn188 PMID: 18424795

27. Guo B, Zhang C, Liu J, Ma X. Improving text classification with weighted word embeddings via a multi-channel TextCNN model. Neurocomputing. 2019; 363:366–374. https://doi.org/10.1016/j.neucom.2019.07.052

28. Sun K, Li Y, Deng D, Li Y. Multi-channel CNN based inner-attention for compound sentence relation classification. IEEE Access. 2019; 7:141801–141809. https://doi.org/10.1109/ACCESS.2019.2943545

29. Cheng X, Lin WZ, Xiao X, Chou KC. pLoc_bal-mAnimal: predict subcellular localization of animal proteins by balancing training dataset and PseAAC. Bioinformatics. 2019; 35(3):398–406. https://doi.org/10.1093/bioinformatics/bty628 PMID: 30010789

30. Chen W, Ding H, Zhou X, Lin H, Chou KC. iRNA (m6A)-PseDNC: identifying N6-methyladenosine sites using pseudo dinucleotide composition. Analytical biochemistry. 2018; 561:59–65. https://doi.org/10.1016/j.ab.2018.09.002 PMID: 30201554

31. Qiu WR, Sun BQ, Xiao X, Xu ZC, Jia JH, Chou KC. iKcr-PseEns: Identify lysine crotonylation sites in histone proteins with pseudo components and ensemble classifier. Genomics. 2018; 110(5):239–246. https://doi.org/10.1016/j.ygeno.2017.10.008 PMID: 29107015

32. Quang D, Xie X. FactorNet: a deep learning framework for predicting cell type specific transcription factor binding from nucleotide-resolution sequential data. Methods. 2019; 166:40–47. https://doi.org/10.1016/j.ymeth.2019.03.020 PMID: 30922998

33. Alipanahi B, Delong A, Weirauch MT, Frey BJ. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. Nature biotechnology. 2015; 33(8):831–838. https://doi.org/10.1038/nbt.3300 PMID: 26213851

34. 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. https://doi.org/10.1093/nar/gkw226 PMID: 27084946

35. Crooks GE, Hon G, Chandonia JM, Brenner SE. WebLogo: a sequence logo generator. Genome research. 2004; 14(6):1188–1190. https://doi.org/10.1101/gr.849004 PMID: 15173120