# DSA4266 - Sense-making Case Analysis: Science and Technology

## Prediction of m6A RNA Modifications from Direct RNA–Seq Data

**Team Name:**
TeamRC4DSA

**Team Members:**
Liu Ernest Hin Yui (A0216284E)
Aaron Lee Wei Qi (A0216341R)
Leon Tan (A0216096A)
Poh Yu Jie (A0216055M)

**Page Count:**
6 (Excluding cover page, content page, Code Availability, and Reference section)

# Table of Contents

## Introduction and Problem Statement

The m6A modification is a prevalent RNA alteration crucial for regulating gene expression. Recent advancements in Direct RNA Sequencing provide a promising method to directly profile m6A sites across the transcriptome. This study addresses the challenge of accurately predicting m6A modifications from RNA-Seq data. We introduce a computational approach that utilizes machine learning to identify m6A modifications via a multiple-instance learning (MIL) framework.

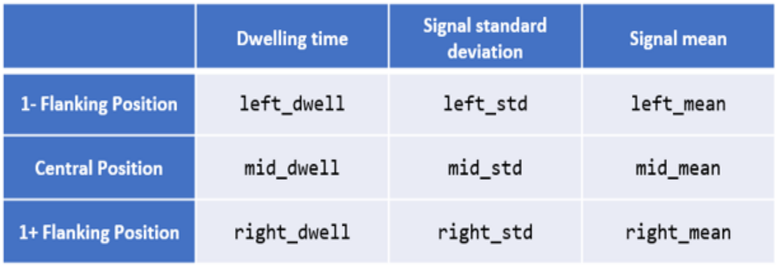
## Methods

Multiple Instance Learning (MIL) is a machine learning paradigm that handles scenarios where the training data is organized into bags, each containing multiple instances. Unlike traditional supervised learning, MIL operates under the assumption that the labels for the bags are known, but the specific labels for the individual instances within the bags are unknown. The goal of MIL is to learn a model that accurately classifies bags based on their collective content, making it useful in situations where instance-level labels are challenging to obtain.

Current approaches to tackling MIL problems can be broken down into three categories (Xiong D, Zhang Z, Wang T, Wang X., 2021). The first category is the so-called "Instance-space method", which focuses on performing classifications on individual instances, rather than on the bag level. Methods in this class typically ignore the spread of data points within each bag. The second category is "Bag-space methods", which focus on the similarity or differences between bags. The third category is "Embedded-space methods", which works by engineering several features that describe each bag before performing classifications on these engineered features. Our group has attempted methods from two of these categories, which will be later explained.

### Terminology

For convenience, we developed key terminology which shall be used for the remainder of this report. An "instance" refers to the data representing a single read, whereas a "bag" refers to a collection of reads with the same gene ID, transcript ID, and transcript position. Furthermore, the nine instance-level features will be referred to using the terms given in Table 1.



| | Dwelling time | Signal standard deviation | Signal mean |
|---|---|---|---|
| 1- Flanking Position | left_dwell | left_std | left_mean |
| Central Position | mid_dwell | mid_std | mid_mean |
| 1+ Flanking Position | right_dwell | right_std | right_mean |

Table 1: Terminology for the nine instance-level features

### Method 1: Autoencoder method (Instance-space method)

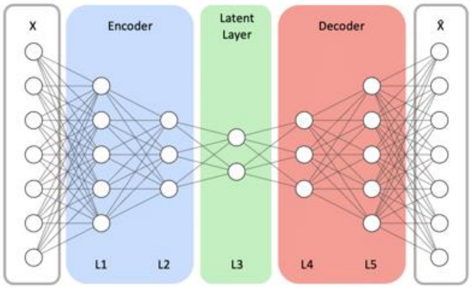


Figure 1: Generic autoencoder architecture

We have adopted the autoencoder anomaly model in our study to identify m6A modifications. Speaking in the language of anomaly models, we shall use the term "normal data point" to refer to an instance that is not m6A-modified, and the term "anomalous data point" to refer to an instance that is.

The objective of the autoencoder is to produce a reconstruction $\hat{x}$ of an input instance $x$ (L. Bergamin, T. Carraro, M. Polato, F. Aiolli, 2022). Due to its bottlenecked architecture, every forward pass through the model leads to a loss of information in the encoder as data from the original dimension is compressed into a lower dimension. After that, the remaining information in the lower dimension is transformed back to the data space by the decoder. This trait of "lossy compression" is useful for training autoencoders, as it forces them to learn to encode important information about the input instances during training to make their output reconstructions $\hat{x}$ as close to the inputs as possible. The reconstruction quality is quantified by $||x - \hat{x}||_2$.

Thus, autoencoder anomaly models are trained only with normal data points so that they only know how to accurately reconstruct normal data points. In other words, $||x_{normal} - \hat{x}_{normal}||_2$ is small. If tasked to reconstruct an anomalous data point after training, the autoencoder should produce a bad reconstruction as the anomalous data point is likely to come from a different distribution than the normal data points. This causes $||x_{anomalous} - \hat{x}_{anomalous}||_2$ to be large. Classification is then typically done by setting a threshold $\boldsymbol{\tau}$ to $||x - \hat{x}||_2$. We classify a data point to be anomalous if $||x - \hat{x}||_2 > \tau$, and normal otherwise.

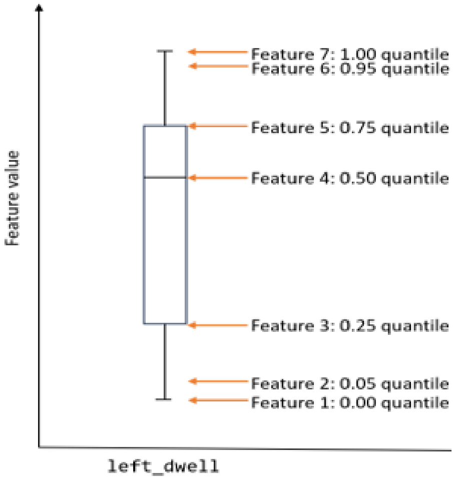
In identifying m6A modifications, we trained the autoencoder with instances from bags labeled 0 (no anomalous instances). Since our task is not to generate a class for each bag $b_i$ but to generate an estimate of $P(b_i\ has\ an\ anomalous\ instance)$, we estimated this using calculation:

$$P(b_i\ has\ an\ anomalous\ instance) = 1 - \prod_{instance\ j \in b_i} P(instance\ j\ is\ not\ anomalous)$$

To generate an estimate of $P(instance\ j\ is\ not\ anomalous)$, we fit a logistic regression model on the list of reconstruction errors for normal and anomalous instances. The labels for instances that come from normal bags is labelled 0, while those that come from anomalous bags are labelled as 1.

### Method 2: Random Forest (Embedding-space method)

Embedding-space methods involve the engineering of bag-level features before performing classification on these engineered features. Our method achieves this simply by calculating the 0, 0.05, 0.25, 0.5, 0.75, 0.95, and 1.00 quantiles for each of the nine features individually. Figure 2 illustrates the engineered bag-level features from the `left_dwell` instance-level feature. Note that this is repeated for all nine instance-level features, for a grand total of 63 bag-level features.



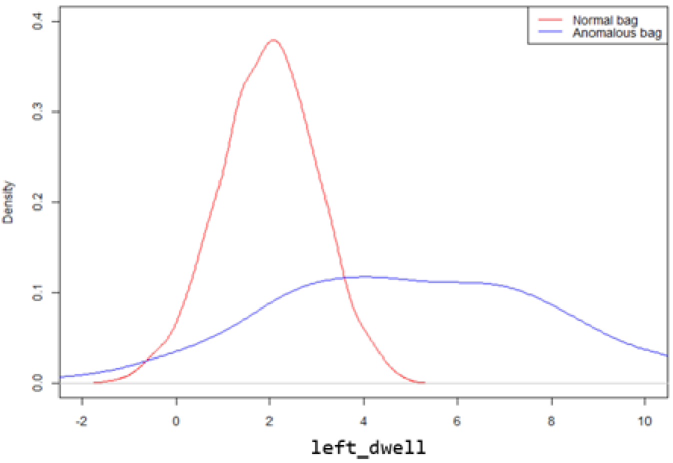


Figure 2 (left) Illustration of generated bag-level feature from one instance-level feature

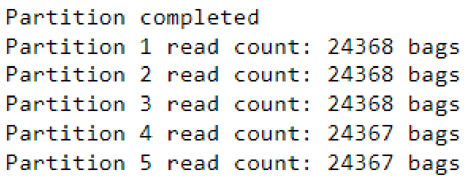
Figure 3 (right): Illustration of different `left_dwell` distributions for normal and anomalous bags

The intuition behind this is that the distribution of some instance-level features might be different in normal bags from anomalous bags, causing the bag-level features generated to be different as well. Figure 3 provides a portrayal of this, though take note that this simply illustrates the idea above and is not an accurate depiction of the observed data.

After feature engineering, we simply performed classical supervised classification on the generated bag-level features and bag labels. The model chosen at this stage to estimate $P(b_i\ has\ an\ anomalous\ instance)$ was the random forest regressor model, with 1000 trees.

## Model Evaluation

Due to the poor performance of the autoencoder model during the intermediate submission, we decided to discontinue the experimentation of that model. As for the random forest model, we evaluated it using 5-fold cross validation. To better estimate how the model will perform on unseen genes, we split dataset0 into 5 approximately equal folds according to the gene ID. This means that if fold $i$ contains bags from gene $g$, then there will not exist any bag from another fold $j \neq i$ which also comes from gene $g$. The number of bags in each fold is shown in the image below.



```
Partition completed
Partition 1 read count: 24368 bags
Partition 2 read count: 24368 bags
Partition 3 read count: 24368 bags
Partition 4 read count: 24367 bags
Partition 5 read count: 24367 bags
```

Using these five folds, we performed cross validation, which yielded the results shown in the Figures 4 and 5 below. From the images, we see that the model performs relatively well in terms of its AUROC score. The performance of the model also does not seem to fluctuate much, suggesting low estimation variance. However, judging from the Precision-Recall plot, there seems to be room for improvement, which is discussed in the "Discussion" section below.

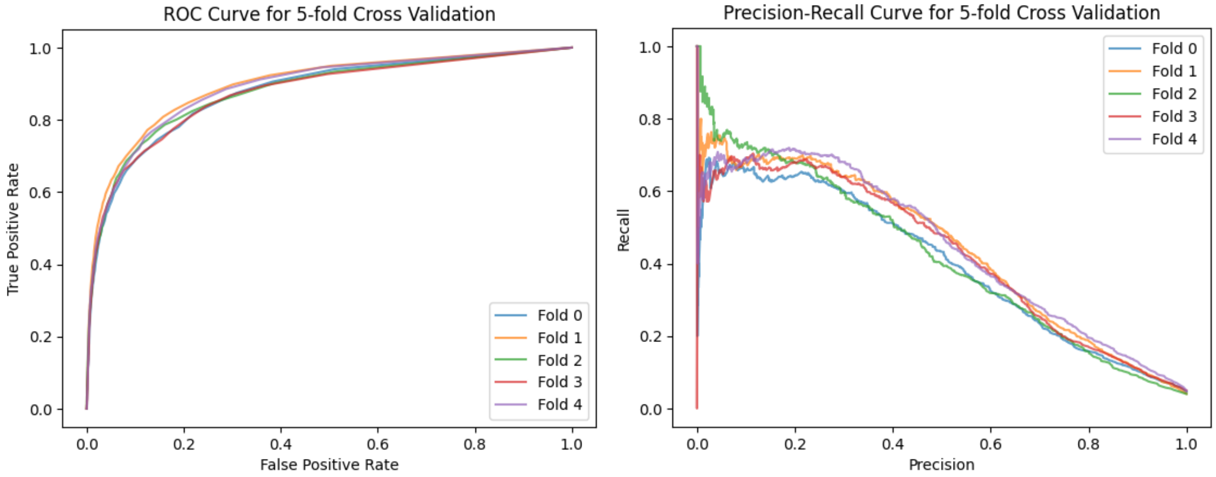
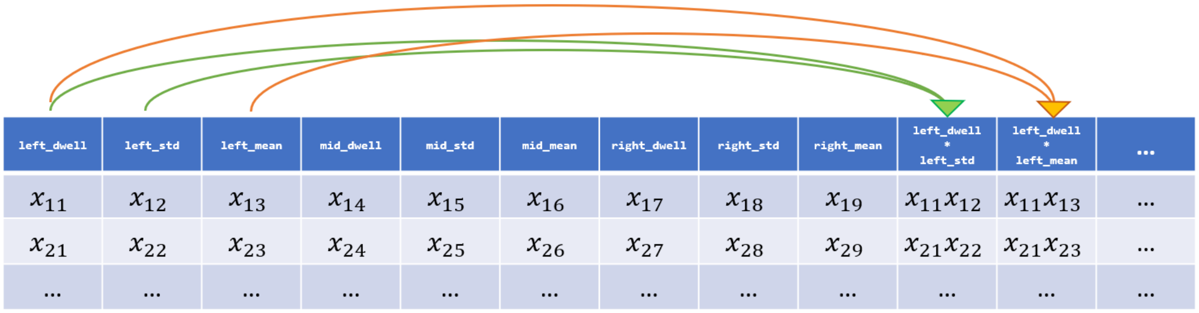


Figure 4 (left): ROC Curve, Average AUROC: 0.88331; Figure 5 (right): Average AU-Precision-Recall Curve: 0.4267

## Discussion

One obvious limitation of our random forest model can be seen through the lens of the Analysis of Variance. From the way that the bag-level features are generated, one may observe that the model completely disregards any interaction effects between the instance-level features. This is because only the quantiles of individual instance-level features are considered. It is possible for m6A modifications to be more easily detected if we also take the interaction effects between the instance-level features into account.

Hence, a natural improvement to the current model could be to consider the interaction effects between instance-level features. This could be done by multiplying pairwise instance-level features together as a proxy of the interaction effect. Figure 6 illustrates this idea. From this, we will be left with $9 + \frac{9*8}{2} = 45$ instance-level features. After calculating the 0, 0.05, 0.25, 0.5, 0.75, 0.95 and 1.00 quantiles for each of the 45 instance-level features, we will have 315 bag-level features, which would then be individually normalized using a min-max scaler. While this may seem like many features, given the fact that the number of bags in dataset0 is in the hundred-thousands, it is unlikely that we will suffer from the curse of dimensionality.



| left_dwell | left_std | left_mean | mid_dwell | mid_std | mid_mean | right_dwell | right_std | right_mean | left_dwell * left_std | left_dwell * left_mean | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_{11}$ | $x_{12}$ | $x_{13}$ | $x_{14}$ | $x_{15}$ | $x_{16}$ | $x_{17}$ | $x_{18}$ | $x_{19}$ | $x_{11}x_{12}$ | $x_{11}x_{13}$ | ... |
| $x_{21}$ | $x_{22}$ | $x_{23}$ | $x_{24}$ | $x_{25}$ | $x_{26}$ | $x_{27}$ | $x_{28}$ | $x_{29}$ | $x_{21}x_{22}$ | $x_{21}x_{23}$ | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

Figure 6: Illustration of generating proxies for interaction effect

As for the autoencoder model, its limitation stems from the way that we estimate the probabilities for each instance. This is because in fitting the logistic regression model, we simply labelled all instances that come from anomalous bags with a value of 1. This might not be the best approach since anomalous bags will contain instances without m6A modifications. Assigning such instances with a label of 1 would not be accurate as these instances should have a low probability of being m6A-modified. Besides this, there could have also been outliers in the training data that should have been removed. A fix to this could be to perform outlier removal techniques before training, such as the Mahalanobis Distance outlier identification technique.

## Application on SG-NEx Data

Predictions from our model have been made on the data available from the SG-NEx project. You can find the predictions in the `/data/curated` directory from our public GitHub repository linked in the section below.

### Threshold Determination

Based on the prediction scores, we deemed it prudent to create a 5-fold precision and recall plot against various thresholds, as depicted in Figure 7. The aim was to identify the optimal threshold for categorizing the bags as 1 or 0. We prioritized precision over recall to ensure that users have confidence in our model's mutation identifications, as lower precision might erode trust in the predictions. At the same time, even though lower recall score might lead to more false negatives, it is likely that if many samples were taken from a cancerous cell line, at least a few of them would be identified as containing m6A modifications, which would warrant further investigation by healthcare professionals. Ultimately, we settled on the 0.5 threshold since that is the point where precision levels off, indicating the best trade-off between precision and recall.

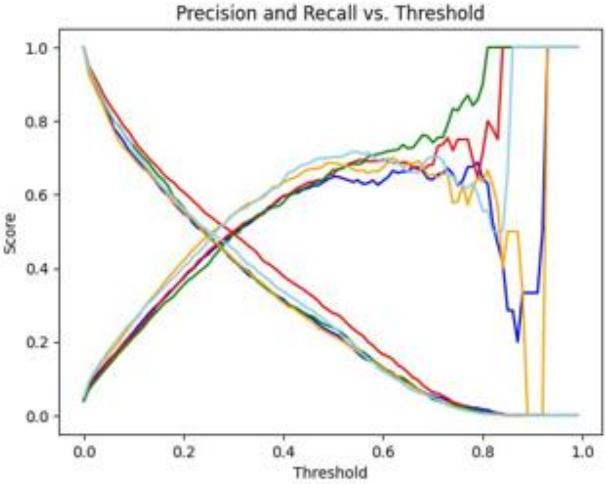
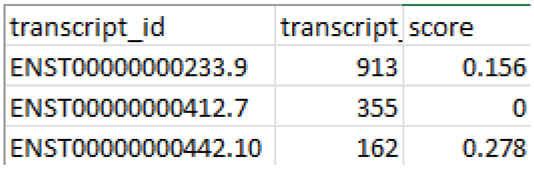


Figure 7: Precision & Recall against Threshold for different CV Models

### Data Cleaning

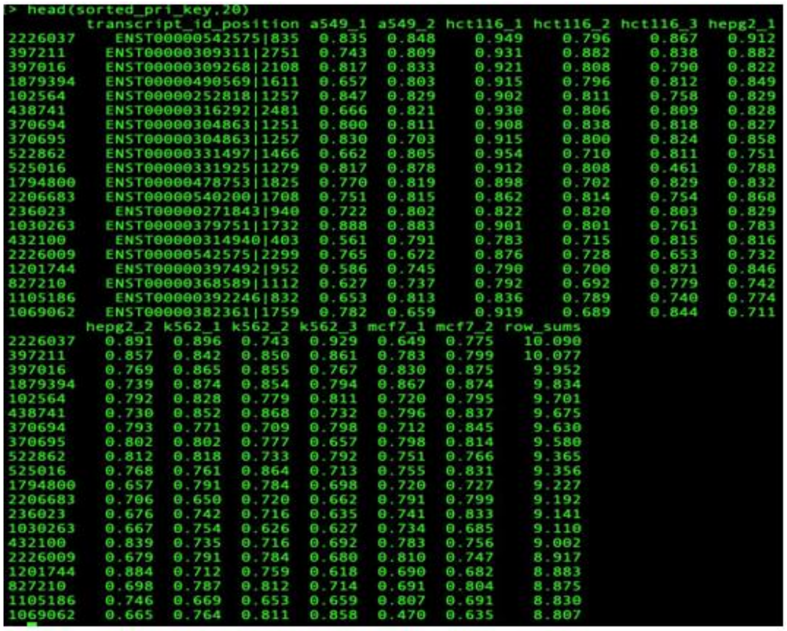
Before performing analysis, we noticed some datasets had different formats of transcript IDs than the rest. Notably, only the datasets for the MCF7 cell line formatted transcript ID in the "ENSTXXXXXXXXXXX.X" format as shown by the example on the right, whilst the other datasets formatted it using the "ENSTXXXXXXXXXXX" format. We standardized the formatting by removing the ".X" part of the transcript ID string from the MCF7 datasets. We then used our model to make predictions on the cleaned dataset. Here are some of our key findings from the result.



| transcript_id | transcript_ | score |
|---|---|---|
| ENST00000000233.9 | 913 | 0.156 |
| ENST00000000412.7 | 355 | 0 |
| ENST00000000442.10 | 162 | 0.278 |

### Analysis 1 – Susceptibility of Transcript IDs and Positions to m6A Modification

The aim of this analysis is to identify specific transcript IDs and positions that may be more susceptible to m6A modification. To do so, we need to examine whether such modifications occur consistently across all cell lines. This allows us to draw conclusions that the findings can be broadly applicable to other unobserved cell lines.



```
> head(sorted_pri_key,20)
        transcript_id_position a549_1 a549_2 hct116_1 hct116_2 hct116_3 hepg2_1
2226037    ENST00000542575|835  0.835  0.848    0.949    0.796    0.867   0.912
397211    ENST00000309311|2751  0.743  0.809    0.931    0.882    0.838   0.882
397016    ENST00000309268|2108  0.817  0.833    0.921    0.808    0.790   0.822
1879394   ENST00000490569|1611  0.657  0.803    0.915    0.796    0.812   0.849
102564    ENST00000252818|1257  0.847  0.829    0.902    0.811    0.758   0.829
438741    ENST00000316292|2481  0.666  0.821    0.930    0.806    0.809   0.828
370694    ENST00000304863|1251  0.800  0.811    0.908    0.838    0.818   0.827
370695    ENST00000304863|1257  0.830  0.703    0.915    0.800    0.824   0.858
522862    ENST00000331497|1466  0.662  0.805    0.954    0.710    0.811   0.751
525016    ENST00000331925|1279  0.817  0.878    0.912    0.808    0.461   0.788
1794800   ENST00000478753|1825  0.770  0.819    0.898    0.702    0.829   0.832
2206683   ENST00000540200|1708  0.751  0.815    0.862    0.814    0.754   0.868
236023     ENST00000271843|940  0.722  0.802    0.822    0.820    0.803   0.829
1030263   ENST00000379751|1732  0.888  0.883    0.901    0.801    0.761   0.783
432100     ENST00000314940|403  0.561  0.791    0.783    0.715    0.815   0.816
2226009   ENST00000542575|2299  0.765  0.672    0.876    0.728    0.653   0.732
1201744    ENST00000397492|952  0.586  0.745    0.790    0.700    0.871   0.846
827210    ENST00000368589|1112  0.627  0.737    0.792    0.692    0.779   0.742
1105186    ENST00000392246|832  0.653  0.813    0.836    0.789    0.740   0.774
1069062   ENST00000382361|1759  0.782  0.659    0.919    0.689    0.844   0.711
        hepg2_2 k562_1 k562_2 k562_3 mcf7_1 mcf7_2 row_sums
2226037   0.891  0.896  0.743  0.929  0.649  0.775   10.090
397211    0.857  0.842  0.850  0.861  0.783  0.799   10.077
397016    0.769  0.865  0.855  0.767  0.830  0.875    9.952
1879394   0.739  0.874  0.854  0.794  0.867  0.874    9.834
102564    0.792  0.828  0.779  0.811  0.720  0.795    9.701
438741    0.730  0.852  0.868  0.732  0.796  0.837    9.675
370694    0.793  0.771  0.709  0.798  0.712  0.845    9.630
370695    0.802  0.802  0.777  0.657  0.798  0.814    9.580
522862    0.812  0.818  0.733  0.792  0.751  0.766    9.365
525016    0.768  0.761  0.864  0.713  0.755  0.831    9.356
1794800   0.657  0.791  0.784  0.698  0.720  0.727    9.227
2206683   0.706  0.650  0.720  0.662  0.791  0.799    9.192
236023    0.676  0.742  0.716  0.635  0.741  0.833    9.141
1030263   0.667  0.754  0.626  0.627  0.734  0.685    9.110
432100    0.839  0.735  0.716  0.692  0.783  0.756    9.002
2226009   0.679  0.791  0.784  0.680  0.810  0.747    8.917
1201744   0.884  0.712  0.759  0.618  0.690  0.682    8.883
827210    0.698  0.787  0.812  0.714  0.691  0.804    8.875
1105186   0.746  0.669  0.653  0.659  0.807  0.691    8.830
1069062   0.665  0.764  0.811  0.858  0.470  0.635    8.807
```

Figure 8: List of Transcript IDs and Positions, Sorted by Row-wise Sums.

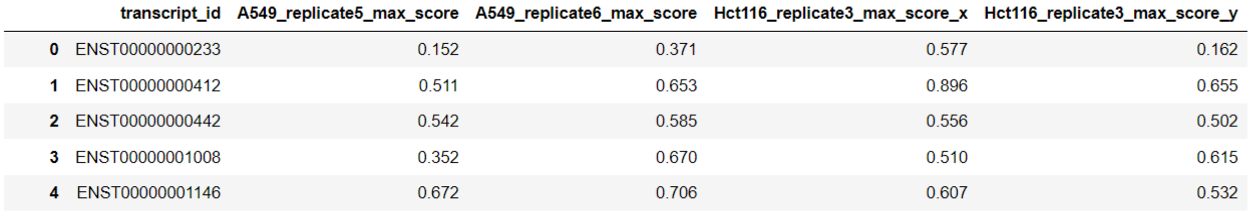
We began first obtained the model's predictions for all 12 datasets provided by SG-NEx. Subsequently, we joined the 12 datasets based on their transcript ID and transcript position, which are the primary keys of all datasets. After this, we calculated the row-wise sums and noticed that certain transcript IDs and positions exhibited significantly high row sums. A snippet of this resulting matrix is shown in Figure 8. This suggests a strong likelihood of mutations occurring at these specific transcript IDs and positions, regardless of the cell line. By identifying these transcript IDs and positions, we can pinpoint sites with extensive m6A modifications.

Finally, we categorized each bag. Predictions above the established 0.5 threshold were labeled as having m6A modifications, whereas those below were classified as lacking such modifications. We followed the same procedure to calculate row sums and identified 141 bags with a row-wise sum of 12. This indicates that our model classified these 141 bags as having m6A modifications across all 12 datasets. Consequently, we infer that these specific 141 <transcript ID, transcript positions> combinations are particularly susceptible to m6A modifications. We recommend conducting further tests to see if these specific transcript IDs and positions are detected in future RNA sequencing data, regardless of the cell lines involved. A copy of these 141 transcript IDs and positions is available here.

### Analysis 2 - Identifier Transcripts

The objective of this analysis is to identify transcripts that reliably indicate the presence or absence of m6A modifications, independent of the cell line type. In simpler terms, we aim to ascertain whether there are any transcripts (denoted as $t$) that consistently exhibit a high probability of m6A detection ($P(Detection\ of\ m6A\ in\ t \mid Actual\ m6A\ in\ t)$) regardless of the specific cell line from which $t$ is extracted. For this analysis, these transcripts shall be termed “identifier transcripts". Likewise, the term “m6A score” will refer to the probability that a transcript $t$ has an m6A mutation.

To achieve this, one safe but key assumption we had to make was that **all cell lines available from the SG-NEx project had m6A modifications**. We first used the model’s predictions on the datasets to estimate the m6A score for each bag $b$. Next, we aggregated the maximum m6A score per cell line and transcript ID to yield a table with the following structure:



| | transcript_id | A549_replicate5_max_score | A549_replicate6_max_score | Hct116_replicate3_max_score_x | Hct116_replicate3_max_score_y |
|---|---|---|---|---|---|
| 0 | ENST00000000233 | 0.152 | 0.371 | 0.577 | 0.162 |
| 1 | ENST00000000412 | 0.511 | 0.653 | 0.896 | 0.655 |
| 2 | ENST00000000442 | 0.542 | 0.585 | 0.556 | 0.502 |
| 3 | ENST00000001008 | 0.352 | 0.670 | 0.510 | 0.615 |
| 4 | ENST00000001146 | 0.672 | 0.706 | 0.607 | 0.532 |

Figure 9: Example table structure (note that only 4 out of 12 SG-NEx datasets are shown here)

We took the maximum score across all transcript positions because we wanted to identify the most significant m6A score for each combination of cell line and transcript ID. A high maximum m6A score indicates strong evidence of the presence of an m6A modification in some position along that transcript.

Lastly, we took the minimum m6A score of each transcript across all 12 samples. This is done by calculating the minimum across each row in the table above. The intuition behind taking the minimum is because identifier transcripts would have high m6A scores across all cell lines. Consequently, the minimum m6A score of an identifier transcript across all cell lines will also be high.

To determine the identifier transcripts, we ranked the transcripts by their resulting m6A score and attained the following plot which shows the transcripts with the top m6A scores:

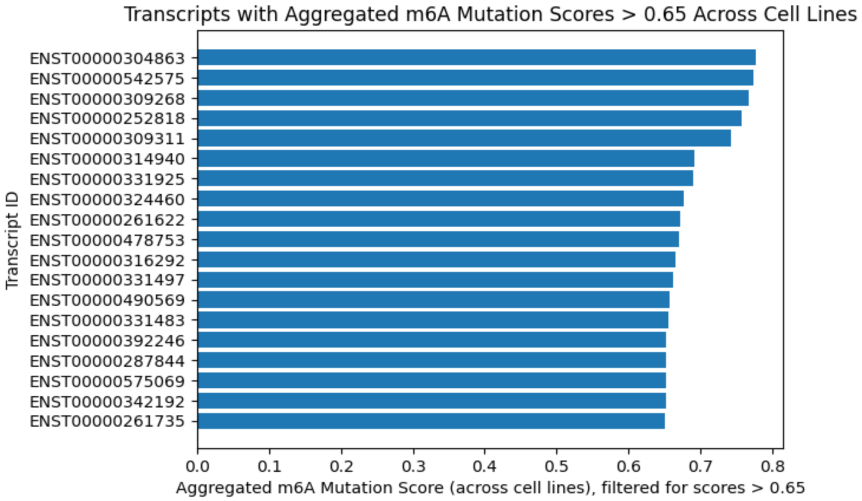


Figure 10: Top identifier transcript candidates

As shown in the bar chart above, the top-5 transcripts ENST00000304863, ENST00000542575, ENST00000309268, ENST00000252818, and ENST00000309311 had scores above 0.7. This means that regardless of cell line, these transcripts showed high probabilities of having m6A modifications. This suggests that these transcripts may be good contenders for detecting cancers in the five types of cell lines made available by the SG-NEx project.

### Summary of our Key Findings

A closer examination of the results in Analysis 2 revealed that the top 10 transcript IDs with high aggregated mutation scores are all found in the list of transcript IDs and positions available in Analysis 1. This leads us to the conclusion that certain transcript IDs are important for indicating the presence of m6A modifications, and certain positions within said transcripts are more susceptible to m6A modifications.

## Summary

We present two MIL-based machine learning models aimed at the identification of m6A sites using RNA-Seq data. While the autoencoder model still has much room for improvement, our embedded-space model has seen decent results on datasets 0, 1, and 2. In addition, we present two key findings using the model's predictions on the datasets made available by the SG-NEx project. The first identifies specific positions on specific transcripts that are more susceptible to m6A modification, while the second identifies five transcripts that could be used to detect cancers in the five cell lines – A549, HCT116, HepG2, K562, and MCF7.

## Code Availability

You may refer to our public GitHub repository which contains the code developed for the project. The public Docker repository which houses the image to create the model container is also provided below.

GitHub: https://github.com/leontanwh/teamrc4dsa
Docker: https://hub.docker.com/repository/docker/elhy1999/quantrf-bagm6a/general

For student testers, the `/deployment` directory within the repository will be of particular relevance to you. You can also refer to this video tutorial we created which goes over how to create the container from a fresh AWS EC2 instance (m6a.large or larger). Also, note that you **do not need to clone** the repository for testing, but you are certainly welcome to. However, you will need to install Git LFS to clone it. Instructions to do so are available in the root directory of the repository.

## References

Xiong, D., Zhang, Z., Wang, T., & Wang, X. (2021). A comparative study of multiple instance learning methods for cancer detection using T-cell receptor sequences. Computational and Structural Biotechnology Journal, 19, 3255–3268. https://doi.org/10.1016/j.csbj.2021.05.038

Bergamin, L. (2022, April 26). Novel Applications for VAE-based Anomaly Detection Systems. arXiv.org. https://doi.org/10.48550/arXiv.2204.12577

Chen, Ying, et al. "A systematic benchmark of Nanopore long read RNA sequencing for transcript level analysis in human cell lines." bioRxiv (2021). doi: https://doi.org/10.1101/2021.04.21.440736

The SG-NEx data was accessed on 12 November 2023 at registry.opendata.aws/sg-nex-data.