Supplementary Notes

1. DARTS BHT statistical modeling

1.1 Benchmarking DARTS BHT on simulated data

1.1.1 Generation of benchmark dataset

In order to better represent the variability inherent in real experimental datasets, while knowing the ground-truth, we employed the flux-simulator[1] software (v1.2.1) to simulate RNA-seq reads. Flux-simulator is a specialized simulation software program that models RNA-seq experiments using a set of modules for different experimental procedures, including RNA fragmentation, library preparation and high-throughput sequencing. The major advantage of simulating data using flux-simulator as opposed to directly drawing reads from a statistical distribution is that the former approach takes the variances/noises at different stages into consideration, whereas the latter assumes all reads are generated by a simple stochastic process and are counted correctly; hence, our approach better captures the real-world variances compared to a naïve simulator.

Flux-simulator simulates RNA-seq reads based on a given molecular profile that contains "number of molecules" for each transcript. We derived the molecular profile from a previously published dataset, E-MTAB-1147 from ArrayExpress, which is an RNA-seq experiment of HeLa cell line upon hnRNPC knockdown[2]. We chose this dataset because our previous analysis had demonstrated that it contained abundant splicing changes[3], and that its sequencing depth was sufficiently deep to ensure robust estimation of the transcript expression. We used Kallisto[4] (v 0.43.0) to estimate the transcript TPM from the raw reads using Gencode[5] V19 as reference GTF. The transcript TPM was subsequently converted to number of molecules by fixing the total number of molecules at 5,000,000 (default setting in flux-simulator) and rounding fractional molecules to the nearest integer.

Taking the customized molecular profile, we ran Flux-simulator using: the fragment distribution derived from the above experiment, sequencing read length equal to 72bp with 100 million paired-end reads, and leaving other parameters at their default settings. Next, we ran STAR (v 2.5.2a) to map the reads to the hg19 version of genome with Gencode v19 as gene annotation file. The resulting outputs were two alignment bam files corresponding to the profiles derived from Control and hnRNPC knockdown.

1.1.2 Evaluation of DARTS BHT, MISO and MATS

We processed the alignment files with rMATS[6] (v4.0.1) to count the junction-spanning reads with Gencode v19 as reference annotation. The inclusion junction counts and skipping junction counts for all detected events were then fed into the DARTS BHT model with a flat prior as input. We ran DARTS BHT with $\tau_1 = 0.3, \tau_0 = 0.03$ and testing for C=0.05. The output of DARTS BHT was subsequently benchmarked using the true delta-PSI values between two conditions.

Note that we only considered simple skipping events in the simulation study, because the complex events are often combinations of multiple alternative splicing events and the true PSI values are often ambiguous to define and hence hard to compute. We define the simple events as events with a unique one-to-one mapping for the 5'- and 3'- splice sites of the middle skipping exon, upstream exon 5'- to middle exon 3'- splice site, and middle exon 5'- to downstream 3'- splice site. After filtering for simple events, we had 7,678 simple exon-skipping events out of 16,676 exons detected by rMATS.

As a comparison, we also ran MISO[7] (v0.5.3) and MATS (v4.0.1) on the simulated datasets. To run MISO exon-centric analysis, we downloaded the Human genome (hg19) annotation file v1.0 from the MISO website (https://miso.readthedocs.io/en/fastmiso/annotation.html) and built the index for MISO using the Skipping Events (SE) in the annotated folder by "index_gff". Next, we ran MISO to quantify the splicing level under each condition

then used "compare_miso" to compute the Bayes Factor for the skipping events in MISO annotation files. We ran the MATS statistical model with setting C=0.05 on the read counts generated by rMATS.

Since MISO analyzes its own internal skipping events annotation which is different from the simple events definition in DARTS and MATS, we took the intersection of the events from these software programs. There were 3,407 common events between the two software programs, with 1,344 events' absolute delta-PSI larger than 5% which we labeled as positive. We measured the accuracy of DARTS BHT, MISO and MATS by AUROC and AUPR. As shown in **Supplementary Fig. 2**, DARTS compares favorably to MISO and MATS, demonstrating its superior inference power to the state-of-the-art splicing inference tools when using only empirical RNA-seq data.

1.2. DARTS BHT statistical model for unpaired or paired replicates

1.2.1 Illustration of replicate DARTS BHT statistical model

Thanks to the rapid development of sequencing technology, it has become practical and common for transcriptomic studies to carry out RNA-seq experiments with multiple replicates to quantify biological variances and improve reproducibility. Previously we had demonstrated that pooling the reads from different replicates is not recommended[6]. Motivated by the replicate analysis, we sought to develop the replicate DARTS model that considers replicates in its likelihood function while still being capable of taking the informative prior into account.

Following the notations in the DARTS main text, we extend the DARTS BHT model to include read counts from different replicates into the following hierarchical model (we are abusing the subscripts k here to index replicate; whereas k was used to index different experimental conditions in the main text):

$$
\begin{aligned}
I_{ijk} \mid \psi_{ijk} &\sim \text{Binomial}(n = I_{ijk} + S_{ijk}, p = f_i(\psi_{ijk})) \\
\psi_{ijk} &= \mu_i + 1(j = 2) \cdot \delta_i + \epsilon_{ik}, \qquad \epsilon_{ik} \sim N(0, \sigma^2) \\
\mu_{ik} &= \mu_i + \epsilon_{ik}, \qquad \mu_{ik} \sim N(\mu_i, \sigma^2) \\
\mu_i &\sim \text{Unif}(0,1) \\
\delta_i &\sim N(0, \tau^2)
\end{aligned}
\qquad \text{[S1]}
$$

I_{ijk}, S_{ijk} and ψ_{ijk} are the inclusion read counts, the skipping read counts and the exon inclusion level for exon i, sample group j=1,2, in replicate k; f_i is the length normalization function for exon i; μ_i is the baseline inclusion level for exon i, and δ_i is the difference of the exon inclusion levels between the two conditions. Without loss of generality, we let $\psi_{i1\text{k}} = \mu_i + \epsilon_{ik}, \psi_{i2\text{k}} = \mu_i + \delta_i + \epsilon_{ik}$; that is, we assume that the effect size δ_i is the same across different replicates; and that ψ_{ijk} values in each replicate k have a random replicate-specific deviation from the group mean μ_i by ϵ_{ik}. The term ϵ_{ik} captures the within group variance of PSI values in different replicates and has an expectation of 0.

It is worthwhile to point out that the above replicate DARTS framework is applicable for both paired replicates and unpaired replicates. The subscript k indexes for the samples from different/same origins. For paired replicates, the two paired observations under the two corresponding conditions are indexed with the same k, and should therefore share the same starting point/baseline level of $\mu_{ik} = \mu_i + \epsilon_{ik}$, while only differing by the amount δ_i

caused by the treatment. For unpaired replicates, each sample is indexed with a different k, hence the baseline level μ_{ik} was drawn independently from $N(\mu_i, \sigma^2)$ and there is no covariance between samples in the two groups.

1.2.2 Simulated read counts and evaluation

Next, we simulated read counts by drawing reads from binomial distributions as in Eq. **S1**. We did not use flux-simulator for this analysis because it is non-trivial to define the within-group variances at the "number of molecules" level; instead, we imposed a normal distribution to the simulated group mean PSI value, then drew read counts from this hierarchical generating process.

We performed extensive simulation studies using different combinations of parameters. Specifically, we set the model parameters equal to the following values: $\sigma \in \{0.05, 0.35\}$, the within group variance, smaller values of σindicated more consistent patterns across replicates; $K \in \{6,10\}$, the number of replicates, more replicates would help better capture the within group variance; $n \in \{30, 50\}$, the coverage of each replicate, deeper coverage would help estimation of sample-wise PSI; presence of outlier, outlier PSI value was draw randomly from [0,1] to represent one unrelated sample out of the all replicates. We benchmarked the performances of pooled DARTS, replicate DARTS (rDARTS), and rMATS, using the AUROC and AUPR. To obtain a reliable performance estimate, we randomly sampled n=3,000 events under each simulation configuration, with the expected differentially spliced events (positive cases) at 50%.

As shown in **Supplementary Fig. 3**, replicate DARTS showed a consistent gain in power under two specific situations, regardless of the number of replicates: i) when the within-group variance σ is large, and ii) when there is an outlier sample. This is consistent with our previous observation in the rMATS paper[6]. Notably, in all simulations, we fixed the total coverage at 300, i.e. when K=6, each sample has 50 read counts per event; when K=10, each sample has 30 read counts per event. Such configurations emulate a fixed sample-size budget, where researchers hope to get the best scientific outcomes using the optimal experimental design. It is not surprising that increasing the number of 6 replicates by 4 would significantly reduce the loss of power caused by introducing 1 outlier sample. The same effect was true for larger within-group variances, demonstrating the better group variance estimation captured by more replicates with less coverage per replicate. In all comparisons, the replicates DARTS model outperforms the pooled DARTS model under certain conditions, while inflicting no loss of power under regular conditions. Hence, we recommend using the replicate DARTS model whenever possible, and advise against pooling reads from replicates.

1.3. Technical notes on statistical model optimization

1.3.1 Laplacian approximation

The optimization of the DARTS model involves two major steps: i) calculating the Bayes Factor of two competing models/hypotheses, ii) sampling the posterior distribution given the non-conjugate priors. In this part we will first deal with the calculation of the Bayes Factor, where we utilized Laplace's method to approximate the intractable integrals.

Following the notation in the Method section, the essence of DARTS BHT with flat prior is the ratio of the integrated likelihood function, also known as the Bayes Factor. In the DARTS model, the integrated likelihood function takes the form of

$$P(I_{ij}, S_{ij} \,|H_n) = \iint_{\Theta_n} P(I_{ij}, S_{ij} \,|\mu_i, \delta_i\,) \cdot P(\mu_i, \delta_i|H_n)\, d\mu_i d\delta_i$$

$$\propto \int_{-\infty}^{+\infty} \int_{0}^{1} f_i(\psi_{i1})^{I_{i1}} \cdot \left(1 - f_i(\psi_{i1})\right)^{S_{i1}} \cdot f_i(\psi_{i2})^{I_{i2}} \cdot \left(1 - f_i(\psi_{i2})\right)^{S_{i2}} \cdot \mathbf{1}(|2 \cdot (\mu_i + \delta_i - 0.5)| < 1) \cdot e^{-\delta^2/\tau_n^2} d\mu_i \, d\delta_i$$

$$= \iint_{\Theta_n} g(\mu_i, \delta_i; I_{ij}, S_{ij}) d\mu_i d\delta_i$$

$$= \iint_{\Theta_n} \exp\left(\, g_1(\mu_i, \delta_i; I_{ij}, S_{ij})\, \right) d\mu_i d\delta_i$$

[S2]

The above integral cannot be solved in closed form. Instead, we employ Laplace's method to approximate the integral. Let $g_1 = \log g$ be the log posterior density function, the Laplacian approximation can be viewed as the Gaussian approximation to any (posterior) distribution that is smooth and well-peaked around its maximal point. To implement Laplacian approximation for DARTS BHT, we compute both the maximal point of the posterior probability as well as the local curvature/Hessian matrix around the maximal point using the "optim" function in R by feeding its objective function and the gradient function. Then the approximation for the integral, denoted by Z_n, is

$$Z_n = \log\left(P\left(I_{ij}, S_{ij} \,\middle|\, H_n\right)\right) \approx g_1(\hat{\mu}_i, \hat{\delta}_i; I_{ij}, S_{ij}) - 0.5 \times \log\left(|(\mathrm{H}(\hat{\mu}_i, \hat{\delta}_i)|\right) + \frac{d}{2} \cdot \log(2\pi)$$

[S3]

$\hat{\mu}_i, \hat{\delta}_i$ are the parameter values that maximize posterior probability; $g_1(\hat{\mu}_i, \hat{\delta}_i; I_{ij}, S_{ij})$ is the log posterior probability function evaluated at maximal point; $\mathrm{H}(\hat{\mu}_i, \hat{\delta}_i)$ is the Hessian matrix of g_1 evaluated at the maximal point; and d is the total number of parameters in $g_1(\cdot)$. Then, the Bayes Factor (BF) is

$$\mathrm{BF} = \frac{P\left(I_{ij}, S_{ij} \,\middle|\, H_1\right)}{P(I_{ij}, S_{ij} \mid H_0)} = \exp(Z_1 - Z_0)$$

[S4]

1.3.2 MCMC sampling

Next we seek to sample from the posterior distribution of the parameters given the data/observations under a specific hypothesis. Since we do not have the conjugate prior for the likelihood, we employ an MCMC random walk to draw samples from the posterior distribution. Specifically, we designed the transition probability q as a normal distribution with mean equal to the current state and a small variance corresponding to a small step size. For each proposed state, we accept the proposal by a Metropolis-Hasting acceptance probability:

$$\alpha(\theta^t, \theta^{t+1}) = \min\left(1, \frac{q(\theta^t | \theta^{t+1}) \cdot g(\theta^{t+1}; I_{ij}, S_{ij})}{q(\theta^{t+1} | \theta^t) \cdot g(\theta^t; I_{ij}, S_{ij})}\right)$$

[S5]

$g\left(\theta^{t+1}; I_{ij}, S_{ij}\right) = g(\mu_i, \delta_i; I_{ij}, S_{ij})$ is the posterior probability function defined in subsection 1.3.1, and $q(x|y)$ is the transition probability from state y to state x. Note that to maintain the domain of $\psi_{ij} \in [0,1]$, out of domain parameter values were truncated by setting the likelihood function to zero.

In order to shorten the burn-in period, we initialize the Markov Chain at $\hat{\theta}$, i.e. the optimal point obtained from the previous step while computing the Bayes Factor. Moreover, such an initialization ensures that the starting state is close to where the target probability density is concentrated, especially when there are multiple replicates and the target probability density is in high-dimensional space. The initialization scheme can greatly shorten the burn-in period.

Under the above configurations, we noticed that in practice, drawing 1500 samples with a burn-in period of 100 and 10 thinning achieved good balance between estimation accuracy and running time.

1.3.3 Justification on different values of τ parameter

In DARTS BHT, the choice of the parameter τ specifies the two competing hypotheses: differential splicing and unchanged splicing between two biological conditions. Here we show that since the final inference is performed on the probability of $P(|\Delta\psi| > c)$ marginalizing over the hypotheses, DARTS BHT is robust to different choices of τ_k. We started with an example by comparing the inference results on a set of simulated splicing events (n=1000) when setting $\tau_1 = 0.3, \tau_2 = 0.03$ (default setting in our paper) with $\tau_1 = 0.4, \tau_2 = 0.02$ (alternative setting here). We observed the ranks of the final inference $P(|\Delta\psi| > c)$ under these two settings were highly consistent (Spearman's rho=0.99), demonstrating the robustness of DARTS BHT to difference choices of τ. Additionally, comparing the actually posterior probability of these two settings, we observed the values were highly similar for $P(|\Delta\psi| > c) \approx 1$, where is the major region of interest for inference of differential splicing. The alternative setting has a negative bias (more conservative) around $P(|\Delta\psi| > c) \approx 0$ due to stronger regularization effect from a smaller $\tau_2 = 0.02$. This will allow users to reflect their beliefs on data quality through the choices of τ as regularization strength. For example, when data is noisy, users would preferably specify the alternative setting over our default setting. To further understand the impact of the parameter τ, we examined another four alternative settings of τ using various combinations of different τ values. Indeed, the inference results are robust in different scenarios, especially for the ranking/inference of differential alternative splicing events (upper right corner of each panel in **Supplementary Fig. 10**). The model of DARTS BHT is designed to be robust to different specifications as well as flexible enough to account for different dataset-specific requirements.

1.3.4 Running time analysis

The computation of the DARTS BHT model is demanding because of the random sampling of the non-conjugate posterior. Compared to conventional inference methods that only estimate point estimates for the parameters of interest, the DARTS BHT model needs to derive the whole posterior probability distribution using an MCMC sampling. Hence, we re-wrote the MCMC sampler in Rcpp[8]. The source code was compiled during the installation of the DARTS R package and the resulting speed gain was around 10 fold. We also tuned the MCMC sampling (see subsection 1.3.2) to shorten the burn-in period.

In general, the optimized optimization procedure runs in a reasonable amount of time. For the DARTS BHT without replicate mode, an individual event takes 0.23s wall-clock time on average to finish the optimization on an Intel i7-4790 3.60GHz CPU. For the DARTS BHT with replicate, the running time scales linearly with the number of replicates for an individual event. In our benchmarking, an event with 6 replicates takes around 1.38s and an event with 10 replicates takes 2.07s on average.

2. DARTS DNN Machine learning

2.1. Sequence feature extraction and normalization

The DARTS DNN cis sequence features are built upon a previous report[9] that curated 1,393 RNA features. Furthermore, we expanded the feature set by including 1,533 additional features on RBP binding motifs and conservation scores. We compiled cis sequence features for four different types of alternative splicing events, i.e. exon skipping, alternative 5' splice sites, alternative 3' splice sites, and retained introns. Below we briefly describe all the feature annotations of exon skipping events as an example; the full lists of all cis sequence features for the four types of alternative splicing events can be found in **Supplementary Table 1** and are publicly available in the GitHub repository.

For each exon skipping events, let C1, A, and C2 be the upstream exon, skipping exon and downstream exon respectively. I1 denotes the intron region between C1 and A, and I2 denotes the intron region between A and C2. The DARTS DNN cis features are grouped by the following generic categories:

1) Exon length and ratio of length of exons and introns.

2) Nucleosome occupancy scores are computed using NuPoP[10] for the skipping exon and flanking introns. The features are defined as predicting the nucleosome positioning in the first 100 nucleotides of each intron and in the first and last 50 nucleotides of skipping exons.

3) The definition of translatability is whether a sequence can be translated without stop codons under three different reading frames. We are evaluating translatability of C1, C1-C2, C1-A, C1-A-C2.

4) We include 111 curated RBP-binding motifs and count motifs in each of the 7 intronic and exonic regions. In addition to the counting procedure, we also download the RBP binding PSSM matrix from RBPmap[11] and calculate the PSSM scores of each RBP-binding profile.

5) We run two different tools, one from Itoh et al.[12], and maxent[13], to estimate the splicing strength between the three exon-exon junctions: C1-C2, C1-A and A-C2.

6) Conservation scores are computed as average conservation score of the first and last 100 nucleotides of intron I1 and intron I2. The conservation scores are downloaded from UCSC phastCons46way.

7) The secondary structure score is predicted by the maximum availability of intron regions using RNAfold[14].

8) Short motifs are integrated from Xiong et al.[9].

9) Alu repeats annotation is downloaded from UCSC genome browser. Features are defined as counts of Alu repeats on the plus and minus strand of two intronic regions.

10) ESE (exon splicing enhancer), and ESS (exon splicing silencer) are from Burge's and Chasin's work[15, 16]. ISS (intron splicing silencer) and ISE (intron splicing enhancer) are from Wainberg's work[17].

In total, the number of RNA features was 2,926 and the complete list can be found in **Supplementary Table 1**.

Although certain classifiers (e.g. tree-based models) are robust to the feature scaling, it is important to scale the features for neural networks. We followed the feature scaling method described previously[9] and divided each feature by its maximum absolute value across all training sets. This rescaled the features to [-1,1] while preserving the zero values, which has specific biological indications.

2.2 ENCODE data processing

2.2.1 Extraction of junction counts and detection of novel events

Following the descriptions in the Method section, we had downloaded all the alignment files from the ENCODE data portal[18] and processed the bam files with rMATS. Aside from the annotated events in the reference GTF Gencode v19, rMATS detected novel splicing events where edges not annotated in the GTF splicing graph connect two annotated exons. These novel events consist of a large proportion of our training dataset and are crucial for learning the regulatory code between RBP perturbations and alternative splicing. Note that our definition of novel events are novel edges or junction reads that are not present in GTF; we do not detect novel splice sites or novel exons.

2.2.2 RBP expression estimation

The robust performance of DARTS DNN is dependent on the robust estimation of RBP expression levels, given that all sequence features are static. A previous report has demonstrated that 10 million reads per sample was a good depth for differential gene expression analysis[19], hence we reasoned that the gene expression estimates are fairly robust to reduction in sequencing depth, unlike the exon inclusion level estimates that depends on junction-spanning read counts. In practice we re-analyzed gene expression using Kallisto (v.0.43.0) from raw fastq reads downloaded from the ENCODE data portal. We extracted the TPM of all RBPs from the annotated list. The estimated TPM was subsequently divided by the maximum value across all datasets to rescale it range to [0,1].

2.3 Implementation of other machine learning strategies and comparison to DARTS DNN

2.3.1 Logistic regression and Random Forest

To benchmark the performance of our trained DARTS DNN model to other machine learning strategies, we implemented two baseline methods, Logistic regression with L2 penalty and Random Forest. Because these baseline methods were unable to scale up to big data (see 2.3.2 below), they were trained and benchmarked on individual ENCODE leave-out datasets by cross-validation. The identical events with their corresponding labels and features were fed into the baseline classifiers through 5-fold cross-validation and we recorded the performance measured by AUROC in each of the validation sets.

We implemented the two methods using scikit-learn in python. For the logistic regression, we need to tune one parameter, i.e. the penalty strength, or the inverse of the penalty strength C. This parameter controls the complexity of the classifier and hence the severity of overfitting. We chose C=0.1 for our implementation of logistic regression because in practice such a penalty achieves good reasonable generalization over different datasets. Although logistic regression is easy to interpret and a good baseline method for most classification tasks, it cannot effectively detect high-order interaction terms, diminishing its predictive power for such complex tasks.

Another more powerful and robust machine learning strategy we employed as a baseline method was Random Forest. Random Forest is an ensemble learning method where each base classifier is a decision tree that over-fits a set of bootstrapped training samples with a subset of features. The Random Forest classifier has several desirable properties, including being robust to feature scaling and irrelevant features, and being capable of dividing the feature space more flexibly than more conventional partitioning based classification methods. We tuned the hyper-parameter of Random Forest, i.e. the number of trees in the forest. Typically, the more trees in a random

forest, the better predictive power it renders to the ensemble classifier. We noticed that for our datasets, 500 trees achieved the best testing accuracy while increasing the number of trees further did not grant much more gain.

As shown in **Fig. 1d**, Random Forest almost always outperformed Logistic regression given the same training datasets. We can also observe a positive correlation between the performance of Random Forest and Logistic regression, indicating the internal structure of the training data plays an important role in the learning efficiency, despite the fact that the two learning algorithms are based on dramatically different underlying structures. Nevertheless, DARTS DNN showed superior performance compared to the baseline methods, even though these knock-down datasets have never been trained in DARTS DNN. Furthermore, the performance of DARTS DNN does not show strong correlations with the base learners, indicating its generalization over the single datasets to a more generic regulatory code.

2.3.2 Technical notes on DNN training

Below we briefly describe some technical details in training the DARTS DNN model using the ENCODE data. DARTS DNN was implemented in Keras with Theano backend. The DNN model was a 4-hidden layer fully-connected neural network with drop-out (with probability 0.6, 0.5, 0.3 and 0.1, respectively) and batch-normalization layers, and each neuron had ReLU (rectifier linear unit) activation function that maps the input vector x to a non-linear output:

$$\mathrm{ReLU}(x) \;=\; \max\left(0, w^T x + b\right)$$

The weight parameter w and bias b are learned through training on labelled samples and minimizing the loss function, which is the binary cross-entropy between the observed labels Y and predictions $\hat{Y}$:

$$\mathrm{L}\left(\hat{Y}; Y\right) = -\frac{1}{\mathrm{N}}\sum_{i=1}^{N}\left[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log\left(1 - \hat{y}_i\right)\right]$$

We optimized the model parameters using the RMSprop optimizer. RMSprop is a variant of the stochastic gradient descent algorithm, which accounts for the recent momentums of the gradient and adaptively adjusts the learning rate. In our experiments, RMSprop works better than other optimizers for most network architectures.

Because the training dataset was huge and took too much memory (>100G) to be loaded at once, we divided the training samples into different data batches by the knock-down experiments. In each data batch, we randomly picked two different RBP knockdown experiments; due to the way the training datasets were constructed, every RBP selected must have been knocked down in both HepG2 and K562 cell lines. Hence in each data batch, we had at least 4 different datasets, sometimes more if this RBP was knocked-down by more than one shRNA in a certain cell line. The pairing of the same RBP in two different cell lines ensured that there was sufficient variance in the RBP expression features, hence facilitating the classifier to learn from the trans-acting factors.

Next we mixed the training skipping-exon events from the data batch, and held-out 20% of these events as validation set, and the remaining 80% as training set. The training set was then split into positive and negative stacks of cases, and we aimed to construct mini-batches of size 400 to feed into training the model sequentially. Because the training set was very imbalanced and the number of negative cases outweighed the number of positive cases, we balanced the composition of each mini-batch by first extracting 100 (25%) positive cases from the positive stack, then compensating 300 (75%) negative cases from the negative stack. Such biased composition of mini-batches will generate the back-propagation of errors from positive cases and reduce strong negative bias caused by the imbalanced data.

To monitor potential overfitting, we computed the validation loss and the prediction AUROC of the current model every 10 mini-batches of training. Due to the imbalanced composition of the datasets, we noticed that using AUROC as the monitoring criteria performed better than the loss function because the loss function could be stuck in a local optima where all cases were classified as negative. We only saved the parameter values of the best performing models on the validation data; by the end of the training for each data batch, we re-loaded the saved model parameter values. The goal of such a configuration was to avoid overfitting to any particular individual data batch while exploring for the global optimal point(s) in the model energy landscape.

References

1. Griebel, T. et al. Modelling and simulating generic RNA-Seq experiments with the flux simulator. *Nucleic Acids Res* **40**, 10073-10083 (2012).
2. Zarnack, K. et al. Direct competition between hnRNP C and U2AF65 protects the transcriptome from the exonization of Alu elements. *Cell* **152**, 453-466 (2013).
3. Zhang, Z. & Xing, Y. CLIP-seq analysis of multi-mapped reads discovers novel functional RNA regulatory sites in the human transcriptome. *Nucleic Acids Res* **45**, 9260-9271 (2017).
4. Bray, N.L., Pimentel, H., Melsted, P. & Pachter, L. Near-optimal probabilistic RNA-seq quantification. *Nat Biotechnol* **34**, 525-527 (2016).
5. Harrow, J. et al. GENCODE: producing a reference annotation for ENCODE. *Genome Biol* **7 Suppl 1**, S4 1-9 (2006).
6. Shen, S. et al. rMATS: robust and flexible detection of differential alternative splicing from replicate RNA-Seq data. *Proc Natl Acad Sci U S A* **111**, E5593-5601 (2014).
7. Katz, Y., Wang, E.T., Airoldi, E.M. & Burge, C.B. Analysis and design of RNA sequencing experiments for identifying isoform regulation. *Nat Methods* **7**, 1009-1015 (2010).
8. Eddelbuettel, D. Seamless R and C++ integration with Rcpp. (Springer, New York; 2013).
9. Xiong, H.Y. et al. RNA splicing. The human splicing code reveals new insights into the genetic determinants of disease. *Science* **347**, 1254806 (2015).
10. Xi, L. et al. Predicting nucleosome positioning using a duration Hidden Markov Model. *BMC Bioinformatics* **11**, 346 (2010).
11. Paz, I., Kosti, I., Ares, M., Jr., Cline, M. & Mandel-Gutfreund, Y. RBPmap: a web server for mapping binding sites of RNA-binding proteins. *Nucleic Acids Res* **42**, W361-367 (2014).
12. Itoh, H., Washio, T. & Tomita, M. Computational comparative analyses of alternative splicing regulation using full-length cDNA of various eukaryotes. *RNA* **10**, 1005-1018 (2004).
13. Yeo, G. & Burge, C.B. Maximum entropy modeling of short sequence motifs with applications to RNA splicing signals. *J Comput Biol* **11**, 377-394 (2004).
14. Bindewald, E. & Shapiro, B.A. RNA secondary structure prediction from sequence alignments using a network of k-nearest neighbor classifiers. *RNA* **12**, 342-352 (2006).
15. Fairbrother, W.G. et al. RESCUE-ESE identifies candidate exonic splicing enhancers in vertebrate exons. *Nucleic Acids Res* **32**, W187-190 (2004).
16. Zhang, X.H. & Chasin, L.A. Computational definition of sequence motifs governing constitutive exon splicing. *Genes Dev* **18**, 1241-1250 (2004).
17. Wainberg, M., Alipanahi, B. & Frey, B. Does conservation account for splicing patterns? *BMC Genomics* **17**, 787 (2016).
18. Consortium, E.P. An integrated encyclopedia of DNA elements in the human genome. *Nature* **489**, 57-74 (2012).
19. Liu, Y., Zhou, J. & White, K.P. RNA-seq differential expression studies: more sequence or more replication? *Bioinformatics* **30**, 301-304 (2014).