OpenSpliceAI: An efficient, modular implementation of SpliceAI enabling easy retraining on non-human species

Kuan-Hao Chao[1, 2, *, †], Alan Mao[1, 2, 3, †], Anqi Liu[1], Steven L Salzberg[1, 2, 3, 4, *], and Mihaela Pertea[1, 2, 3, *]

[1]Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218, USA
[2]Center for Computational Biology, Johns Hopkins University, Baltimore, MD 21218, USA
[3]Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD 21218, USA
[4]Department of Biostatistics, Johns Hopkins University, Baltimore, MD 21211, USA

[†]These authors contributed equally
[*]*corresponding authors:* kh.chao@cs.jhu.edu, salzberg@jhu.edu, mpertea@jhu.edu

Abstract

The SpliceAI deep learning system is currently one of the most accurate methods for identifying splicing signals directly from DNA sequences. However, its utility is limited by its reliance on older software frameworks and human-centric training data. Here we introduce OpenSpliceAI, a trainable, open-source version of SpliceAI implemented in PyTorch to address these challenges. OpenSpliceAI supports both training from scratch and transfer learning, enabling seamless re-training on species-specific datasets and mitigating human-centric biases. Our experiments show that it achieves faster processing speeds and lower memory usage than the original SpliceAI code, allowing large-scale analyses of extensive genomic regions on a single GPU. Additionally, OpenSpliceAI's flexible architecture makes for easier integration with established machine learning ecosystems, simplifying the development of custom splicing models for different species and applications. We demonstrate that OpenSpliceAI's output is highly concordant with SpliceAI. *In silico* mutagenesis (ISM) analyses confirm that both models rely on similar sequence features, and calibration experiments demonstrate similar score probability estimates.

Introduction

Predicting splice sites within primary DNA sequences has a wide range of uses, including understanding gene regulation, identifying alternative protein isoforms, and detecting sequence variants that affect splicing (Black, 2000; Braunschweig et al., 2013; Wagner et al., 2023, 2023; Xiong et al., 2015). Splicing is a complex and tightly regulated process that enables the production of multiple protein isoforms from a single gene, contributing to cellular complexity, adaptability and diversity across different cells and tissues (Blencowe, 2006; Johnson et al., 2003; Wang et al., 2008).

Aberrant splicing regulation can contribute to a wide range of diseases, including some types of cancer (Bonnal et al., 2020; Jung et al., 2015; Lee and Abdel-Wahab, 2016; Supek et al., 2014; Sveen et al., 2016), neurodegenerative disorders (Li et al., 2021; Mills and Janitz, 2012; Nikom and Zheng, 2023), cardiovascular diseases (Gotthardt et al., 2023; Martí-Gómez et al., 2022), metabolic syndromes (Dlamini et al., 2017; Moore et al., 2010), and various genetic conditions (Segal and Widom, 2009; Xiong et al., 2015). Notably, Duchenne muscular dystrophy (Aartsma-Rus et al., 2002; Mcclorey et al., 2005) and spinal muscular atrophy (Burnett et al., 2009; Lorson et al., 1999; Naryshkin et al., 2014) are well-known examples of disorders arising from splicing defects. It has been estimated that 15% to 50% of disease-causing mutations in humans influence splice site selection (Baralle and Giudice, 2017; Barash et al., 2010; Wang and Cooper, 2007), underscoring the critical need for precise modeling of splicing regulation at the DNA level and accurate interpretation of model predictions.

Building on advancements in deep learning, particularly convolutional neural networks (CNNs), genomics researchers have made substantial progress in modeling complex, long-range dependencies in DNA sequences. These models have driven significant improvements in predictive accuracy across diverse applications, including regulatory grammar (Alipanahi et al., 2015; Kelley et al., 2018, 2016; Zhou et al., 2018; Zhou and Troyanskaya, 2015), 3D genome organization (Fudenberg et al., 2020), mRNA stability (Agarwal and Kelley, 2022), and notably, splice site prediction (Jaganathan et al., 2019; Sokolova et al., 2024). Among these, SpliceAI (Jaganathan et al., 2019) stands out as the leading tool for splice site prediction, applying a deep residual CNN architecture to identify patterns dictating splicing mechanisms directly from primary sequences without relying on human-engineered features.

Despite its success, SpliceAI has limitations that hinder its broader application. The official implementation relies on an outdated version of TensorFlow (Abadi et al., 2016) and Keras, which may not function well with newer machine learning frameworks such as PyTorch (Paszke et al., 2019), which has been widely adopted in recent years. Additionally, SpliceAI's use of human training data limits its performance on non-human species, suggesting that a re-training module could provide substantial advantages for those wishing to use it on model organisms or other species.

To address these limitations, we developed OpenSpliceAI, a trainable open-source implementation of SpliceAI in PyTorch. OpenSpliceAI supports both training from scratch and transfer-learning approaches, making it adaptable to species-specific datasets. As we show in our experiments below, OpenSpliceAI

offers faster processing speed, reduced memory usage, and efficient GPU utilization, enabling analysis of long sequences and large datasets on a single GPU. In silico mutagenesis (ISM) analyses revealed the features that both SpliceAI and OpenSpliceAI rely on for making predictions. Calibration experiments showed that OpenSpliceAI models are well-calibrated, improving the reliability of splice site predictions.

Results

OpenSpliceAI: an open-source splice site prediction framework in Pytorch

Our new system, which we call OpenSpliceAI, is a suite of modular Python scripts that provide researchers with a user-friendly computational framework to study RNA splicing. OpenSpliceAI is an open-source version of SpliceAI, a highly accurate splice site prediction method (Jaganathan et al., 2019). By replacing TensorFlow and Keras with the more-efficient PyTorch framework, OpenSpliceAI offers improved performance, scalability, and compatibility with current machine learning workflows (see Discussion).

The framework faithfully replicates SpliceAI's architecture while extending its functionality. It is important to note that the models produced by OpenSpliceAI and the original SpliceAI are not identical. Variations in weight initialization, data shuffling, batch normalization, and optimizer stochasticity introduce subtle differences between the models, as discussed in detail in the Discussion section. Additionally, we provide new modules for training the network, allowing for easy re-training on other species, which we show provides more accurate performance on those species. OpenSpliceAI supports custom model training on long DNA sequences and offers both training-from-scratch and transfer-learning approaches to adapt models to species-specific datasets. We also conducted experiments to analyze the effects of DNA mutations on OpenSpliceAI's predicted scores for donor and acceptor sites, and show how to use it to identify cryptic splicing events, where a mutation can activate a normally dormant splice site.

To streamline its use, OpenSpliceAI offers six subcommands for data preprocessing, model training, transfer learning, calibration, prediction, and variant analysis (see Figure 1). Detailed functionalities of each module are described in Methods.

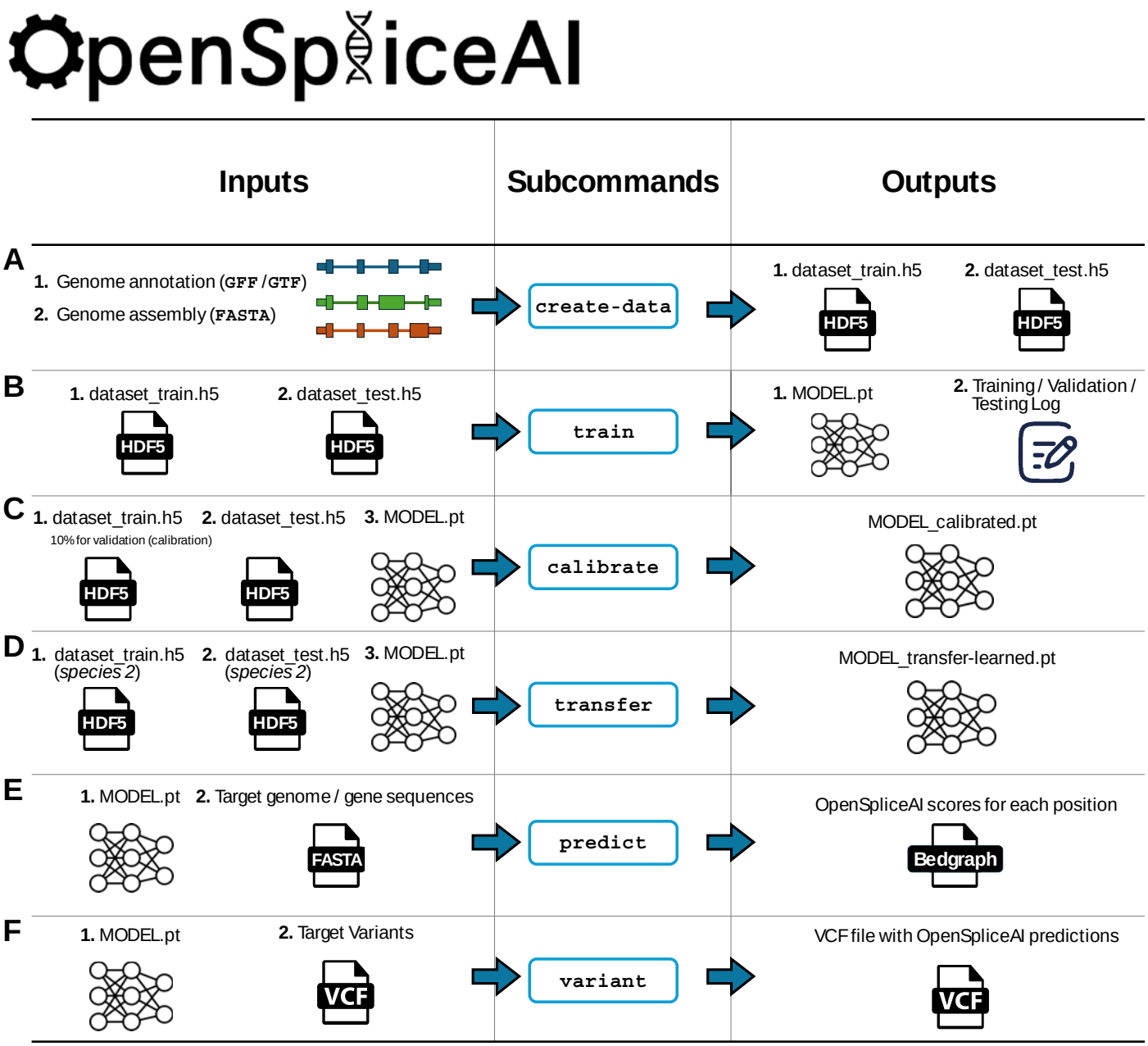


Figure 1 Overview of the OpenSpliceAI design. This toolkit features six primary subcommands: **(A)** The "`create-data`" subcommand processes genome annotations in GFF/GTF format and genome sequences in FASTA format to produce one-hot encoded gene sequences (`X`) and corresponding labels (`Y`), both stored in HDF5 format. **(B)** The "`train`" subcommand utilizes the HDF5 files generated by "`create-data`" to train the SpliceAI model using PyTorch, resulting in a serialized model in PT format. This process also generates logs for training, testing, and validation. **(C)** The "`calibrate`" subcommand takes both training and test datasets along with a pre-trained model in PT format. It randomly allocates 10% of the training data as a validation (calibration) set, which is then used to adjust the model's output probabilities so that they more accurately reflect the observed empirical probabilities during evaluation on the test set. **(D)** The "`transfer`" subcommand allows for model customization using a dataset from a different species, requiring a pre-trained model in PT format and HDF5 files for transfer learning and testing. **(E)** The "`predict`" subcommand enables users to predict splice site probabilities for sequences in given FASTA files. **(F)** The "`variant`" subcommand assesses the impact of potential SNPs and indels on splice sites using VCF format files, providing predicted cryptic splice sites.

Training OpenSpliceAI with Human MANE annotation

Using the OpenSpliceAI framework, we trained a new Pytorch version of SpliceAI using protein-coding genes annotated in the RefSeq MANE v1.3 database (https://ftp.ncbi.nlm.nih.gov/refseq/MANE/MANE_human/release_1.3/MANE.GRCh38.v1.3.refseq_genomic.gff.gz). MANE provides a standardized set of human gene annotations covering nearly all known protein-coding genes, with one transcript per gene, and ensures that the transcripts are represented identically in the RefSeq and Ensembl/GENCODE annotations of the GRCh38 human reference genome (Morales et al., 2022).

Gene sequences and splice site labels from MANE annotations were extracted and one-hot encoded into tensors for OpenSpliceAI training. Models were trained using flanking sequences of 80, 400, 2,000, and 10,000 nucleotides, with five models trained for each sequence length (see Figure 2A). OpenSpliceAI assigns a score to each position that is an estimate of the probability that the position is a donor site, acceptor site, or neither (see Figure 2B and Methods). In this setup, "OpenSpliceAI" refers to the reimplemented framework, "OSAI" refers to the model, and "$\text{OSAI}_{\text{MANE}}$" denotes the model trained specifically with the MANE annotation. "SpliceAI-Keras" denotes the original published SpliceAI model, which was trained using the canonical transcripts from GENCODE version V24lift37 from the hg19/GRCh37 reference genome.

To compare the performance of $\text{OSAI}_{\text{MANE}}$ and SpliceAI-Keras, both models were evaluated on a held-out test set comprising genes from MANE annotations on GRCh38 chromosomes 1, 3, 5, 7, and 9. Paralogous genes were excluded to prevent data leakage by aligning the test set against the training sets of both models (Methods). $\text{OSAI}_{\text{MANE}}$ (Figure 2C-D, orange curve) showed performance comparable to SpliceAI-Keras (Figure 2C-D, blue curve) across metrics, including top-1 accuracy, area under the precision-recall curve (AUPRC), precision, recall, and F1 score for donor and acceptor splice sites (Figure S1, Methods). Results are presented as error bar plots showing the mean and ± one standard error for each metric.

The best-performing $\text{OSAI}_{\text{MANE}}$ model, trained with 10,000-nt flanking sequences, demonstrated slight yet consistent improvements over SpliceAI-Keras across all metrics (Figure 2 and Figure S1). Specifically, it achieved a top-1 accuracy increase of 1.25% for donor sites and 1.56% for acceptor sites, an F1 score gain of 1.20% for donor sites and 1.04% for acceptor sites, and an AUPRC improvement of 1.90% for donor sites and 1.76% for acceptor sites.

Performance improved with longer flanking sequences, consistent with previous SpliceAI findings. The largest gains occurred between 80 and 400 nt, with accuracy increasing by 62% for donor sites and 74% for acceptor sites. In comparison, improvements were smaller between 400 nt and 2000 nt (3.2% for donors and 2.5% for acceptors) and between 2000 nt and 10,000 nt (3.8% for donors and 4.2% for acceptors). Cross-species evaluations with mouse (Figure S2), honeybee (Figure S3), zebrafish (Figure S4), and Arabidopsis (Figure S5) test sets confirmed comparable performance across species.

OpenSpliceAI supports both targeted splice site prediction and genome-wide predictions across full chromosomes. Its variant submodule enables researchers to assess the splicing impacts of specific variants, such as acceptor and donor site gains or losses, using pre-trained models. Our performance benchmarks of the `predict` (Figure 2E and Figure S6A-F) and `variant` (Figure S6G-L) submodules demonstrate that OSAI outperforms SpliceAI in processing speed, memory usage, and GPU efficiency (see Discussion).

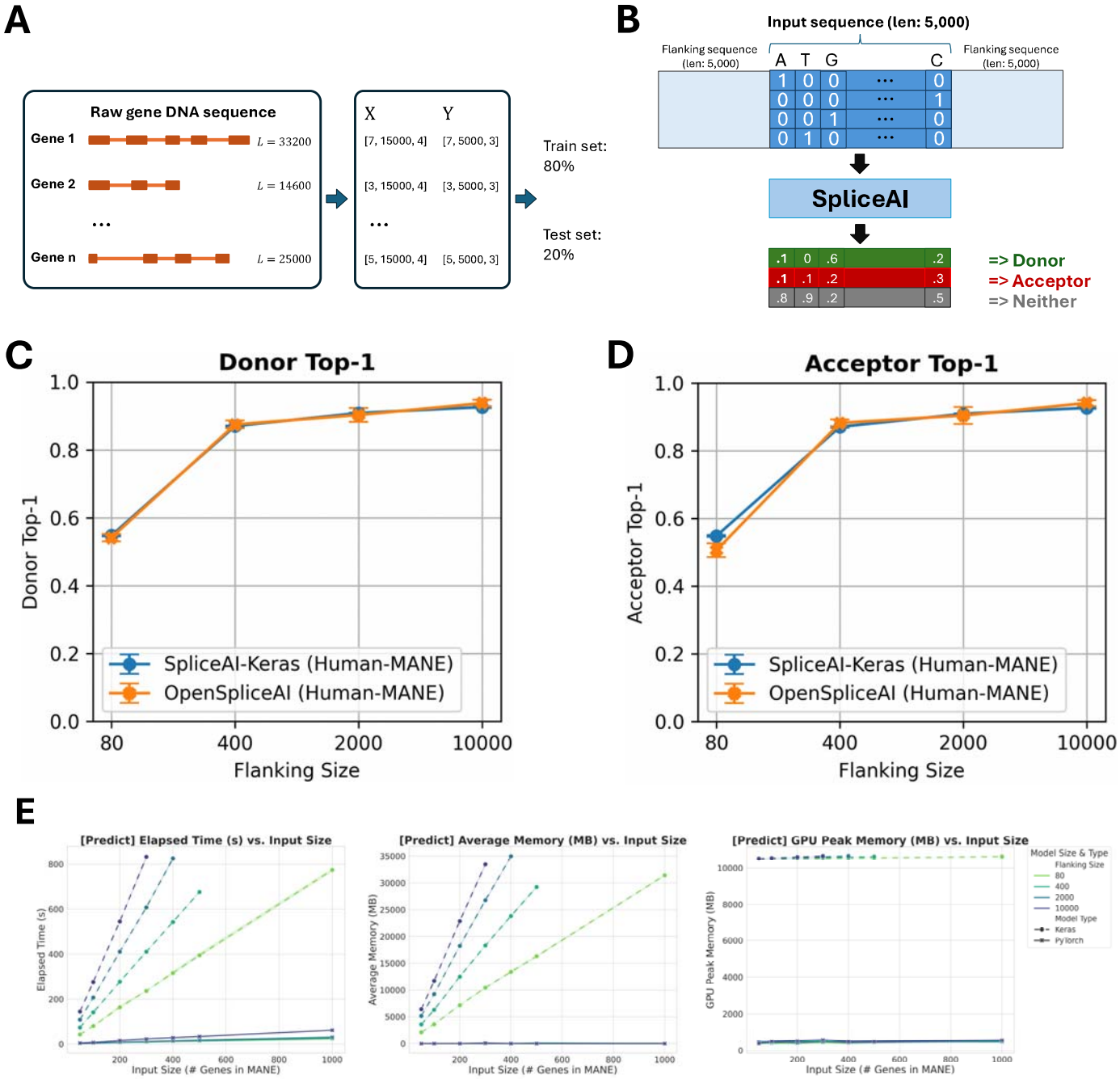


Figure 2 **(A)** Schematic overview of OpenSpliceAI's approach. Gene sequences are first extracted from the genome FASTA file and one-hot encoded (X). Splice sites are identified and labeled using the annotation file (Y). The resulting paired data (X, Y) for each gene is then compiled for model training (80% of the sequences) and testing (20% of the sequences). **(B)** Workflow of the $\text{OSAI}_{\text{MANE}}$ 10,000 model. Input sequences are one-hot encoded and padded with 5,000 Ns ([0,0,0,0]) on each side, totaling 10,000 Ns. The model processes the input and outputs, for each position, the probability of that position being a donor site, an acceptor site, or neither. **(C-D)** Performance comparison between $\text{OSAI}_{\text{MANE}}$ and SpliceAI-Keras on splicing donor and acceptor

sites, trained with 80nt, 400nt, 2,000nt, and 10,000nt flanking sequences. Evaluation metrics include top-1 accuracy for both donor and acceptor sites. Blue curves represent SpliceAI-Keras, while orange curves represent $OSAI_{MANE}$. Each dot represents the average score with error bars indicating ± one standard error. Performance is compared across test datasets from human. **(E)** Benchmarking results for elapsed time, average memory usage, and GPU peak memory for the `prediction` submodule.

Retraining models with different species using OpenSpliceAI framework

To assess whether SpliceAI can generalize across different species and to demonstrate the ease of retraining models with OpenSpliceAI, we selected four model organisms representing diverse taxa: a mammal (mouse, *Mus musculus*), an insect (honeybee, *Apis mellifera*), a freshwater fish (zebrafish, *Danio rerio*), and a flowering plant (thale cress, *Arabidopsis thaliana*). Using OpenSpliceAI with the same training hyperparameters as the human model, we trained species-specific models that we designated $OSAI_{Mouse}$, $OSAI_{Zebrafish}$, $OSAI_{Honeybee}$, and $OSAI_{Arabidopsis}$ (see Table 1and Supplemental Note S1-S5).

Training and test sets for each species were generated using the "`create-data`" submodule (Methods). Due to differences in genome sizes across species, we report the number of protein-coding genes used for training and testing in each model (Figure 3A), along with statistics on the ratio of canonical to non-canonical splice sites (Figure S7A) and intron length distributions (Figure S7B) for human-MANE and the four other selected species. To ensure that the test sets did not contain paralogs of the training sets, OpenSpliceAI aligns them using minimap2 (Li, 2018) and excludes test sequences with over 80% similarity and coverage to the training sequences, and enforces a 20% sequence diversity threshold (see Methods). In its original paper, SpliceAI was evaluated on a test set containing genes from human chromosomes 1, 3, 5, 7, and 9, which the Ensembl database (http://grch37.ensembl.org/biomart/martview) classifies as free of paralogs. However, applying our paralog removal criteria, we found that 0.71% of the MANE transcripts from these chromosomes were paralogous to training set sequences. In other species, the proportion of removed paralogous sequences was 3.86% for mouse, 31.97% for zebrafish, 0.08% for honeybee, and 2.26% for *Arabidopsis* (Figure 3B).

As we did with $OSAI_{MANE}$, we retrained each species model five times using different random seeds (10–14) and evaluated performance based on top-1 accuracy, F1 scores, precision, recall, and AUPRC for donor and acceptor sites. The results for $OSAI_{Mouse}$, $OSAI_{Zebrafish}$, $OSAI_{Honeybee}$, and $OSAI_{Arabidopsis}$ are shown in Figure 3 and Figures S8–S11. We calculated the average percentage improvement across all flanking sequence sizes for donor and acceptor sites under four flanking sequence lengths. On average, OSAI outperformed SpliceAI-Keras by 2% in mouse, 54% in honeybee, 19% in zebrafish, and 57% in *Arabidopsis*.

The human and mouse genomes share a majority of their protein-coding genes (Mouse Genome Sequencing Consortium, 2002). This conserved evolutionary relationship likely explains the comparable performance of $OSAI_{Mouse}$ and SpliceAI-Keras. In contrast, OSAIs that had been re-trained substantially outperformed SpliceAI-Keras in more distantly related species, particularly in honeybee, zebrafish, and *Arabidopsis* (see Discussion).

Table 1. Genome assembly and annotation details for species used for OpenSpliceAI training and transfer-learning in this study. Note: For each species, the table includes the GenBank accession number, assembly name, ftp sites for assembly and annotation downloads, and annotation release dates.

Species	Name	Genbank accession	Download link	Annotation Release date
Homo sapiens	GRCh38.p14	GCA_000001405.29	https://ftp.ncbi.nlm.nih.gov/genomes/all/annotation_releases/9606/GCF_000001405.40-RS_2023_03/	21-March-2023
Mus musculus	GRCm39	GCA_000001635.9	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/635/GCF_000001635.27_GRCm39/	08-February-2024
Apis mellifera	Amel_HAv3.1	GCA_003254395.2	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/003/254/395/GCF_003254395.2_Amel_HAv3.1/	30-September-2022
Arabidopsis thaliana	TAIR10.1	GCA_000001735.2	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/735/GCF_000001735.4_TAIR10.1/	16-June-2023
Danio rerio	GRCz11	GCA_000002035.4	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/002/035/GCF_000002035.6_GRCz11/	15-August-2024

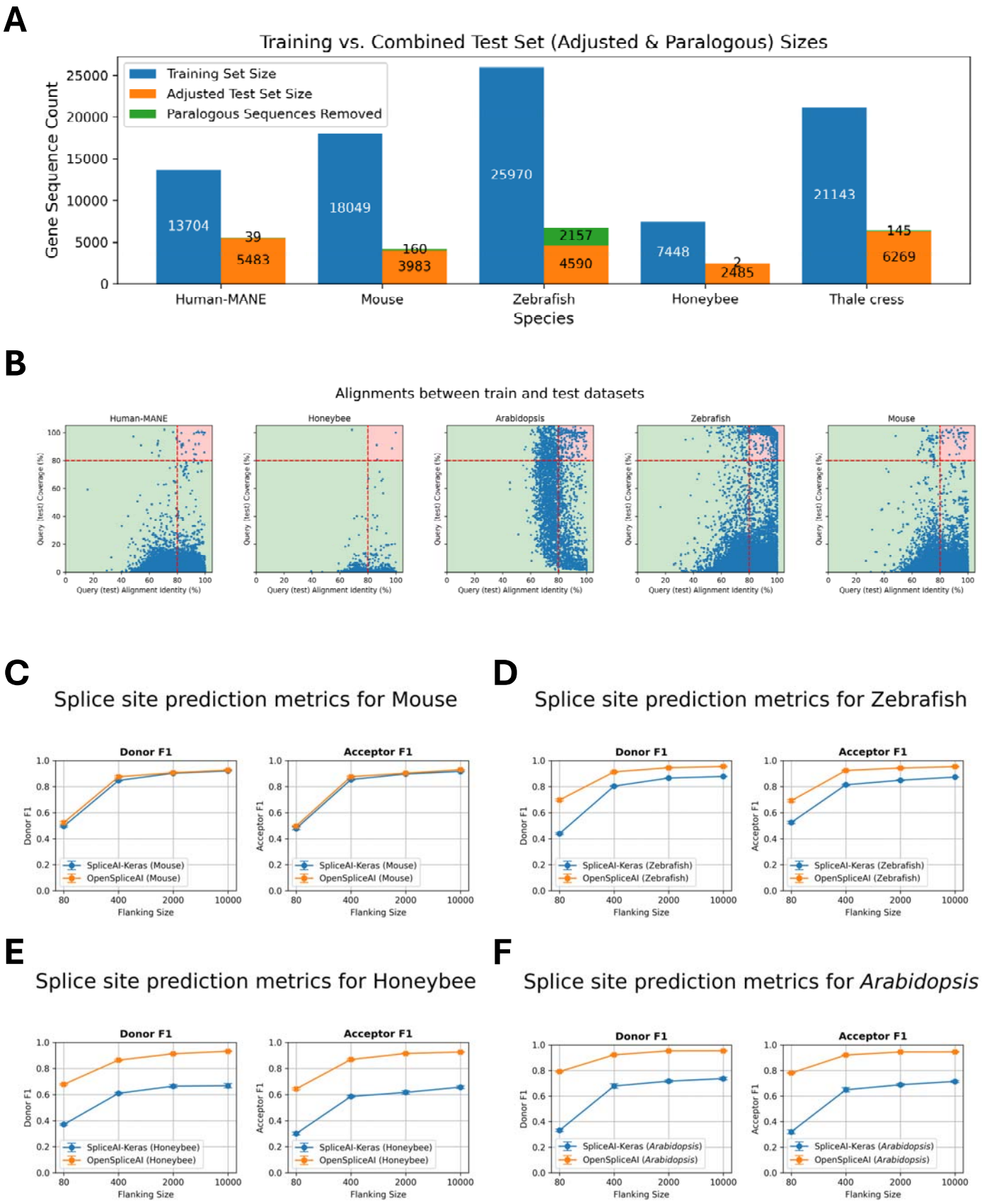


Figure 3 **(A)** The number of protein-coding genes in the training and test sets, along with the count of paralogous genes removed for each species: Human-MANE, mouse, zebrafish, honeybee, and *Arabidopsis*. **(B)** Scatter plots of DNA sequence alignments between testing and training sets for Human-MANE, mouse, honeybee, zebrafish, and *Arabidopsis*. Each dot

represents an alignment, with the x-axis showing alignment identity and the y-axis showing alignment coverage. Alignments exceeding 80% for both identity and coverage are highlighted in the red-shaded region and excluded from the test sets. **(C-F)** Performance comparisons of OSAIs trained on species-specific datasets (mouse, zebrafish, honeybee and *Arabidopsis*) versus SpliceAI-Keras, original published SpliceAI models, trained on human data. The orange curves represent OSAI metrics, while the blue curves show SpliceAI-Keras metrics. Each subplot **(C-F)** includes F1 score evaluated separately for donor and acceptor sites.

Adapting $\mathbf{OSAI_{MANE}}$ to different species via transfer learning

Transfer learning can improve model performance by leveraging knowledge from related source domains (Zhuang et al., 2021). In the context of splice site prediction, we tested whether $OSAI_{MANE}$, initially trained on human splice annotations, could be effectively adapted to predict splice sites across other species.

We evaluated four species – *M. musculus*, *A. mellifera*, *D. rerio*, and *A. thaliana* – by fine-tuning five distinct pre-trained $OSAI_{MANE}$ models for each species. For every species, each pre-trained model was fine-tuned using the same training and test datasets, yielding five transfer-trained variants. These variants – collectively referred to as $OSAI_{Mouse}$-transfer, $OSAI_{Honeybee}$-transfer, $OSAI_{Zebrafish}$-transfer, and $OSAI_{Arabidopsis}$-transfer – were directly compared with models trained from scratch ($OSAI_{Mouse}$, $OSAI_{Honeybee}$, $OSAI_{Zebrafish}$, and $OSAI_{Arabidopsis}$) to assess the benefits of transfer learning. For each species, we trained transfer-learned models using flanking sequences of 80 nt, 400 nt, 2,000 nt, and 10,000 nt. We then compared the performance of transfer-trained and scratch-trained models by evaluating top-1 accuracy for donor and acceptor splice site predictions across four lengths of flanking sequences: 80, 400, 2000, and 10000 nt (Figure 4A-D). Full results, including top-1 accuracy, F1 score, and AUPRC for donor and acceptor splice sites are provided in supplementary Figures S12-S15.

Across all configurations, transfer-trained models consistently outperformed scratch-trained models in both accuracy and training stability, as evidenced by higher top-1 accuracies and lower standard errors across the five pre-trained models. Notably, transfer-trained models achieved near-optimal performance after just one epoch, while scratch-trained models required ten epochs to reach comparable results and showed substantial performance gaps between one and ten epochs (Figure 4E–H). Moreover, transfer learning also solved a convergence issue for one data set: in the *A. thaliana* scratch-training experiments using 10k flanking sequences, the CosineAnnealingWarmRestarts scheduler led to unstable optimization. Although switching to MultiStepLR with learning rate decay improved stability, convergence still took a very long time. Notably, transfer learning did not display this problem. See the Methods section for training parameters.

After ten epochs, transfer-trained models slightly outperformed their scratch-trained counterparts for *Arabidopsis* and honeybee, the two species with the smallest genome sizes among those tested. These results suggests that pretraining improves generalization, particularly for compact genomes.

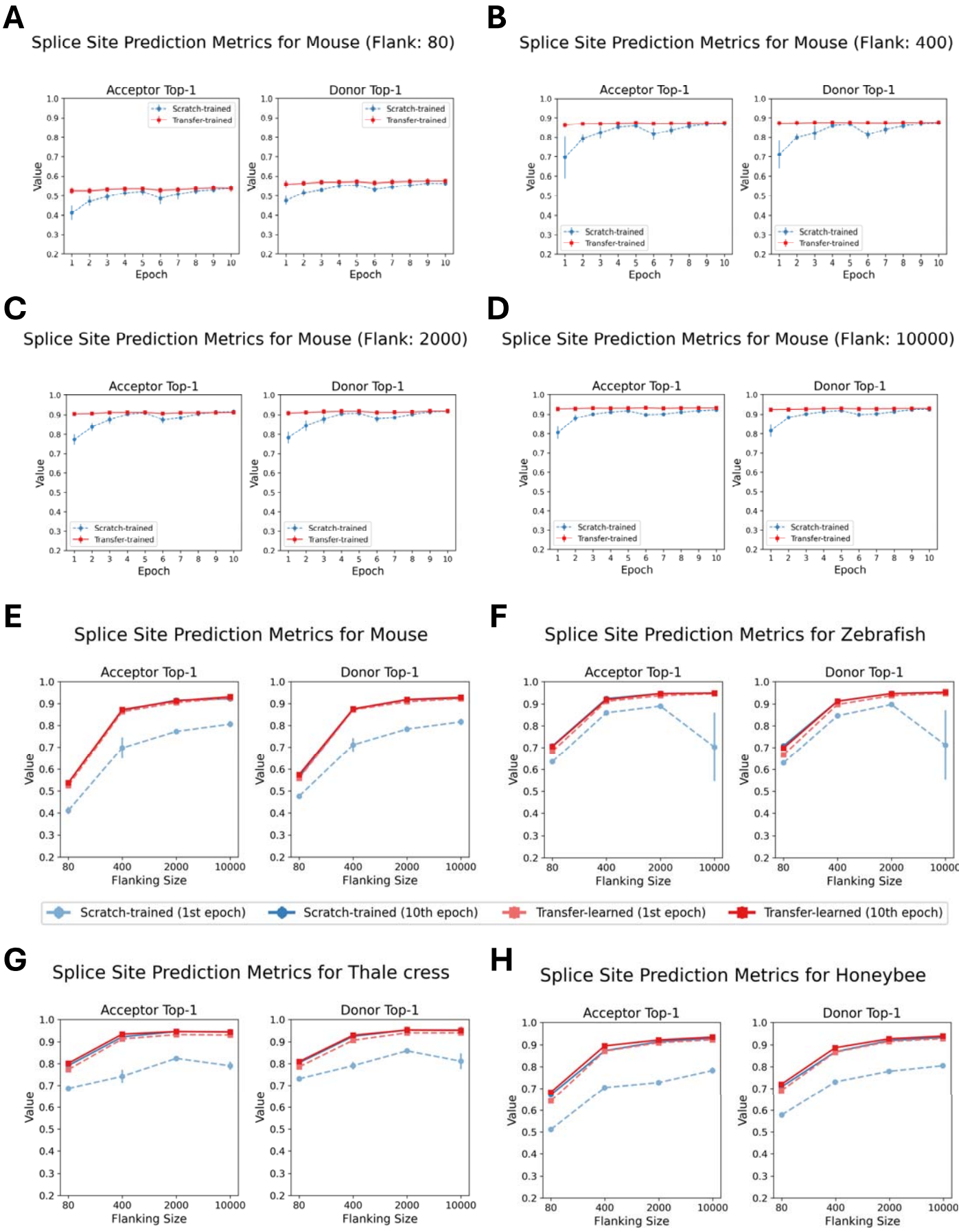

A
Splice Site Prediction Metrics for Mouse (Flank: 80)
Acceptor Top-1
Donor Top-1
Scratch-trained
Transfer-trained
Value
Epoch
B
Splice Site Prediction Metrics for Mouse (Flank: 400)
C
Splice Site Prediction Metrics for Mouse (Flank: 2000)
D
Splice Site Prediction Metrics for Mouse (Flank: 10000)
E
Splice Site Prediction Metrics for Mouse
F
Splice Site Prediction Metrics for Zebrafish
Flanking Size
Scratch-trained (1st epoch)
Scratch-trained (10th epoch)
Transfer-learned (1st epoch)
Transfer-learned (10th epoch)
G
Splice Site Prediction Metrics for Thale cress
H
Splice Site Prediction Metrics for Honeybee

Figure 4 Performance comparison of scratch-trained and transfer-trained OSAIs across species and sequence lengths. **(A-D)** Top-1 accuracy for donor and acceptor splice sites of 80 nt, 400 nt, 2,000 nt, and 10,000 nt models, comparing $OSAI_{Mouse}$ (scratch-trained) and $OSAI_{Mouse}$-transferred (transfer-trained) models over epochs 1 to 10 on the test dataset. **(E-H)** Top-1 accuracy after one epoch of training versus after ten epochs for both scratch-trained and transfer-trained models across the same sequence lengths. Each plot represents one species and its corresponding transfer-trained model: **(E)** $OSAI_{Mouse}$ vs. $OSAI_{Mouse}$-transferred, **(F)** $OSAI_{Zebrafish}$ vs. $OSAI_{Zebrafish}$-transferred, **(G)** $OSAI_{Arabidopsis}$ vs. $OSAI_{Arabidopsis}$-transferred, and **(H)** $OSAI_{Honeybee}$ vs. $OSAI_{Honeybee}$-transferred.

Calibrating OpenSpliceAI models

Model calibration helps align predicted probabilities with the true likelihood of observed outcomes, thereby mitigating the risk of overconfident or underconfident predictions. Here, we applied class-wise temperature scaling, a single-parameter variant of Platt scaling, to adjust each class's predicted probabilities without altering the model's classification performance (see Methods). We calibrated $OSAI_{MANE}$ models on the validation set and subsequently evaluated them on the test set.

We then compared $OSAI_{MANE}$ models before and after calibration using reliability diagrams (Figure 5A and Figure S16), which show reliability curves for non-splice, acceptor, and donor sites of $OSAI_{MANE}$ trained with flanking sequence lengths of 80, 400, 2000, and 10,000 nt, with the calibration temperature (Ts) in the legend. Calibration quality was quantified using negative log-likelihood (NLL) loss and expected calibration error (ECE). For each species, metrics were averaged over five calibrated models, and the results indicated slight improvements in both measures following calibration (Figure 5B, Figure S17-S18; see Methods). Temperature parameters greater than one indicated overconfidence, whereas values below one indicated underconfidence. After calibration, score distributions for donor and acceptor sites shifted slightly away from extreme values (1 and 0), resulting in smoother probability distributions (Figure 5C).

We observed similar outcomes when calibrating OSAIs on mouse (Figure S19), honeybee (Figure S20), zebrafish (Figure S21), and *Arabidopsis* (Figure S22). Collectively, these results demonstrate that OSAIs were already well-calibrated, and this consistency across species underscores the robustness of OpenSpliceAI's training approach in diverse genomic contexts.

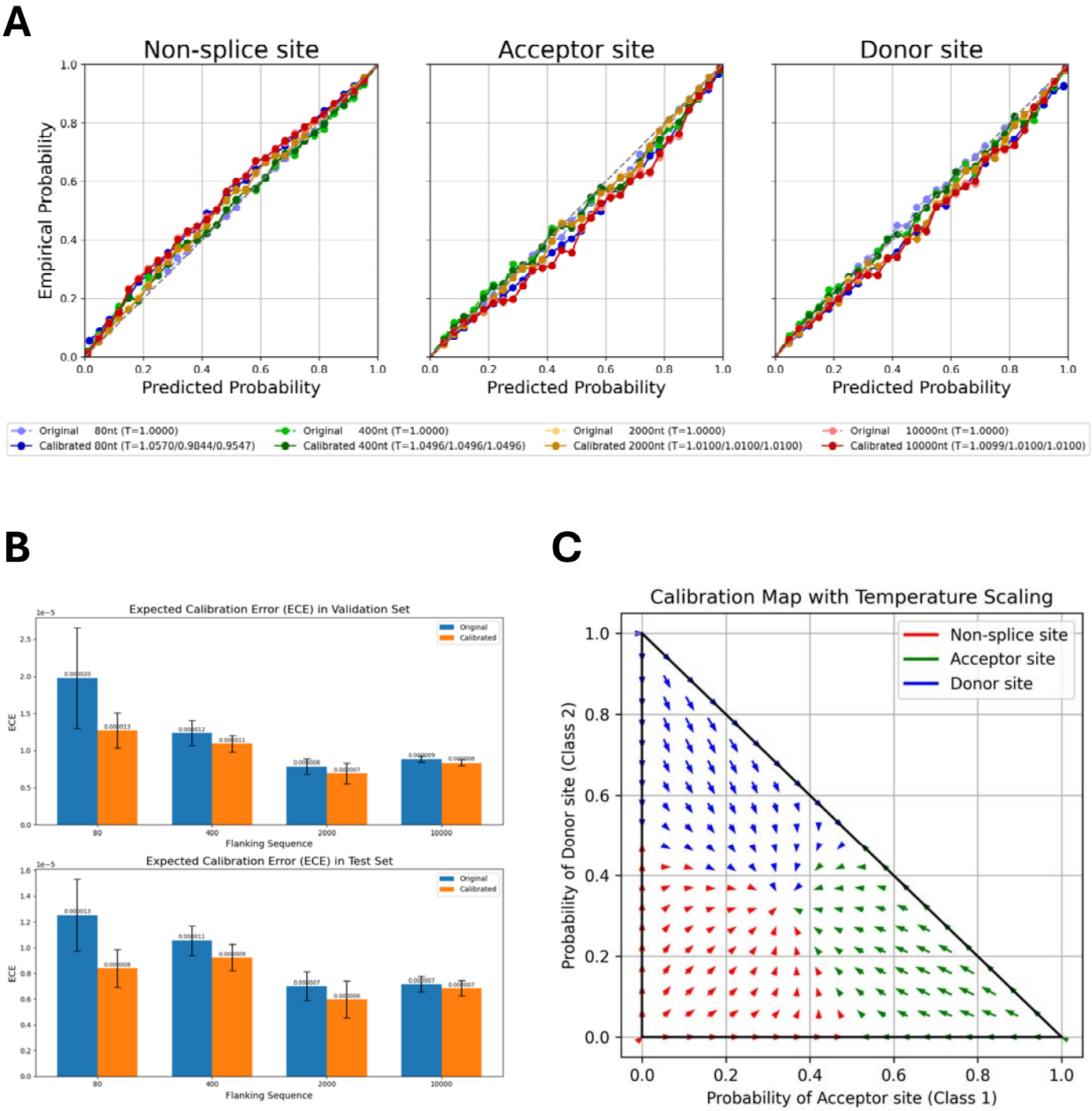


Figure 5 (A) Calibration results for $OSAI_{MANE}$ across non-splice sites, acceptor sites, and donor sites. Models trained with different flanking sequence lengths are represented by color: 80 nt (blue), 400 nt (green), 2,000 nt (orange), and 10,000 nt (red). Dotted curves in lighter colors denote pre-calibration results, while solid curves in darker shades show post-calibration results. **(B)** Expected calibration error (ECE) on the validation set (top) and test set (bottom), comparing the $OSAI_{MANE}$'s performance before (blue bars) and after (orange bars) calibration. For each flanking sequence $OSAI_{MANE}$, five calibration experiments were performed, with the mean loss and ± one standard error. **(C)** Two-dimensional calibration map for $OSAI_{MANE}$, illustrating how raw predicted probabilities for acceptor (x-axis) and donor (y-axis) sites are transformed after calibration. Arrows indicate the shift from pre- to post-calibration states in two-dimensional probability space, resulting in a smoother probability distribution.

Comparing $OSAI_{MANE}$ and SpliceAI via variant effects of *in silico* mutagenesis

A crucial finding of *Jaganathan et al.* (2019) was that SpliceAI was capable of capturing non-local effects of genomic mutations on splice site location and strength. In order to show that $OSAI_{MANE}$ has the same capabilities, we recreated several of their studies, as well as a large-scale *in silico* mutagenesis (ISM) experiment aimed at elucidating the model's learned splice site recognition pattern.

First, we recreated the experiment from Jaganathan *et al*. in which they mutated every base in a window around exon 9 of the U2SURP gene and calculated its impact on the predicted probability of the acceptor site. We repeated this experiment on exon 2 of the DST gene, again using both SpliceAI and $OSAI_{MANE}$. In both cases, we found a strong similarity between the resultant patterns between SpliceAI and $OSAI_{MANE}$, as shown in Figure 6A.

These two examples suggest that OpenSpliceAI and SpliceAI are learning very similar internal models, but we also wanted to run a larger-scale ISM comparison. We randomly sampled 100 donor and 100 acceptor sites from the testing dataset (Chromosomes 1, 3, 5, 7, and 9) and evaluated the difference in the predicted probability of the splice site following a point mutation in a given base within an 80 bp window around the site, for each of the three possible mutations. Figure 6B shows, as a DNA logo, the average decrease in splice site strength for each mutation for both tools. The logos for both donor and acceptor are highly similar between the two tools, with the strongest signals exhibited for mutations at the donor/acceptor sites outside of the canonical GT/AG dinucleotide motif.
and evaluated the difference in the predicted probability of the splice site following a point mutation in a given base within an 80 bp window around the site, for each of the three possible mutations. Figure 6B shows the average decrease in splice site strength for each mutation in the format of a DNA logo, for both tools. Notably, the logos for both donor and acceptor sites show a similar pattern between the two tools, with the strongest signals exhibited for mutations at the donor/acceptor sites outside of the canonical GT/AG dinucleotide motif. The acceptor sites additionally show relatively higher sensitivity to A and G just upstream of the acceptor site, which is expected due to the CT-richness of the polypyrimidine tracts common in this region (Majewski and Ott, 2002).

Jaganathan *et al*. also demonstrated SpliceAI's ability to predict cryptic splicing mutations – intronic mutations that create alternatively spliced transcripts. We recreated their experiment in which they investigated a MYBPC3 intron mutation associated with cardiomyopathy. Both $OSAI_{MANE}$ and SpliceAI predict very similar changes in the location and strength of acceptor site gain and loss events (Figure 6C). We then extended this experiment by examining an intronic splicing mutation in the OPA1 gene which has been shown to cause alternative splicing of a cryptic pseudoexon upstream (Qian et al., 2021). Again, both $OSAI_{MANE}$ and SpliceAI correctly predicted this event with similarly high accuracy.

We then replicated Jaganathan *et al.*'s experiment on the CFTR gene, in which they showed that SpliceAI predicted all of the splice sites accurately without any false positives. Using the full gene sequence from the GRCh38 assembly and a score threshold of 0.5, we found that $OSAI_{MANE}$ and SpliceAI predict the

exact same set of donor and acceptor sites, and accurately capture all but the first donor site, using the MANE Select annotation as reference (Figure 6D).

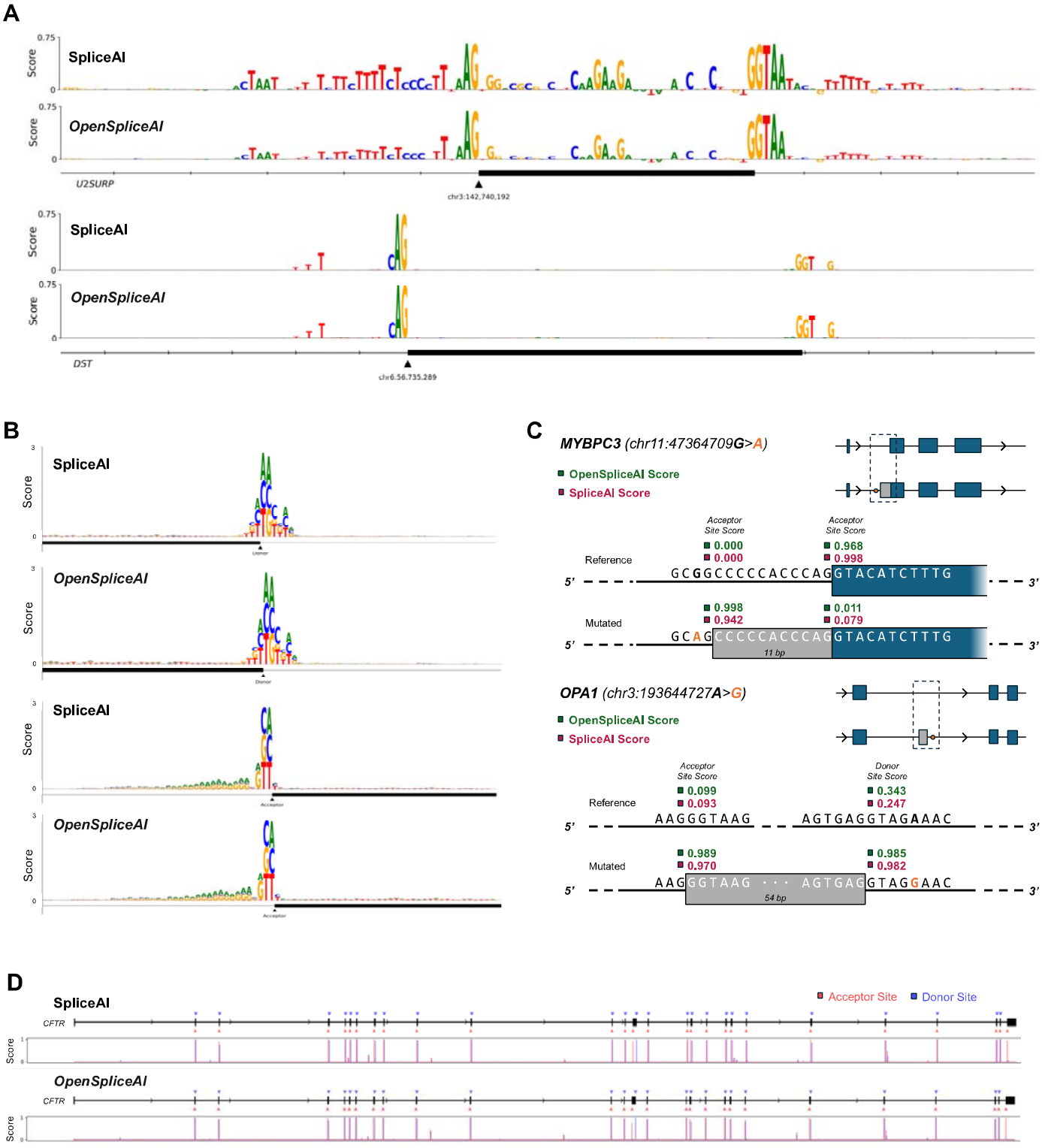


Figure 6 (A) Plot of importance scores for nucleotides near the acceptor site of exon 9 of U2SURP (top) and DST (bottom), for both SpliceAI and $OSAI_{MANE}$. The importance score is calculated by taking the average decrease in acceptor site score across the three possible point mutations at a given base position. **(B)** Plot of the impact of each possible point mutation within 80 bp of a donor (top) site or acceptor (bottom) site, for both SpliceAI and $OSAI_{MANE}$. The impact is the raw decrease in predicted splice

site score after mutating a given base to a different one. **(C)** Visualization of cryptic splicing variants being predicted for the MYBPC3 gene (top), with an acceptor site gain and loss event, from SpliceAI's original analysis, and the OPA1 gene (bottom), where a cryptic exon inclusion event was recently reported (Qian et al., 2021). **(D)** Predicted splice sites for the entire CFTR gene, with the corresponding predicted probability distribution by base position plotted below, for both SpliceAI and $OSAI_{MANE}$.

Discussion

We developed OpenSpliceAI to be a modular Python toolkit designed as an open-source implementation of SpliceAI, to which we added several key enhancements. The framework replicates the core logic of the SpliceAI model while optimizing prediction efficiency and variant effect analysis, such as acceptor and donor gains or losses, using pre-trained models. Our benchmarks show substantial computational advantages over SpliceAI, with faster processing, lower memory usage, and improved GPU efficiency (Figure 2B and Figure S6). These improvements are driven by our optimized PyTorch implementation that employs dynamic computation graphs and on-demand GPU memory allocation – allowing memory to be allocated and freed as needed – in contrast to SpliceAI's static, Keras-based TensorFlow approach, which pre-allocates memory for the worst-case input size. In SpliceAI, this rigid memory allocation leads to high memory overhead and frequent out-of-memory errors when handling large datasets through large loop iteration prediction. Additionally, OpenSpliceAI leverages streamlined data handling and enhanced parallelization through batch prediction and multiprocessing, automatically distributing tasks across available threads. Together, these features prevent the memory pitfalls common in SpliceAI and make OpenSpliceAI a more scalable and efficient solution for large-scale genomic analysis (see Supplemental Note S6).

It is important to note that even though OpenSpliceAI and SpliceAI share the same model architecture, the released trained models are not identical. The variability observed between our models and the original SpliceAI – and even among successive training runs using the same code and data – can be attributed to several sources of inherent randomness. First, weight initialization is performed randomly for many layers, which means that different initial weights can lead to distinct convergence paths and final model parameters. Second, the process of data shuffling alters the composition of mini-batches during training, impacting both the training dynamics and the statistics computed in batch normalization layers. Although batch normalization is deterministic for a fixed mini-batch, its reliance on batch statistics introduces variability due to the random sampling of data. Finally, OpenSpliceAI employs the AdamW optimizer (Loshchilov and Hutter, 2019), which incorporates exponential moving averages of the first and second moments of the gradients. This mechanism serves a momentum-like role, contributing to an adaptive learning process that is inherently stochastic. Moreover, subtle differences in the order of operations or floating-point arithmetic, particularly in distributed computing environments, can further amplify this stochastic behavior. Together, these factors contribute to the observed non-deterministic behavior, resulting in slight discrepancies between our trained models and the original SpliceAI, as well as among successive training sessions under identical conditions.

OpenSpliceAI empowers researchers to adapt the framework to many other species by including modules that enable easy re-training. For closely related species such as mouse, our re-trained model demonstrated

comparable or slightly better precision than the human-based SpliceAI model. For more distant species such as *Arabidopsis thaliana*, whose genomic structure differs substantially from humans, retraining OpenSpliceAI yields much greater improvements in accuracy. Our initial release includes models trained on the human MANE genome annotation and four additional species: mouse, zebrafish, honeybee, and *A. thaliana*.

OpenSpliceAI also includes modules for transfer learning, allowing researchers to initialize models with weights learned on other species. In our transfer learning experiments, models transferred from human to other species displayed faster convergence and higher stability, with potential for increased accuracy. We also incorporate model calibration via temperature scaling, providing better alignment between predicted probabilities and empirical distributions.

The ISM study revealed that $\text{OSAI}_{\text{MANE}}$ and SpliceAI made predictions using very similar sets of motifs (Figure 6B). Across several experiments, we note that SpliceAI exhibits an inherent bias near the starts and ends of transcripts which are padded with flanking N's (as was done in the original study), predicting donor and acceptor sites in these boundaries with an extremely high signal that disappears when the sequence is padded with the actual genomic sequence. For example, the model correctly predicted the first donor site of the CFTR gene when the gene's boundaries were flanked with N's; however, when replaced those N's with the actual DNA sequence upstream of the gene boundary, the signal all but disappeared, as seen in Figure 6D. This suggests a bias resulting from the way the model is trained. In our ISM benchmarks, we thus chose not to use flanking N's unless explicitly recreating a study from the original SpliceAI paper.

Additionally, we note that both the SpliceAI and $\text{OSAI}_{\text{MANE}}$ "models" are the averaged result of five individual models, each initialized with slightly different weights. During the prediction process, each individual model was found to have discernably different performance. By averaging their outputs leveraging the deep ensemble approach (Fort et al., 2019; Lakshminarayanan et al., 2017), the overall performance of both SpliceAI and OpenSpliceAI improved while reducing sensitivity to local variations. In essence, this method normalizes the inherent randomness of the individual models, resulting in predictions that are more robust and better represent the expected behavior, ultimately yielding improved average performance across large datasets. OpenSpliceAI's "`predict`" module averages across all five models by default, but it also supports prediction using a single model.

In summary, OpenSpliceAI is a fully open-source, accessible, and computationally efficient deep learning system for splice site prediction. Its modular architecture, enhanced performance, and adaptability to diverse species make it a powerful tool for advancing research on gene regulation and splicing across diverse species.

Online Methods

OpenSpliceAI is designed with modular subcommands that allow users to preprocess genomic data into training and test sets, train models, perform model calibration, make efficient predictions, and conduct variant analysis. The following sections summarize the usage and technical implementation of each subcommand.

OpenSpliceAI `create-data` subcommand

The `create-data` subcommand converts standard genomics data formats into a machine-readable form suitable for training machine learning models. It processes genomic sequences (FASTA) and genome annotations (GFF/GTF) to produce gene sequences and splice site labels stored in Hierarchical Data Format version 5 (HDF5).

In this standard supervised sequence-to-sequence machine learning framework, a dataset comprising input features (Xs) and corresponding labels (Ys) is constructed. Here, X represents the one-hot-encoded pre-mRNA sequences, which serve as the input variables for prediction, while Y denotes the labels, specifically the donor and acceptor splice sites derived from the genome annotations.

For each gene locus, the longest transcript is selected as the canonical transcript, consistent with the canonical-transcript-labeling approach of SpliceAI. By default, the "`--biotype`" argument is set to "protein-coding", which means only protein-coding genes are included in the feature set and label set. Users can change this setting to "all" to include both protein-coding and non-coding genes.

Splitting gene sequences into training and testing sets.

For generating datasets for $\text{OSAI}_{\text{MANE}}$, OpenSpliceAI adopts SpliceAI's chromosome-based partitioning strategy. In human datasets, the training set is defined by chromosomes 2, 4, 6, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, X, and Y, while the testing set is defined by chromosomes 1, 3, 5, 7, and 9. For non-human species, OpenSpliceAI defaults to a random splitting method (specified by the "`--split-method`" parameter). In this approach, the algorithm first computes the total chromosome length, randomly shuffles the chromosomes, and then iteratively assigns them to the training or testing set until the desired split ratio (defaulting to 80% training) is achieved, with any remaining chromosomes allocated to the test set.

Pseudogenes and paralogous gene sequences removal

To ensure the integrity and accuracy of model testing, pseudogenes – segments of DNA that resemble functional genes but are incapable of coding for a protein – are removed from the test dataset. This is accomplished by filtering out genes in the GFF file that either have "pseudogene" as the feature type in the third column or specify "pseudogene", "transcribed_pseudogene", or "processed_pseudogene" for the "gene_biotype" or "biotype" fields.

The removal of paralogous genes is also critical, as sequence similarity between training and test sets can lead to data leakage. OpenSpliceAI performs DNA sequence alignment to detect paralogous sequences. Specifically, OpenSpliceAI uses mappy, a Python wrapper for minimap2 (Li, 2018), to align test sequences to the training set, applying the "`--asm20`" argument to allow a sequence divergence of up to 20%. Following alignment, OpenSpliceAI examines each result and excludes any test sequence that shows more than 80% sequence similarity and 80% coverage compared to any sequence in the training set.

One-hot encoding scheme

The one-hot encoding procedure for the input sequence for model training, testing, and prediction uses the following representation: $A = [1, 0, 0, 0]$, $C = [0, 1, 0, 0]$, $G = [0, 0, 1, 0]$, $T\ (or\ U) = [0, 0, 0, 1]$. Any ambiguous nucleotide (denoted as N or other non-standard symbols) is encoded as $[0, 0, 0, 0]$. The encoding of the labels for model training uses the scheme: none-splice site $= [1, 0, 0]$, acceptor site $= [0, 1, 0]$, donor site $= [0, 0, 1]$, padding $= [0, 0, 0]$. The predictions use the same scheme, where the three output channels sum to one, representing a probability score.

Gene sequence segmentation for one-hot encoded features (Xs) and labels (Ys)

Following the approach used by SpliceAI, OpenSpliceAI divides gene sequences into overlapping segments, each spanning 15,000 nucleotides. Each segment comprises a central region of 5,000 nucleotides, flanked on both sides by 5,000-nucleotide extensions, thereby providing essential upstream and downstream context. A step size of 5,000 nucleotides is used to ensure comprehensive coverage with overlapping windows. For instance, a gene that is 22,000 nucleotides long is partitioned into five segments. Each segment is represented as a tensor with dimensions $(15{,}000, 4)$; when a segment lacks sufficient nucleotides, the remaining positions are padded with 'N' bases to maintain uniform tensor dimensions. Notably, the final segment may contain fewer real nucleotides – only 2,000 in this example – with the deficit filled by padding. Thus, the one-hot-encoded feature matrix (X) for the gene has a shape of $(5, 15{,}000, 4)$, while the corresponding label tensor (Y), which focuses on the central 5,000 nucleotides, has a shape of $(5, 5{,}000, 3)$. The batch size is set to 100 by default, meaning that OpenSpliceAI concatenates the first dimension of each tensor from 100 genes into a single matrix and performs the same concatenation for the label tensor matrix.

Selecting splice sites for inclusion in dataset labels (Ys)

Even with curated annotations, some splice sites in the annotation file may still be misannotated. To improve the accuracy of splice site labeling, OpenSpliceAI provides a "`--canonical-only`" argument that restricts analysis to canonical splice sites. By default, this option is disabled, so all splice sites in the annotation file are evaluated. These include the U2-snRNP-type motifs "GT-AG" and "GC-AG" (Brow, 2002) and the U12-snRNP-type motifs "GT-AG" and "AT-AC" (Frilander and Steitz, 1999; Patel and Steitz, 2003; Wassarman and Steitz, 1992).

OpenSpliceAI `train` subcommand

After the training and test sets are created, this subcommand takes the HDF5 outputs from the `create-data` subcommand and enables users to train their OpenSpliceAI model. Users can train different OpenSpliceAI models with various flanking sequence lengths, including 80 nt, 400 nt, 2,000 nt, and 10,000 nt.

OpenSpliceAI adaptive learning

OpenSpliceAI uses the AdamW optimizer (Loshchilov and Hutter, 2019) with a default learning rate of 0.001. The training dataset is further split into 90:10 for training and validation. By default, OpenSpliceAI trains a model for 10 epochs, with an early stopping patience of 2.

The "`--scheduler`" argument enables users to choose between two built-in PyTorch learning rate schedulers – MultiStepLR and CosineAnnealingWarmRestarts (Loshchilov and Hutter, 2016)– to dynamically adjust the learning rate during training. By default, OpenSpliceAI employs "MultiStepLR" with a learning rate of 0.001, beginning with a 0.5 decay from the 6th epoch, the same approach used in the SpliceAI model.

As detailed in the Results section, training the OSAI model on *Arabidopsis* data using the MultiStepLR scheduler resulted in a more stable training process. In contrast, all other OpenSpliceAI models, except $\text{OSAI}_{\text{Arabidopsis}}$, were trained using the "CosineAnnealingWarmRestarts" scheduler, configured with "`T_0=5`", "`T_mult=1`", "`eta_min=1e-5`", and "`last_epoch=-1`". This scheduler gradually reduces the learning rate from an initial value of 1e-3 to a minimum of 1e-5 in a smooth, wave-like (cosine) pattern over each cycle. The parameter "`T_0=5`" sets the initial period for the cosine decay, meaning the learning rate completes one full cycle – from the starting rate down to "`eta_min`" and back – within 5 epochs. After the first cycle, the learning rate "restarts" at its initial value, creating a "warm restart".

OpenSpliceAI loss function

By default, OpenSpliceAI uses the categorical cross-entropy loss function (Equation 1) to compute the loss at every nucleotide position in the input DNA sequence. This loss function measures the discrepancy between the predicted probability distribution and the true distribution for each position, which is standard practice for multi-class classification tasks. Alternatively, users can opt for the focal loss (Equation 2) (Lin et al., 2018).

Focal loss enhances the standard cross-entropy loss by adding a modulating term, $(1 - P_{class})^{\gamma}$, where P_{class} represents the model's predicted probability for the correct class. This term down-weights the loss assigned to well-classified examples, allowing the model to concentrate more on the misclassified or harder-to-classify cases. For instance, setting γ to 2 amplifies the focus on challenging predictions, which is particularly beneficial in scenarios with class imbalance or when the signal in the data is subtle. This dynamic weighting of loss can enhance overall model accuracy.

$$Loss_{CEL} = \sum_{class \in \{donor, acceptor, neither\}} I_{class} \times \log(P_{class}) \quad \ldots(1)$$

$$Loss_{FL} = \sum_{class \in \{donor, acceptor, neither\}} I_{class} \times (1 - P_{class})^{\gamma} \times log(P_{class}) \ , \quad where\ \gamma = 2 \quad \ldots(2)$$

OpenSpliceAI `transfer` subcommand

Instead of training a model entirely from scratch, users can leverage transfer learning to adapt a human-trained model for a target species using the `transfer` subcommand. This process resembles standard model training but starts from a pre-trained model specified with the "`--pretrained-model`" argument. We recommend $OSAI_{MANE}$ as the pre-trained base model. Once the pre-trained weights are loaded, the `transfer` subcommand enables flexible fine-tuning. Users can either unfreeze all layers (using the "`--unfreeze-all`" flag) or selectively train the final layers (with "`--unfreeze <INT>`") to adapt the model more effectively to the target species data. In addition, similar to the `train` subcommand, the `transfer` subcommand integrates adaptive learning rate scheduling and early stopping to optimize convergence and prevent overfitting, all while using the same loss function configuration.

While transfer learning employs the same underlying OpenSpliceAI architecture, optimizer, scheduler, and loss function as training from scratch, it differs primarily in its initialization step, which is based on a fully trained model. For optimal results, we recommend selecting species with high-quality genome assemblies and comprehensive annotations, such as *Homo sapiens*. This approach substantially reduces training time and can improve accuracy on the target species. See **Figure S12-S15** for further details.

OpenSpliceAI `calibrate` subcommand

One improvement in OpenSpliceAI over SpliceAI is the incorporation of model calibration, which refines model-predicted probabilities to align more closely with actual outcome likelihoods. This is achieved by calibrating the model's output so that a prediction with a probability of 0.6, for example, accurately reflects a 60% chance of being correct. The `calibrate` subcommand evaluates scores around this value, identifies deviations from expected probabilities, and applies non-linear adjustments to correct the score distribution without altering the model's performance. After the OpenSpliceAI model was trained, we used the validation dataset to calibrate the model and evaluated the calibrated results using the test dataset. Such calibration is crucial in predictive modeling, particularly for classification, as it ensures that predicted probabilities are consistent with observed outcomes. Uncalibrated models can be overconfident or underconfident, potentially compromising decision-making quality.

There are various methods for calibrating models, including Platt Scaling (Platt, 1999), isotonic regression (Zadrozny and Elkan, 2002), and histogram binning (Zadrozny and Elkan, 2001). Here, we implemented class-wise temperature scaling, a variant of Platt Scaling often used in knowledge distillation and statistical mechanics (Hinton et al., 2015; Jaynes, 1957). Temperature scaling is a post-hoc adjustment that modifies model output probabilities to more accurately reflect true class likelihoods.

OpenSpliceAI calibration optimization procedure

For model calibration in OpenSpliceAI, we froze the trained OpenSpliceAI model weights and augment the network with a class-specific temperature scaling layer. Instead of using a single scalar temperature parameter, we employ a vector of temperature parameters $T = [T_0, T_1, T_2]$ corresponding to the non-splice site, acceptor site, and donor site, respectively. This design allows each class's logit to be scaled individually, thereby addressing the inherent class imbalance and the sparsity of splice site signals. The logits are divided by their corresponding temperature parameters before applying the softmax function (Bridle, 1989), thereby aligning the predicted probabilities with the empirical likelihoods.

The temperature vector T was optimized using the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.01 for its adaptive capabilities. To further enhance convergence, we employ PyTorch's ReduceLROnPlateau scheduler, which reduces the learning rate by a factor of 0.1 if the validation loss does not improve over two consecutive epochs. In addition, early stopping was implemented with a patience of two epochs and a minimum improvement threshold (delta) of 10^{-6}. If the validation loss did not decrease by at least 10^{-6} over two epochs, optimization halts early, ensuring calibration efficiency and preventing overfitting.

Temperature scaling modifies the logits z (the raw outputs of the model before the softmax function) by scaling them with T (Equation 3). The adjusted logits z' are computed as:

$$z' = \frac{z}{T} \qquad \ldots(3)$$

where z represents the original logits. The calibrated probabilities $\hat{p}$ are then obtained by applying the softmax function (Equation 4):

$$\hat{p} = softmax(z') \qquad \ldots(4)$$

A higher temperature $T > 1$ spreads out the probability distribution, reducing confidence, while a lower temperature $T < 1$ sharpens it, increasing confidence. To optimize calibration, we use the negative log-likelihood (NLL) loss function defined as (Equation 5):

$$\mathcal{L}_{NLL} = -\sum_{i=1}^{N}\sum_{c=1}^{C} I_{i,c} \log(\hat{p}_{i,c}) \qquad \ldots(5)$$

Where N is the number of samples, C is the number of classes, in our case, acceptor site, donor site, and non-splice site. $I_{i,c}$ is the indicator function, which equals 1 if sample i belongs to class c and 0 otherwise. $\hat{p}_{i,c}$ is the model's predicted probability of the sample i belongs to class c. A lower NLL indicates that the predicted probabilities are more closely aligned with the true labels, reflecting better calibration and overall model performance.

The optimal temperature T^* is determined by minimizing the NLL loss over the validation dataset (Equation 6):

$$T^* = \arg\min_T \mathcal{L}_{NLL}(\hat{p}, \mathrm{y}) \quad \ldots(6)$$

where y are the true labels.

The temperature parameter T was initialized to one and constrained between 0.05 and 5.0 to prevent extreme scaling. We employed gradient-based optimization to minimize the cross-entropy loss (nn.CrossEntropyLoss, i.e., NLL loss) on the validation set while keeping the original model weights fixed.

OpenSpliceAI calibration evaluation

The first metric, which we used for temperature optimization, is NLL (Equation 5), which measures the match between predicted probabilities and the true labels; lower NLL values indicate better calibration.

We also evaluated the temperature-scaled new probabilities using the Expected Calibration Error (ECE) (Equation 7). ECE quantifies the discrepancy between confidence estimates and actual accuracy over a range of probability bins. It does so by partitioning the predictions into M bins, by default OpenSpliceAI sets to 30, and computing the weighted average of the absolute differences between the confidence (predicted probability) and accuracy within each bin. The ECE is defined as:

$$\mathcal{L}_{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} |acc(B_m) - conf(B_m)| \quad \ldots(7)$$

B_m is the set of indices of samples whose predicted probabilities fall into the mth bin; $|B_m|$ is the number of samples in the mth bin; N is the total number of samples; $acc(B_m)$ is the average accuracy in the mth bin (Equation 8).

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} 1(\hat{y}_i = y_i) \quad \ldots(8)$$

$conf(B_m)$ is the average confidence in the mth bin (Equation 9), where $\hat{y}_i$ is the predicted class for sample i, where $\hat{p}_i$ is the predicted probability associated with the predicted class $\hat{y}_i$.

$$conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i \quad \ldots(9)$$

An ECE of 0 indicates perfect calibration, where confidence and accuracy are aligned across all bins.

OpenSpliceAI reliability curve and confidence interval

We generated calibration curves for each class using the `calibration_curve` function from scikit-learn, employing 30 bins with a uniform binning strategy. For input, the logits and labels were reshaped into two-dimensional tensors. This function calculates the mean predicted probability and the true

frequency of the positive class within each bin. To visualize the uncertainty in these estimates, we computed confidence intervals for each bin using the normal approximation method (Raschka, 2018)(Equation 10, 11, and 12):

$$SE = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \quad \ldots (10)$$

$$\mathrm{CI}_{\mathrm{lower}} = \max\left(\hat{p} - z \cdot SE, 0\right) \quad \ldots (11)$$

$$\mathrm{CI}_{\mathrm{upper}} = \min\left(\hat{p} + z \cdot SE, 1\right) \quad \ldots (12)$$

where $\hat{p}$ is the empirical probability and n is the number of samples in the bin. Equation 10 defines the standard error (SE) of the estimated parameter. For a 95% confidence level, z is set to 1.96. Equations 11 and 12 provide the lower and upper bounds of the confidence interval, respectively (See Figure S16-S22 for all reliability curve results).

OpenSpliceAI `predict` subcommand

After the OpenSpliceAI model is trained, users can execute this subcommand to predict splice sites in DNA sequences provided in FASTA format. This command also supports limiting predictions to protein-coding genes by using a GFF annotation file for the given genome. It outputs the results in BED format, collecting all probable donor and acceptor site locations into separate files.

OpenSpliceAI data pre-processing

Depending on the inputs of the subcommand, OpenSpliceAI will extract the input sequences differently. If only a FASTA file is provided, OpenSpliceAI will collect all sequences within the file for prediction. If a FASTA and GFF file are both provided, OpenSpliceAI will extract all features of type ‘gene’ from the GFF file and use those coordinates to extract sequences from the FASTA file for prediction.

To aid in memory management, OpenSpliceAI splits any sequence with length greater than the “split-threshold” (default: 1,500,000 bases), into chunks that are no longer than this threshold. This parameter can be adjusted and ensures that each chunk can be loaded entirely into memory during the one-hot encoding process. Additionally, to optimize speed, if the total length of sequences in the FASTA file is below the “hdf_threshold” (default: 5,000 bases), the tool bypasses HDF5 compression and processes the input directly as text, achieving a slight performance speedup.

After all sequences are collected, the tool pre-processes the inputs using a method similar to that employed during training, but without handling any true labels. Each sequence is split into overlapping windows of size 5000 + “flanking_size”, where the overlap is equal to half of the “flanking_size”. No clipping is allowed for the input, so if the final subsequence is shorter than 5000 + “flanking_size”, it is right-padded with N’s. Similarly, the first subsequence is left-padded with “flanking_size” / 2 N’s. If a sequence is split according to the “split_threshold”, it is divided so that adjacent subsequences share an overlap equal to half

the ‘flanking_size’, preventing N-padding from interfering with predictions. This ensures that the model predicts every single base of the provided input sequences.

After window-based splitting, all 5000 + ‘flanking_size’ subsequences generated from the input FASTA entry are grouped together and one-hot encoded in parallel, yielding an entry with dimensions (N, 5000 + ‘flanking_size’, 4), where N is the number of subsequences generated from the given FASTA entry. These entries are further grouped into chunks of size ‘chunk_size’ (default 100), which ensures that they are processed together, reshaping the input to size (<=‘chunk_size’, N, 5000 + ‘flanking_size’, 4). The resultant chunks are saved together as an HDF5-compressed file.

OpenSpliceAI prediction algorithm design

To dynamically manage memory and optimize speed, OpenSpliceAI offers two modes of prediction: standard and turbo, which is controlled by the ‘predict_all’ flag. In standard mode, predictions for all bases are stored (which can be memory-intensive) and written to a BED file. In contrast, turbo mode skips storing individual predictions and directly converts them to a BED file without storing them, reducing memory usage. By default, turbo mode is enabled (See Figure S23).

To start the prediction process, OpenSpliceAI first loads the appropriate pre-trained models. Like SpliceAI, each pre-trained OpenSpliceAI model can include multiple individual models that are averaged to produce a final, higher-quality prediction. Users can specify either a single model file or a directory containing multiple models, in which case the tool automatically averages predictions from all provided models. Depending on the user’s system, OpenSpliceAI selects the best available computing device for model loading and prediction, prioritizing CUDA, MPS, CPU in that order.

OpenSpliceAI performs batch prediction with parallelized prediction, significantly reducing prediction time. The ‘batch_size’ parameter is determined based on the ‘flanking_size’ and the computing device used. Chunked sequences are loaded into a PyTorch DataLoader object, which batches the one-hot-encoded chunk into dimension (<=‘batch_size’, 4, 5000 + ‘flanking_size’). Each batch is processed through all provided models, producing averaged, batched predictions, which are then accumulated by chunk and flattened to reconstruct the full input sequence (FASTA entry) of dimension (<=‘split_threshold’, 3). The second dimension represents predictions for whether each base position is a donor site, acceptor site, or neither.

OpenSpliceAI’s memory handling varies based on the prediction mode. In standard mode, the output tensor is periodically flushed to a file, controlled by the ‘flush_predict_threshold’ parameter, which specifies how many sequences are stored in memory before flushing to an HDF5 file. Predictions are then converted into a BED file that explicitly identifies donor and acceptor sites. In turbo mode, the prediction and BED-file-writing steps are performed simultaneously, and the raw tensor predictions for each base are discarded from memory rather than saved. This substantially reduces memory and processing time.

OpenSpliceAI prediction outputs

The prediction step generates two BED files – one for donor sites and one for acceptor sites – containing the coordinates and scores of all splice sites that exceed a specified score threshold (set by the 'threshold' parameter, default 0.5). The tool automatically extracts relevant information from input files to determine splice site coordinates in the BED file. If not enough annotation data is provided (e.g., the FASTA header does not have transcript start and end coordinates and no annotation file is provided), the coordinates are reported relative to the FASTA sequence, with position 0 corresponding to the first nucleotide. If an annotation file is provided, the tool extracts protein-coding gene loci and calculates the coordinates of splice sites within each locus.

OpenSpliceAI `variant` subcommand

The 'variant' subcommand reimplements SpliceAI's publicly-available utility (Jaganathan et al., 2019) to evaluate the effects of genetic variants on the location and strength of splice sites. It does so by comparing predictions made on wild-type and mutatnt sequences to determine the impact of single nucleotide polymorphisms (SNPs) and insertions or deletions (INDELs) on the resulting mRNA transcript. The tool outputs "delta" scores for four events: donor site gain, donor site loss, acceptor site gain, and acceptor site loss, along with the most probable position of each event relative to the mutation. It accepts a variant call format (VCF) file as input and returns an output VCF file annotated with the delta scores and positions.

This subcommand supports variant effect prediction using both PyTorch and Keras models to maintain compatibility with upstream workflows. However, PyTorch-based models are strongly recommended for faster prediction and lower memory overhead.

OpenSpliceAI variant delta score calculation

The "delta" score is defined similarly to *Jaganathan et al.* (2019), and refers to the maximum change in splicing score within a fixed window on each side of the mutation. By default, the window size is 50, meaning it will consider the donor and acceptor scores for the 101 positions around the variant. Supposing the array of donor and acceptor scores of the wild-type sequence are d_{ref} and a_{ref}, and those of the mutated sequence are d_{alt} and a_{alt}, then the delta scores (DS) are calculated as follows (Equation 13 -16):

$$DS\ (Acceptor\ Gain) = \max\ (a_{alt} - a_{ref}) \qquad \ldots(13)$$

$$DS\ (Acceptor\ Loss) = \max\ (a_{ref} - a_{alt}) \qquad \ldots(14)$$

$$DS\ (Donor\ Gain) = \max\ (d_{alt} - d_{ref}) \qquad \ldots(15)$$

$$DS\ (Donor\ Gain) = \max\ (d_{alt} - d_{ref}) \qquad \ldots(16)$$

In *Jaganathan et al.* (2019), the term 'delta score' specifically refers to the maximum value among the four events. However, we do not use this score in our output. Instead, the output VCF file reports separate scores for each of the four events.

OpenSpliceAI splice site variant scoring process

The 'variant' tool of OpenSpliceAI takes as input a VCF file, a reference genome in FASTA format, the model, the flanking size, and a custom annotation file. It annotates each variant in the provided VCF with four delta scores and four corresponding "delta positions," which represent the relative nucleotide location of each delta score. By default, the delta position is ±50, with negative values indicating positions upstream of the variant and positive values downstream. Variants outside of genes defined by the annotation file, those that are less than 'flanking_size' from the ends of the chromosome, and deletions greater than 2 x 'distance' are excluded from annotation. The tool returns a VCF file with OpenSpliceAI annotations for all valid variants (Figure S24).

$\mathbf{OSAI_{MANE}}$ training

We generated training and test datasets using the `create-data` subcommand. Following the SpliceAI approach, chromosomes 1, 3, 5, 7, and 9 were held out for testing. Paralogous genes were stringently removed from the test dataset based on sequence alignment results between the training and test sets (see Methods: Pseudogenes and paralogous gene sequences removal). The "`--canonical-only`" argument was used to label only donor and acceptor sites in U2-snRNP-type and U12-snRNP-type introns.

Next, we trained $OSAI_{MANE}$ with the `train` subcommand, employing a cosine annealing scheduler ("`--scheduler CosineAnnealingWarmRestarts`") and a categorical cross-entropy loss function ("`--loss cross_entropy_loss`") over 10 epochs ("`--epochs 10`").

We trained $OSAI_{MANE}$ with four different flanking sequence lengths: 80, 400, 2,000, and 10,000 nt. For each flanking sequence length, five models were trained with different random seeds to enable ensemble score predictions, following the SpliceAI-Keras approach. Model performance was evaluated on the held-out chromosomes (1, 3, 5, 7, and 9) using top□k accuracy, AUPRC, overall accuracy, precision, recall, and F1 score for both donor and acceptor sites.

Commands to train $\mathbf{OSAI_{MANE}}$

The following commands reproduce the creation of $OSAI_{MANE}$ using the OpenSpliceAI framework. To do so, an annotation GFF file, specifically the Human RefSeq MANE v1.3 annotation (https://ftp.ncbi.nlm.nih.gov/refseq/MANE/MANE_human/release_1.3/MANE.GRCh38.v1.3.refseq_genomic.gff.gz), and a genome FASTA file, the GRCh38.p14 genome (https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/405/GCF_000001405.40_GRCh38.p14/GCF_000001405.40_GRCh38.p14_genomic.fna.gz), are required.

- Creating training and test dataset:

```
openspliceai create-data --genome-fasta
GCF_000001405.40_GRCh38.p14_genomic.fna --annotation-gff
MANE.GRCh38.v1.3.refseq_genomic.gff --output-dir
train_test_dataset_MANE_test/ --remove-paralogs --min-identity 0.8 --min-
coverage 0.8 --parse-type canonical --write-fasta --split-method human --
canonical-only
```

- Training $OSAI_{MANE}$

```
openspliceai train --flanking-size 10000 --train-dataset dataset_train.h5 --
test-dataset dataset_test.h5 --output-dir model_train_outdir/ --project-name
OSAI-MANE --loss cross_entropy_loss --scheduler CosineAnnealingWarmRestarts
--epochs 10
```

- Calibrating $OSAI_{MANE}$ (optional)

```
Openspliceai calibrate –flanking-size 10000 --train-dataset dataset_train.h5
--test-dataset dataset_test.h5 --output-dir model_calibrate_outdir/ --
project-name OSAI-MANE-calibrate --pretrained-model model_best.pt --loss
cross_entropy_loss
```

Similar to training $OSAI_{MANE}$, the OpenSpliceAI framework can also be used to train species-specific models using different genomes and genome annotations. See Supplemental Note S2-S5 for more details.

Hardware resources for training in this study

OpenSpliceAI study was conducted on the Rockfish cluster. For data preprocessing, OpenSpliceAI was run with a single thread on a 24-core Intel Xeon Cascade Lake 6248R processor, with a base frequency of 3.0GHz and a 1 TB NVMe local drive. The five OpenSpliceAI models – $OSAI_{MANE}$, $OSAI_{Mouse}$, $OSAI_{Zebrafish}$, $OSAI_{Honeybee}$, and $OSAI_{Arabidopsis}$ – were each trained with a single Nvidia A100 GPU with 40 GB of memory and 192 GB of DDR4 memory. Slurm jobs were submitted with "--mem=64G".

Model architecture and training hyperparameters

Building on SpliceAI's model architecture, we re-implemented the deep residual convolutional neural network using PyTorch to improve flexibility and extensibility (see Figure S25; https://github.com/Kuanhao-Chao/OpenSpliceAI/blob/main/openspliceai/train_base/spliceai.py). The network processes an input tensor of shape (batch size, input length, 4) that encodes one-hot nucleotide sequences. Four different flanking sequence lengths – 80, 400, 2,000, and 10,000 nucleotides – are used to train four separate models, providing flexibility in capturing splicing signals at varying genomic contexts.

At the core of OpenSpliceAI, an initial 1D convolution (mapping from 4 channels to L channels) projects the nucleotide embedding into a higher-dimensional feature space. The resulting features pass through a series of ResidualUnit blocks, each consisting of two dilated convolutional layers with LeakyReLU ($\alpha = 0.1$) activation and batch normalization. These dilated convolutional layers employ increasing dilation rates (AR vector) and kernel sizes (W vector) to enlarge the receptive field without requiring extremely deep networks. Every fourth residual block is followed by a Skip layer that merges skip-connection features via a 1D convolution, ensuring better gradient flow and stabilizing training.

To accommodate the shrinking of the sequence length necessitated by large receptive fields, we employ a Cropping1D layer that removes extra padding introduced by convolutional dilation. Specifically, the amount of cropping CL is computed as twice the sum of $AR \times (W - 1)$. By slicing out $CL/2$ nucleotides from each end of the sequence, the Cropping1D layer aligns the network output with the desired prediction length. The final layer is a 1D convolution mapping the output features into three channels – representing

probabilities for the donor site, acceptor site, or neither. We apply a softmax activation over these three positions at each nucleotide, yielding position-wise splice site prediction probabilities. See Table 2 for a summary of the model architectures trained with four different flanking sequence lengths.

While model training is performed with fixed-length sequences (for example, 15,000 nucleotides to produce 5,000 prediction positions), the dilated convolutional structure and final Cropping1D layer allow each trained model to process variable-length sequences at inference time. Specifically, any sequence longer than the network's receptive field can be fed to the network, which will output predictions aligned to the valid region (i.e., the input length minus the cropping region). For flanking regions, users can either pad with Ns or include relevant upstream and downstream genomic context. Memory permitting, this design grants flexibility in analyzing genomic segments of varying lengths in a single pass, without the need to retrain separate models for each new sequence length.

In total, these architectural components allow OpenSpliceAI to capture sequence contexts spanning up to 10,000 nucleotides. By varying kernel sizes, dilation rates, and cropping, the model can learn both local and long-range patterns important for accurate identification of canonical splice signals.

Table 2. Summary of the four OpenSpliceAI model architectures, each trained with a distinct flanking sequence length (80, 400, 2,000, and 10,000 nucleotides). The table lists the kernel sizes (W), dilation rates (AR), number of residual and skip blocks, and total cropping length (CL).

Parameter	Flanking = 80	Flanking = 400	Flanking = 2000	Flanking = 10000
Kernel sizes (W)	[11, 11, 11, 11]	[11, 11, 11, 11, 11, 11, 11, 11]	[11, 11, 11, 11, 11, 11, 11, 11, 21, 21, 21, 21]	[11, 11, 11, 11, 11, 11, 11, 11, 21, 21, 21, 21, 41, 41, 41, 41]
Dilated rates (AR)	[1, 1, 1, 1]	[1, 1, 1, 1, 4, 4, ,4 ,4]	[1, 1, 1, 1, 4, 4, ,4 ,4, 10, 10, 10, 10]	[1, 1, 1, 1, 4, 4, ,4 ,4, 10, 10, 10, 10, 25, 25, 25, 25]
Residual blocks	4	8	12	16
Skip connections	1 (inserted after residual block 4)	2 (inserted after residual blocks 4 & 8)	3 (inserted after residual blocks 4, 8, & 12)	4 (inserted after residual blocks 4, 8, 12, & 16)
Cropping length (CL); $(\mathbf{2} \times \sum[\boldsymbol{AR} \times (\mathbf{W} - \mathbf{1})])$	80	400	2,000	10,000

For model training, we introduced enhancements to the adaptive learning rate schedule. Specifically, we set a maximum learning rate for the initial 10 epochs and implemented an early stopping criterion to prevent overfitting. The adaptive learning rate decreases by a factor of 0.5 starting from the 6th epoch in

the original implementation. Additionally, we incorporated the CosineAnnealingWarmRestarts scheduler to enable periodic learning rate restarts, which can help escape local minima and improve convergence.

Evaluation metrics on model performance

To evaluate both scratch-trained and transfer-trained models, we used outputs from the "`create-data`" subcommand derived from a test set that included only protein-coding genes and excluded paralogous sequences. For consistency with the SpliceAI study, we held out the same chromosomes for testing. For other species, we reserved approximately 20% of the data for testing and used the remaining 80% for training. These datasets were then used to assess (1) the original SpliceAI models, (2) all OSAI models trained from scratch, and (3) all OSAI models fine-tuned from $\text{OSAI}_{\text{MANE}}$.

Top-k accuracy

In line with the definition used in the SpliceAI paper, we evaluate the model's performance using the top-k accuracy metric. This metric is computed by examining each dimension of the model's predictions. For each DNA sequence of length L, the model outputs an $L \times 3$ matrix: the first channel indicates non-splice sites, the second indicates acceptor splice sites, and the third indicates donor splice sites.

Top-k splice site accuracy evaluation is computed as follows: For a gene sequence of length L, containing 10 true donor splice sites ($n_{true}^{donor} = 10$) and 10 true acceptor splice sites ($n_{true}^{acceptor} = 10$), the model generates probability scores for donor and acceptor sites across the sequence. For top-k accuracy, the $k \times n_{true}^{class}$ highest-scoring predictions are extracted per class (donor/acceptor), where $n_{true}^{class} = 10$ for each splice site type. Class-specific accuracy is calculated as follows (Equation 17):

$$Top\text{-}k^{class} = \frac{\sum_{i=1}^{k \times n_{true}^{class}} \mathrm{I}(p_i^{class} \in S_{true}^{class})}{k \times n_{true}^{class}}, \qquad class \in \{donor, acceptor\} \qquad \ldots(17)$$

where p_{true}^{class} denotes the i-th ranked prediction for the class, S_{true}^{class} is the set of true splice sites for the class, and I is an indicator function.

For $k = 1$, this evaluates whether true splice sites are present among the top 10 donor and top 10 acceptor predictions (20 total). Accuracy is defined as the proportion of true sites correctly identified within this subset. For $k = 2$, the evaluation expands to the top 20 predictions per class (40 total). This metric quantifies the model's ability to prioritize true splice sites within ranked candidate positions

Accuracy, precision, recall, F1-score, and AUPRC

Unlike top-k accuracy, which requires knowing the number of ground-truth splice sites, we can determine the predicted class for each position based on the highest probability across each dimension and evaluate predictions as true positives (TP), true negatives (TN), false positives (FP), or false negatives (FN) based on a preset threshold. After labelling each site in a given sequence, we calculate accuracy, precision, recall,

F1-score, and AUPRC with the threshold 0.5 for the sequence prediction. For example, for donor sites, the metrics are calculated as follows (Equation 18-20):

$$Accuracy_{donor\ site} = \frac{TP_{donor\ site} + FP_{donor\ site}}{TP_{donor\ site} + FP_{donor\ site} + TN_{donor\ site} + FN_{donor\ site}} \quad \ldots(18)$$

$$Precision_{donor\ site} = \frac{TP_{donor\ site}}{TP_{donor\ site} + FN_{donor\ site}} \quad \ldots(19)$$

$$Recall_{donor\ site} = \frac{TP_{donor\ site}}{TP_{donor\ site} + TN_{donor\ site}} \quad \ldots(20)$$

Benchmarking SpliceAI-Keras and $OSAI_{MANE}$

To compare $OSAI_{MANE}$ with SpliceAI (Jaganathan et al., 2019), we benchmarked the computational efficiency and performance of both tools across various metrics in two tasks: large-scale prediction (using the 'predict' module) and variant effect prediction (using the 'variant' module). The results are shown in **Figure 2**G, H.

Time and CPU/GPU memory profiling

The SpliceAI-Keras and $OSAI_{MANE}$ were benchmarked using the Scalene profiling tool (https://github.com/plasma-umass/scalene), a Python-specific profiler which handles CPU, GPU, and memory profiling and evaluates code line-by-line. We measured the following metrics: elapsed CPU time, peak CPU memory, peak GPU memory, percentage of CPU time in low-level C code, CPU memory growth rate, and average memory usage. The first three metrics are visualized in Figure 2E, while the remaining are presented in the Figure S6.

Predict benchmark design

The objective of this benchmark was to provide a fair comparison of the computational efficiency of SpliceAI and $OSAI_{MANE}$, with the key difference being that SpliceAI is implemented in Keras (https://github.com/keras-team/keras) and $OSAI_{MANE}$ is implemented in PyTorch (https://github.com/pytorch/pytorch). While SpliceAI includes a variant effect prediction utility, it lacks a dedicated tool for large-scale predictions. To address this, we extracted the core Keras-wrapped prediction code from SpliceAI's variant tool and integrated it into our 'predict' utility, which has our specific data pre-processing and BED file generation . We call this tool "SpliceAI-Keras."
For this experiment, we randomly sampled 1,000 protein-coding genes from MANE and benchmarked the tools on increasingly large subsets of the genes. The prediction task is to identify all splice sites within the gene locus.

With the Scalene profiler active, we ran both SpliceAI-Keras and $OSAI_{MANE}$ with the 'predict' subcommand across all five models (per default usage), extracting the averaged predictions, and repeated the process for a total of 5 trials. We further benchmarked every model size (80, 400, 2000, and 10k bp flanking size). The graph visualizations depict the mean metrics as a solid line, while the shaded region represents the variance between trials.

If the computation encountered an out-of-memory error that caused the prediction to stall, we discarded that trial. Note that running Scalene also requires dedicated memory usage, which was not included in the graphs. Some input sizes yielded no successful trials for SpliceAI-Keras, reflected in missing datapoints.

Variant benchmark design

We compared SpliceAI 'variant' tool (https://github.com/Illumina/SpliceAI) with OpenSpliceAI 'variant' command (in default mode) in an analogous manner to the 'predict' benchmark. This experiment compared every model size across both SpliceAI-Keras and $OSAI_{MANE}$, for 5 trials. For the input VCF file, we used the Mills and 1000 Genomes Project gold standard dataset of known indels in GRCh38, provided by the Broad Institute (https://storage.googleapis.com/genomics-public-data/resources/broad/hg38/v0/Mills_and_1000G_gold_standard.indels.hg38.vcf.gz). We randomly sampled 1,000 indels and benchmark on increasingly large subsets of this data. The resultant graph is shown in Figure S6G-L**.**

In Silico Mutagenesis analysis

The *in silico* mutagenesis (ISM) study compares the prediction patterns of $OSAI_{MANE}$ with SpliceAI (Jaganathan et al., 2019) to demonstrate their similarity and biological relevance. The ISM experiments investigate the effect of mutations on predicted splicing patterns, and we replicate several key experiments from *Jaganathan et al.* (2019) to illustrate this.

Importance score

For assessing the impact of a mutation in a given base position on the strength of the splice site, we calculate an "importance score" as follows (Equation 21):

$$Importance = S_{ref} - \frac{S_A + S_C + S_G + S_T}{4} \qquad \ldots(21)$$

Where S_{ref} denotes the splice site score of the wild-type sequence, and S_A, S_C, S_G, and S_T denote the splice site's score with each of the corresponding base substitutions. The importance score reflects the decrease in the predicted strength of the splice site when the target nucleotide is mutated and can be regarded as the significance of the target base in contributing towards the activation of the splice site (Equation 20).

Single-nucleotide variation in short exons

Pursuant to the protocol of *Jaganathan et al.* (2019), we investigate the importance of various base positions around two known short exons in the human genome, U2SURP exon 9 and DST exon 2. We

evaluate the scores using both SpliceAI-Keras and $OSAI_{MANE}$ to compare the predicted splicing patterns. The findings are summarized in Figure 6A.

For each example, we extract the entire exon as well as part of the intronic region upstream and downstream of the exon to illustrate the full range of base positions. We then mutate each base of the input sequence to the three possible single nucleotide variations at that position. For the reference sequence and each mutated sequence, we collect the acceptor site score and calculate an importance score for that position. We finally visualize the acceptor site importance scores, corresponding to the vertical size of the DNA logo of the reference sequence.

Note that for the U2SURP investigation, we use the same input sequence as *Jaganathan et al.* (2019), which is extracted from the hg19 assembly, to ensure reproducibility. The DST exon used is updated with the more recent GRCh38 assembly. DST was selected to ensure representation across the train and test datasets. Additionally, we note that the DST gene is on the reverse strand, but we display the forward strand (the strand that the splicing models use as input) in our visualization for ease of comparison. Lastly, for this experiment and all subsequent ISM studies, we extract an additional 10,000 bp around the input sequence (as opposed to N-padding) and use the SpliceAI-10k and $OSAI_{MANE}$-10k models to perform prediction.

Batch mutagenesis for donor and acceptor site motif recognition

To establish a more representative idea of the splicing pattern around splice sites, we scale up the ISM experiment across multiple donor and acceptor sites. The data consists of 100 randomly sampled donor and 100 randomly sampled acceptor sites from the testing dataset (Chromosomes 1, 3, 5, 7, and 9). Each sequence is 400 bp with the donor and acceptor splicing motifs located at the midpoint of the sequence.

For each position in the sequence, we mutate the base to every other base and measure the decrease in the strength of the central splice site score. Again, the measurement is taken as the average across all five models for both SpliceAI-Keras and $OSAI_{MANE}$ (10k model size). We consider each point mutation separately here. We repeat this for every sequence and take the average result across the 100 samples.

Finally, we display the averaged scores in a DNA logo (Figure 6B), where the vertical size of a base denotes the magnitude of decrease in splice site strength when the original base at that position is mutated to that base. We show the central 80 base positions in our visualization for readability.

We note that SpliceAI-Keras runs many orders of magnitude slower than $OSAI_{MANE}$. In order to speed up the computation, we split up the input into ten smaller batches of 10 transcripts each and process them in parallel on a GPU cluster. The results are aggregated, then averaged.

Cryptic splicing mutation analysis

To reproduce another key experiment from *Jaganathan et al.*, 2019 (Figure 2A), we selected a specific G to A point mutation in intron 14 of the MYBPC3 gene which results in a cryptic splicing variant. We

further select another example of a cryptic splicing variant which is validated *in vitro*, from *Qian et al.* (2021). This A to G point mutation occurs deep in intron 16 of OPA1 and was shown to have the highest aberrant-only splicing in minigene assays, with the inclusion of an entire cryptic exon upstream of the mutation.

For each variant, we score the wild-type sequence and the mutated sequence and calculate the change in donor and acceptor scores for all bases around the mutation (specifically at the expected locations of cryptic splice site gain), for both SpliceAI-Keras and $OSAI_{MANE}$. We visualize the most significant splice site gain or loss events, along with their raw splice site scores, in Figure 6C.

Because the sequence near the acceptor site differs in the MYBPC3 gene between the hg19 assembly (used in *Jaganathan et al.*, 2019) and GRCh38 assembly, we opted to retain the hg19 annotation for this example, for the purpose of reproducing SpliceAI's results.

Full prediction of the CFTR gene

We finally reproduce the full prediction of splice sites in the CFTR gene using both SpliceAI-Keras and $OSAI_{MANE}$. Notably different, however, is the fact that we are using the updated GRCh38 assembly, and that we use a fixed score threshold of 0.5 (default for 'predict') to identify splice sites.

We extract the full CFTR gene and run both tools, taking the averaged score across all five models for each tool. The findings are summarized in Figure 6D. The exon plot displays the reference MANE annotation with the locations of the predicted donor and acceptor sites marked in color. The histogram below visualizes the corresponding donor and acceptor score distributions.

We further note that we extract the true 10k DNA flanking sequence around the gene (as opposed to N-padding), which is why our results do not appear to exactly replicate the original study from *Jaganathan et al.* (2019).

Data and Code Availability

OpenSpliceAI is implemented as a Python package, using Pytorch framework. OpenSpliceAI project is freely available on github at: https://github.com/Kuanhao-Chao/OpenSpliceAI, and is available on PyPi: https://pypi.org/project/openspliceai/. The OpenSpliceAI documentation is available at: https://ccb.jhu.edu/openspliceai/.

Author Contribution

K.C., S.L.S., and M.P. designed the research.
K.C. and A.M. co-developed the OpenSpliceAI framework.
K.C., A.M., A.L., S.L.S., and M.P. analyzed the results.
K.C. trained OpenSpliceAI models and conducted experiments comparing OSAI to SpliceAI.
K.C. conducted transfer learning experiments.

A. M. and K.C. conducted the ISM experiments.
K.C. and A.M. wrote the OpenSpliceAI documentation.
K.C., A. M., A.L., S.L.S., and M.P. wrote the manuscript.

Funding

This research was supported in part by the U.S. National Institutes of Health under grants R01-HG006677, R35-GM130151, and R35-GM156470, and by the U.S. National Science Foundation under and DBI 2412449. Computational work was carried out at the Advanced Research Computing at Hopkins (ARCH) core facility, supported in part by NSF grant OAC 1920103.

Acknowledgements

We thank all the members of the Salzberg and Pertea Labs for their valuable discussions and insights.

References

Aartsma-Rus A, Bremmer-Bout M, Janson AAM, Den Dunnen JT, Van Ommen G-JB, Van Deutekom JCT. 2002. Targeted exon skipping as a potential gene correction therapy for Duchenne muscular dystrophy. *Neuromuscular Disorders* **12**:S71–S77. doi:10.1016/S0960-8966(02)00086-X

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mane D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viegas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. doi:10.48550/ARXIV.1603.04467

Agarwal V, Kelley DR. 2022. The genetic and biochemical determinants of mRNA degradation rates in mammals. *Genome Biol* **23**:245. doi:10.1186/s13059-022-02811-x

Alipanahi B, Delong A, Weirauch MT, Frey BJ. 2015. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat Biotechnol* **33**:831–838. doi:10.1038/nbt.3300

Baralle FE, Giudice J. 2017. Alternative splicing as a regulator of development and tissue identity. *Nat Rev Mol Cell Biol* **18**:437–451. doi:10.1038/nrm.2017.27

Barash Y, Calarco JA, Gao W, Pan Q, Wang X, Shai O, Blencowe BJ, Frey BJ. 2010. Deciphering the splicing code. *Nature* **465**:53–59. doi:10.1038/nature09000

Black DL. 2000. Protein Diversity from Alternative Splicing. *Cell* **103**:367–370. doi:10.1016/S0092-8674(00)00128-8

Blencowe BJ. 2006. Alternative Splicing: New Insights from Global Analyses. *Cell* **126**:37–47. doi:10.1016/j.cell.2006.06.023

Bonnal SC, López-Oreja I, Valcárcel J. 2020. Roles and mechanisms of alternative splicing in cancer — implications for care. *Nat Rev Clin Oncol* **17**:457–474. doi:10.1038/s41571-020-0350-x

Braunschweig U, Gueroussov S, Plocik AM, Graveley BR, Blencowe BJ. 2013. Dynamic Integration of Splicing within Gene Regulatory Pathways. *Cell* **152**:1252–1269. doi:10.1016/j.cell.2013.02.034

Bridle JS. 1989. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parametersProceedings of the 3rd International Conference on Neural Information Processing Systems, NIPS'89. Cambridge, MA, USA: MIT Press. pp. 211–217.

Brow DA. 2002. Allosteric Cascade of Spliceosome Activation. *Annu Rev Genet* **36**:333–360. doi:10.1146/annurev.genet.36.043002.091635

Burnett BG, Muñoz E, Tandon A, Kwon DY, Sumner CJ, Fischbeck KH. 2009. Regulation of SMN Protein Stability. *Molecular and Cellular Biology* **29**:1107–1115. doi:10.1128/MCB.01262-08

Dlamini Z, Mokoena F, Hull R. 2017. Abnormalities in alternative splicing in diabetes: therapeutic targets. *Journal of Molecular Endocrinology* **59**:R93–R107. doi:10.1530/JME-17-0049

Fort S, Hu H, Lakshminarayanan B. 2019. Deep Ensembles: A Loss Landscape Perspective. doi:10.48550/ARXIV.1912.02757

Frilander MJ, Steitz JA. 1999. Initial recognition of U12-dependent introns requires both U11/5' splice-site and U12/branchpoint interactions. *Genes & Development* **13**:851–863. doi:10.1101/gad.13.7.851

Fudenberg G, Kelley DR, Pollard KS. 2020. Predicting 3D genome folding from DNA sequence with Akita. *Nat Methods* **17**:1111–1117. doi:10.1038/s41592-020-0958-x

Gotthardt M, Badillo-Lisakowski V, Parikh VN, Ashley E, Furtado M, Carmo-Fonseca M, Schudy S, Meder B, Grosch M, Steinmetz L, Crocini C, Leinwand L. 2023. Cardiac splicing as a diagnostic and therapeutic target. *Nat Rev Cardiol* **20**:517–530. doi:10.1038/s41569-022-00828-0

Hinton G, Vinyals O, Dean J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:150302531*.

Jaganathan K, Kyriazopoulou Panagiotopoulou S, McRae JF, Darbandi SF, Knowles D, Li YI, Kosmicki JA, Arbelaez J, Cui W, Schwartz GB, Chow ED, Kanterakis E, Gao H, Kia A, Batzoglou S, Sanders SJ, Farh KK-H. 2019. Predicting Splicing from Primary Sequence with Deep Learning. *Cell* **176**:535-548.e24. doi:10.1016/j.cell.2018.12.015

Jaynes ET. 1957. Information theory and statistical mechanics. *Physical review* **106**:620.

Johnson JM, Castle J, Garrett-Engele P, Kan Z, Loerch PM, Armour CD, Santos R, Schadt EE, Stoughton R, Shoemaker DD. 2003. Genome-Wide Survey of Human Alternative Pre-mRNA Splicing with Exon Junction Microarrays. *Science* **302**:2141–2144. doi:10.1126/science.1090100

Jung H, Lee D, Lee J, Park D, Kim YJ, Park W-Y, Hong D, Park PJ, Lee E. 2015. Intron retention is a widespread mechanism of tumor-suppressor inactivation. *Nat Genet* **47**:1242–1248. doi:10.1038/ng.3414

Kelley DR, Reshef YA, Bileschi M, Belanger D, McLean CY, Snoek J. 2018. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. *Genome Res* **28**:739–750. doi:10.1101/gr.227819.117

Kelley DR, Snoek J, Rinn JL. 2016. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res* **26**:990–999. doi:10.1101/gr.200535.115

Kingma DP, Ba J. 2014. Adam: A Method for Stochastic Optimization. doi:10.48550/ARXIV.1412.6980

Lakshminarayanan B, Pritzel A, Blundell C. 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. Advances in Neural Information Processing Systems. Curran Associates, Inc.

Lee SC-W, Abdel-Wahab O. 2016. Therapeutic targeting of splicing in cancer. *Nat Med* **22**:976–986. doi:10.1038/nm.4165

Li D, McIntosh CS, Mastaglia FL, Wilton SD, Aung-Htut MT. 2021. Neurodegenerative diseases: a hotbed for splicing defects and the potential therapies. *Transl Neurodegener* **10**:16. doi:10.1186/s40035-021-00240-7

Li H. 2018. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**:3094–3100. doi:10.1093/bioinformatics/bty191

Lin T-Y, Goyal P, Girshick R, He K, Dollár P. 2018. Focal Loss for Dense Object Detection.

Lorson CL, Hahnen E, Androphy EJ, Wirth B. 1999. A single nucleotide in the *SMN* gene regulates splicing and is responsible for spinal muscular atrophy. *Proc Natl Acad Sci USA* **96**:6307–6311. doi:10.1073/pnas.96.11.6307

Loshchilov I, Hutter F. 2019. Decoupled Weight Decay Regularization.

Loshchilov I, Hutter F. 2016. SGDR: Stochastic Gradient Descent with Warm Restarts. doi:10.48550/ARXIV.1608.03983

Majewski J, Ott J. 2002. Distribution and Characterization of Regulatory Elements in the Human Genome. *Genome Res* **12**:1827–1836. doi:10.1101/gr.606402

Martí-Gómez C, Larrasa-Alonso J, López-Olañeta M, Villalba-Orero M, García-Pavía P, Sánchez-Cabo F, Lara-Pezzi E. 2022. Functional Impact and Regulation of Alternative Splicing in Mouse Heart Development and Disease. *J of Cardiovasc Trans Res* **15**:1239–1255. doi:10.1007/s12265-022-10244-x

Mcclorey G, Fletcher S, Wilton S. 2005. Splicing intervention for Duchenne muscular dystrophy. *Current Opinion in Pharmacology* **5**:529–534. doi:10.1016/j.coph.2005.06.001

Mills JD, Janitz M. 2012. Alternative splicing of mRNA in the molecular pathology of neurodegenerative diseases. *Neurobiology of Aging* **33**:1012.e11-1012.e24. doi:10.1016/j.neurobiolaging.2011.10.030

Moore MJ, Wang Q, Kennedy CJ, Silver PA. 2010. An Alternative Splicing Network Links Cell-Cycle Control to Apoptosis. *Cell* **142**:625–636. doi:10.1016/j.cell.2010.07.019

Morales J, Pujar S, Loveland JE, Astashyn A, Bennett R, Berry A, Cox E, Davidson C, Ermolaeva O, Farrell CM, Fatima R, Gil L, Goldfarb T, Gonzalez JM, Haddad D, Hardy M, Hunt T, Jackson J, Joardar VS, Kay M, Kodali VK, McGarvey KM, McMahon A, Mudge JM, Murphy DN, Murphy MR, Rajput B, Rangwala SH, Riddick LD, Thibaud-Nissen F, Threadgold G, Vatsan AR, Wallin C, Webb D, Flicek P, Birney E, Pruitt KD, Frankish A, Cunningham F, Murphy TD. 2022. A joint NCBI and EMBL-EBI transcript set for clinical genomics and research. *Nature* **604**:310–315. doi:10.1038/s41586-022-04558-8

Mouse Genome Sequencing Consortium. 2002. Initial sequencing and comparative analysis of the mouse genome. *Nature* **420**:520–562. doi:10.1038/nature01262

Naryshkin NA, Weetall M, Dakka A, Narasimhan J, Zhao X, Feng Z, Ling KKY, Karp GM, Qi H, Woll MG, Chen G, Zhang N, Gabbeta V, Vazirani P, Bhattacharyya A, Furia B, Risher N, Sheedy J, Kong R, Ma J, Turpoff A, Lee C-S, Zhang X, Moon Y-C, Trifillis P, Welch EM, Colacino JM, Babiak J, Almstead NG, Peltz SW, Eng LA, Chen KS, Mull JL, Lynes MS, Rubin LL, Fontoura P, Santarelli L, Haehnke D, McCarthy KD, Schmucki R, Ebeling M, Sivaramakrishnan M, Ko C-P, Paushkin SV, Ratni H, Gerlach I, Ghosh A, Metzger F. 2014. *SMN2* splicing modifiers improve motor function and longevity in mice with spinal muscular atrophy. *Science* **345**:688–693. doi:10.1126/science.1250127

Nikom D, Zheng S. 2023. Alternative splicing in neurodegenerative disease and the promise of RNA therapies. *Nat Rev Neurosci* **24**:457–473. doi:10.1038/s41583-023-00717-6

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Köpf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. doi:10.48550/ARXIV.1912.01703

Patel AA, Steitz JA. 2003. Splicing double: insights from the second spliceosome. *Nat Rev Mol Cell Biol* **4**:960–970. doi:10.1038/nrm1259

Platt J. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* **10**:61–74.

Qian X, Wang J, Wang M, Igelman AD, Jones KD, Li Y, Wang K, Goetz KE, Birch DG, Yang P, Pennesi ME, Chen R. 2021. Identification of Deep-Intronic Splice Mutations in a Large Cohort of Patients With Inherited Retinal Diseases. *Front Genet* **12**:647400. doi:10.3389/fgene.2021.647400

Raschka S. 2018. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. doi:10.48550/ARXIV.1811.12808

Segal E, Widom J. 2009. From DNA sequence to transcriptional behaviour: a quantitative approach. *Nat Rev Genet* **10**:443–456. doi:10.1038/nrg2591

Sokolova K, Chen KM, Hao Y, Zhou J, Troyanskaya OG. 2024. Deep Learning Sequence Models for Transcriptional Regulation. *Annual Review of Genomics and Human Genetics* **25**:105–122. doi:10.1146/annurev-genom-021623-024727

Supek F, Miñana B, Valcárcel J, Gabaldón T, Lehner B. 2014. Synonymous Mutations Frequently Act as Driver Mutations in Human Cancers. *Cell* **156**:1324–1335. doi:10.1016/j.cell.2014.01.051

Sveen A, Kilpinen S, Ruusulehto A, Lothe RA, Skotheim RI. 2016. Aberrant RNA splicing in cancer; expression changes and driver mutations of splicing factor genes. *Oncogene* **35**:2413–2427. doi:10.1038/onc.2015.318

Wagner N, Çelik MH, Hölzlwimmer FR, Mertes C, Prokisch H, Yépez VA, Gagneur J. 2023. Aberrant splicing prediction across human tissues. *Nature Genetics* **55**:861–870.

Wang ET, Sandberg R, Luo S, Khrebtukova I, Zhang L, Mayr C, Kingsmore SF, Schroth GP, Burge CB. 2008. Alternative isoform regulation in human tissue transcriptomes. *Nature* **456**:470–476. doi:10.1038/nature07509

Wang G-S, Cooper TA. 2007. Splicing in disease: disruption of the splicing code and the decoding machinery. *Nat Rev Genet* **8**:749–761. doi:10.1038/nrg2164

Wassarman KM, Steitz JA. 1992. The Low-Abundance U11 and U12 Small Nuclear Ribonucleoproteins (snRNPs) Interact To Form a Two-snRNP Complex. *Molecular and Cellular Biology* **12**:1276–1285. doi:10.1128/mcb.12.3.1276-1285.1992

Xiong HY, Alipanahi B, Lee LJ, Bretschneider H, Merico D, Yuen RK, Hua Y, Gueroussov S, Najafabadi HS, Hughes TR, others. 2015. The human splicing code reveals new insights into the genetic determinants of disease. *Science* **347**:1254806.

Zadrozny B, Elkan C. 2002. Transforming classifier scores into accurate multiclass probability estimates. Presented at the Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 694–699.

Zadrozny B, Elkan C. 2001. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. Presented at the Icml. pp. 609–616.

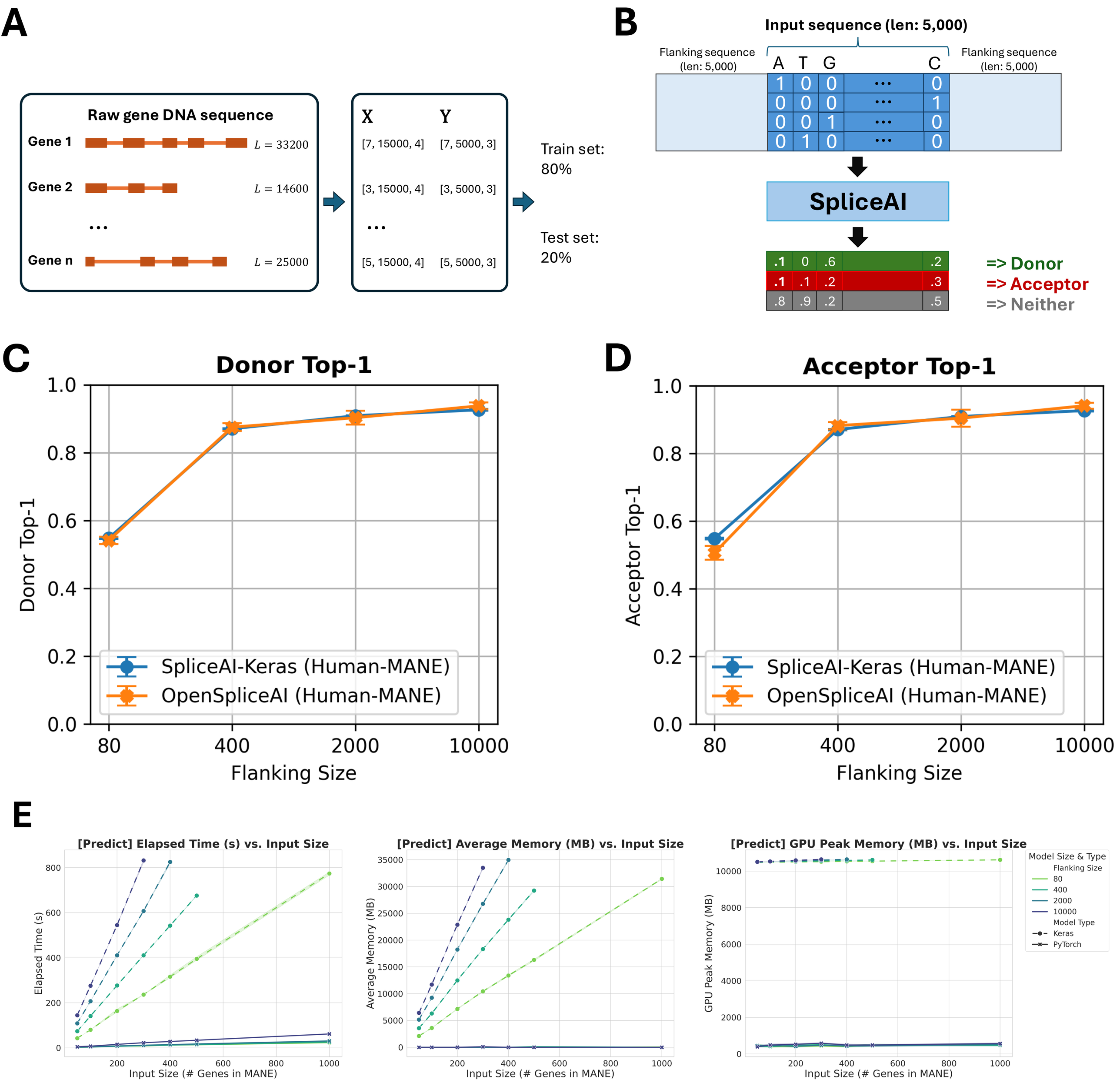
Zhou J, Theesfeld CL, Yao K, Chen KM, Wong AK, Troyanskaya OG. 2018. Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. *Nat Genet* **50**:1171–1179. doi:10.1038/s41588-018-0160-6

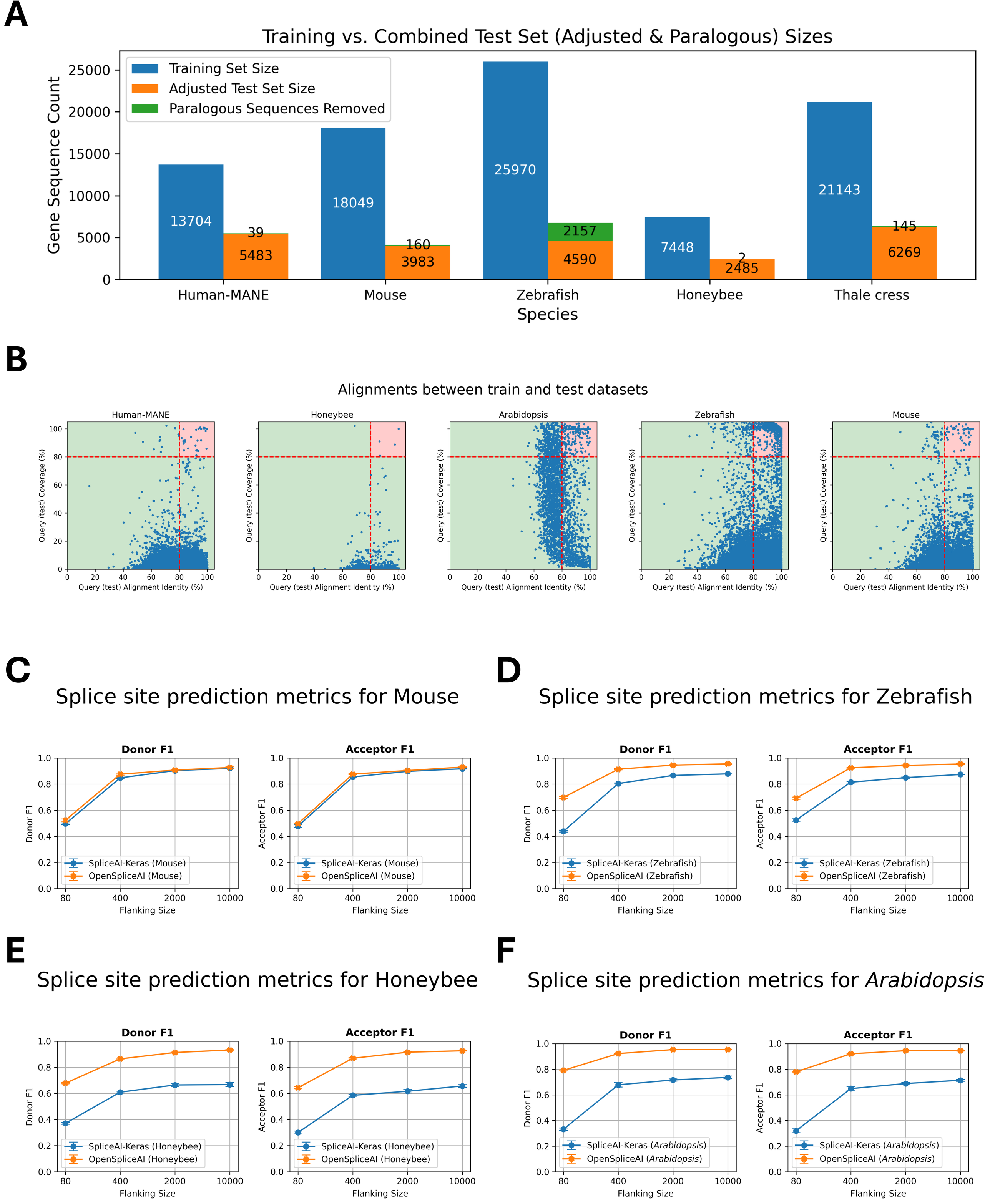
Zhou J, Troyanskaya OG. 2015. Predicting effects of noncoding variants with deep learning–based sequence model. *Nat Methods* **12**:931–934. doi:10.1038/nmeth.3547

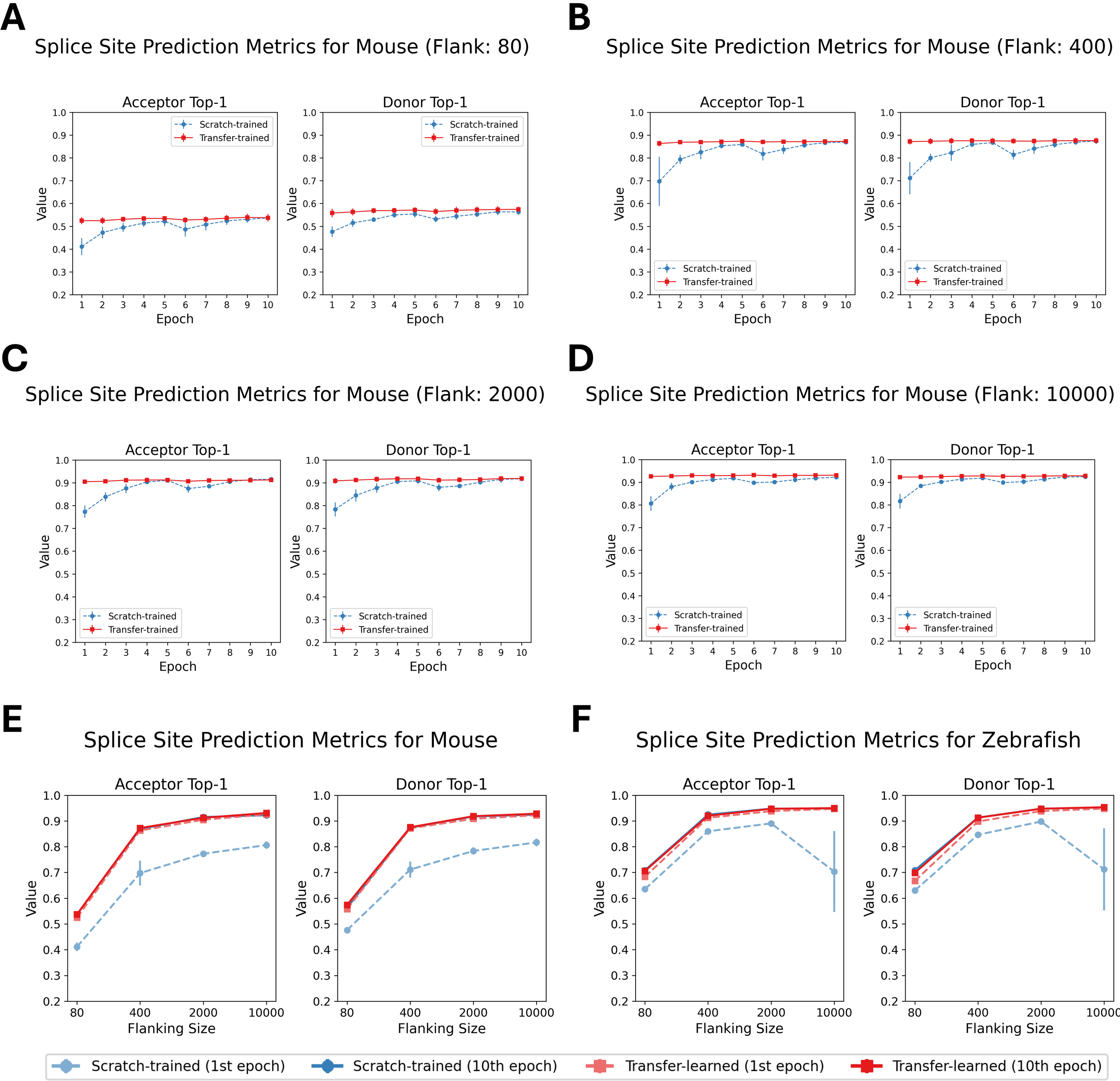
Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H, He Q. 2021. A Comprehensive Survey on Transfer Learning. *Proc IEEE* **109**:43–76. doi:10.1109/JPROC.2020.3004555

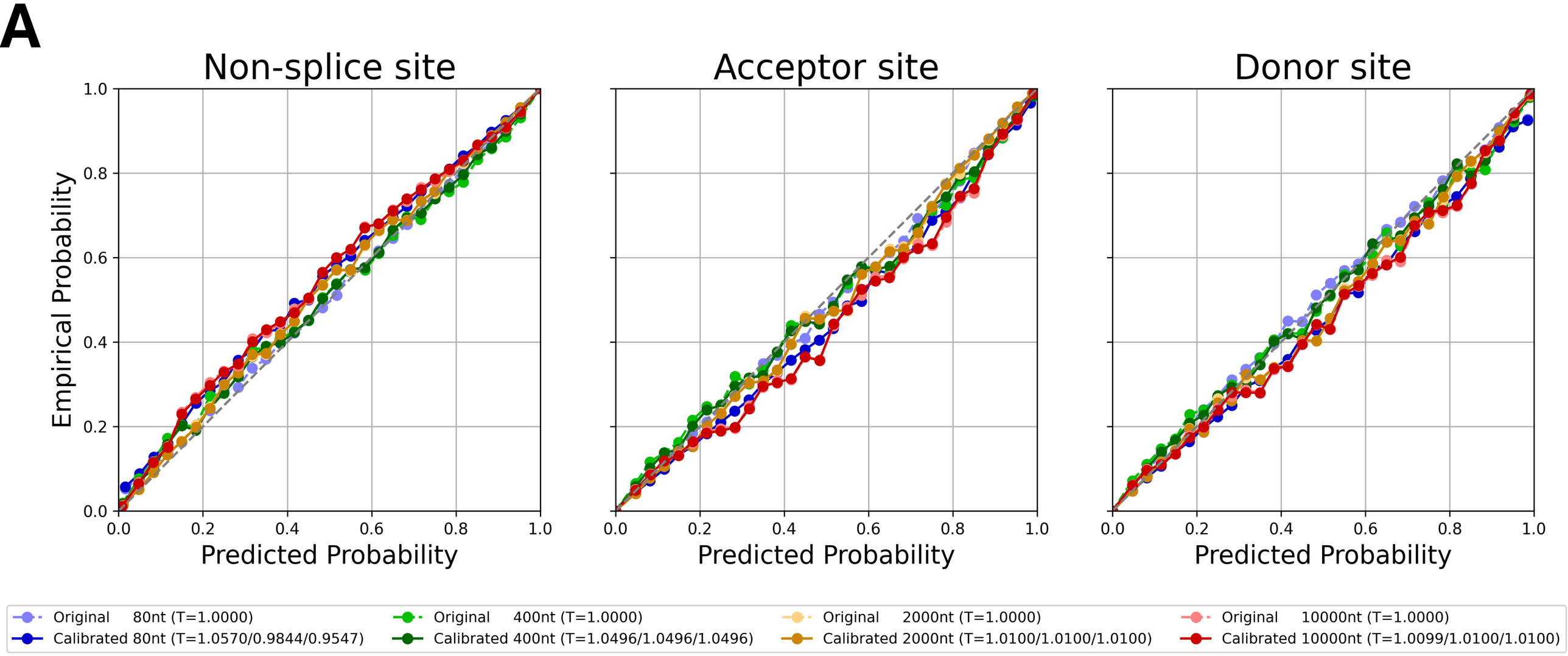
OpenSpliceAI

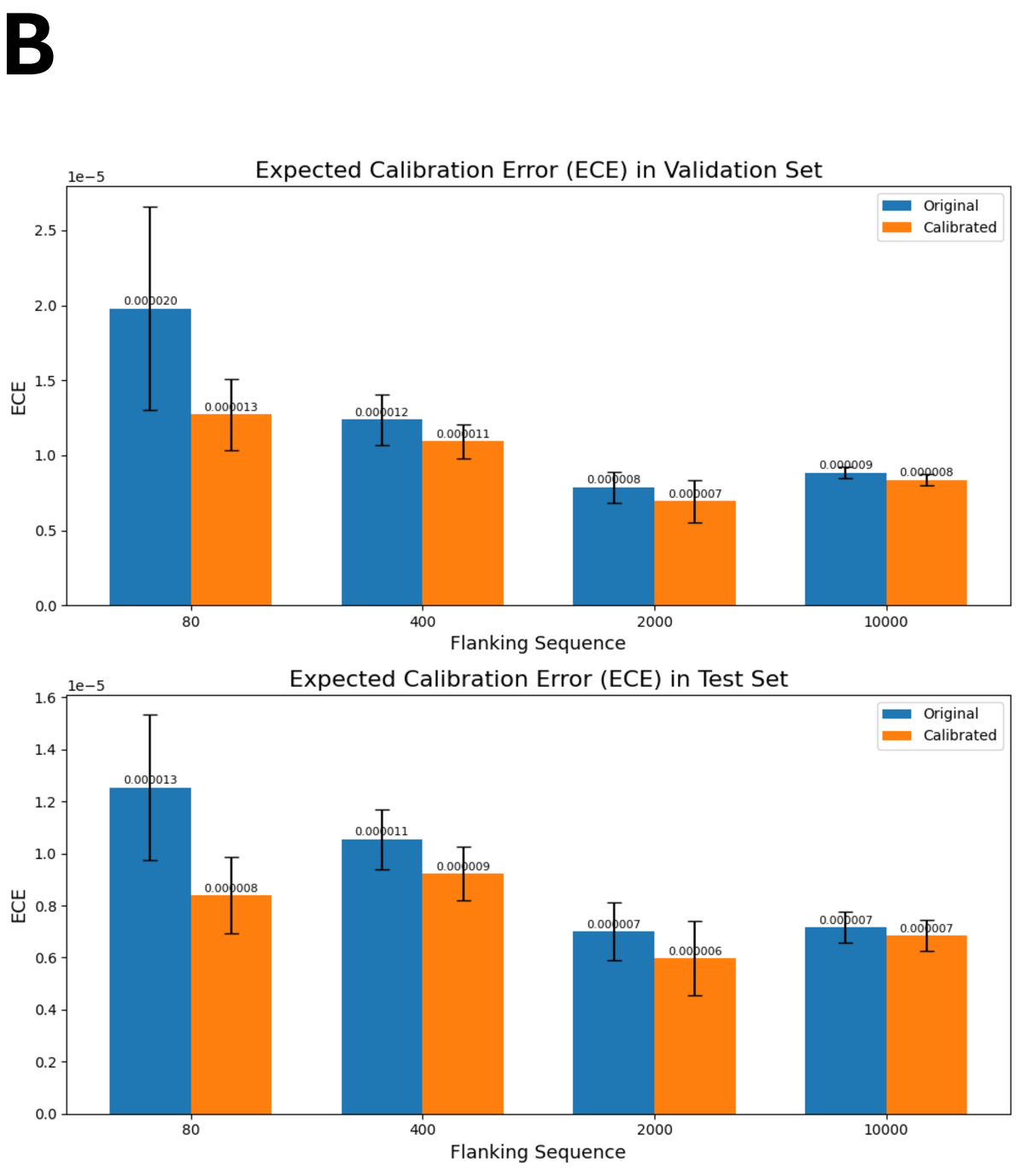
	Inputs	Subcommands	Outputs
A	**1.** Genome annotation (**GFF** /**GTF**) **2.** Genome assembly (**FASTA**)	`create-data`	**1.** dataset_train.h5 **2.** dataset_test.h5
B	**1.** dataset_train.h5 **2.** dataset_test.h5	`train`	**1.** MODEL.pt **2.** Training / Validation / Testing Log
C	**1.** dataset_train.h5 10% for validation (calibration) **2.** dataset_test.h5 **3.** MODEL.pt	`calibrate`	MODEL_calibrated.pt
D	**1.** dataset_train.h5 (*species 2*) **2.** dataset_test.h5 (*species 2*) **3.** MODEL.pt	`transfer`	MODEL_transfer-learned.pt
E	**1.** MODEL.pt **2.** Target genome / gene sequences	`predict`	OpenSpliceAI scores for each position
F	**1.** MODEL.pt **2.** Target Variants	`variant`	VCF file with OpenSpliceAI predictions

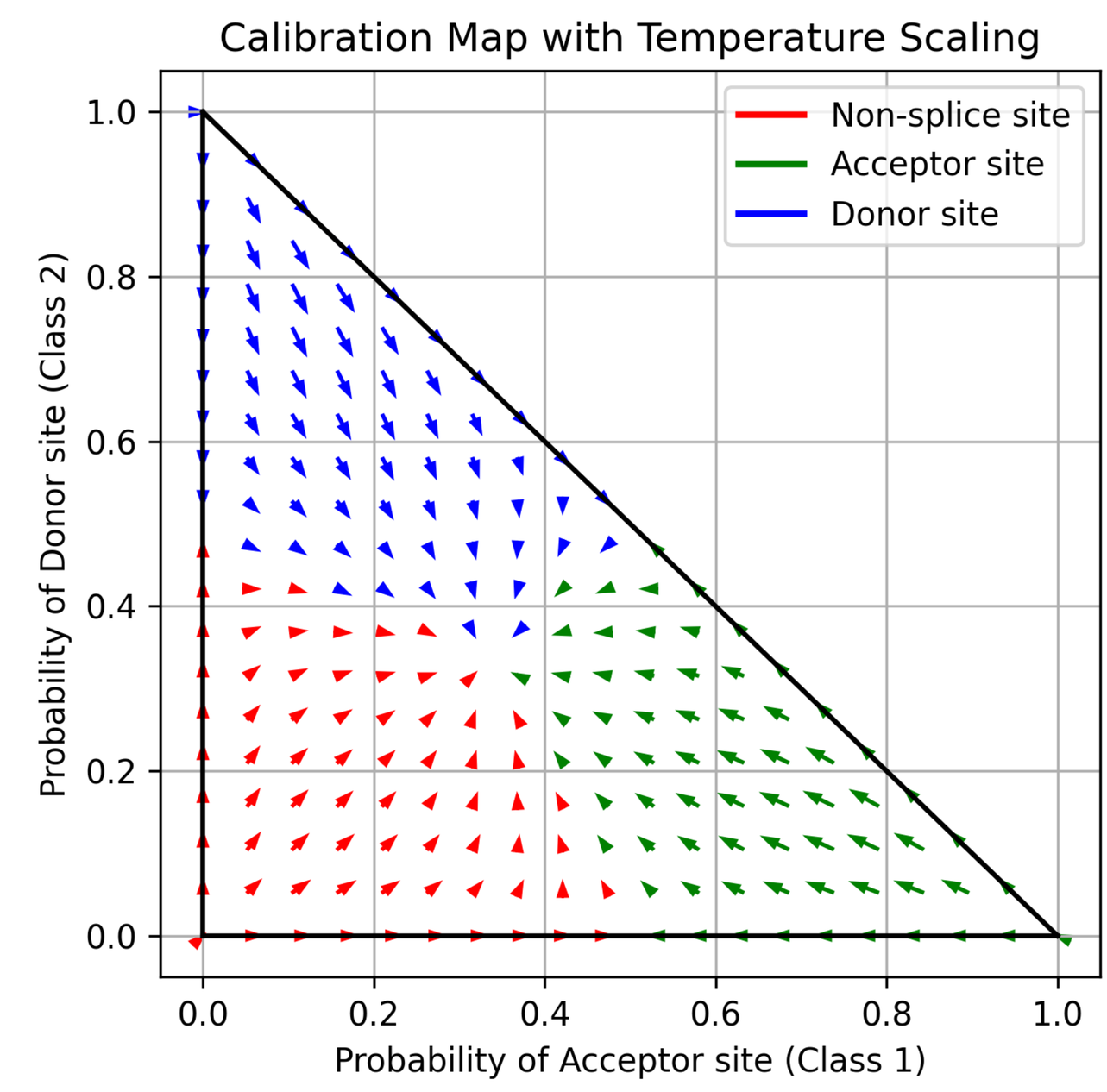

A
Raw gene DNA sequence
Gene 1
L = 33200
Gene 2
L = 14600
...
Gene n
L = 25000
X
Y
[7, 15000, 4]
[7, 5000, 3]
[3, 15000, 4]
[3, 5000, 3]
...
[5, 15000, 4]
[5, 5000, 3]
Train set: 80%
Test set: 20%
B
Input sequence (len: 5,000)
Flanking sequence (len: 5,000)
Flanking sequence (len: 5,000)
A T G C
SpliceAI
=> Donor
=> Acceptor
=> Neither
C
Donor Top-1
Donor Top-1
Flanking Size
SpliceAI-Keras (Human-MANE)
OpenSpliceAI (Human-MANE)
D
Acceptor Top-1
Acceptor Top-1
Flanking Size
SpliceAI-Keras (Human-MANE)
OpenSpliceAI (Human-MANE)
E
[Predict] Elapsed Time (s) vs. Input Size
Elapsed Time (s)
Input Size (# Genes in MANE)
[Predict] Average Memory (MB) vs. Input Size
Average Memory (MB)
Input Size (# Genes in MANE)
[Predict] GPU Peak Memory (MB) vs. Input Size
GPU Peak Memory (MB)
Input Size (# Genes in MANE)
Model Size & Type
Flanking Size
80
400
2000
10000
Model Type
Keras
PyTorch

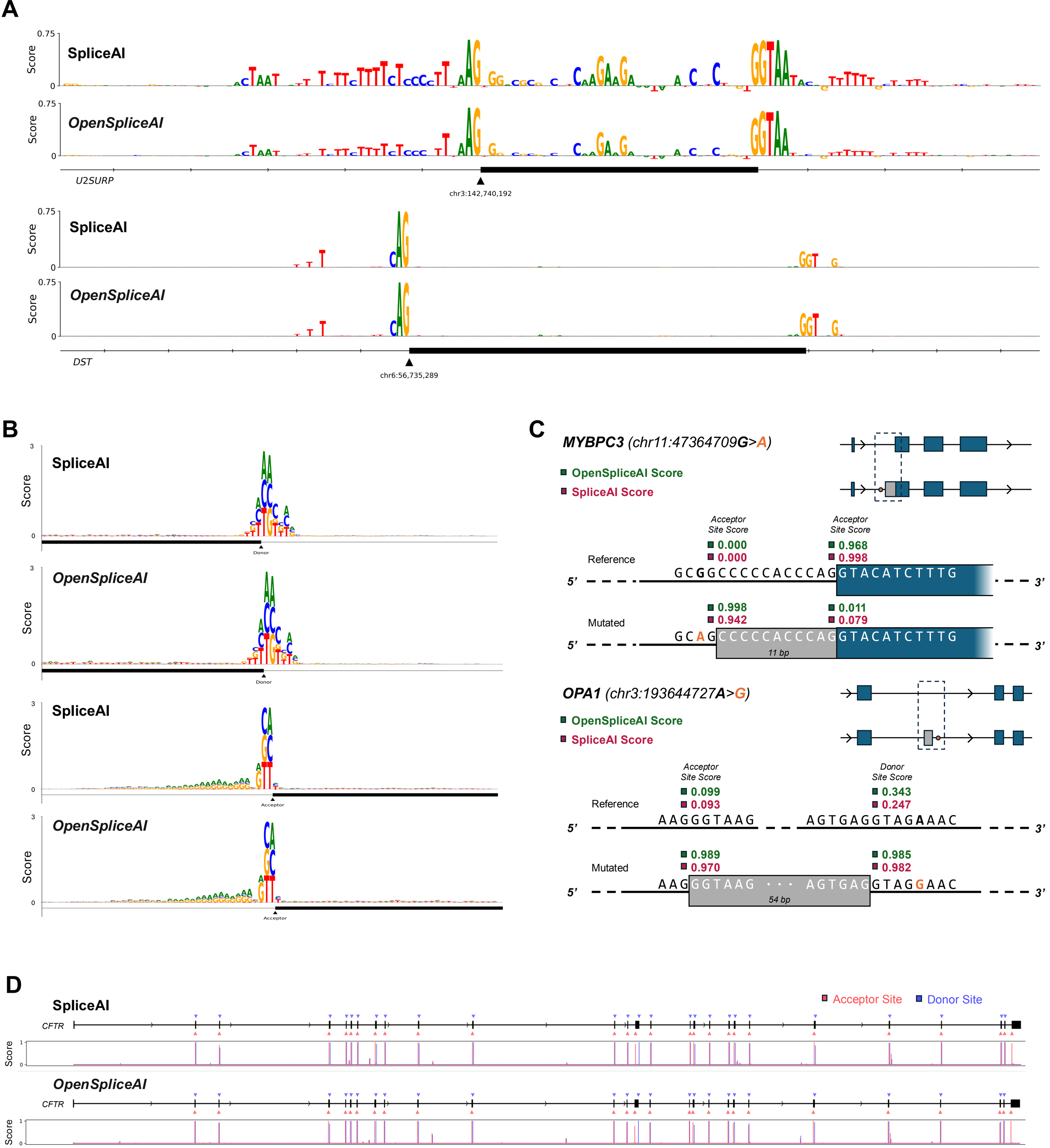

A
Training vs. Combined Test Set (Adjusted & Paralogous) Sizes
Training Set Size
Adjusted Test Set Size
Paralogous Sequences Removed
Gene Sequence Count
13704
39
5483
18049
160
3983
25970
2157
4590
7448
2
2485
21143
145
6269
Human-MANE
Mouse
Zebrafish
Honeybee
Thale cress
Species
B
Alignments between train and test datasets
Human-MANE
Honeybee
Arabidopsis
Zebrafish
Mouse
Query (test) Coverage (%)
Query (test) Alignment Identity (%)
C
Splice site prediction metrics for Mouse
Donor F1
Acceptor F1
SpliceAI-Keras (Mouse)
OpenSpliceAI (Mouse)
Flanking Size
D
Splice site prediction metrics for Zebrafish
SpliceAI-Keras (Zebrafish)
OpenSpliceAI (Zebrafish)
E
Splice site prediction metrics for Honeybee
SpliceAI-Keras (Honeybee)
OpenSpliceAI (Honeybee)
F
Splice site prediction metrics for *Arabidopsis*
SpliceAI-Keras (*Arabidopsis*)
OpenSpliceAI (*Arabidopsis*)


A
Splice Site Prediction Metrics for Mouse (Flank: 80)
Acceptor Top-1
Donor Top-1
Scratch-trained
Transfer-trained
Value
Epoch
B
Splice Site Prediction Metrics for Mouse (Flank: 400)
C
Splice Site Prediction Metrics for Mouse (Flank: 2000)
D
Splice Site Prediction Metrics for Mouse (Flank: 10000)
E
Splice Site Prediction Metrics for Mouse
F
Splice Site Prediction Metrics for Zebrafish
Flanking Size
Scratch-trained (1st epoch)
Scratch-trained (10th epoch)
Transfer-learned (1st epoch)
Transfer-learned (10th epoch)
G
Splice Site Prediction Metrics for Thale cress
H
Splice Site Prediction Metrics for Honeybee


A
Non-splice site
Acceptor site
Donor site
Empirical Probability
Predicted Probability
Original 80nt (T=1.0000)
Calibrated 80nt (T=1.0570/0.9844/0.9547)
Original 400nt (T=1.0000)
Calibrated 400nt (T=1.0496/1.0496/1.0496)
Original 2000nt (T=1.0000)
Calibrated 2000nt (T=1.0100/1.0100/1.0100)
Original 10000nt (T=1.0000)
Calibrated 10000nt (T=1.0099/1.0100/1.0100)


B
Expected Calibration Error (ECE) in Validation Set
Original
Calibrated
ECE
Flanking Sequence
Expected Calibration Error (ECE) in Test Set


C
Calibration Map with Temperature Scaling
Non-splice site
Acceptor site
Donor site
Probability of Donor site (Class 2)
Probability of Acceptor site (Class 1)


A
SpliceAI
OpenSpliceAI
Score
0.75
0
U2SURP
chr3:142,740,192
DST
chr6:56,735,289
B
Donor
Acceptor
C
MYBPC3 (chr11:47364709G>A)
OpenSpliceAI Score
SpliceAI Score
Acceptor Site Score
Reference
0.000
0.000
0.968
0.998
GCGGCCCCCACCCAGGTACATCTTTG
5'
3'
Mutated
0.998
0.942
0.011
0.079
GCAGCCCCCACCCAGGTACATCTTTG
11 bp
OPA1 (chr3:193644727A>G)
Donor Site Score
0.099
0.093
0.343
0.247
AAGGGTAAG
AGTGAGGTAGAAAC
0.989
0.970
0.985
0.982
AAGGGTAAG ··· AGTGAGGTAGGAAC
54 bp
D
Acceptor Site
Donor Site
CFTR