Recreating DeepSTARR Model

Anna Szymik

June 25, 2025

Abstract

Regulatory DNA elements called enhancers play a critical role in controlling gene expression, yet their identification and functional characterization remain challenging due to their complex sequence features. DeepSTARR is a deep learning model trained on UMI-STARR-seq data from *Drosophila melanogaster* cells that predicts enhancer activity from DNA sequence alone. While being highly effective, the original DeepSTARR implementation uses a slightly outdated Keras framework, making further development and interpretation more difficult. In this study, DeepSTARR was reimplemented in PyTorch 2.5.1, preserving its original architecture and functionality. The reimplementation was validated by ensuring identical layer-wise outputs between models using Keras weights. Training on the original dataset achieved comparable results, achieving Pearson correlations of 0.661 and 0.745 for developmental and housekeeping enhancer activity, respectively. Further investigation of the model architecture using Bayesian optimization and the Weights & Biases framework identified a slightly simpler architecture that achieved nearly identical performance. While the model requires the application of decision explainability techniques to verify if it learns biologically relevant enhancer sequence features like the original DeepSTARR, this reimplementation improves compatibility with modern deep learning tools and enables further interpretation and adaptation of the model.

Introduction

Enhancers are sequences that affect gene expression by increasing the probability of binding transcription factors in the promoter region, thereby facilitating transcription initiation. They usually span several hundred base pairs and are located at a distance — usually from several dozen to several hundred kilobases — from the gene they regulate. They can be located both upstream and downstream of the regulated gene [1].

Since enhancers do not possess uniform characteristics and are not strictly limited by their distance to genes, identifying and validating them remains difficult. Therefore, even with numerous experimental and computational efforts, the relationship between DNA sequence and regulatory activity is still not well understood [2].

In recent years, there emerged publications with deep learning models that can predict the activity of regulatory sequences with high accuracy [2, 3]. One of these is DeepSTARR [2], a model designed to predict the activity of enhancers identified by UMI-STARR-seq in *Drosophila melanogaster* S2 cells, a culture of late-stage *Drosophila* embryos [4].

UMI-STARR-seq is a modified version of the STARR-seq (Self-Transcribing Active Regulatory Region Sequencing) technique designed to quantify enhancer activity in low-complexity candidate libraries. Like STARR-seq, it involves inserting candidate DNA fragments into the 3' UTR of a reporter gene in a plasmid so that active enhancers are transcribed as part of the reporter mRNA. UMI-STARR-seq additionally introduces unique molecular identifiers (UMIs) prior to amplification, to allow accurate quantification of individual transcripts through PCR duplicate

correction [5]. The activity of a candidate enhancer can then be measured as the log2 ratio between the normalized number of transcripts and plasmid DNA fragments.

The authors generated genome-wide libraries in *Drosophila* cells using two distinct core promoters: a developmental promoter (DSCP) and a housekeeping promoter (RpS12) and were able to define these two enhancer activities for 484,052 sequences. They selected the first and second halves of chr2R as validation and test sets, with 40,570 (8.4%) and 41,186 (8.5%) sequences, respectively. The DeepSTARR model was able to achieve a Pearson correlation coefficient of 0.68 for developmental and 0.74 for housekeeping enhancer activity in the test set, outperforming methods based on known TF motifs or unbiased k-mer counts [2, 6].

This model is noteworthy for its robustness, as well as explainability – it learned the valid TF motifs and higher-order syntactic rules such as functionally nonequivalent occurrences of the same TF motif, which were experimentally verified [2]. However, it was originally implemented using Keras 2.2.4 [7] with Python 3.7, an older setup that lacks some convenient features like the `to_numpy()` function, which is useful for model interpretation. Therefore, the primary aim of this work was to reimplement DeepSTARR using a more up-to-date framework. For this, PyTorch 2.5.1 [8], compatible with Python 3.12, was chosen.

In addition, this reimplementation provides an opportunity to explore whether the model's performance can be even further improved through minor changes in architecture, such as adjusting the number or size of neurons in the convolutional and fully connected layers. Ultimately, this work aims to both modernize DeepSTARR for continued use and lay the groundwork for future enhancements in regulatory sequence modeling.

Materials and Methods

Dataset

The dataset used in this work is the same as in the original publication [2], based on UMI-STARR-seq enhancer activity screens performed in *Drosophila melanogaster* S2 cells. Genome-wide candidate enhancer libraries were generated by fragmenting genomic DNA from a sequenced *Drosophila* strain (y; cn bw sp) into 200 bp fragments, which were then cloned into STARR-seq plasmids containing either a developmental (DSCP) or housekeeping (RpS12) core promoter. These were then electroporated into S2 cells, and after 24 hours of incubation, polyadenylated RNA was extracted to quantify enhancer-driven transcription. Each transcript was barcoded with its unique molecular identifier (UMI), which allowed for minimizing bias in transcript quantification after PCR amplification. Next-generation sequencing was performed and reads were aligned to the *Drosophila* genome (dm3) using Bowtie v1.2.2 [9], with strict filtering and UMI-based deduplication [2].

Enhancer activity was defined as the log2 fold change in RNA relative to input DNA, and peaks with an adjusted enrichment relative to input greater than 3 and a hypergeometric p-value $\leq$ of 0.001 were selected as active enhancers. These sequences were resized to 249 bp, based on the genomic sequence, to match the input data for model training. The final dataset consisted of 242,026 unique sequences, augmented by their reverse complements to obtain 484,052 input examples. Sequences from the first and second halves of chromosome 2R were used for evaluation as validation (40,570 sequences; 8.4%) and test (41,186 sequences; 8.5%) sets.

Model Translation

DeepSTARR model is a multi-task convolutional neural network that receives one-hot encoded 249 bp long DNA sequences (where A = [1,0,0,0], C = [0,1,0,0], G = [0,0,1,0], T = [0,0,0,1]) as input and outputs two continuous values corresponding to developmental and housekeeping enhancer

activities. Its architecture is adapted from the well-established Basset model, designed to classify regulatory regions based on a DNA sequence [10]. DeepSTARR comprises four 1D convolutional layers (with 246, 60, 60, and 120 filters and kernel sizes of 7, 3, 5, and 3, respectively). Each convolutional layer is followed by batch normalization, a ReLU activation, and max pooling (with window size = 2). The convolutional block is followed by two fully connected layers of 256 neurons each, with batch normalization, ReLU, and dropout (dropout rate = 0.4). A final linear layer returns two outputs representing predicted enhancer activities.

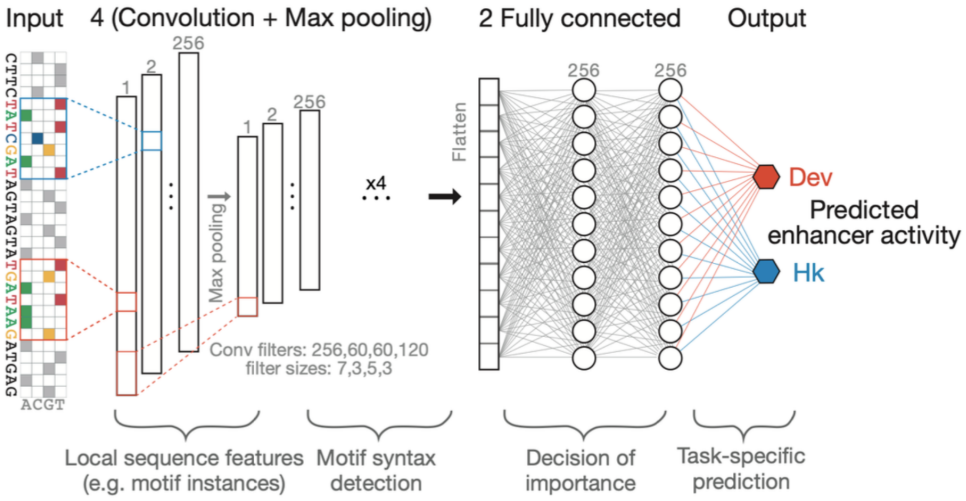


Figure 1: DeepSTARR model architecture diagram. Graphics adapted from [2].

The original model was implemented in Keras (version 2.2.4), whereas the reimplementation presented in this work uses PyTorch (version 2.5.1). These two deep learning frameworks differ in several important aspects that needed to be accounted for during the model translation.

Firstly, convolutional layers in Keras and PyTorch expect input tensors in different shapes. In Keras, the expected input shape is `(batch_size, length, channels)`, whereas in PyTorch it is `(batch_size, channels, length)`. This had to be taken into account during validation to ensure that the re-implemented model behaves identically to the original. The inputs and outputs of the convolutional layers had to be transposed in the PyTorch model to be able to compare them with the outputs of the Keras model layers. A similar issue arose with the flattening operation: Keras's 'Flatten()' layer flattens the 3D tensor in the order of dimensions 0, 2, 1, while PyTorch's 'squeeze()' flattens it in the default 0, 1, 2 order. These differences had to be handled during validation to ensure consistency.

Secondly, there are slight differences in the default settings of the Adam optimizer used during training. One key difference is the value of the epsilon parameter, a small constant added for numerical stability to prevent division by zero. In Keras, the default value of epsilon is 1e-07, whereas in PyTorch it is 1e-08. To ensure consistency with the original DeepSTARR implementation, the epsilon value in the PyTorch version was manually set to 1e-07.

The remaining differences between the two frameworks were mostly syntactic. ChatGPT [11] was helpful during the translation process, especially in mapping the architecture components and syntax between the frameworks, although naturally, everything was manually examined and adjusted to ensure that the final implementation was correct and equivalent to the original.

Training and Performance Evaluation

During the training phase, components that were previously added for validation purposes, such as transposing outputs after each convolutional layer and custom tensor flattening to match Keras behavior, were removed to improve training efficiency. The model training setup was 100 epochs with an early stop patience parameter of 10, the same as in the original implementation [2]. Values of all the other training hyperparameters were also the same, namely the Adam optimizer with a learning rate of 0.002, mean square error (MSE) as loss function, and batch size of 128.

Performance evaluation was performed separately for developmental and housekeeping enhancer predictions using a pre-defined test set. In line with the original work, the primary metric used was the Pearson correlation coefficient (PCC) between predicted and observed enhancer activities across all test sequences.

Architecture Tuning

Once it was verified that the reimplemented model achieved comparable performance to the original one (see Results section), the architecture was further explored to determine whether the model's performance could be fine-tuned. For this purpose, the Weights & Biases platform [12] was used to perform a systematic search over a defined parameter grid. The search configuration included hyperparameters such as batch size, number of training epochs, early stopping patience, learning rate, number of convolutional layers, the number and size of filters in each convolutional layer, the number and size of fully connected layers, and dropout probability.

A Bayesian optimization strategy was used to maximize the Pearson correlation coefficient between predicted and observed developmental enhancer activity values on the validation dataset. This value was chosen due to the model's performance for developmental enhancers being lower than for housekeeping ones, suggesting more room for improvement.

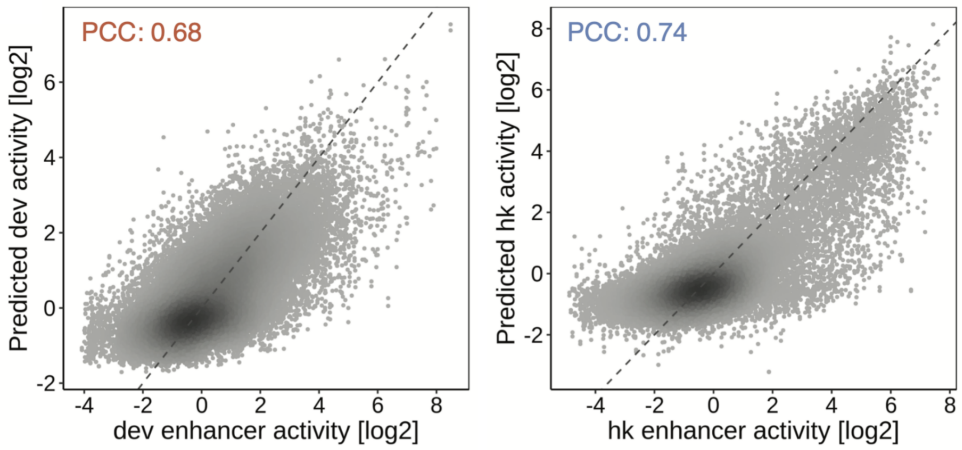
Results

Model Validation

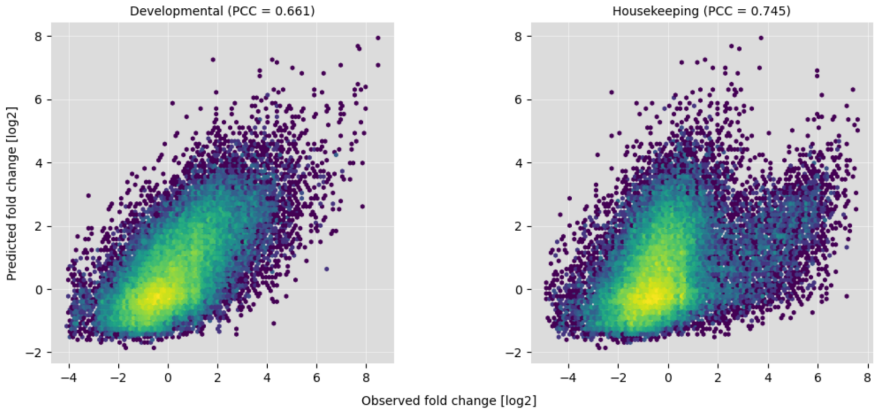
For the example input, after minor modifications described in Model Translation section, after loading Keras weights into the PyTorch model, the outputs of all layers for two models were the same. This was a confirmation that the models are equivalent and the next step was to proceed to training the PyTorch model.

Training Results

The trained model achieved results comparable to those presented in the original work [2], achieving Pearson correlation coefficients of 0.661 and 0.745 for developmental and housekeeping enhancers, respectively (see Fig. 2). The only thing that may be concerning is the shape of the plot for developmental enhancers, where we can see a splitting of the data into two trends, a bifurcation that is not visible in the prediction plot for the original DeepSTARR model. This may potentially indicate the presence of two subtypes or a non-linear relationship between predicted and measured activity. This could potentially indicate the presence of two subtypes or a nonlinear relationship between predicted and measured activity. This effect could be a target for further investigation, such as unsupervised sequence clustering or motif enrichment analysis.



(a) Predictions of the original DeepSTARR model on the test set. Graphics adapted from [2].



(b) Predictions of the reimplemented model on the test set.

Figure 2: Comparison of the predictions of the original DeepSTARR model (a) and its reimplementation (b). Scatter plots of predicted versus observed enhancer activity across all sequences in the test set chromosome for developmental (left) and housekeeping (right) enhancers. Color reflects density.

Architecture Tuning

The model with the highest Pearson correlation coefficient on the validation set after grid search on hyperparameters turned out to have a slightly different architecture from the original model. It differs from it in the number and size of filters in the convolutional layers (having 128, 60, 60, and 30 filters and kernel sizes of 7, 3, 3, and 7, respectively) and has one fully connected layer with 256 neurons instead of two. However, the differences in the results of this model on the test set compared to the results of the model without changes to the architecture are negligible (Pearson correlations of 0.665 vs. 0.661 for developmental enhancers, see Fig. 3).

Nevertheless, despite the modest difference, the architecture search process revealed that simpler models, with smaller convolutional layers and fewer fully connected layers, could achieve near-optimal results. This suggests that some of the complexity in the original architecture may be redundant and that model size can potentially be reduced without sacrificing accuracy, although this would need to be further investigated, as the tuned model's results still show trend splitting as was the case with the model with the original architecture (see Fig. 2 for reference).

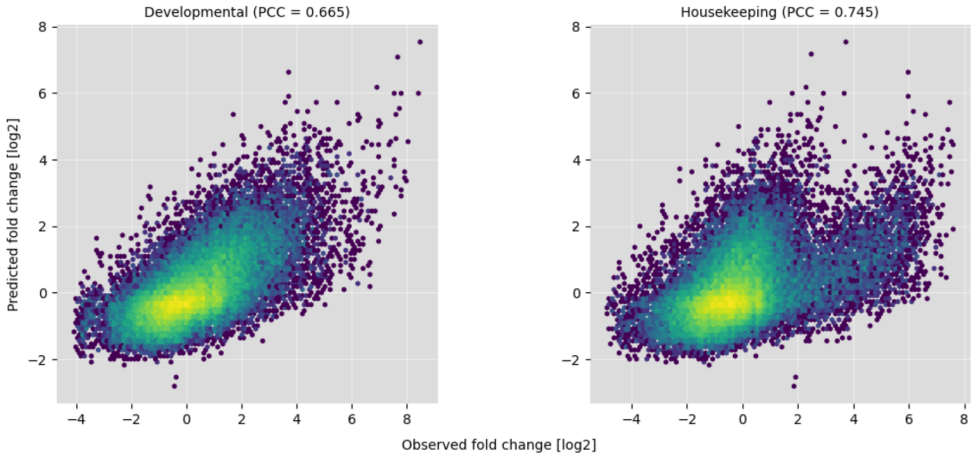


Figure 3: Predictions of the modified DeepSTARR model on the test set. Scatter plots of predicted versus observed enhancer activity across all sequences in the test set chromosome for developmental (left) and housekeeping (right) enhancers. Color reflects density.

Discussion

This work successfully reimplemented the DeepSTARR enhancer activity prediction model using the PyTorch framework, demonstrating its functional equivalence to the original Keras version. The PyTorch model reproduces key performance metrics and simplifies further development thanks to the improved compatibility with modern Python and machine learning ecosystems.

The reimplemented model also provides flexibility to experiment with architecture. Although detailed hyperparameter tuning did not yield significant performance improvements, it suggested that simpler models could achieve comparable results. This finding raises questions about the minimum architectural requirements for accurate enhancer prediction. This might be useful when developing models for other species or enhancer types.

However, some challenges remain. The split of the trend observed in developmental enhancer

predictions, absent from the original model results, may reflect subtle differences in the training process or data preprocessing. Further analysis is required to determine whether this pattern reflects biological heterogeneity or modeling artifacts.

This is only one of the numerous reasons why it would also be worthwhile to conduct future model explanation, as the authors of the original paper did. It could verify if the PyTorch model also learns biologically relevant elements of enhancer syntax.

In summary, this work has achieved its primary goal of reimplementing the DeepSTARR model in PyTorch, validating its performance and architectural flexibility. In addition to its technical validation function, this reimplementation provides a foundation for future studies aimed at analysing the regulatory code of enhancers. By being up-to-date with modern tools, the PyTorch model opens new possibilities for investigating how neural networks learn to decode the complex nature of regulatory sequences.

Code availability

The code used to download UMI-STARR-seq data, train models, grid search architecture hyperparameters, and predict enhancer activity from DNA sequences is available on GitHub (https://github.com/aszymik/deep-starr).

References

[1] Gasperini, M., Tome, J. M., & Shendure, J. (2020). Towards a comprehensive catalogue of validated and target-linked human enhancers. *Nature Reviews Genetics, 21*(5), 292–310.

[2] De Almeida, B. P., Reiter, F., Pagani, M., & Stark, A. (2022). DeepSTARR predicts enhancer activity from DNA sequence and enables the de novo design of synthetic enhancers. *Nature Genetics, 54*(5), 613–624.

[3] Deng, C., Whalen, S., Steyert, M., Ziffra, R., Przytycki, P. F., Inoue, F., Pereira, D. A., Capauto, D., Norton, S., Vaccarino, F. M., PsychENCODE Consortium‡, Pollen, A. A., Nowakowski, T. J., Ahituv, N., Pollard, K. S., Akbarian, S., Abyzov, A., Ahituv, N., Arasappan, D., . . . Zintel, T. M. (2024). Massively parallel characterization of regulatory elements in the developing human cortex. *Science, 384*(6698), eadh0559.

[4] Schneider, I. (1972). Cell lines derived from late embryonic stages of Drosophila melanogaster. *Development, 27*(2), 353–365.

[5] Neumayr, C., Pagani, M., Stark, A., & Arnold, C. D. (2019). STARR-seq and UMI-STARR-seq: Assessing Enhancer Activities for Genome-Wide-, High-, and Low-Complexity Candidate Libraries. *Current Protocols in Molecular Biology, 128*(1), e105.

[6] Ghandi, M., Lee, D., Mohammad-Noori, M., & Beer, M. A. (2014). Enhanced Regulatory Sequence Prediction Using Gapped k-mer Features. *PLoS Computational Biology, 10*(7), e1003711.

[7] Chollet, F., & others. (2015). Keras (Version 2.2.7) [Computer software]. https://github.com/keras-team/keras

[8] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2024). PyTorch: An imperative style, high-performance deep learning library (Version 2.5.1) [Computer software]. https://pytorch.org

[9] Langmead, B., Trapnell, C., Pop, M. & Salzberg, S. L. (2009). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol, 10*(R25).

[10] Kelley, D. R., Snoek, J., & Rinn, J. L. (2016). Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Research, 26*(7), 990–999.

[11] OpenAI. (2025). *ChatGPT* (May 14 version) [Large language model]. https://chat.openai.com/chat

[12] Biewald, L (2020). *Experiment Tracking with Weights and Biases* [Computer software]. https://www.wandb.com/