# Surveying Machine Learning Methods for Predicting the Outcomes of Prime Editing

1592800

April, 2024

# Contents

# 1 Background and Introduction

Genome editing is a powerful tool for understanding the genetic basis of life and for developing new therapies. The ability to precisely edit the genome of living organisms has been a long-standing goal of genetic engineering and inspired the development of various technologies.

The discovery of CRISPR(Clustered Regularly Interspaced Short Palindromic Repeats) and its associated family of Cas9 proteins revolutionized the field of genetic engineering. CRISPR-Cas9 is an adaptive prokaryotic immune system that detects invading viruses and plasmids, and damages their gene sequences. It was harnessed by researchers to provide a precise method for editing the genome of living organisms, leveraging its ability to recognize specific gene sequences and produce scission[1]. A number of CRISPR-Cas9 based tools have been developed since the discovery of the original system, including CRISPR-Cas9 HDR(Homology Directed Repair), base editors, and prime editors.

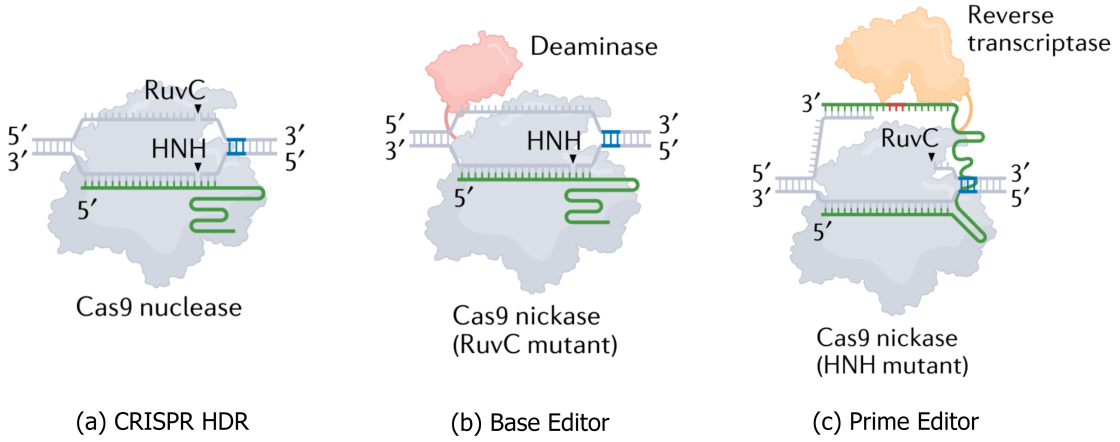


Figure 1: Structure of CRISPR-Cas9, Base Editors, and Prime Editors[2]

## 1.1 CRISPR-Cas9 HDR

Shown in Figure 1 (a), the CRISPR-Cas9 system consists of two components: a single guide RNA(sgRNA) that binds to a target DNA sequence(the protospacer) complementary to the sgRNA, and a Cas9 protein that cuts both strands of the DNA at the target site. To install intended edits, an exogenous DNA template is also provided to the cell. The cell's endogenous repair pathway Homology Directed Repair(HDR) can then uses the template to repair the broken DNA sequence, resulting in the desired edit.

However, a competing repair pathway, Non-Homologous End Joining (NHEJ), is the preferred pathway in many cell types and can introduce unwanted insertions and deletions (indels) at the cut site. This can lead to undesirable effects and limit the therapeutic potential of CRISPR-Cas9 HDR system. As a result, CRISPR-Cas9 HDR is now mostly used for the disruption of the target genome instead of precise editing[3].

## 1.2 Base Editors

Developed by David R. Liu and colleagues, base editors eliminate the involvement of NHEJ by directly converting one base pair to another without introducing a double-strand break (DSB). The components of a typical base editor are shown in Figure 1 (b). The system consists of a Cas9 nickase fused to a deaminase enzyme that converts a base pair to another, and a sgRNA that binds to the protospacer. The Cas9 nickase nicks the DNA strand complementary to the sgRNA at target location, then the deaminase enzyme chemically converts the target base pair. Finally, the cell's endogenous repair system repairs the nicked DNA strand using the modified strand as template, installing the desired edit into the genome[4].

The base editors significantly increased the editing efficiency and reduced the off-target effects compared to the CRISPR-Cas9 HDR system. However, limited by the deaminase enzyme, base editors can only introduce point mutations. Additionally, most base editors can only introduce transition mutations(A-T $\Longleftrightarrow$ G-C) and not transversion mutations(G-C $\Longleftrightarrow$ C-G, A-T $\Longleftrightarrow$ T-A), further limiting their viabilities[3].

## 1.3 Prime Editors

Determined to fix the limitations of CRISPR HDR and base editors, David R. Liu and colleagues developed prime editors, a new class of genetic editing tools that can introduce a wide range of edits, including varying lengths of insertions and deletions. Figure 1 (c) shows the structure of a prime editor.

The system consists of a SpCas9 nickase fused to a reverse transcriptase and a prime editing guide RNA(pegRNA). The pegRNA is similar to sgRNA used in CRISPR-Cas9 and base editors at its 5′ ends(and thus this section will be referred to as sgRNA in the rest of the survey), but is extended in its 3′ end to include a prime binding site(PBS) and a reverse transcriptase template(RTT) complementary to the desired edit. The two functional ends are connected together with a tracrRNA scaffolding sequence that should have no significant effects during the editing process.

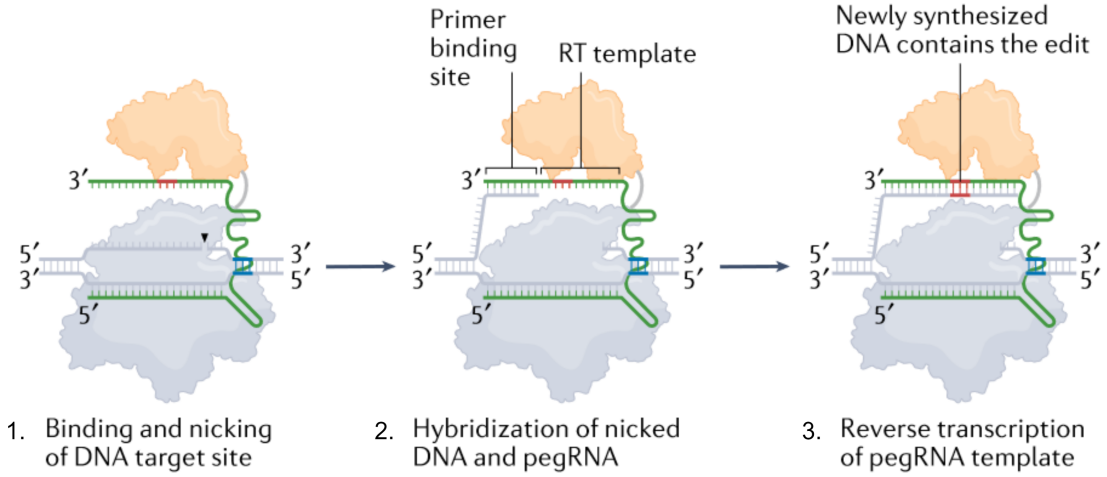


Figure 2: Prime Editing Process[2]

The prime editing process is a multi-stage procedure consisting of[5]:

1. The sgRNA at the 5′ end of pegRNA binds to the protospacer, then the Cas9 nickase introduces a scission in the exposed strand.
2. The PBS at the 3′ end of pegRNA binds to the now displaced strand.
3. The reverse transcriptase introduces the edits according to the template provided.
4. The edited strand hybridizes with the non-edited stand, reforming the double helix.
5. The cell's endogenous repair system installs the edits into the non-edited strand, permanently adding the modification into the genome.

The first three steps of the process are illustrated in Figure 2.

This concludes the editing process for prime editors PE1 and PE2, where PE2 is the much more efficient version of PE1 using a different reverse transcriptase[5]. The more efficient PE3 includes an extra ngRNA(nicking guide RNA) that introduces a nick in the non-edited strand to encourage the cell's DNA repair system to use the edited strand as the template[5].

More advanced versions of the prime editors have been developed after the publication of the original prime editors(PE1-PE3), two important examples are PE4 and PE5. PE4 and PE5 are the results of combining an additional MLH1dn protein with the original PE2 and PE3 respectively.

The MLH1dn protein is a dominant negative mutant of the human MutL homolog 1(MLH1) protein, which is involved in the mismatch repair pathway(MMR). MMR can revert the nicked heteroduplex formed when the edited 3′ DNA flap anneals to the genome in the final step, reducing editing efficiency when the edits produce mismatches known to be repaired by MMR.

The edits repairable by MMR include single base substitution as well as short sequence insertion or deletion. The MLH1dn protein inhibits MMR activities shown to antagonize the prime editing efficiency in those cases, increasing the overall editing efficiency of the prime editors[6].

## 1.4 pegRNA Design

For all versions of the prime editors, the design of pegRNA is the most crucial step of the editing process. Performing the same edit on the same target sequence with different pegRNAs can result in different editing efficiencies. However, the experimental process of testing the efficiency of different pegRNAs on a specific target sequence is laborious and expensive, significantly limiting the therapeutic potential of prime editing[7].

A number of in silico methods have thus been developed, using the hardcoded design guideline suggested by the Liu lab to recommend a number of pegRNA sequences for a given target sequence[8]. However, how to unbiasedly optimize the combination of the features from the design guide and identify the most suitable sequence from a list of candidates remains problematic[9].

pegRNA design has the additional challenge of prime editing being a multi-stage process. The resulting sequence of each stage will have an impact on the efficiencies of the subsequent stages, creating a significantly larger search space for the traditional optimization methods when compared to less complex methods such as base editing.

These problems inspired a cohort of researchers to apply machine learning methods to predict the outcomes of prime editing using a given pegRNA, leveraging a number of known determinants of prime editing efficiency as well as the pattern finding capabilities of machine learning algorithms. This survey aims to review the methods currently available and to identify new opportunities in the field.

# 2 Methods For pegRNA Design

## 2.1 Determinants of Prime Editing Efficiency

Although the exact determinants of prime editing efficiency are not known to us due to the complex nature of prime editors[7], high throughput screening from a number of studies have revealed a number of features that have strong correlation with the efficiencies of prime editing[7, 10, 11, 12, 6]. I hereby summarize the findings of these studies for better understanding of the machine learning methods that may utilize them in their models.

### 2.1.1 Prime Editor Features

The composition of each component of the prime editor as well as the mutations to install(indicated by the RTT) will clearly have an impact on the efficiency of the editing process. Specifically, the features that have been found to be correlated with the editing efficiency include:

1. **RHA Length**: The right homology arm(RHA) is the section of RTT after all the edits(starting from the $3'$ end), complementary to the non-edited strand. They help the edited strand hybridize with the non-edited strand, increasing the success rate of the edits being installed into the genome. As a result, a RHA too short would result in a lower editing efficiency[10].
2. **pegRNA Secondary Structure**: It was shown that stronger secondary structure of the pegRNA homology arm can increase the editing efficiency[12].
3. **GC Content and Melting Temperature of PBS**: Higher GC content and melting temperature of the PBS can increase the efficiency of the editing process[10]
4. **PAM Sequence Disruption**: The protospacer adjacent motif(PAM) is a short sequence that is required for the binding of the Cas9 protein to the target DNA downstream of the protospacer. The PAM sequences were shown as blue base pairs in Figure 1 and Figure 2.

   The prime editors have the possibility of rebinding to the edited DNA sequence before the cellular repair mechanism installs the edits into both stands. This can lead to the introduction of unwanted edits.

   The disruption of the PAM sequence can prevent the rebinding and thus improve the efficiency of the editing process[2, 11, 12].
5. **Length of Modifications to Insert**: As expected, substituting, deleting or inserting a longer sequence is more difficult and thus less efficient than a shorter sequence[13].

The efficiency remains relatively consistent for up to 3-5 base pairs(bp), but starts to decrease after that[10].

6. **Prime Editor Type**: Obviously, the different versions of prime editors have different efficiencies due to whether beneficial components such as the MLH1dn protein and the extra ngRNA are included or not.

### 2.1.2 Target Sequence Features

The exact composition of the target sequence(protospacer) itself also has a significant impact on the result of editing. The sequence features that have been found to be correlated with the editing efficiency include:

1. **Poly-T sequences**: Consecutive A-T sequences in the protospacer can reduce the effiency of the editing process, as poly-U sequences in the spacer(gRNA) of the pegRNA are known to cause catalytic inactivation and backtracking of RNA polymerase III[7].
2. **GC Content**: Having G/C nucleotides directly flanking the target base pair during single base editing can increase the efficiency of the editing process[7].
3. **Single Nucleotide Composition**: The appearance of certain nucleotides at specific positions in the protospacer can significantly increase or decrease editing efficiency. Suppose that position 1 is the 20th base pair from the PAM sequence, a T at the 16th position can hamper PE efficiency, while a G at the 17th position can increase the efficiency[11]. This is consistent with previous results showing that these positions are important for Cas9 nickase activity[14].

### 2.1.3 Target Context Features

The in vitro or in vivo editing process is not isolated, and the environment the editors work under should also be considered when making predictions on editing efficiencies. Examples of the context features are:

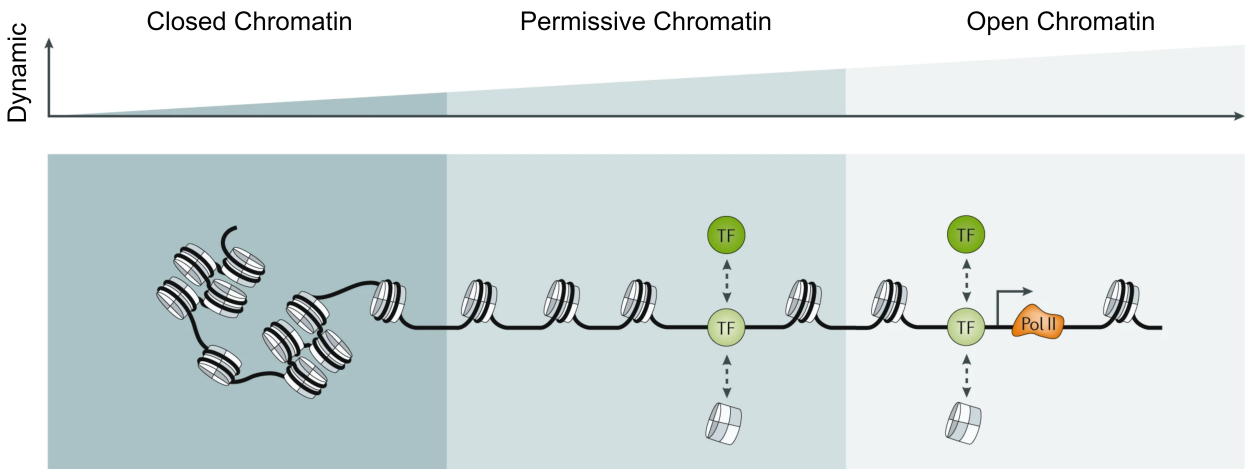


Figure 3: Levels of Chromatin Accessibility[15]

1. **Chromatin Accessibility of the Target Sequence**: The chromatin is the complex of DNA and proteins that make up the chromosomes in the Eukaryotic cells, such as the human embryonic kidney(HEK) cell lines often used in the experiments testing the capability of editors in human cells[16]. Different levels of chromatin accessibility are illustrated in Figure 3, where higher editing efficiency should be expected with more open chromatin[7].
2. **Target Cell Type**: Different cell types used in the testing have different MMR activities. The HEK293T cell line is partly MMR deficient, while other cell lines such as HAP1(derived from a chronic myelogenous leukemia patient) are not. This results in PEs without the additional MLH1dn protein in PE4 and 5 having lower editing efficiency when doing short sequence substitution or insertion/deletion in the HAP1 cell line, compared to the HEK293T cell line[12].

## 2.2 Machine Learning Methods

Table 1 summarizes and compares some useful aspects of the machine learning methods to be discussed in this review. The supported cell types and PE versions are the datasets the models were trained and achieved the good performance on in the original studies, while their corresponding online web interfaces may be continuously updated with new supported cell lines and PEs.

### 2.2.1 Sequence Data Embedding

The embedding of the sequence data is very similar among the models. The sequences are one-hot encoded as the presence of a base pair at each location, producing a $4 \times X$ or $3 \times X$ matrix, where $X$ is the length of the input sequence. An example is shown in Figure 4.


Figure 4: One-Hot Encoding of Target Sequence

With this in mind, we can now start the introduction of the machine learning methods developed for predicting the outcomes of prime editing.

### 2.2.2 DeepPE

Developed by Kim et al, DeepPE is one of the earliest attempt at predicting the outcomes of prime editing using machine learning and illustrated many possible determinants of PE efficiency. Most of the determinants discovered in their study are still valid today, but the model itself is not as relevant in terms of performance due to the constraints in datasets at the time of their publication. The model is also very limited in terms of editing types supported, focusing on predicting the efficiency of G to C substitution at the position +5 nick site of the target sequence.

| Method | Supported Cell Types | Supported Edits | Supported PEs |
|---|---|---|---|
| DeepPE | HEK293T | G to C substitution at position +5 nick site | PE2 |
| EasyPrime | HEK293T | GWAS Variants | PE2, PE3, PE3b |
| MinsePIE | HEK293T, HAP1 | Insertion | PE2, PE4 |
| DeepPrime(-FT) | HEK293T, HCT116, DLD1, MDA-MB-231, A549, HeLa, NIH3T3 | Substitution, Insertion, Deletion | PE2, PE2max, PE2max-e (PE2max with epegRNAs), PE4max, PE4max-e, NRCH-PE2, NRCH-PE2max, NRCH-PE4max |
| PRIDICT | HEK293T, K562, U2OS | Substitution, Insertion, Deletion | PE2, PE2-max,PE3, PE3b, PE4 |

| Method | Code Availability | Web Tool | Reference |
|---|---|---|---|
| DeepPE | `https://github.com/hkimlab-PE/PE_SupplementaryCode` | `https://deepcrispr.info/DeepPE/` | [11] |
| EasyPrime | `https://github.com/YichaoOU/easy_prime` | `http://easy-prime.cc/` | [9] |
| MinsePIE | `https://github.com/julianeweller/MinsePIE` | `https://elixir.ut.ee/minsepie/` | [12] |
| DeepPrime(-FT) | `https://github.com/hkimlab/DeepPrime` | `https://deepcrispr.info/DeepPrime/` | [10] |
| PRIDICT | `https://github.com/uzh-dqbm-cmi/PRIDICT` | `https://www.pridict.it/` | [7] |

Table 1: Comparison of Machine Learning Methods for Predicting Prime Editing Outcomes

After the users provided the target sequence to edit, the web tool will propose a number of pegRNA sequences and evaluate their efficiency using the DeepPE model. The 47-nt long target sequence and the 17 to 37nt RTT+PBS sequences, as well as 20 explicit features including the GC content and melting temperature of the PBS are used as input to the model. The two nucleotide sequences are one-hot encoded into four dimensional matrices as mentioned in section 2.2.1. The two embedded sequences and the explicit features are then concatenated(stacked) together and fed into a convolutional neural network with 10 $3 \times 4$ filters. The output is pooled using a deep reinforcement learning model instead of a traditional pooling layer, and then input into a fully connected layer with 1000 units. The result is linearly transformed to the DeepPE prediction score, indicating the efficiency of the editing process on the target sequence using the provided PBS and RTT.

A sketch of the model architecture is shown in Figure 5.

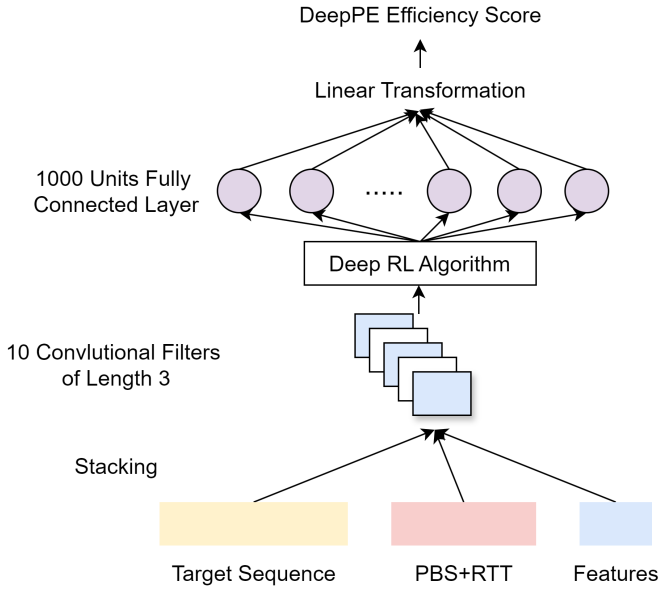


Figure 5: DeepPE Model Architecture

This architecture had the highest performance among the methods reviewed by the authors, although not significantly higher than L1 Lasso regression model. The model achieved a Spearman's R of 0.7 to 0.8, and a Pearson's r of 0.6 to 0.7 on the held-out as well as generalization(unseen) datasets.

The authors also developed another multi-layer perceptron(MLP) model to predict the editing efficiency of more general editing types, including single-nucleotide insertions, deletions and substitutions. However, random forest achieved better performances and was thus used in the additional PE_Type and PE_Position models. PE_Type model proposes pegRNA for 24 possible edits, including specific single-nucleotide deletions, insertions, and substitutions at designated locations. PE_Position model proposes pegRNA optimized to perform substitutions at ten more positions, namely positions 1, 2, 3, 4, 6, 7, 8, 9, 11 and 14.

### 2.2.3 Easy Prime

EasyPrime is a XGBoost regression model developed by Liu et al to produce design for RTT, PBS and ngRNA. Instead of arbitrary mutations, Easy-Prime predicts the editing efficiency of the variants logged in the Genome-Wide Association Studies(GWAS) database. The GWAS variants are the single nucleotide polymorphisms(SNPs) that have been associated with particular traits or diseases.

Similar to DeepPE, for each variant, the web tool proposes a number of pegRNA design using the constrains on PBS, RTT and ngRNA length provided by the user. The proposed pegRNAs are then evaluated by the XGBoost models to find the optimal candidate. When producing the efficiency score, EasyPrime takes the extracted features from pegRNA and target sequences as input to the model instead of the sequences themselves. The extracted features include GC content of the PBS, PAM sequence disruption, as well as several target mutation features describing the mutations to insert.

Cas9 activity score produced by DeepSpCas9 is also used as a feature in the model. DeepSpCas9 is a convolutional neural network model that predicts the activity of the SpCas9 protein on a given target sequence and sgRNA pair. The model architecture is very similar to DeepPE, with one convolutional layer followed by three fully connected layers. The unique feature of the model is the use of filters of different sizes in the convolutional layer(3, 5, and 7nt), allowing the model to capture the dependencies between the base pairs at different distances[17].

Albeit limited in the type of edits supported, EasyPrime is one of a few methods that provides official support for ngRNA design required by PE3 and PE3b. Note that PE3b is the optimized version of PE3, where the ngRNA is selected to avoid possible DSB by not targeting the nucleotide complementary to the nicked position[5].

Constrained by the data available and the simple architecture of the model, the performance of EasyPrime is relatively low compared to the other models reviewed. The model achieved a Spearman's R and Pearson's r of 0.5 to 0.6 on their held-out datasets.

### 2.2.4 MinsePIE

Koeppel et al. developed MinsePIE, a model that focuses on predicting the efficiency of insertions of varying lengths. It supported editing of cells with or without MMR deficiency, which can be interpreted as the result of using PE2 or PE4 with additional MLH1dn protein.

The authors compared the performance of Lasso regression, Ridge regression, Random Forest, XGBoost regression, and a MLP model with hidden layers of size $(1000, 100)$. The Gradient Boosting model with the XGBoost library achieved the best performance, and was thus selected as the final model.

The web tool works nearly identically to DeepPrime, with the additional constraint in homology arm length. The sequence to insert is surrounded by curly brackets inside of the target sequence. The user can also explicitly define a set of PBS, RTT and spacer, omitting the step of the web interface proposing candidate pegRNA for the model to evaluate.

It also only takes ten extracted features as input to the model instead of the one-hot encoded sequences. This significantly reduces the computational cost of the model, as the one-hot encoding of the sequences can be very large for long sequences. On the other hand,

it also limits the model's ability to capture certain unknown features that are present in the sequences. This resulted in a relatively weaker performance compared to the other models published around the same time(DeepPrime, PRIDICT). However, with the significantly larger training datasets, it still outperformed EasyPrime designed with a similar architecture.

The final model trained on the full training set achieved a correlation of 0.68 on held-out sequences, with performance ranging from R = 0.44 to 0.92 when restricted to datasets from individual screens.

### 2.2.5 DeepPrime

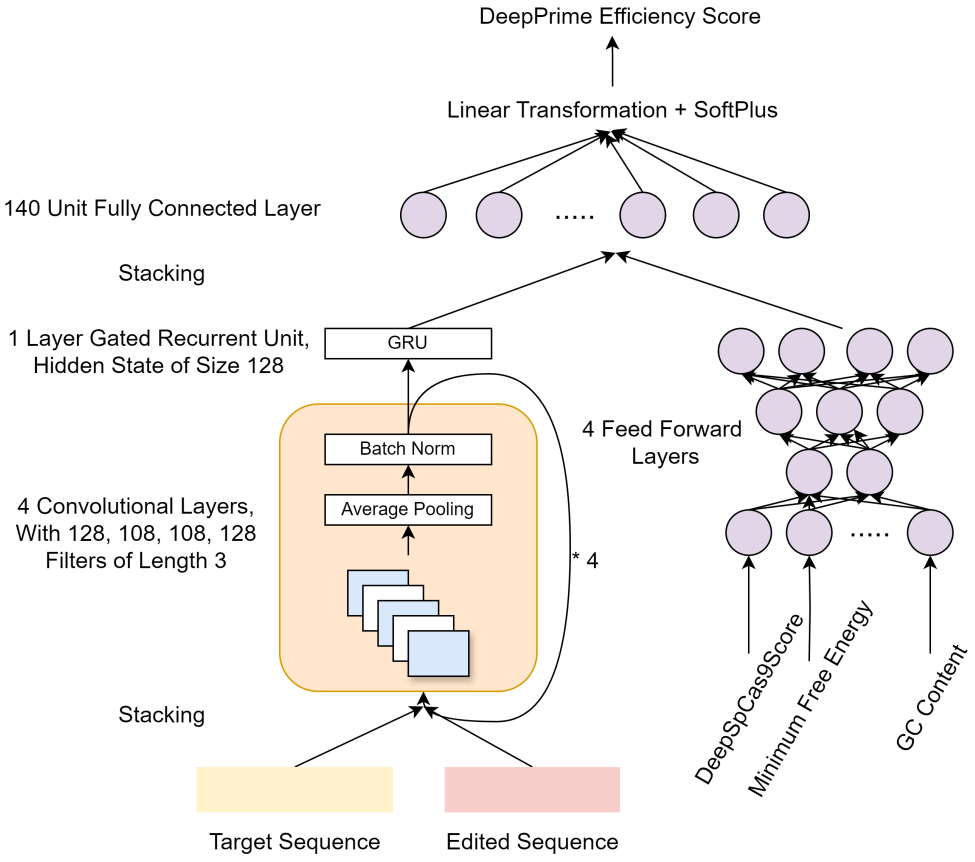


Figure 6: DeepPrime Model Architecture

Also developed by Kim et al., DeepPrime is the updated version of DeepPE with an upgraded model architecture.

Illustrated in Figure 6, instead of PBS + RTT with target sequence, it takes the wild type(unedited) as well as the edited sequences as input to the CNN. The convolutional network is significantly larger than DeepPE, containing four convolutional layers with 128, 108, 108, and 128 filters, respectively. Moreover, conventional average pooling is used this time round after each convolutional layer instead of a deep RL algorithm. Batch normalization is also applied to accelerate training. The output of the convolutional layer is then processed by bidirectional Gated Recurrent Units(GRU).

Instead of processed together with the embedded sequence, features extracted from the proposed pegRNA and target context sequence are processed using a separate four-layer feed forward neural network. The outputs from the two networks are then stacked and fed into a fully connected layer. The result is linearly transformed and processed by a SoftPlus activation function to produce the final prediction score.

DeepPrime is effectively a multi-task learning model, with a base model trained using the combination of 18 datasets of different PE and cell line combinations. The base model is then fine tuned using each of the 18 datasets to produce a task specific model for each setting, collectively named DeepPrime-FT.

Shown in Table 1, the max PEs(PE2max, PE4max) are the versions with updated reverse transcriptase and SpCas9 nickase. epegRNAs(engineered pegRNAs) are pegRNAs with additional 3′ RNA structural motif that increases prime editing efficiency[6]. NRCH-PEs are PEs that supports the additional NRCH PAM.

The amount of training data used makes DeepPrime the most comprehensive method in terms of the number of cell lines and PE versions covered. At the same time, the multi-task design allows DeepPrime-FT to utilize the share features between the different PEs and cell lines, and thus achieve very high performance.

The model has made significant improvement compared to DeepPE, with a Spearman's R of 0.8 to 0.9, and a Pearson's r of 0.7 to 0.9 on most of the testing datasets, including the generalization datasets unseen during training.

### 2.2.6 PRIDICT

Developed by Gerald Schwank et al, PRIDICT utilizes a sophisticated attention-based bidirectional RNN model with a similar pegRNA recommendation pipeline to DeepPE and DeepPrime.

The model overall is a three encoder one decoder architecture. Two of the encoders are attention-based bidirectional RNN models, learning the vector representation of the sequence data. The third encoder is a feed forward neural network taking explicit features derived from the proposed pegRNA, such as the length of the modifications to insert and melting temperature of the PBS, as inputs, similar to DeepPrime.

The target and mutation sequences are one-hot encoded as described in section 2.2.1, alongside three additional binary encoding indicating whether the nucleotide belongs to the protospacer, RTT or PBS. The four embeddings are stacked together into a vector of length 9 for each token in the target sequence and 7 for the mutated sequence(the protospacer embedding is omitted for mutated sequence) and fed into the model.

The bidirectional RNN model is used to capture the dependencies between the base pairs within the whole sequence, instead of only past information captured by unidirectional RNN models. Two separate attention query vectors then pools(compresses) sequence of token-level representations into one fixed length vector using the calculated attention weights. One query vector pools all tokens of the sequences, providing context. The other pools only RTT tokens, focusing on the part where the edits are made.

The decoder is another feed forward neural network with residual connections and layer normalization, taking the pooled vectors from the encoders and calculating the probability distribution of possible outcomes of the edits when using the proposed guide.

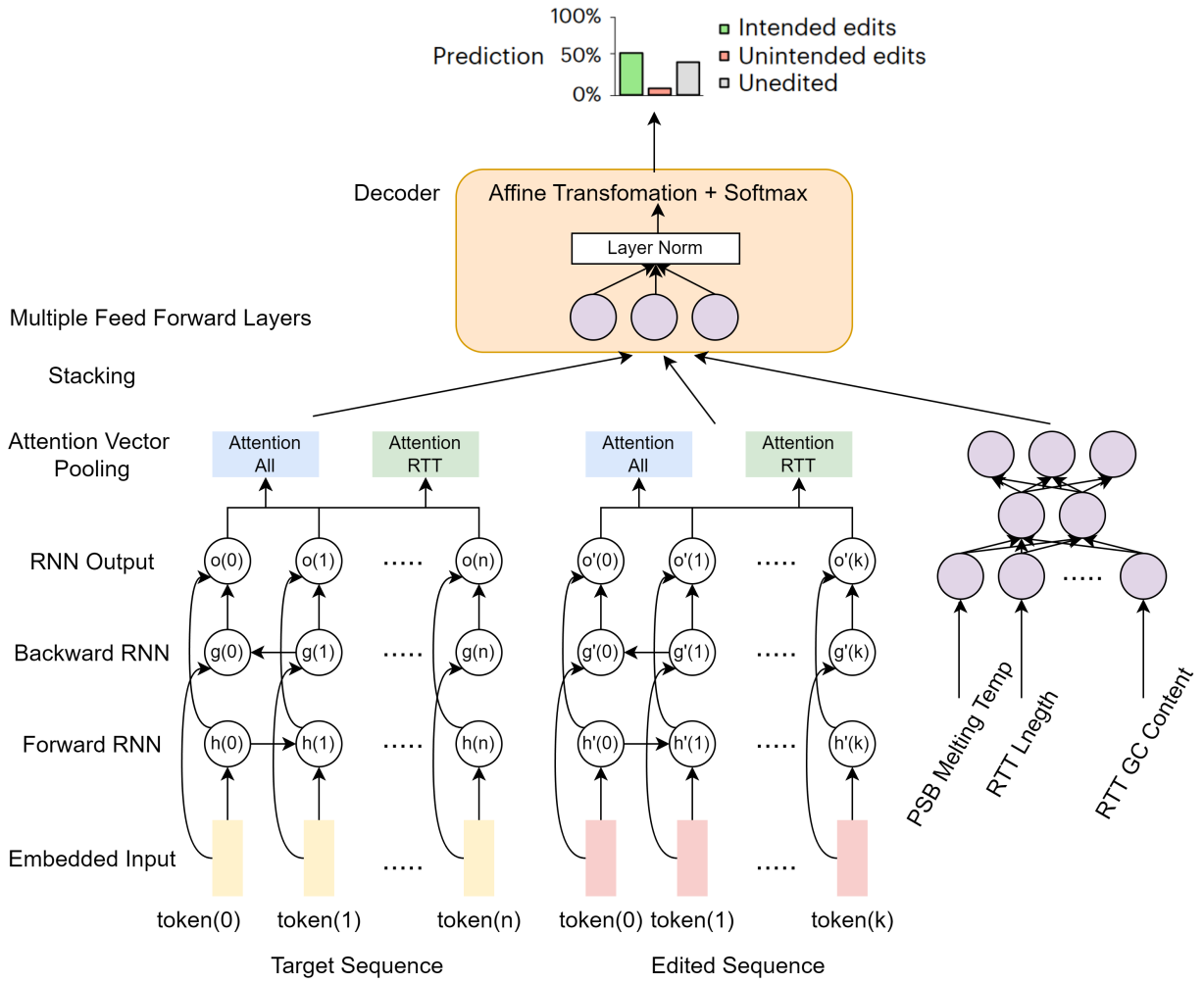


Figure 7: PRIDICT Model Architecture[7]

The model architecture is illustrated in Figure 7.

Significantly higher performance was achieved by PRIDICT when compared to DeepPE (including PE_Type and PE_Position) and EasyPrime, with 2-3 fold increase in Spearman's R and Pearson's r on the generalization datasets curated by Gerald et al. PRIDICT also achieved comparable or better results on the datasets DeepPE and EasyPrime were originally trained on. In terms of current generation of models, it was on par with DeepPrime on the HEK293T datasets when predicting intended edits, with a Spearman's R of 0.81. At the same time, PRIDICT outperformed the MinsePIE model as mentioned in section 2.2.4.

# 3 Discussion

A clear increasing trend in the performance of the models as well as the size and variety of the datasets can be observed over the years. DeepPrime supports an enormous amount of prime editors and cell types, while PRIDICT shows very high and consistent performance

across different datasets. Limits on editing types are also being pushed, from the simple G to C substitution in DeepPE to arbitrary edits supported by PRIDICT and EasyPrime.

However, many improvements can still be made to further advance the field.

## 3.1 Future Directions

### 3.1.1 Incoorperation of Advanced Tokenizer

First of the possible directions is the possibility of using a newly emerged tokenizer instead of one-hot encoding during data embedding. Chen et al proposed a 'fingerprinting' tokenization method that can capture the essential of RNA-DNA hybrids in Cas9 system. It had been shown to improve the performance of machine learning systems including XGBoost classifiers in the task of predicting the efficiency of CRISPR-Cas9 HDR[18].

The method can potentially be applied to the aforementioned prime editing models, as the RNA-DNA hybrid is also a crucial part of the prime editing process.

### 3.1.2 Application of Other Advanced Model Architecture

State of the art model architectures including the transformer can also be applied to the task. Transformer has shown to be very effective in capturing the dependencies between the base pairs in the sequence, and has been applied to the task of predicting base editing efficiency with great success, outperforming RNN and CNN[13]. As a result, the model can potentially be applied to the task of predicting prime editing efficiency as well.

### 3.1.3 Using Ensemble Learning to Improve Overall Accuracy

Moreover, more than one models can be utilized at the same time during pegRNA design. As discussed in section 2.2, each model has its own features and design choices, resulting in individual limitations and advantages. Ensemble learning is a technique used to mitigate errors or biases that may exist in individual models by leveraging the collective intelligence of the ensemble, and possibly result in a more accurate and robust model.

### 3.1.4 Development of Benchmarking Datasets

At the same time, it was very difficult to directly gauge the performance of the models due to the lack of a consistent benchmarking dataset during the survey. To validate whether their methods produced significant improvements, the authors had to train and test all previous representative models on their own datasets. This is feasible at the current stage of the field, but as the number of models and datasets grow, it will become increasingly difficult to compare the performance of the models.

A benchmarking dataset can be curated for each PE and cell line combination, and the studies can include the performance of the models on the relevant datasets in their publications. This will allow for a more direct comparison of the models and provide a clearer picture of the current state of the field.

# References

[1] Fuguo Jiang and Jennifer A. Doudna. "CRISPR–Cas9 Structures and Mechanisms". In: *Annual Review of Biophysics* 46 (Volume 46, 2017 May 22, 2017), pp. 505–529. ISSN: 1936-122X, 1936-1238. DOI: 10.1146/annurev-biophys-062215-010822. URL: https://www.annualreviews.org/content/journals/10.1146/annurev-biophys-062215-010822 (visited on 04/21/2024).

[2] David R. Liu and Peter J. Chen. "Prime Editing for Precise and Highly Versatile Genome Manipulation". In: *Nature Reviews Genetics* 24.3 (Mar. 2023), pp. 161–177. ISSN: 1471-0056, 1471-0064. DOI: 10.1038/s41576-022-00541-1. URL: https://www.nature.com/articles/s41576-022-00541-1 (visited on 02/16/2024).

[3] Ariel Kantor, Michelle McClements, and Robert MacLaren. "CRISPR-Cas9 DNA Base-Editing and Prime-Editing". In: *International Journal of Molecular Sciences* 21.17 (Aug. 28, 2020), p. 6240. ISSN: 1422-0067. DOI: 10.3390/ijms21176240. URL: https://www.mdpi.com/1422-0067/21/17/6240 (visited on 02/08/2024).

[4] Holly A. Rees and David R. Liu. "Base Editing: Precision Chemistry on the Genome and Transcriptome of Living Cells". In: *Nature reviews. Genetics* 19.12 (Dec. 2018), pp. 770–788. ISSN: 1471-0056. DOI: 10.1038/s41576-018-0059-1. pmid: 30323312. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6535181/ (visited on 04/12/2024).

[5] Liu David R. et al. "Search-and-Replace Genome Editing without Double-Strand Breaks or Donor DNA". In: *Nature* 576.7785 (Dec. 5, 2019), pp. 149–157. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-019-1711-4. URL: https://www.nature.com/articles/s41586-019-1711-4 (visited on 02/08/2024).

[6] Peter J. Chen et al. "Enhanced Prime Editing Systems by Manipulating Cellular Determinants of Editing Outcomes". In: *Cell* 184.22 (Oct. 2021), 5635–5652.e29. ISSN: 00928674. DOI: 10.1016/j.cell.2021.09.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0092867421010655 (visited on 03/03/2024).

[7] Nicolas Mathis et al. "Predicting Prime Editing Efficiency and Product Purity by Deep Learning". In: *Nature Biotechnology* 41.8 (Aug. 2023), pp. 1151–1159. ISSN: 1087-0156, 1546-1696. DOI: 10.1038/s41587-022-01613-7. URL: https://www.nature.com/articles/s41587-022-01613-7 (visited on 04/24/2024).

[8] Gue-Ho Hwang et al. "PE-Designer and PE-Analyzer: Web-Based Design and Analysis Tools for CRISPR Prime Editing". In: *Nucleic Acids Research* 49.W1 (July 2, 2021), W499–W504. ISSN: 0305-1048. DOI: 10.1093/nar/gkab319. URL: https://doi.org/10.1093/nar/gkab319 (visited on 04/16/2024).

[9] Yichao Li et al. "Easy-Prime: A Machine Learning–Based Prime Editor Design Tool". In: *Genome Biology* 22.1 (Dec. 2021), p. 235. ISSN: 1474-760X. DOI: 10.1186/s13059-021-02458-0. URL: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-021-02458-0 (visited on 02/08/2024).

[10] Goosang Yu et al. "Prediction of Efficiencies for Diverse Prime Editing Systems in Multiple Cell Types". In: *Cell* 186.10 (May 2023), 2256–2272.e23. ISSN: 00928674. DOI: 10.1016/j.cell.2023.03.034. URL: https://linkinghub.elsevier.com/retrieve/pii/S0092867423003318 (visited on 03/02/2024).

[11] Hui Kwon Kim et al. "Predicting the Efficiency of Prime Editing Guide RNAs in Human Cells". In: *Nature Biotechnology* 39.2 (Feb. 2021), pp. 198–206. ISSN: 1546-1696. DOI: 10.1038/s41587-020-0677-y. URL: https://www.nature.com/articles/s41587-020-0677-y (visited on 04/16/2024).

[12] Jonas Koeppel et al. "Prediction of Prime Editing Insertion Efficiencies Using Sequence Features and DNA Repair Determinants". In: *Nature Biotechnology* 41.10 (Oct. 2023), pp. 1446–1456. ISSN: 1087-0156, 1546-1696. DOI: 10.1038/s41587-023-01678-y. URL: https://www.nature.com/articles/s41587-023-01678-y (visited on 02/07/2024).

[13] Kim F. Marquart et al. "Predicting Base Editing Outcomes with an Attention-Based Deep Learning Algorithm Trained on High-Throughput Target Library Screens". In: *Nature Communications* 12.1 (Aug. 25, 2021), p. 5114. ISSN: 2041-1723. DOI: 10.1038/s41467-021-25375-z. URL: https://www.nature.com/articles/s41467-021-25375-z (visited on 01/21/2024).

[14] John G. Doench et al. "Optimized sgRNA Design to Maximize Activity and Minimize Off-Target Effects of CRISPR-Cas9". In: *Nature Biotechnology* 34.2 (Feb. 2016), pp. 184–191. ISSN: 1546-1696. DOI: 10.1038/nbt.3437. URL: https://www.nature.com/articles/nbt.3437 (visited on 04/29/2024).

[15] Sandy L. Klemm, Zohar Shipony, and William J. Greenleaf. "Chromatin Accessibility and the Regulatory Epigenome". In: *Nature Reviews Genetics* 20.4 (Apr. 2019), pp. 207–220. ISSN: 1471-0064. DOI: 10.1038/s41576-018-0089-8. URL: https://www.nature.com/articles/s41576-018-0089-8 (visited on 04/22/2024).

[16] Philip Thomas and Trevor G. Smart. "HEK293 Cell Line: A Vehicle for the Expression of Recombinant Proteins". In: *Journal of Pharmacological and Toxicological Methods*. Electrophysiological Methods in Neuropharmacology 51.3 (May 1, 2005), pp. 187–200. ISSN: 1056-8719. DOI: 10.1016/j.vascn.2004.08.014. URL: https://www.sciencedirect.com/science/article/pii/S1056871905000110 (visited on 04/22/2024).

[17] Hui Kwon Kim et al. "SpCas9 Activity Prediction by DeepSpCas9, a Deep Learning–Based Model with High Generalization Performance". In: *Science Advances* 5.11 (Nov. 6, 2019), eaax9249. DOI: 10.1126/sciadv.aax9249. URL: https://www.science.org/doi/10.1126/sciadv.aax9249 (visited on 04/16/2024).

[18] Qinchang Chen et al. "Genome-Wide CRISPR off-Target Prediction and Optimization Using RNA-DNA Interaction Fingerprints". In: *Nature Communications* 14.1 (Nov. 18, 2023), p. 7521. ISSN: 2041-1723. DOI: 10.1038/s41467-023-42695-4. URL: https://www.nature.com/articles/s41467-023-42695-4 (visited on 03/08/2024).