# Predicting CRISPR Gene-Editing Efficiency with Deep Learning

**Max O'Meara**
mxomeara@bu.edu

**Rehan Samaratunga**
rdsam@bu.edu

## 1 Task

We want to build a CNN that can tell ahead of time whether a tiny "address tag" (RNA string) for genes will work well. In the lab, these tags are short strings that help turn off a chosen part of DNA. Scientists have already tried many tags in cells and measured how strong their effect was. Therefore the goal for the model is to learn these relationships automatically from experimental examples, so that it can make reliable predictions for new RNA sequences it has never seen before. This choice matters because it pushes the model to generalize to genes it hasn't seen before and it must rely on sequence based rules that transfer to other genes rather than relying on properties of specific genes.

In practice, the CNN will learn simple position-specific patterns near the PAM and along the guide (e.g., seed region signals and GC balance) that track with effect strength, producing a calibrated score that ranks candidates across different cell lines and readouts without using gene-level features.

## 2 Input and Output

User input to our CNN model will be an example CRISPR-Cas9 screen without gene level features. This forces the model to learn universal guide RNA design principles for new genes and RNA sequences it hasn't seen before. The user will input a 23 letter sequence consisting of A/C/G/T, the cell line model, the phenotype, chromosome and the strand.

Concretely, inputs are (seq23, celline, phenotype, chr, strand). seq23 is 23 letters as it includes the guide RNA of 20 characters with the 3 PAM characters which act as a binding and recognition signal for the Cas. celline encodes the experimental context and lets the model adjust baselines across cell types while still relying on sequence rules. phenotype tells the model what outcome is being measured. chr adds minimal genomic context since we exclude coordinates but allows it to contribute to the output without leaking gene specific information. strand indicates the target orientation and if strand is "$-$", we reverse-complement so the CNN always sees the PAM at positions 21–23.

An example input can look like: $GCAGCATCCCAACCAGGTGGAGG$, Jiyoye, viability, $10$, $+$. The model will take the input and will output a single number showing the expected strength of the effect using the score: $0$ = weak to $1$ = strong. An example output can look like: $0.31590732393855947$

We one-hot encode the sequence and include small learned vectors for celline, phenotype, chr, and strand; no gene-level inputs are used. Basic checks enforce length =23 and A/C/G/T only. The output is a single value in [0,1] (higher = stronger expected effect), obtained with a sigmoid and lightly calibrated so scores are comparable within the same experiment type.

## 3 Purpose

The purpose of this project is an attempt to save resources in labs. CRISPR-Cas9 a gene-editing tool that can treat genetic diseases, fight cancer, and combat infectious diseases. When biotech companies are designing new therapies, they're targeting genes they've never screened before. Instead of testing dozens of DNA tags to find one that works, researchers can start with the most promising options our model suggests. That means fewer trial and error experiments, faster progress on basic biology studies, and a smoother path for teams using this tool for disease research, drug studies, or classroom labs.

This reduces library size, sequencing, and hands-on work, freeing time for confirmatory tests and follow-up biology. It also gives a clear, repeatable ranking method that helps small labs and teaching settings. Because the setup is configurable by cell line and phenotype, teams can run quick design–test–update cycles to steadily improve guide selection.

## 4 Data Used

The data we will be using for our model is provided by GenomeCRISPR which is a database for high-throughput CRISPR/Cas9 screening experiments. The current data set contains data on the performance of approximately 700 000 single guide RNAs (sgRNAs) used in  500 different experiments performed in 421 different human cell lines. The data set is publicly downloadable and can be found at this link: `https://genomecrispr.dkfz.de/#!/`.

The data contains data on the sgRNA start and end position, the target chromosome, the target strand, the PubMed ID of the screen's original publication, the cell line model used in the screen, the phenotype that was measured in the screen, the sgRNA sequence including PAM, the HUGO gene symbol of the targeted gene, the ENSEMBL gene identifier of the targeted gene, an array of read counts at the 'initial' time point and counts of the treated sample, a value of the sgRNA effect determined by GenomeCRISPR, the Cas variant used in the screen and the type of screen used.

For our model we will not need all of this data since we are training our model to learn generalizable sequence based rules of CRISPR guide efficiency rather than memorizing gene specific effects. This allows the model to learn patterns that can be applied to other genes. To do so we will restrict the data we give our model to just (seq23, cellline, phenotype, chr, strand) which only adds minimal genomic context in the form of including the chr data point. However we are choosing to including the chr data point as it does not teach the model gene identity and excludes genomic coordinates, so it doesn't learn from where a guide is located, only the chromosome level context.

We will divide the GenomeCRISPR data set into 80 percent training, 10 percent validation and 10 percent testing. The data will be split up in a way in which there will be no data leakage as no gene in the training set will appear in the testing set. This reinforces learning of universal genetic patterns as the main idea is to make sure the model is tested on completely new genes that it hasn't seen before, so we can measure how well it generalizes.

## 5 Performance Evaluation

For model performance, we will use the following metrics to help us evaluate our model: (add equations)

- Spearman Rank Correlation - This will be our primary metric. It measures how well the model's predictions preserve the relative ranking of the true effect scores. This will help the user answer the question of whether or not the model correctly identifies the best-performing guides and the worst-performing (also ranking the best higher than the worst).

$$\rho_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \tag{1}$$

- Pearson Correlation - This will help us measure the linear relationship between the predicted scores and the true effect scores. A high Pearson correlation indicates that the model's output is directly proportional to the true experimental value.

$$r = \frac{n\sum xy - (\sum x)(\sum y)}{\sqrt{(n\sum x^2 - (\sum x)^2)\,(n\sum y^2 - (\sum y)^2)}} \tag{2}$$

  - $n$ = number of data points
  - $x$ and $y$ = variables being compared
  - $\sum$ = summation symbol

- Mean Squared Error (MSE): This will measure the average of the squares of the prediction errors and will give us an outline of the magnitude of a specific error. Also this will penalize larger outlier errors.

$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

We will use all of these metrics but will place a big emphasize on Spearman correlation.

## 6 Baseline Method

For our baseline model, we used a simple 1D convolutional neural network that takes inputs of shape $(\text{batch}, 4, 23)$, where the 4 channels are one-hot encodings of A/C/G/T. A single Conv1d$(4 \to 16,\ \text{kernel} = 5,\ \text{padding} = 0)$ scans for 5 metrics, producing $(\text{batch}, 16, 19)$. A ReLU applies nonlinearity, and an adaptive max-pool over the result reduces each channel to its strongest positional response, giving $(\text{batch}, 16, 1)$. We then squeeze to $(\text{batch}, 16)$ and pass through a linear layer $(16 \to 1)$ to produce a scalar prediction per sample. This architecture (353 parameters total) induces position invariance for metric detection while remaining small and quick to train.

## 7 Baseline Performance

We trained for 3 epochs with Adam (lr $= 10^{-3}$) and MSE loss (batch size $= 256$), the baseline CNN achieves on the validation split: MSE $= 0.6495$, Pearson $r = 0.1399$, Spearman $\rho = 0.1433$, and sign-accuracy $= 0.5220$. On the test split: MSE $= 0.6518$, Pearson $r = 0.1398$, Spearman $\rho = 0.1432$, and sign-accuracy $= 0.5223$. Given the close alignment between validation and test metrics, the results indicate stable generalization.