Proceedings of CONF-SEML 2026 Symposium: Learning and Decision Making in Multi Agent Software Systems
DOI: 10.54254/2755-2721/2026.BA31676

CNN–GRU Hybrid Network for Predicting sgRNA On-Target Activity in Bacteria

Ziyang Ni

Department of Biology, Pennsylvania State University, Pennsylvania, USA
zfn5034@psu.edu

Abstract. The precise prediction of CRISPR–Cas9 single-guide RNA (sgRNA) on-target activity is important for designing an efficient gene editing experiment. Even though there is a large amount of deep learning models that have been raised and proved. When these models are transferred to other species like bacteria, the performance will decrease significantly. There is fundamental difference between genome background and DNA repair/selection. Based on multiple Cas9 variants, we propose CNNGRUHybrid, a hybrid neural architecture that (i) extracts local motif features using multi-scale 1D convolutions, (ii) models longer-range dependencies using stacked bidirectional GRUs (BiGRUs), and (iii) aggregates position-wise representations via a lightweight attention-style pooling mechanism. This model uses one-hot encoded 43-nt or 28-nt windows as input, and incorporates numeric or tabular features optionally like GC content and assay-derived descriptors. We further define and compare our pretraining–finetuning regimes across datasets/variants. The results of our experiment comparing CNNGRUHybrid with representative CNN based models show consistent improvements in rank-based metrics (e.g., Spearman correlation) on bacterial sgRNA activity prediction tasks.

Keywords: CRISPR–Cas9, sgRNA, on-target activity, deep learning, transfer learning

1. Introduction

The CRISPR–Cas9 system is a dominant programmable genome-editing platform. An sgRNA guides Cas9 to a certain locus in genome by Watson–Crick pairing, typically adjacent to a protospacer adjacent motif (PAM), inducing a double-strand break and downstream repair outcomes that determine editing efficiency. Then, selecting high-activity sgRNA is a core computing problem in experiment design.

Early sgRNA design depends on manually built features (position-specific nucleotides, GC content, thermodynamic heuristics) and classical models. As the development of deep learning, end-to-end sequence models—especially CNN-based architectures—have become prominent in sgRNA activity predictors. While it is still a challenge to predict sgRNA activity in bacteria, models tuned for eukaryotic contexts do not transfer reliably, motivating bacterial-specific predictors and evaluation methods.

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.BA31676

2. Related works

2.1. Eukaryotic sgRNA activity prediction

A large amount of research focus on eukaryotic on-target efficacy prediction. Doench et al. introduced large-scale screening-driven models and optimized design rules in human/mouse contexts [1]. Moreno-Mateos et al. proposed CRISPRscan, explaining sequence determinants and practical scoring rules for in vivo activity prediction [2]. CNN-based end-to-end methods became mainstream methods, including DeepCRISPR [3] and DeepCas9 [4]. DeepSpCas9 is a related deep learning model trained on large paired sgRNA–target datasets and showed strong generalization [5]. More recent models incorporate additional context beyond sequence; for instance, CRISPRon integrates sequence with chromatin-related features and reports improved efficiency prediction in eukaryotes [6].

2.2. Bacterial sgRNA activity prediction

Comparatively, there is relatively little amount of research on sgRNA on-target prediction in bacteria. To this question, Wang and Zhang made large-scale E. coli sgRNA activity datasets across multiple Cas9 conditions and designed a CNN-based predictor, showing that bacterial-specific training improves performance relative to naïve transfer [7]. In bacteria research, CNN based architecture modeling methods on sgRNA activity is still a competitive baseline model.

2.3. Motivation for hybrid CNN–RNN + attention

CNNs perform well at capturing local sequence motifs (k-mer-like patterns), but pure CNN stacks may not explicitly model longer-range dependencies along the protospacer and its flanking context. Hybrid CNN–RNN architectures with attention mechanisms have been widely explored in biological sequence modeling tasks [8]. Hybridizing CNNs with recurrent layers (e.g., GRU/LSTM) provides a clear mechanism for sequential dependencies, at the same time, attention pooling provides a learnable method to highlight sequential positions which have the most information for activity prediction.

3. Datasets and preprocessing

3.1. Data sources and enzyme conditions

The study uses E. coli CRISPR–Cas9 activity datasets measured under three enzyme/host conditions: Cas9 (wild-type SpCas9), eSpCas9, and Cas9 in a recA-deletion background. Two dataset versions are provided: Set 1 (full) and Set 2 (redundancy-reduced). Sample sizes in Set 1 are approximately 44k (Cas9), 45k (eSpCas9), and 48k (recA-deletion condition).

3.2. File organization and cross-validation protocol

Data are organized into a fold-based cross-validation directory structure, for example:data/Cross_validation/Set1/<ENZ>/<cv>/traindata.txt and .../testdata.txt, with <cv> $\in$ {1,…,5}.This structure supports consistent model selection and evaluation across folds and across dataset variants.

DOI: 10.54254/2755-2721/2026.BA31676

3.3. Parsing, label normalization, and feature types

Each train/test file is tab-separated. The dataset loader treats column 0 as an sgRNA identifier and automatically detects the label column (activity score) if not specified, preferring columns that parse as floats for the largest fraction of rows. Sequence columns are detected by nucleotide-string heuristics (ACGT/U patterns). Labels are min–max normalized within each training split to stabilize optimization.

4. Model architecture

This section detailly explains the CNN–GRU hybrid architecture (CNNGRUHybrid) used to predict sgRNA on-target activity based on a fixed-length nucleotide window around each candidate target site. As showed in Figure 1, the model maps a one-hot encoded sequence tensor to a scalar activity score via: (i) a multi-scale 1D convolutional encoder (kernel sizes 3/5/7) to extract local motif features; (ii) a 1×1 projection to a shared model dimension; (iii) stacked bidirectional GRU (BiGRU) layers to capture long-range dependencies along the protospacer and flanking context; (iv) attention-style pooling to form a sequence-level embedding; and (v) a regression head. Additionally, normalized numeric/tabular features are embedded by an MLP and they can be fused with the sequence embedding before the final regressor.

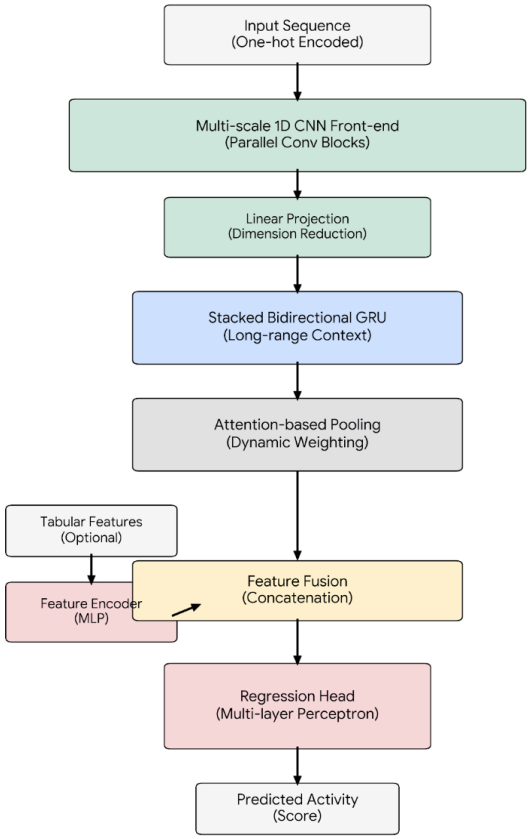


Figure 1. Overview of the proposed CNNGRUHybrid architecture for sgRNA on-target activity prediction. The model combines multi-scale 1D convolution, bidirectional GRU layers, attention-based pooling, and an optional tabular-feature fusion module to output a scalar activity score

DOI: 10.54254/2755-2721/2026.BA31676

4.1. Model overview

Our CNNGRUHybrid model is designed to predict bacterial sgRNA on-target activity using an end-to-end method. The model uses a fixed-length nucleotide window as input and uses target sites as a center. Sequence information is encoded using one-hot representation and then processed by a multi-scale convolutional neural network (CNN) to extract local motif features. These features are then passed to a stacked bidirectional gated recurrent unit (BiGRU) network. It is to capture long-range sequential dependencies. An attention-style pooling mechanism aggregates position-wise representations into a sequence-level embedding. When it is available, numeric or tabular features are encoded by a multilayer perceptron (MLP) and fused with the sequence embedding before final regression.

The one-hot encoding of the input sequence is defined as follows:

$$X_{t,d} = \{ 1,\ if\ nucleotide\ at\ position\ t\ is\ d;\ 0,\ otherwise \} \tag{1}$$

where X denotes the encoded input tensor, t indexes the sequence position, and d represents the nucleotide channel (A, C, G, or T).

4.2. Multi-scale CNN feature extraction

In order to capture local sequence motifs at different spatial resolutions, three parallel one-dimensional convolutional branches are used with kernel sizes 3, 5, and 7, respectively. Each convolution branch is followed by batch normalization and a ReLU activation. The resulting feature maps from all branches are concatenated along the channel dimension. It forms a unified representation.

The convolution operation at a given position is defined as:

$$z_{t,c}^{k} = b_c^k + \Sigma_{i=0}^{k-1} \Sigma_d W_{i,d,c}^{k} X_{t+i-p,d} \tag{2}$$

where $z_{t,c}^{(k)}$ is the pre-activation output for kernel size k at position t and channel c, $W_{i,d,c}^{(k)}$ denotes convolution weights, $b_c^{(k)}$ is a bias term, p is the padding size, and X is the one-hot encoded input.

4.3. Channel projection

After concatenation of the multi-scale CNN outputs, a point-wise linear projection is applied to map the concatenated features into a fixed model dimension. This projection ensures dimensional consistency for subsequent recurrent processing.

The projection is defined as:

$$\widetilde{H}_t = H_t W_p + b_p \tag{3}$$

DOI: 10.54254/2755-2721/2026.BA31676

where H_t is the concatenated CNN feature vector at position t, W_p and b_p are trainable projection parameters, and $\widetilde{H}_t$ is the projected feature representation.

4.4. Stacked bidirectional GRU

The projected sequence features are fed into a stacked bidirectional GRU to model long-range dependencies along the sequence. As followed, the GRU update mechanism consists of reset, update, and candidate hidden state computations.

The GRU cell is defined by the following equations:

$$r_t = \sigma\left(W_r x_t + U_r h_{t-1} + b_r\right) \tag{4}$$

$$u_t = \sigma\left(W_u x_t + U_u h_{t-1} + b_u\right) \tag{5}$$

$$\widetilde{h}_t = \tanh\left(W_h x_t + U_h\left(r_t \odot h_{t-1}\right) + b_h\right) \tag{6}$$

$$h_t = \left(1 - u_t\right) \odot h_{t-1} + u_t \odot \widetilde{h}_t \tag{7}$$

where x_t is the input at position t, h_t is the hidden state, r_t and u_t are the reset and update gates, $\sigma(\cdot)$ denotes the sigmoid function, and $\odot$ denotes element-wise multiplication.

4.5. Attention-style pooling

To aggregate position-wise sequence representations into a single embedding, an attention-style pooling mechanism is added. This allows the model to assign different importance weights to different sequence positions.

The attention weights and pooled embedding are computed as:

$$\alpha_t = \frac{\exp\left(\frac{q^T k_t}{\sqrt{D}}\right)}{\Sigma_{j=1}^{L} \exp\left(\frac{q^T k_j}{\sqrt{D}}\right)} \tag{8}$$

$$c = \Sigma_{t=1}^{L} \alpha_t S_t \tag{9}$$

where α_t is the attention weight at position t, q is a learnable query vector, k_t is a key vector derived from the BiGRU output S_t, D is the model dimension, and c is the pooled sequence embedding.

DOI: 10.54254/2755-2721/2026.BA31676

4.6. Feature fusion and regression head

When numeric features are available, they are encoded by an MLP and concatenated with the sequence embedding. The fused representation is then passed to a regression head composed of fully connected layers to produce the final prediction.

The fusion and prediction process is defined as:

$$\hat{y} = f_{\theta([c;\, u])} \tag{10}$$

where c is the sequence embedding, u is the encoded numeric feature vector, $[cu]$ denotes concatenation, $f_\theta(\cdot)$ is the regression function, and $\hat{y}$ is the predicted sgRNA activity score.

5. Training objective and optimization

5.1. Combined loss function

The model is trained using a composite objective that combines regression accuracy, correlation alignment, and ranking consistency. This design reflects the importance of both absolute prediction accuracy and relative ranking in sgRNA selection.

The total loss is defined as:

$$L = \lambda_{reg}\mathscr{L}_{reg} + \lambda_{corr}\mathscr{L}_{corr} + \lambda_{rank}\mathscr{L}_{rank} \tag{11}$$

where $\mathscr{L}_{\text{reg}}$ is the regression loss, $\mathscr{L}_{\text{corr}}$ is the correlation-based loss, $\mathscr{L}_{\text{rank}}$ is the ranking loss, and λ_* are weighting coefficients.

5.2. Optimization details

Model parameters are optimized using the Adam or AdamW optimizer with mini-batch training. Learning rate scheduling and early stopping based on validation performance are applied to stabilize convergence and prevent overfitting.

5.3. Data augmentation

To improve robustness and generalization, a sequence reversal augmentation strategy is employed. The one-hot encoded sequence is reversed along the length dimension while preserving nucleotide channels.

The augmented sequence is defined as:

$$X'_{t,d} = X_{L+1-t,d} \tag{12}$$

where X is the original one-hot encoded sequence, X' is the augmented sequence, L is the sequence length, and t and d index positions and nucleotide channels, respectively.

DOI: 10.54254/2755-2721/2026.BA31676

6. Training strategy and experimental design

6.1. Training–validation protocol

All experiments use a five-fold cross-validation protocol in evaluation. It strictly follows original partition structure. For each fold, sgRNA samples are divided into mutually exclusive training and testing subsets, ensuring that no identical sequence appears in both sets. Model parameters learning process are all based on the training data of each fold. The performance is evaluated on test data.

To reduce the fluctuation of result introduced by random initialization, each fold is trained using identical hyperparameter settings. The final performance uses the average of 5 folds.

6.2. Pretraining and finetuning regimes

To investigate model generalization and transferability across bacterial conditions, four training regimes are defined:

• Regime 1 (Direct training):

The model is trained and evaluated on Dataset Cas9 using cross-validation.

• Regime 2 (Pretrain–finetune):

The model is first pretrained on Dataset eSpCas9 and subsequently finetuned on Dataset Cas9 before evaluation.

• Regime 3 (Dataset eSpCas9 only):

Training and evaluation are conducted exclusively on Dataset eSpCas9.

• Regime 4 (Merged pretraining):

Dataset eSpCas9 and Dataset knoRecA_Cas9 are merged for pretraining, followed by finetuning and evaluation on Dataset Cas9.

This design enables systematic assessment of whether shared sequence–activity patterns across Cas9 variants can be leveraged to improve prediction accuracy.

6.3. Optimization details

In the training process, the model parameters are updated by using the Adam-based optimizer with weight decay. The stability is improved by incorporating a learning rate scheduler. Then in order to reduce overfitting, dropout layers are added to recurrent and fully connected layers. Model training continues until it converges or it stopped before it reaches the best performance.

7. Results

7.1. Evaluation metrics

This research uses Spearman rank correlation as the primary evaluation. It is to measure the consistency between prediction result and the activity from the experiment. Because sgRNA design focuses more on relative performance ranking instead of accurate numeric predictions, this index is more meaningful in this task. While Pearson correlation and regression-based errors are used as complimentary reference, not as a primary evaluation.

DOI: 10.54254/2755-2721/2026.BA31676

7.2. Benchmark comparison across models and datasets

We compare our hybrid CNN–GRU model with different exist representative deep learning methods. These methods are used widely in sgRNA on-target activity prediction tasks. The models we used to compare include DeepCRISPR [3], a deep convolutional network originally developed for sgRNA activity modeling, CNN_Lin [9], a shallow convolutional model with linear-style regression layers, and DeepCas9 [4], a multi-branch convolutional architecture specifically designed for Cas9 activity prediction.

All models are evaluated under identical cross-validation protocols and dataset splits to ensure fair comparison.

Table 1. CNNGRUHybrid is consistently the best (or tied-best) across nearly all columns, supporting the claim that adding BiGRU + attention pooling improves bacterial on-target activity prediction beyond CNN-only baselines

	Cas9 Set1	Cas9 Set2	eSpCas9 Set1	eSpCas9 Set2	eSp_knoRecA+ Cas9 Set1	eSp_knoRecA+ Cas9 Set2	eSpCas9+ Cas9 Set1	eSpCas9+Cas9 Set2
DeepCRISPR [3]	0.51	0.51	0.66	0.66	0.57	0.57	0.57	0.57
CNN_Lin [9]	0.52	0.52	0.67	0.67	0.58	0.57	0.58	0.25
DeepCas9 [4]	0.56	0.55	0.70	0.69	0.62	0.61	0.62	0.61
CNNGRUHybrid	0.57	0.57	0.70	0.69	0.63	0.62	0.63	0.63

7.3. Overall performance trends

From Table 1, hybrid model preform the highest or near-highest Spearman correlation in most of experiment conditions. In particular, the hybrid architecture shows clear improvements over convolution-only baselines under both Set 1 and Set 2 conditions for Cas9 and eSpCas9 datasets.

Compared with DeepCRISPR [3] and CNN_Lin [9], the hybrid model is more stable across dataset variants, indicating improved robustness. Even DeepCas9 [4] performs competitively in several settings, the result of hybrid model consistently shows higher correlations, especially in combined or more challenging conditions.

7.4. Performance under different enzyme and dataset conditions

In the condition of Cas9, eSpCas9, and recA-deficient, our CNN-GRU hybrid model shows stable and relatively high prediction efficiency then others. It has more advantage when it is put in a more complicated setting in the relationship between sequence and activity. It explains that based on using convolution to extract local features, using recurrent sequence modeling better explain the dependency in sequence.

The result also shows that the sgRNA activity among different Cas9 variants is regulated by the same sequence factors. And these partially conserved models can be learned and used effectively by our model.

8. Discussion

8.1. Contribution of architectural components

In this architecture, the multi-scale convolutional layers are used to extract different dimensional local sequence features, while the bidirectional GRU layers model additionally models longer range

DOI: 10.54254/2755-2721/2026.BA31676

context dependency in sgRNA target regions. Attention-based pooling further improves the performance of model and let the model focus on more crucial sequences positions to activity prediction instead of averaging all features.

Overall, these modules balance with each other on functions and let the model have a better performance of local pattern extraction with global sequence context modeling.

8.2. Transferability across bacterial conditions

The improvement of pretraining shows that in different Cas9 variants and genetic backgrounds, bacteria sgRNA activity has common pattern in sequence level. At the same time, finetuning is still important to adjust the model to pair with the data distribution in real experimental conditions. This results shows that the efficiency of sgRNA is influenced by sequence factors across conditions and also by the regulation of unique factor in certain backgrounds.

8.3. Limitations

Even though the model has clear improvements in prediction performance, the input is still limited on sequence information and small amounts of complimentary numeric features. Broader genomic context, such as transcriptional state or chromosomal accessibility in bacteria, is not considered into modeling. While attention weights provide limited interpretability, they do not provide directly causal explanations for sgRNA activity.

9. Conclusion

This study presents a hybrid CNN–GRU neural architecture for predicting sgRNA on-target activity in bacterial systems. By integrating multi-scale convolutional feature extraction, recurrent sequence modeling, and attention-based pooling, our model consistently outperforms CNN-only baselines across multiple datasets and Cas9 variants.

Further pretraining and finetuning experiments demonstrate that transfer learning is an effective strategy for bacterial sgRNA activity prediction. Overall, it is important to model local sequence motifs and long-range dependencies together to improve prediction accuracy. And it provides a scalable framework for designing CRISPR guide in prokaryotic organisms.

Declarations

Source code is freely available at https://github.com/lanceskull/HybridBacteria. The source code repository includes software application, detailed user manual and all relevant data.

References

[1] J. G. Doench et al., "Optimized sgRNA design to maximize activity and minimize off-target effects of CRISPR-Cas9, " Nature Biotechnology, vol. 34, no. 2, pp. 184–191, 2016. DOI: 10.1038/nbt.3437

[2] M. A. Moreno-Mateos et al., "CRISPRscan: designing highly efficient sgRNAs for CRISPR-Cas9 targeting in vivo, " Nature Methods, vol. 12, no. 10, pp. 982–988, 2015. DOI: 10.1038/nmeth.3543

[3] G. Chuai et al., "DeepCRISPR: optimized CRISPR guide RNA design by deep learning, " Genome Biology, vol. 19, article 80, 2018. DOI: 10.1186/s13059-018-1459-4

[4] L. Xue et al., "Prediction of CRISPR sgRNA activity using a deep convolutional neural network (DeepCas9), " Journal of Chemical Information and Modeling, vol. 59, no. 1, pp. 615–624, 2019. DOI: 10.1021/acs.jcim.8b00365

[5] H. K. Kim et al., "SpCas9 activity prediction by DeepSpCas9, a deep learning–based model with high generalization performance, " Science Advances, vol. 5, no. 11, eaax9249, 2019. DOI: 10.1126/sciadv.aax9249

DOI: 10.54254/2755-2721/2026.BA31676

[6] X. Xiang et al., "CRISPRon: enhanced prediction of gRNA efficiency using sequence and chromatin features, " Nature Communications, vol. 12, article 4343, 2021. DOI: 10.1038/s41467-021-24676-5
[7] L. Wang and J. Zhang, "Prediction of sgRNA on-target activity in bacteria by deep learning, " BMC Bioinformatics, vol. 20, article 517, 2019. DOI: 10.1186/s12859-019-3151-4
[8] V. Konstantakos et al., "CRISPR–Cas9 gRNA efficiency prediction: an overview of predictive tools and features, " Frontiers in Genetics, vol. 13, article 798830, 2022. DOI: 10.3389/fgene.2022.798830
[9] J. Lin and K. C. Wong, "Off-target predictions in CRISPR-Cas9 gene editing using deep learning, " Bioinformatics, vol. 34, no. 17, pp. i656–i663, 2018. DOI: 10.1093/bioinformatics/bty554