HIGH SCHOOL EDITION
Journal of Student Research
Volume 11 Issue 3 (2022)

Examining Machine Learning Models That Predict sgRNA Cleavage Efficiencies

Krish Kawle[1], Dr. Rajagopal Appavu[2#] and Jothsna Kethar[2#]

[1]Union County Academy for Allied Health Sciences
[2]Gifted Gabber
[#]Advisor

ABSTRACT

Ever since its discovery, CRISPR-Cas9 has taken over the world in gene editing. By providing a single guide RNA to the cas9 enzyme, CRISPR-Cas9 can immediately pinpoint the target gene location in the genome and slice it. Scientists discovered a revolutionary way to use this method for gene editing. Yet, the challenge is that the CRISPR-Cas9 system is lenient with the matching precision of the guide RNA to the target sequence As a result, the CRISPR-cas9 system may also cleave certain healthy sequences that are almost identical to the target sequence. This paper aims to find the best model that uses machine learning to predict an optimal sgRNA design.

Introduction

When it comes to gene editing, Clustered Regularly Interspaced Palindromic Repeats (CRISPR) is often the topic that is widely discussed within the field. CRISPR was originally discovered as a component in bacterial immune systems. It was used as a way to fight bacteriophages (viruses that infect bacteria). When the virus injects its genetic material into the bacteria, the bacteria releases Cas9 nucleases. These nucleases have a sequence of RNA which is used to navigate to the viral sequence. The Cas9 latches onto the viral genes by the protospacer adjacent motif (PAM) sequence. Subsequent to latching on, it unzips the gene to guide RNA binding. If the sequences of the guide RNA and the viral sequence match, the Cas9 nuclease cuts the viral sequence. This action disables the gene, and hinders its ability to infect the bacteria. In 2012, Jennifer Doudna, Emmanuelle Charpentier, and their team figured out a way to apply this process for gene editing on human DNA. When the Cas9 enzyme cuts the DNA at the desired sequence, repair proteins arrive to try to repair the break. At this point, the scientist can control the way the gene repairs, and in the process edit the gene. For example, if scientists want to disable a gene, they can allow the repair proteins to attempt repairing the break-a process prone to mutations. These mutations could yield base-pair changes that are enough to disable a gene. If scientists want to turn a defective gene into a healthy one, they could introduce template DNA that guides repair proteins to fix the break in the sequence. These proteins would proceed to change, add, and delete base pairs that match the template DNA. This process can essentially turn the defective gene into a healthy one.

The problem that arises from this process is the tolerance of the Cas9 enzyme. Unfortunately, when the Cas9 enzyme is given a single-guide RNA (sgRNA) to locate the target sequence, there is a certain level of leniency. When the guide RNA binds to the target DNA, certain mismatches might arise. If the number of mismatches is small enough then the Cas9 system might still proceed to cut the target sequence. This

means healthy genes that are very similar to the target sequence, may also be cleaved by the Cas9 enzyme. These are called off-target cleavages.

To solve this problem, scientists are trying to predict how well a sgRNA sequence could guide the cas9 nuclease to its target, while minimizing the off-target effects. However, manually finding the best sgRNA sequence can be onerous. The most efficient way would be to use machine learning. This paper examines four machine learning models in depth. It will go over the algorithms, the results, and more to figure out the best machine learning model.

The Hierarchy of Machine Learning:

Before examining in depth the different models scientists have made, an overview of the different types of machine learning algorithms must be covered in order to understand them. Machine learning is a type of artificial intelligence (AI) built on the idea that a machine could learn and think as cognitively as humans. Machine learning focuses on training a computer to write its own algorithm rather than having the whole algorithm written by a human. **Figure 1** shows the hierarchy of the different types of machine learning algorithms.

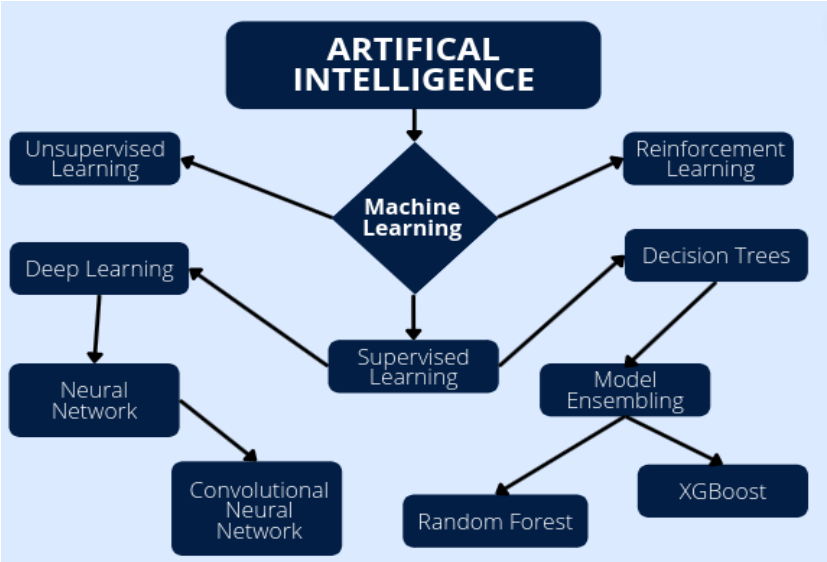

Figure 1. The different types of machine learning algorithms

Types of Machine Learning Techniques:

There are many different ways to train a machine. There is supervised learning, unsupervised learning, reinforcement learning, and much more. This section will look at supervised and unsupervised learning in depth. Supervised learning is a type of machine learning where the machine is given a set of labeled data. This is data where the input values and their respective output values are known. Based on this data, the computer is able to make predictions. For example, say the machine is trying to figure out different fruits based on viewing different images. The machine is given a picture of an apple, with the picture labeled as apple. Then the machine is fed an input of an orange with the label shown as orange. The machine remembers the image and its associated label. Once training done, the machine is given a picture of an orange, this

time with no label of the name of the orange. The machine is able to tell that the picture shown is a picture of an orange. In this case, the input, the pictures of the fruit, are labeled with the names of the fruit. The goal is to enable the machine distinguish between fruits by showing what each of them are. There are two different supervised learning techniques. They are regression and classification. Regression is where a machine learns to identify a relation between an independent variable and a dependent variable. That way it can use this information to predict the outputs of unknown inputs. Classification is where the machine learns to sort data based on certain patterns.

Unsupervised learning is a type of learning where a machine is given unlabeled data. This is primarily a method used to train models to look more for patterns, and hidden correlations rather than make predictions. A good example of this is a spinoff to the previous example above. Imagine the machine is fed with a bunch of images of apples and oranges in a random order. Each picture is not labeled whether it is an apple or an orange. That means the input data is not labeled data, so the computer does not know what are the fruits in the pictures. The machine is asked to sort these images, and amazingly, it sorts them correctly. In this case, the machine was not trying to make a prediction, but rather look for a pattern amongst the images in order to sort them correctly.

Decision Trees:

Decision trees are a type of classification, a supervised learning technique. This is where an outcome is decided based on the conditions of a series of factors that come into play with that decision. For example, a person has to decide how to get to work. If the car is full with fuel, then take the car, if not then take either bus or train. If the bus is not delayed, then take the bus, if it is delayed then take the train, etc. This example uses only one decision tree to decide an output. There are algorithms that take in the decisions of multiple decision trees, and in a majority-vote fashion, decide the final output. This is called model ensembling. Two primary examples of model ensembling are Random Forest and Extreme Gradient Boosting (XGBoost). **Figure 2** illustrates this process best.

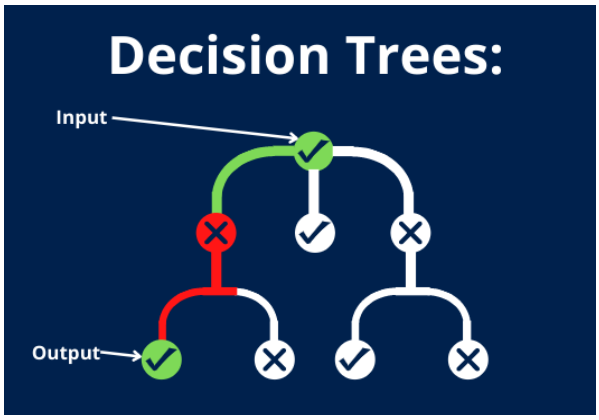

Figure 2. The structure of a decision tree

Random Forest:

Random Forests are a type of model ensembling with parallel decision trees. Parallel decision trees are decision trees that run independently of each other. Each tree runs its own path of decisions before reaching a mini-output of its own. First a set of input data is sent. Different randomized samples of the input

data are taken and sent as input data into a set of independent decision trees. Each decision tree runs its own path of decisions and decides an output. A majority vote is dictated based on the outputs and a final output is yielded. **Figure 3** best shows the different decision trees interacting independently below.

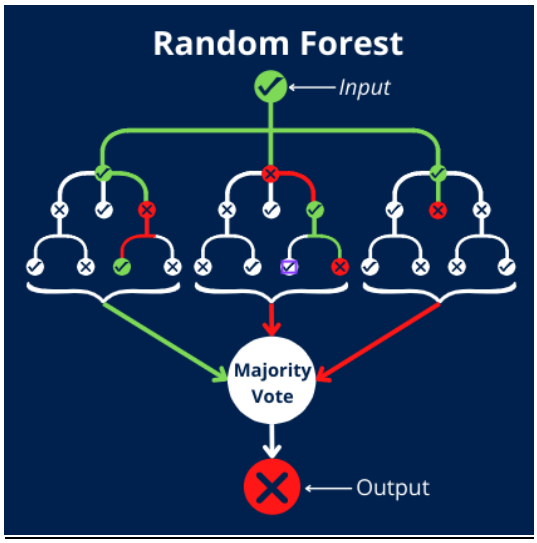

Figure 3. How Random Forest Works.

Extreme Gradient Boosting (XGBoost):

Extreme gradient boosting, (XGBoosting) is a type of model ensembling that uses multiple models. It is a type of Gradient boosting. Gradient Boosting is a framework of building models on top of each other. Each sub-model is supposed to address the shortcomings of the previous sub-model. After a model has been trained, the difference between its predictions and the correct outputs of the training dataset are evaluated. These differences are used to calculate gradients based on the degree a prediction deviates from the actual output. The higher the difference between the prediction and output, the higher the gradient. Gradients are computed by calculating an error rate from the differences in the model's predictions and outputs. Then that error rate is used to calculate the loss function. The loss function is a function that indicates how well a model models a dataset by assigning a value between zero and one. A good algorithm has this value close to zero. A partial derivative (first order partial derivative) of this loss function is calculated to get the gradients. This paper does not cover partial derivatives. However, consider it a computation performed on the loss function to get the gradients. High gradient data is then used as training data for the next model, and the cycle repeats until there are enough sub-models, adequately trained, to yield the expected results. When these models are used to weigh in on the final output, certain weights are applied to each model to determine its influence. The weights are added up to help determine the final result.

XGBoost follows this model, but there are a few differences. XGBoost finds the second order partial derivative of the loss function instead of a first order partial derivative. This paper does not cover second-order partial derivatives. However, it simply means finding the partial derivative of a partial derivative. Another difference is XGBoost uses advanced regularization. This is a method that reduces the complexity of a model to help the model better generalize all types of data. This is typically achieved by adding a term to the loss function.

Deep Learning:

Deep learning is a type of machine learning based on a multi-layered way of learning. The idea is each layer takes in input, and processes it into output. Then that output is sent to the next layer as input and does the same operation. This keeps happening until the output layer is reached, which then dictates the final output.

Neural Networks:

Neural networks are a type of deep learning inspired by the way the brain works. It follows the principles of deep learning where each layer processes input from the previous layer. It contains an input layer (the first layer in a neural network). It contains a network of middle layers (also called *hidden layers*). Lastly, it contains an output layer (the last layer of a neural network). Each layer is filled with nodes (like the neurons in our brain). Channels connect nodes across layers (like the dendrites that connect neurons). Each of these channels carries a certain numerical weight. When an input is sent, it activates some nodes of the input layer while some remain deactivated. These activated neurons send output through the channels. These channels then send this data as input for the hidden layers. This process repeats for all the hidden layers all the way until it activates certain neurons in the output layer. These activated neurons are processed as the final output. See **Figure 4** for an illustration of this process, and observe how different neurons are activated in each layer.

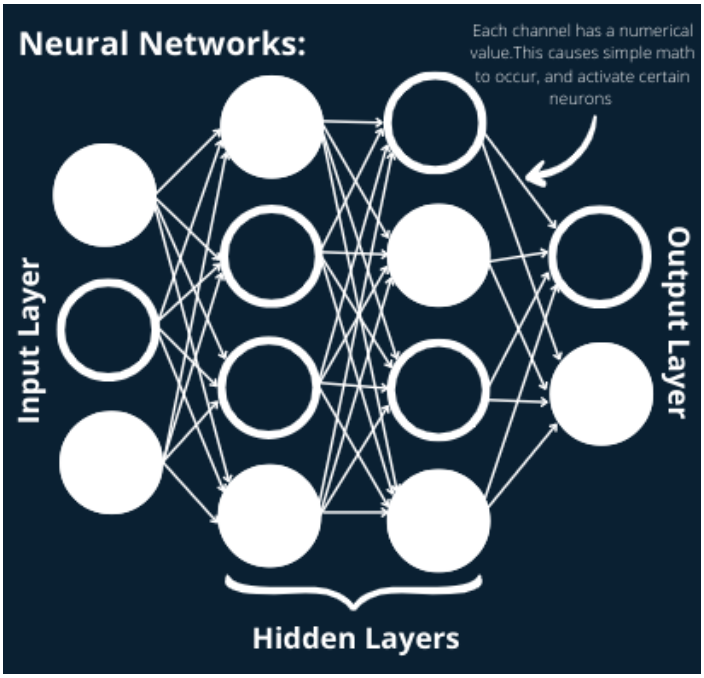

Figure 4. The underlying structure behind a neural network

Convolutional Neural Network (CNN):

Convolutional Neural Networks (CNN), are a type of neural network which uses filters. They are particularly helpful in pattern recognition, especially in identifying images by using these filters. When an input image is sent, it goes through a series of convolutional layers. These layers contain the filters. The way the filter works is that the filter itself would be a scanner for a certain pixelated area of the image. In

this case, the filter would scan a 3x3 area of pixels on the image. In the first layer, the filter would look for something simple within that 3x3 area, e.g. a straight, vertical line. The filter would start in the top left corner of the image, and check to see if it identifies a line, and stores information based on the results. Then the filter shifts by one pixel, and does the same thing (See **Figure 5**).. The filter keeps on scanning chunks of the image in this way until the whole image has been scanned. After that, the information that the filter is passed on to the next convolutional layer where the image is scanned again by a different filter where the filter might look for something more complex, e.g. a square, circle, or triangle, and does the same process as the previous filter. This cycle keeps occurring with filters of each following layer looking for more and more complex characteristics until it is flattened as input for the neural network part (See **Figure 6**). Flattening is a very simple concept. Since these CNNs typically work with 2d images, the most optimal way of storing data is in a matrix. The filters in a CNN truly work by performing simple math on this matrix based on its findings. Flattening takes the final matrix set and stacks them into one column since the input layer in a neural network can only take a column of inputs.

Some CNNs might have pooling feature maps coming after the convolutional layer. They are layers that act like the filters in the convolutional layer, but they perform different computations. These pooling layers will have a set size. However, unlike convolutional filters, these pooling filters are not limited to convolve or slide by only pixel. These filters can convolve by one pixel, two pixels, or as many needed in a model. This is called the stride. For example, a CNN contains a pooling filter of size 2x2 pixels and stride of 2, meaning the filter will slide by 2 pixels after each computation. The pooling filter convolves over a 2x2 section of the input image. At this point, it can do a few things. The filter can perform max pooling where the maximum pixel value in the section is stored. The filter can also perform average pooling where an average of the numbers assigned to each pixel in a section is computed. The filter may instead follow a different approach. There are a myriad of different ways to perform pooling, but the overall goal is to reduce the size of the input image. Processing a smaller image means less computational work needs to be performed, overfitting is reduced, the convolutional layers examine a smaller image more thoroughly, and in general it is a lot more beneficial for a CNN.

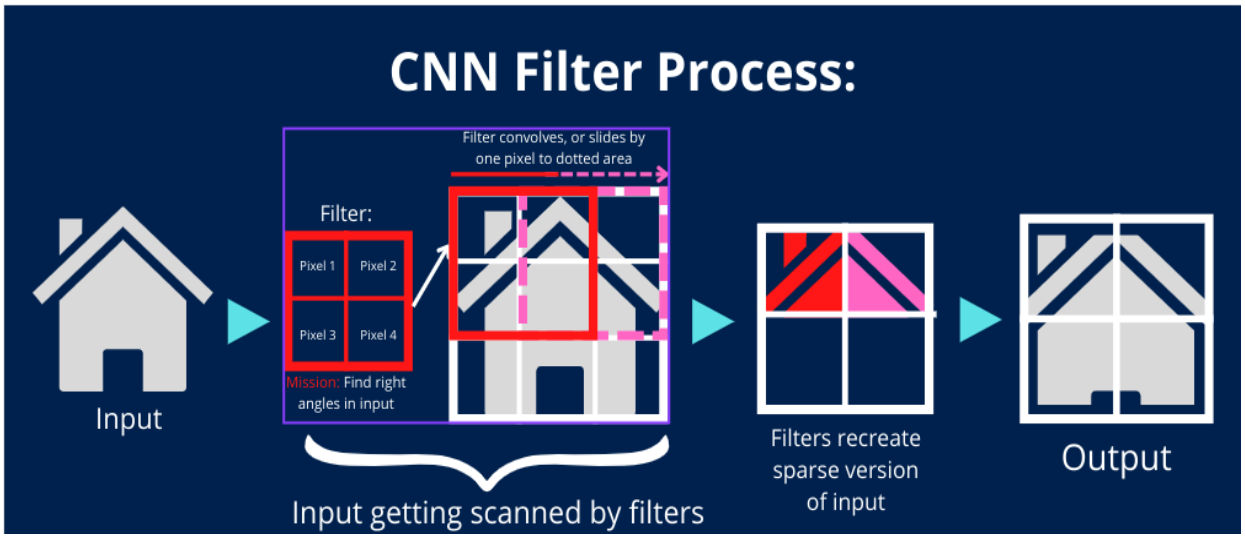

Figure 5. The filter is scanning for certain attributes in an image, recreating the image with those attributes focused.

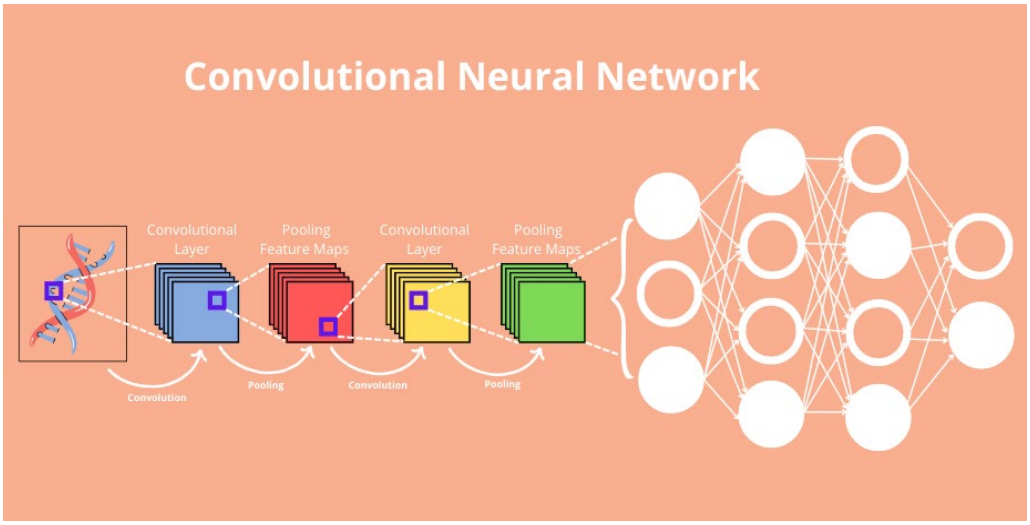

Figure 6. The overall step-by-step process with how a CNN works. The input goes through a filter, pooling occurs, before this data is sent as input to the neural network part.

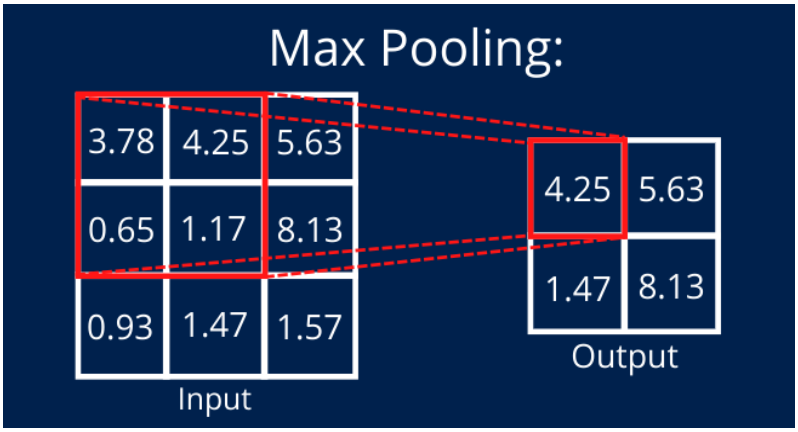

Figure 7. The maximum number inside the filter matrix is taken as part of the output matrix to reduce the size of the input

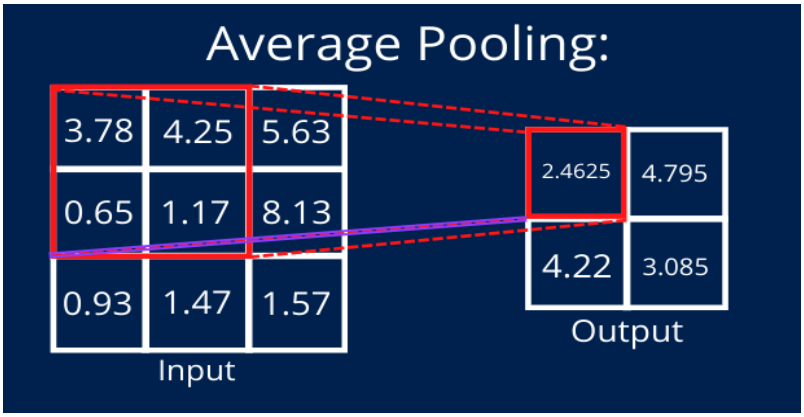

Figure 8. The average of the numbers inside the filter matrix is taken as part of the output to reduce the size of the input

One-hot Encoding:

While One-hot encoding is not a machine learning model, it is still a very important concept to understand. Computers do not understand human language. They only know how to process a series of ones and zeros. Human languages, and other features must all be represented as binary information. This information has to be encoded in a way for computers to understand. The same thing applies for machine learning models. A common method for encoding such data is through one-hot encoding. This type of encoding is tailored specifically for the machine learning model, and it creates a set of ones and zeros with the length dictated by the number of categories to classify. For example, a person is trying to build a machine learning model that identifies certain fruits: apples, bananas, strawberries, and oranges. These are the 4 categories the machine learning model has to sort the fruits into. The machine will take in pictures of fruits, along with their labels for training. To encode this data for processing, an encoder will take in 4 bits (a bit is a one or zero) since there are 4 categories, each bit represents if an image is of a certain fruit depending on the order. In this case, the first bit represents if the image is an apple, the second checks if the image is banana, the third for strawberry, and the fourth bit for oranges. A zero is assigned if the image is not of that fruit, otherwise a one is assigned. Depending on the image sent as input, the encoder will set out this string of ones and zeros as a label for that image.

The Different Models

The previous section covered a lot of information on the different concepts of machine learning. The next section will examine the different sgRNA machine learning models using the concepts covered in the previous section.

CRISTA

CRISTA is a python machine-learning model that uses Random Forest to predict the efficacy of sgRNA designs. When scientists were faced with the daunting task of accurately predicting sgRNA designs, they had to figure out a few things. Primarily they had to figure out what mismatches prompted the cas9 enzyme to avoid cleaving a certain genomic sites and what mismatches were looked away by the enzyme. They learned about a few rules that the cas9 enzyme followed. As mentioned in a paper written in 2017, “Such rules asserted that the number of mismatches should not exceed a specified bound, that mismatches at PAM-proximal positions are more influential than those occurring at PAM-distal positions, that spatially-dispersed mismatches are better tolerated, and that cleavage would not occur at sites that follow PAM sequences other than the canonical NGG (and occasionally NAG)” (Abadi et al., 2017). For clarification, the PAM sequence as mentioned previously is the sequence that helps the cas9 nuclease lock onto a certain genomic site. Typically the sequence is NGG (where N represents any nucleotide). This is called the canonical NGG PAM sequence. Any other PAM sequence that is different from canonical NGG (such as NAG) is called non-canonical NGG PAM sequence. Not only did scientists have to consider the base-pair sequence of the PAM, they also had to account for DNA or RNA bulges. These are cases where one strand contains an extra nucleotide that is not paired to another nucleotide. In DNA, this can cause a kink and make one of the strands “bulge” as a result. Previous studies neglected these cases and built algorithms that focused mainly on the mismatches of the sgRNA sequence and the target sequence, causing the accuracy to suffer. These scientists took this information and built their CRISTA machine learning model to predict for these attributes, and much more and implemented them in the form of decision trees in Random Forest

fashion. There are three things that the CRISTA algorithm does. It can take in an sgRNA sequence, and a couple of genome target sequences as input. Then it can compute the resulting score for each target sequence as to how well it matches to the sgRNA. It can also take an sgRNA sequence, assign a score, and determine certain target sequences it could match and rank them. Finally, the CRISTA algorithm can take in an sgRNA sequence and a genomic target sequence as input and figure out the different off-target sequences an sgRNA sequence can yield Abadi et al. (2017). To get the score, scientists used GUIDE-SEQ to search for potential genome sites. They then found the length of this target sequence (called a sequence read) by counting the base pairs. Then they found the log of this length number, and used this to represent the cleavage frequency. To train this model, they took different uncleaved and cleaved sites. The cleaved sites were validated based on previous experiments. The uncleaved sites were taken from the UCSC genome database where they were compared with how well they aligned to the sgRNA sequence. A score was found, and if it passed a certain threshold, it was used as part of the training data. The results were positive. A scatter plot was used that compared the correlation of the actual cleavage frequencies to the predicted frequency from the model. This was compared against other similar models. The results showed that the CRISTA predictions resembled the closest to the actual cleavage frequencies than other similar models (See **Figure 9**). The true, and false positive ratios were measured as well with the CRISTA algorithm showing to have the highest true:false positive ratio (See **Figure 9**). The source code can be found here: http://crista.tau.ac.il/ Abadi et al. (2017).

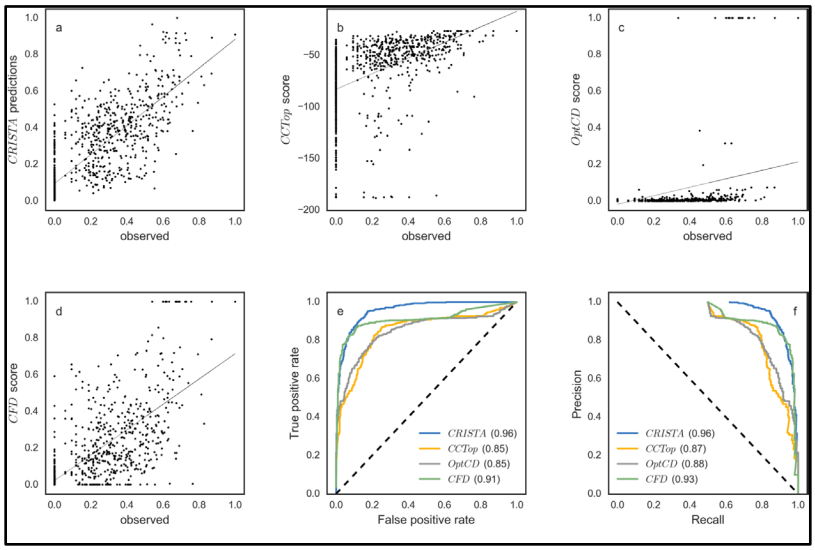

Figure 9. Results from testing CRISTA against other models.

According to the four scatterplots Figure 9a-9d, CRISTA had the strongest correlation with the CRISTA having a Pearson correlation coefficient closest to 1 where $r^2 = 0.65,$ compared to other models (CCTop, OptCD, CFD has Pearson correlation coefficients of $r^2 = 0.23, r^2 = 0.13,$ and $r^2 = 0.52$ respectively. (1 means very strong positive correlation). **9e.** shows the true:false positive ratio with CRISTA having the highest true:false positive ratio. **9f.** shows the ratio between precision and recall. CRISPR has the highest ratio of 0.96. Abadi et al. (2017). *Comparison of four prediction algorithms on the assembled dataset* [Graph]. https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005807#sec011

DeepCRISPR

DeepCRISPR is a deep-learning neural network. It is the few algorithms that can make on-target and off-target predictions based on sgRNA sequences. It is also seemingly one of the most advanced, complex models created for this type of application. To get the training data needed, these scientists had to do a couple of different things. For on-target data sources, it "contains seed sgRNAs with experimentally validated known knockout efficacy, comprising ~ 15,000 sgRNAs containing 1071 genes from four different cell lines (*hct116*, *hek293t*, *hela*, and *hl60*) with redundancy removed" (Chuai et al., 2018). For off-target data sources it contains "two different cell types: 293-related cell lines" (Chuai et al., 2018). To create the algorithm, the team first built an image processing-like code where it split a "picture" of the DNA region into pixels. Each pixel sorts the number of A,G,C,&T nucleotides. Then to make on-target predictions, DeepCRISPR uses an autoencoder. An autoencoder is a type of encoder used especially in image processing. This is often a neural network where unlike many other neural networks, the number of neurons in the input and output layers are the same, because the output is just the input image recreated. The hidden layers will contain a "bottleneck layer," a layer containing the least amount of neurons. This forces the neurons there to process less information which is what causes the output to be a much more simple recreation of the input image. This is particularly useful for things such as image compression, or when trying to decode important aspects in an image from a noisy background. This autoencoder in DeepCRISPR is used as a "pre-trained model" with a Convolutional Neural Network (CNN) to process inputs of information about sgRNA regions. These inputs represent sgRNA regions through one-hot encoding. For off-target predictions, the input goes through two encoders, a merging layer, then the CNN layers to process the output. However, the results payoff with results showing "*DeepCRISPR* generalized generally well in new cell types for sgRNA on-target knockout efficacy prediction;...*DeepCRISPR* efficiently learns the high-level feature representation by avoiding manual feature engineering for sgRNA design, indicated by the apple-to-apple comparisons with the retrained *sgRNA designer* (the gradient-boost-based classification or regression models) with the same training data, while with different features;...*DeepCRISPR* is robust with superior performance for both classification and regression models" (Chuai et al., 2018). **Figure 9** shows the results of the performance of DeepCRISPR below. The source code of DeepCRISPR can be found on this website: http://www.deepcrispr.net/, and on github: https://github.com/bm2-lab/DeepCRISPR. (Chuai et al., 2018).

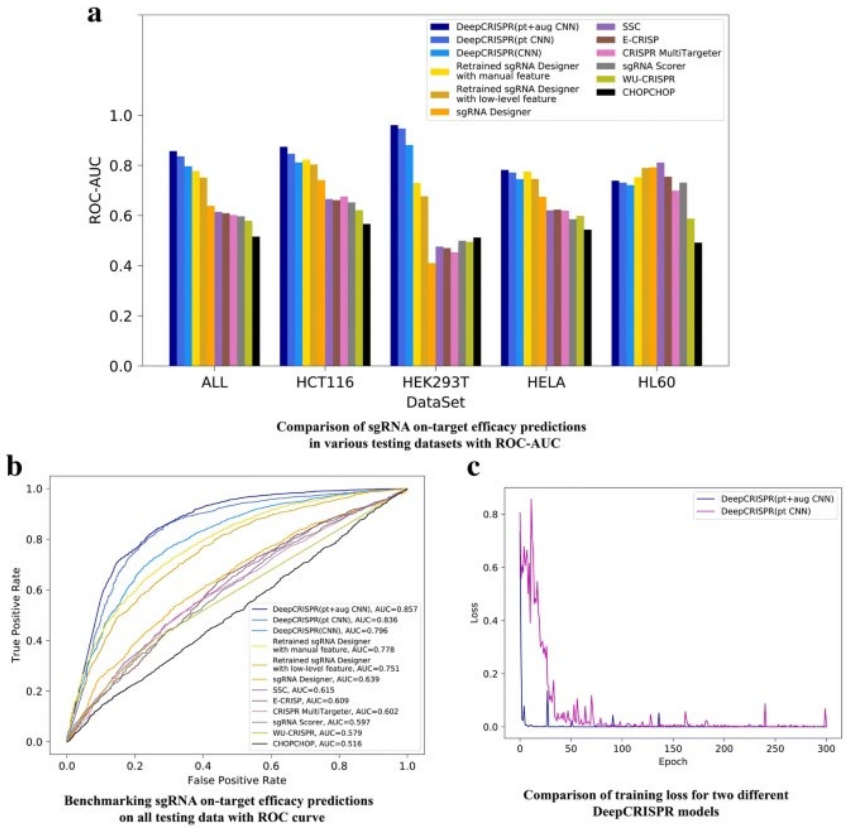

Figure 10. The first graph above, **10a.,** shows the ROC-AUC curve for many different sgRNA prediction models for different cell lines. DeepCRISPR outperforms all other methods in all cell lines. **10b.** shows DeepCRISPR having the highest true: false positive ratio compared to other models. **10c.** shows the the training loss between two versions of DeepCRISPR. Chuai et al. (2018). *Evaluation of DeepCRISPR for on-target efficacy prediction* [Graph]. https://genomebiology.biomedcentral.com/articles/10.1186/s13059-018-1459-4

CNN-XG

CNN-XG is another approach to predicting sgRNA cleavage efficiencies which uses a Convolutional Neural Network (CNN) and Extreme Gradient Boosting (XGBoost). The algorithm is split into two parts. The CNN part is meant to get data based on the sgRNA and other input supplied. The XGBoost side is used to actually predict the cleavage efficiency of the sgRNA. The algorithm takes in an input of an sgRNA sequence and 4 epigenetic sequences. These sequences are encoded using one-hot encoding. Then, they are fed as inputs into two convolutional and max pooling layers. After that, the final outputs generated are flattened as inputs for the neural network part. Computations are performed. The outputs from both of these neural networks are generated. The two outputs are combined with a final output layer that generates the CNN final outputs as a list of distinctive features of the sgRNA and epigenetic sequences. This information is sent to a Random Forest algorithm for further selection. This information is sent to an XGBoost classifier which assigns a score. To train their model, scientists had to obtain data from a variety of places. “a total of 1071 sgRNAs from four experimental, validated, sgRNA on-target cleavage efficacy, independent human datasets, HCT116, HEK293T, HELA and HL60" (Li et. al., 2022). However, scientists did not just use human cell lines for data collection. They used datasets from previous studies. They looked at other

literature, and they used cas9 nucleases. They compared their model to other models, such as a separate CNN model, and a separate XGBoost model. They also tested the CNN-XG model performance against the DeepCrispr, CNN-SVR models and much more. The results showed that in general, the CNN-XG model was good at classification, and performed well with the 10-fold cross validation set (See **figure 11**). The learning ability was also measured. "CNN-XG took about 10 min on average to train over tens of thousands of data per dataset, achieving a performance close to that of the best models" (Li et. al., 2022). The source code for this learning model is located on github: https://github.com/MoonLBH/CNN-XG

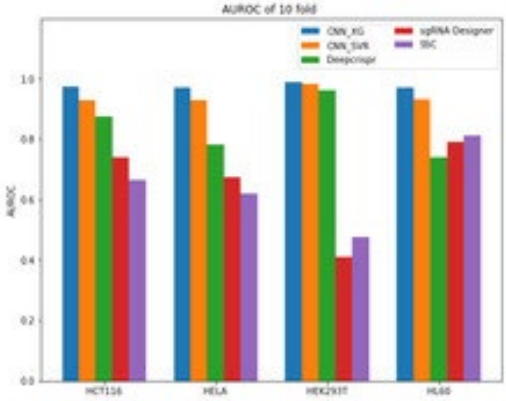

Figure 11. The figure above shows the AUROC results of CNN-XG with other models on different cells. In each case, CNN-XG outperformed the other models, but it outperformed DeepCRISPR by a small margin. Li et. al. (2022). *AUROC of 10 fold* [Bar Graph]. https://doi.org/10.3390/biom12030409

DL-CRISPR

DL-CRISPR is a much more advanced machine learning algorithm. Instead of using decision trees, such as CRISTA, it uses neural networks. More specifically, it uses Convolutional Neural Networks (CNN). DL-CRISPR first had to obtain a lot of data. Scientists "collected off-target sequences as well as their sgRNAs from *in vitro* – and cell-based genome-wide assays" (Zhang et. al., 2020). They released some of their sources that might not yield the most accurate, reliable results. So they had to conduct further cross-checking to have a dataset of 1128 reliable off-target sequences to use to train the model. Scientists also "downloaded the [another] dataset from [a scientist named Dr. Peng], which [was] found by Cas-OFFinder in human gene hg38 related to 29 sgRNAs with no more than 6 mismatches" (Zhang et. al., 2020). 403, 953 samples were obtained from this dataset to use for training. After obtaining this data, scientists built the DL-CRISPR on a 4-step process. First, scientists took the target sequences and the sgRNA sequence and converted each sequence in the form of a one-hot encoder where the categories were the adenine, guanine, cytosine, and thiamine nucleotides. Then this data was stored in a matrix in a special arrangement. This matrix had its values rotated four times by 90 degrees. This was used as a way to generate more data values to add to the training dataset to any imbalances. This data was sent as input to a multilayer model that uses 4 CNN algorithms combined to produce a score. This process was done for 10 such identical models, yielding 10 scores. An average score was taken and was shown as the final output. (See **Figure 12**) These scientists looked for performance with a 5-fold cross validation dataset test (a test where the training dataset is split and the model trains on this split data in order to avoid overfitting), ability to identify as many off-target sequences possible with two sgRNA sequences, and much more. They found that their

model performed the best in the 5-fold cross validation dataset test, and it looked for a lot more reliable off-target sequences from sgRNA sequences than previous models.

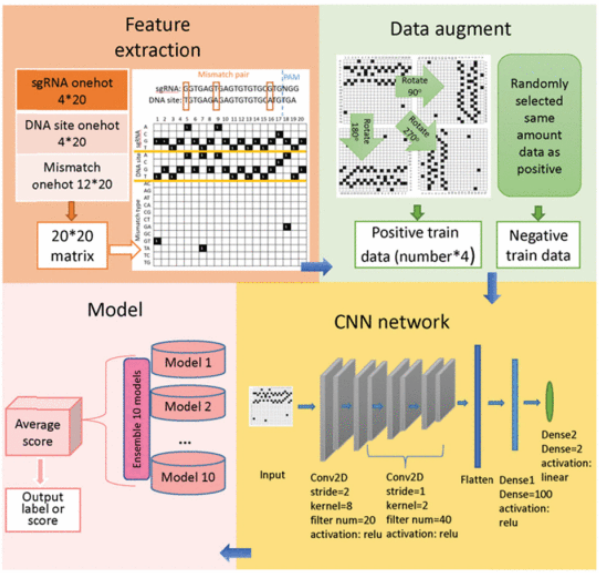

Figure 12. The figure above shows the structure with how the algorithm works.
Zhang et. al. (2020). *Overview of DL-CRISPR* [Diagram]. https://ieeexplore.ieee.org/document/9076075

The Best One

Compared to other models the CNN-XG model seems to be the best algorithm among the four. This is because of a couple of reasons. To begin with, the model looked for another set of data that the other models did not account for. The CNN-XG model took into account the epigenetic information related to the target sequence. This is highly valuable because while the base pairs in a gene express how the protein will function, these epigenetic factors decide if that gene will even be expressed in the first place. The model is also very thorough and efficient when it comes to extracting features of the sgRNA and epigenetic sequences. The model *compartmentalized* parts of its process to different machine learning algorithms rather than have one, overarching algorithm. For example, the model contained 2 CNNS, one for its input sgRNA sequence and one for its input epigenetic sequences. Meanwhile if the scientists went with just one CNN, then that neural network would be processing 2 different sets of data, increasing the load on the CNN. At the same time, each CNN had pooling layers for each of its convolutional lakers. Since the pooling layers reduce the size of the input, this improved efficiency because each convolutional layer was able to process more information from its filter. At the same time, the reduced size meant the CNN had to perform less computational work as well. The model goes a step further and feeds the results from the CNNs to a Random Forest system for further feature processing. Finally, assigning the cleavage score is delegated solely to an XGBoost algorithm. This model uses 3 different types of machine learning algorithms to produce reliable cleavage efficiency scores while other models use only one, maybe two. The results were also pleasing. While DL-CRISPR performed best in a 5-fold cross validation dataset test, the CNN-XG model was based to perform best at a 10-fold cross validation dataset test (This is a more thorough cross validation test than the cross validation test the DL-CRISPR model performed). The model as stated previously, is also able to train with massive amounts of data in a relatively short time. At the same time, the model performed better

than DeepCRISPR on cell lines (See **Figure 11**). These reasons make the CNN-XG Model the best model for sgRNA cleavage efficiency predictions compared to the other 3 models.

Conclusion

A common problem when dealing with CRISPR-Cas9 is finding a sgRNA design that can effectively guide the Cas9 enzyme. These designs are not perfect and yield off-target cleavages. However, scientists around the world have created CRISTA, DL-CRISPR, DeepCRISPR, CNN-XG, and many more machine learning models to address this issue. This paper also covered which model seems to work best for this problem. The implications with the models mentioned above are huge because CRISPR-Cas9 is used in so many applications. CRISPR-Cas9 is being used as a form of cancer treatment, a form of treatment for many genetic diseases, a way to further study the genetic network of plants and animals, agriculture, and much more. A good sgRNA design is key to a successful CRISPR operation, and using these models will go a long way.

There is more that can be done with this research. Scientists can use machine learning to build models of mRNA chains to design proteins. They can use machine learning to accurately study how gene editing affects different traits. They could also study the effects of weak sgRNA designs and off-target cleavages. This research paper highlights that machine learning is a useful tool that scientists can use to navigate through the billions of base pairs that make people, people.

Acknowledgements

Special thanks to Dr. Rajagopal Appavu, and Ms. Jothsna Kethar for their guidance, advice, and overall help in creating this research paper.

References

Abadi, S., Yan, W. X., Amar, D., & Mayrose, I. (2017). A machine learning approach for predicting CRISPR-Cas9 cleavage efficiencies and patterns underlying its mechanism of action. *PLOS Computational Biology*, *13*(10), e1005807. https://doi.org/10.1371/journal.pcbi.1005807

Bentéjac, Candice & Csörgő, Anna & Martínez-Muñoz, Gonzalo. (2019). A Comparative Analysis of XGBoost.

Chicco, D. (2017). Ten quick tips for machine learning in computational biology. *BioData Mining*, *10*(1). https://doi.org/10.1186/s13040-017-0155-3

Choi RY, Coyner AS, Kalpathy-Cramer J, Chiang MF, Campbell JP. Introduction to Machine Learning, Neural Networks, and Deep Learning. Transl Vis Sci Technol. 2020 Feb 27;9(2):14. doi: 10.1167/tvst.9.2.14. PMID: 32704420; PMCID: PMC7347027.

Chuai, G., Ma, H., Yan, J., Chen, M., Hong, N., Xue, D., Zhou, C., Zhu, C., Chen, K., Duan, B., Gu, F., Qu, S., Huang, D., Wei, J., & Liu, Q. (2018). DeepCRISPR: optimized CRISPR guide RNA design by deep learning. *Genome Biology*, *19*(1). https://doi.org/10.1186/s13059-018-1459-4

Deo, R. C. (2015). Machine Learning in Medicine. *Circulation*, *132*(20), 1920–1930. https://doi.org/10.1161/circulationaha.115.001593

Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S., & Dehmer, M. (2020). An Introductory Review of Deep Learning for Prediction Models With Big Data. *Frontiers in Artificial Intelligence*, *3*. https://doi.org/10.3389/frai.2020.00004

Fan, J., Ma, C., & Zhong, Y. (2021). A Selective Overview of Deep Learning. *Statistical Science*, *36*(2). https://doi.org/10.1214/20-sts783

Li, B., Ai, D., & Liu, X. (2022). CNN-XG: A Hybrid Framework for sgRNA On-Target Prediction. *Biomolecules*, *12*(3), 409. https://doi.org/10.3390/biom12030409

Mengstie, M. A., & Wondimu, B. Z. (2021). Mechanism and Applications of CRISPR/Cas-9-Mediated Genome Editing. *Biologics: Targets and Therapy*, *Volume 15*, 353–361. https://doi.org/10.2147/btt.s326422

Nayarisseri, Anuraj. (2019). Machine Learning, Deep Learning and Artificial Intelligence approach for predicting CRISPR for the Cancer treatment. 10.3390/mol2net-05-06258.

Vargas, Rocio & Mosavi, Amir & Ruiz, Ramon. (2017). DEEP LEARNING: A REVIEW. Advances in Intelligent Systems and Computing. 5.

Zhang, G., Zeng, T., Dai, Z., & Dai, X. (2021). Prediction of CRISPR/Cas9 single guide RNA cleavage efficiency and specificity by attention-based convolutional neural networks. *Computational and Structural Biotechnology Journal*, *19*, 1445–1457. https://doi.org/10.1016/j.csbj.2021.03.001

Zhang, Y., Long, Y., Yin, R., & Kwoh, C. K. (2020). DL-CRISPR: A Deep Learning Method for Off-Target Activity Prediction in CRISPR/Cas9 With Data Augmentation. *IEEE Access*, *8*, 76610–76617. https://doi.org/10.1109/access.2020.2989454