THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF STATISTICS

PREDICTING CRISPR-CAS9 GENOME EDITING OUTCOMES WITH MACHINE LEARNING METHODS

ANGELA TING
SPRING 2020

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Mathematics and Statistics
with honors in Statistics

Reviewed and approved* by the following:

Qunhua Li
Associate Professor of Statistics
Thesis Supervisor

David Hunter
Professor of Statistics
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

Abstract

CRISPR-Cas9 technology is an innovative tool that has revolutionized genome editing in the last decade. This technology targets specific stretches of genetic code to edit DNA at specific locations, allowing researchers to edit parts of the genome by removing, adding, or altering sections of the DNA sequence. However, the editing outcomes of CRISPR technology are often heterogeneous and stochastic. Thus, to remedy this incongruity, machine learning methods are used to model genome outcomes based on the guide RNAs (gRNAs) used in CRISPR experiments. In this way, users who wish to use CRISPR-Cas9 technology can design proper gRNAs to maximize the chance of generating the desired genome editing outcomes.

A functional neural network was implemented that is able to accurately predict genome editing outcomes in a controllable manner. However, such a neural network operates successfully only with complete training data sets. Complete datasets are datasets in which all variables are present and no data is missing. Nonetheless, complete datasets are rare given the restraints of reality. Rather, time, monetary expenses, practicality restrictions, as well as data compilation from disparate sources often brings about training data that is incomplete. Incomplete data refers to datasets with a combination of missing values and deleted observations. They pose a problem because the essence of the neural network training process requires that predicted values from the neural network be compared and made close to their true values provided in the dataset by minimizing a function of those predicted and true values called the loss function. However, without the presence of these true values, as is the case in incomplete datasets, the loss function cannot be minimized and the neural network fails to be trained.

In order to rectify the inconsistency of incomplete datasets with the neural network, additional

machine learning models using data imputation methods are applied to these incomplete datasets in order to allow them to resemble complete datasets. In this way, through an additional process, the neural network will be able to train data originally arising from both complete and incomplete datasets. This combination of machine learning techniques results in a robust resolution that allows predictable distributions of genome editing outcomes on a wide array of imperfect datasets.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my thesis supervisor, Dr. Qunhua Li, for her support, supervision, and suggestions throughout the process of this thesis as well as my honors advisor, Dr. David Hunter, for taking the time to read this thesis and giving excellent advice throughout my time at Penn State. I'd also like to thank Kutubuddin Molla, Vicki Hsieh, and Dr. Yinong Yang for providing additional biological reference and data sources. Lastly, I'd like to thank Xinyi Yu for providing great insight and advice throughout this project.

Chapter 1:

Introduction

1.1 CRISPR-CAS9 Technology

CRISPR-Cas9 technology has revolutionized genome editing in the last decade. This technology targets stretches of genetic code to edit DNA at specific locations, allowing researchers to edit parts of the genome by removing, adding, or altering sections of the DNA sequence. The CRISPR-Cas9 system consists of two important molecules that introduce a mutation into the DNA: the Cas9 enzyme and guide RNA (gRNA). Cas9 is a DNA-cutting enzyme which can cut the two strands of DNA at a specific location in the genome such that DNA nucleotides can be added or removed. The gRNA is a predesigned RNA sequence designed to 'guide' the Cas9 enzyme to the desired cut site in the genome.[4] The CRISPR-Cas9 mechanism is shown in Figure 1.1. It can be seen that the gRNA binds to a specific portion of the DNA known as the target site. The gRNA in essence "guides" the Cas9 enzyme to the same location in the DNA and makes a cut so that a sequence of nucleotides or genes may be inserted or deleted. Following a Cas9-induced cut, the DNA will recognize its damage and try to repair itself.

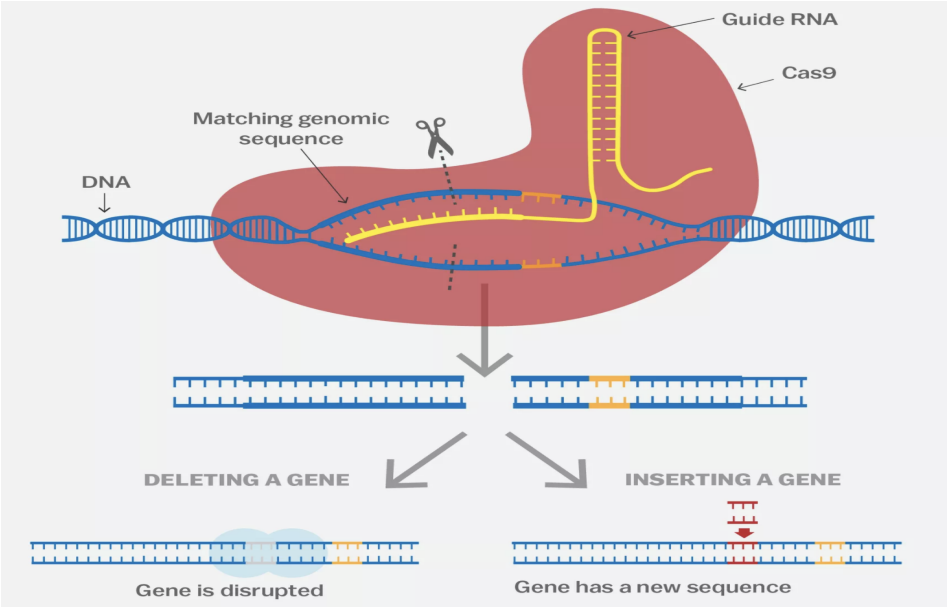


Figure 1.1: The biological mechanism of CRISPR-CAS9 technology.[1]

At this stage, scientists can take advantage of DNA repair machinery to introduce changes to one or more genes in the genome of a cell of interest. Major pathways involved in the repair of Cas9-mediated double-stranded breaks include non-homologous end joining (NHEJ) and microhomology-mediated end joining (MHEJ). These two pathways are shown in Figure 1.2. As depicted in NHEJ, after a double strand break, the ends of the DNA are joined with no repair template, resulting in error prone repair. As depicted in MHEJ, after a double strand break, michrohomology regions (5-25 bp) are matched DNA ends are removed to reveal homology, thus allowing the strands to anneal and DNA synthesis to fill in the gaps. Although these pathways are efficient, they both also result in highly stochastic and heterogenous repair outcomes comprising hundreds of repair genotypes. For this reason, NHEJ and MMEJ have generally not been regarded as useful for precision genome editing applications.[5]

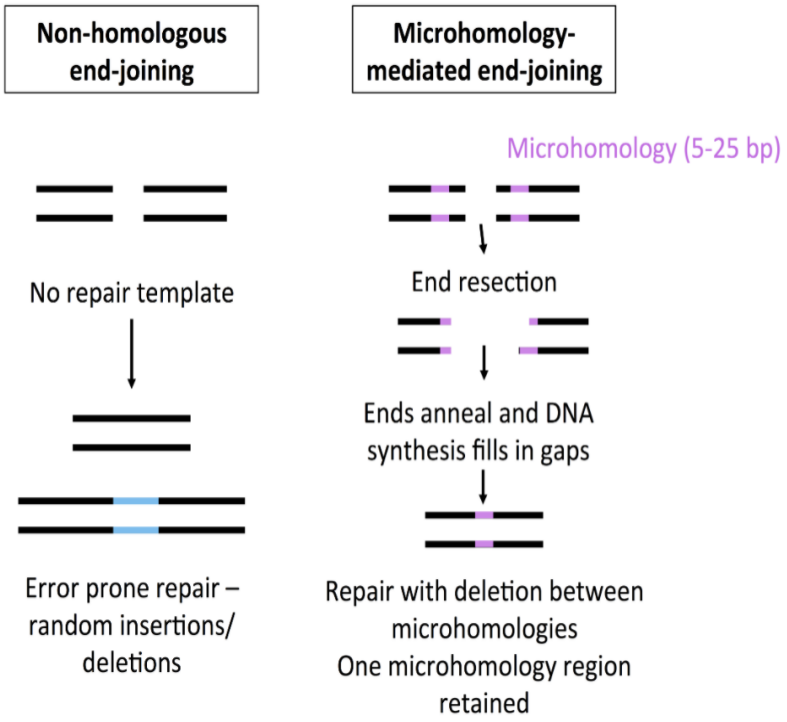


Figure 1.2: Biological process of non-homologous end joining (NHEJ) and microhomology-mediated end joining (MHEJ).[2]

1.2 Machine Learning Models for CRISPR Technology

Several machine learning models for predicting genome editing outcomes have been implemented before. One of these models is named SPROUT.[6] SPROUT takes the gRNA along with a list of 33 genomic features to output features of interest such as the insertion to deletion ratio, the average insertion and deletion length, and the most likely inserted base pair. The underlying machine learning method used is gradient boosted tree ensembles. Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak decision tree models. A decision tree predicts the value of a target variable by learning simple decision rules inferred from data features. Hence, a model is built in an iterative manner by ”adding” several decision trees, where each additional decision tree attempts to correct the errors of its predecessor.[7]

Another machine learning model that has been implemented is named FORECasT.[8] FORECasT takes the target DNA sequence and the index of the PAM (protospacer adjacent motif) to output a repair profile distribution detailing the percentage of various insertion and deletion repair events. In this model, the machine learning method used is multinomial logistic (softmax) regression. Softmax regression is a generalization of logistic regression to the case where we want to handle multiple classes. That is, softmax regression estimates the probability of the class label taking on each of its possible different values.[9]

A further machine learning model used in CRISPR technology is inDelphi.[3] inDelphi takes the gRNA, DNA target site, and PAM sequence to output a distribution of genotypes and frequencies of insertions and deletions. inDelphi uses a neural network to perform its modeling. A neural network is a series of algorithms that are designed to recognize underlying patterns and relationships in a set of data through a process that mimics the way the human brain operates.[10] Given the neural network's unmatched power, intricacy, and ability to predict complex nonlinear patterns, the inDelphi neural network will be of most interest in this research project.

1.3 inDelphi Model

inDelphi is a machine learning model that accurately predicts the frequencies of the substantial majority of template-free Cas9-induced insertion and deletion events (indels) at single-base resolution–particularly, indel genotype frequencies as well as indel length frequencies. While the aforementioned template-free end joining pathways do appear heterogeneous, the inDelphi machine learning model establishes that these repair outcomes can indeed be predicted. inDelphi strives to reinterpret end joining repair from an undesirable repair pathway into an efficient DNA repair pathway such that heterogeneity can be predicted and controlled.

The full inDelphi model has 3 modules: one for microhomology deletions (MMEJ), one for microhomology-less deletions (NHEJ), and one for 1-bp insertions. The deletion events are modeled with a neural network while the insertions are modeled using k-nearest neighbors regression.

Since deletions represent the majority ($63 - 87\%$) of all indels for all datasets used in inDelphi, we will focus our attention on deletion events. For these deletions, the inDelphi neural network is a modular neural network in a multitask learning framework. That is, two neural networks–one representing deletion genotype frequency distributions (MH-NN) and the other representing deletion length frequency distributions (MHless-NN)–are jointly trained with shared parameters. MH-NN takes as inputs the microhomology lengths and GC fractions of a sequence. Two hidden layers follow, each with 16 nodes, which are then passed to an output layer of 1 node. Both hidden layers have sigmoidal activations. MHless-NN takes as input the deletion lengths of a sequence. Two hidden layers also follow, each with 16 nodes, which are again passed to an output layer of 1 node. Both hidden layers also have sigmoidal activations.[3] Figure 1.3 depicts the structure of the multitask neural network.

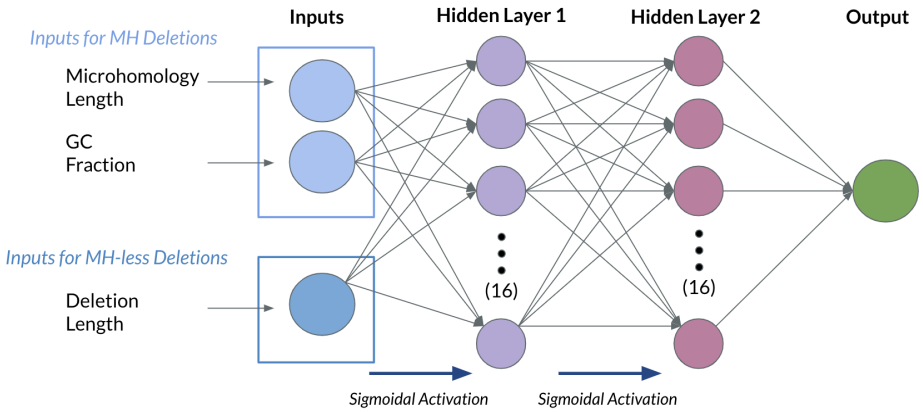
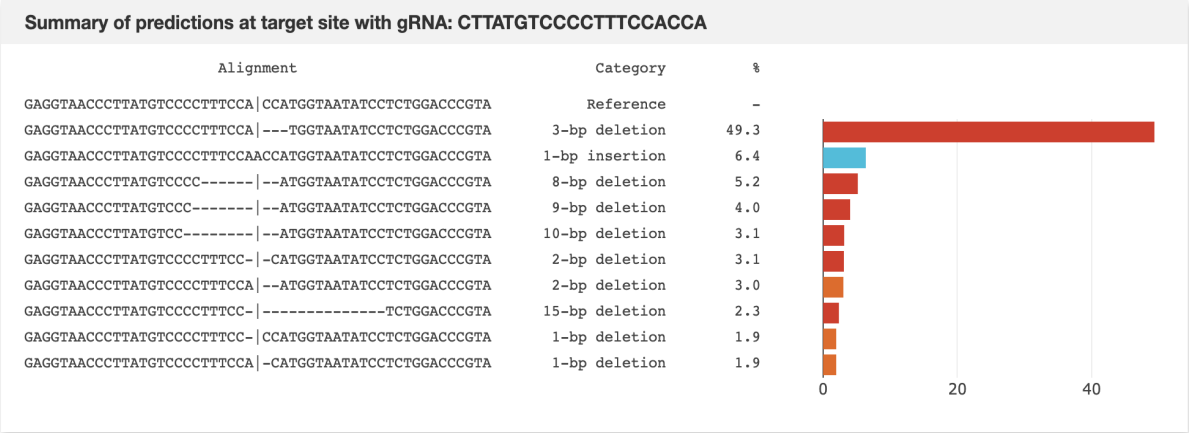


Figure 1.3: inDelphi neural network with multitask learning.[3]

1.3.1 inDelphi Distribution Outcomes

The inDelphi model allows users to obtain a distribution of genotypes as well as a distributions of insertion/deletion outcomes for any CRISPR Cas-9 edited genome. The modeling outcome of inDelphi on an arbitrary CRISPR edited DNA sequence is shown in Figure 1.4. The set of nucleotide sequence outcomes with their inserted or deleted nucleotides is shown on the left side of the figure. Each nucleotide sequence outcome is considered a unique genotype. The corresponding frequency of occurrence for each specific genotype outcome is displayed as a percentage

and depicted in a bar chart on the right. The ”Category” column specifies whether a deletion or insertion caused the corresponding genotype, and how many base pairs were inserted or deleted. This constitutes the distribution of genotypes for that particular genome. Resultingly, by adding all percentages of genotypes from the same ”Category” (genotypes that have the same number of base pairs inserted or deleted), then the distribution of insertion/deletion outcomes is obtained as a function of ”Category.”



Summary of predictions at target site with gRNA: CTTATGTCCCCTTTCCACCA

Alignment	Category	%
GAGGTAACCCTTATGTCCCCTTTCCA|CCATGGTAATATCCTCTGGACCCGTA	Reference	-
GAGGTAACCCTTATGTCCCCTTTCCA|---TGGTAATATCCTCTGGACCCGTA	3-bp deletion	49.3
GAGGTAACCCTTATGTCCCCTTTCCAACCATGGTAATATCCTCTGGACCCGTA	1-bp insertion	6.4
GAGGTAACCCTTATGTCCCC------|--ATGGTAATATCCTCTGGACCCGTA	8-bp deletion	5.2
GAGGTAACCCTTATGTCCC-------|--ATGGTAATATCCTCTGGACCCGTA	9-bp deletion	4.0
GAGGTAACCCTTATGTCC--------|--ATGGTAATATCCTCTGGACCCGTA	10-bp deletion	3.1
GAGGTAACCCTTATGTCCCCTTTCC-|-CATGGTAATATCCTCTGGACCCGTA	2-bp deletion	3.1
GAGGTAACCCTTATGTCCCCTTTCCA|--ATGGTAATATCCTCTGGACCCGTA	2-bp deletion	3.0
GAGGTAACCCTTATGTCCCCTTTCC-|----------TCTGGACCCGTA	15-bp deletion	2.3
GAGGTAACCCTTATGTCCCCTTTCC-|CCATGGTAATATCCTCTGGACCCGTA	1-bp deletion	1.9
GAGGTAACCCTTATGTCCCCTTTCCA|-CATGGTAATATCCTCTGGACCCGTA	1-bp deletion	1.9

Figure 1.4: Genotype and Insertion/Deletion outcome distributions using the inDelphi model.[3]

1.4 mESC Data

The data used throughout this research project was the mESC dataset collected by MIT’s Gifford Laboratory, as used in the development of the original inDelphi model.[11] CRISPR-Cas9 technology was applied to genetically edit mESC DNA, and the results from the editing were recorded in a dataset. This unprocessed mESC dataset contains the category of each genotype editing outcome (insertion or deletion), length of the insertion or deletion (in base pairs), count for the number of times the genotype outcome occurred, and a measure of the genotype’s position in the DNA sequence. Additionally, the mESC dataset is accompanied by a list of corresponding target sequences and gRNAs for each experiment. The structure of the original and unprocessed mESC dataset as described above is shown in Figure 1.5.

1	Category	Ins/Del Length	Count	Genotype Position	Experiment
2	del	7	5	0	0
3	del	1	5	1	0
4	del	6	1	2	0
5	del	4	1	3	0
6	del	15	5	4	0
7	del	8	9	5	0
8	del	28	5	11	0
9	del	35	1	16	0
10	del_notcrispr	1	1	-55	0
11	del_notcrispr	1	6	-52	0
12	del_notcrispr	1	1	-52	0
13	del_notcrispr	2	1	-42	0
14	del_notcrispr	1	188	-40	0
15	del_notcrispr	1	35	-40	0

Figure 1.5: Unprocessed mESC dataset.

1.5 Data Processing

As shown in the neural network structure in Figure 1.3, the inputs of the model are microhomology length, GC fraction, and deletion length. Using the mESC dataset, in order to obtain these inputs, the data must be processed accordingly. Recall that deletions represent the majority of all indels. Thus, we focus on deletion modeling and first remove all insertion events from the dataset. After this, we filter the dataset for only CRISPR repair products. Occasionally, deletions may occur outside of the desired cleavage base pair window, and in this case we deem the result to not be a CRISPR repair product and remove such an observation. Microhomology lengths of each genotype were then obtained by passing the target sequence, gRNA, and genotype position for each genotype outcome into the CRISPR RGEN microhomology predictor.[12] GC fraction was obtained for each genotype outcome by calculating the fraction of G and C nucleotides out of all nucleotides in the corresponding gRNA. At this point, the dataset contains microhomology length, GC fraction, and deletion length for each genotype–allowing passage into the neural network. The structure of the processed mESC dataset is shown in Figure 1.6.

1	MH Length	GC Fraction	Deletion Length	Genotype Frequency	Experiment
2	1	0	1	0.303448276	2
3	2	0	2	0.337931034	2
4	1	0	3	0	2
5	1	0	6	0	2
6	1	1	6	0	2
7	4	0.5	7	0.172413793	2
8	1	0	8	0	2
9	1	1	8	0	2
10	1	0	10	0	2
11	1	0	11	0.006896552	2
12	1	0	11	0	2
13	1	0	12	0	2
14	1	0	13	0	2
15	1	1	13	0	2

Figure 1.6: Processed mESC dataset.

1.6 Causes and Challenges of Missing Data

The mESC dataset as described in Section 1.4, with all data variables present and no missing data is considered complete. Versions of the mESC dataset simulated with a combination of missing values and deleted observations are considered incomplete.

Although complete datasets are always the most desired for having an abundance of data present and not needing to resort to speculation on the cause and effects of missing data on the resulting analysis, they are not always the case. In reality, most datasets are not perfectly complete. Given the constraints of reality, it is much more likely that time, money, and practicality restrictions bring about data that is incomplete to some capacity. In our case, the time and monetary expenses of using gene editing technology often results in a small number of experiments. Clearly, excessively small amounts of data is considered inadequate since it lacks the ability to allow for results that accurately represent the population in question. In these cases, it is wise to augment the available data with data from literature. When searching for data in literature, missing data is very likely–especially when researchers must adapt the data in literature to fit the needs of their own data. In our case, missing data always presented itself in literature in the lack of genotype and deletion length frequencies. That is, the count of editing outcomes for each particular genotype and deletion length was not recorded or not measured in literature. If included in a dataset,

the missingness of these frequency values would render the dataset incomplete. An example of the structure of an incomplete mESC dataset is shown in Figure 1.7. We may note that all frequency values are missing since they are marked with ”NA” for not available. Datasets compiled such that values from a combination of other feature variables are missing are considered incomplete as well.

1	MH Length	GC Fraction	Deletion Length	Genotype Frequency	Experiment
2	1	0	1	NA	2
3	2	0	2	NA	2
4	1	0	3	NA	2
5	1	0	6	NA	2
6	1	1	6	NA	2
7	4	0.5	7	NA	2
8	1	0	8	NA	2
9	1	1	8	NA	2
10	1	0	10	NA	2
11	1	0	11	NA	2
12	1	0	11	NA	2
13	1	0	12	NA	2
14	1	0	13	NA	2
15	1	1	13	NA	2

Figure 1.7: Incomplete mESC dataset.

Incomplete datasets present many challenges in the scope of machine learning with supervised neural networks. The supervised inDelphi neural network model shown in Figure 1.3 requires that inputs and outputs be completely present in the dataset for training. Clearly, the three inputs: microhomology length, GC fraction, and deletion length are present in the dataset as they are simple measurements that may be easily obtained by researchers. When passed through the neural network, frequency value predictions are outputted. However, the essence of the neural network training process requires that predicted frequency values be compared to true frequency values by minimizing a function of the predicted and true frequency values called the loss function. Without the true frequency values–the ones often unattainable in literature, the loss function cannot be minimized and the neural network fails. In an attempt to further examine and alleviate these issues of incomplete and small amounts of data, versions of the mESC dataset will be simulated with a combination of missing values and deleted observations, and performance of the data using both the neural network as well as data imputation methods will be examined in Chapters 2 and 3.

Chapter 2:

Methods and Results of the Neural Network

2.1 Pytorch Implementation of inDelphi

The original inDelphi neural network has implementation in pure Python. Several benefits exist for a re-implementation of inDelphi using a deep learning framework–particularly for optimization and flexibility. Deep learning frameworks are optimized for performance and parallelized to reduce computations. This allows efficient model flexibility and scalability that is especially useful for those with limited computational resources.

The inDelphi neural network was hence reimplemented in PyTorch. PyTorch uses dynamic computational graphs with automatic gradient computations. It is a flexible and efficient deep learning framework suited well for the purposes of inDelphi. Loss values, correlation values, and convergence rates were examined in both the pure Python and PyTorch version implementations to ensure the agreement of both models. All following tests will be run with the PyTorch implementation of inDelphi.

2.2 Experimental Design

2.2.1 Loss Function

As noted in Chapter 1, a neural network is series of algorithms that are designed to recognize underlying patterns and relationships in a set of data. Since the inDelphi neural network is a modular neural network, the microhomolgy length and GC fraction input variables are passed through the network to get an output of genotype frequencies and the deletion length variable is passed through the network to get an output of deletion length frequencies. Neural networks are optimized by a loss function, which calculates model prediction error based on the true values. During the training process, the input data is passed through the neural network many times until the loss function has stabilized at a minimum. This stabilitzation tends to happen uniformly at around 30 epochs, where one full epoch occurs when the entirety the training data has been passed through the neural network once. The loss function used in the inDelphi neural network is the negative sum of the Pearson *r* correlations of the genotype frequencies and deletion length frequencies. We may define the Pearson *r* correlation value as

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}$$

where x is a vector of the predicted frequencies (either for genotype or deletion length), $\bar{x}$ is the mean of all x values, y is a vector of the true frequencies, and $\bar{y}$ is the mean of all y values. The Pearson correlation value is calculated for both genotype frequencies and for deletion length frequencies. If we denote r_g as the Pearson correlation value for genotype frequencies and r_d as the Pearson correlation value for deletion length frequencies, then our loss function we seek to minimize is defined as

$$loss = -(r_g + r_d).$$

We may note that the Pearson correlation *r* is a measure of the strength of linear association between two variables, where the value $r = 1$ indicates a perfect positive correlation and $r = 0$ indicates no correlation. Intuitively, we strive for the Pearson *r* value between the predicted and actual values to be as close to 1 as possible since this dictates an exact matching between predicted and actual values. The closer to 0 the Pearson *r* value becomes, the less true and predicted values coincide, and the less accurate the neural network's predictions are.

2.2.2 Using inDelphi on complete and incomplete mESC datasets

Recall from Section 1 that the original mESC dataset is considered complete as it has no missing variables and no missing observations. The complete mESC dataset will be simulated to create incomplete datasets, which are ones that have missing values or deleted observations. In the following sections in this chapter, inDelphi will be tested on the complete mESC dataset as well incomplete versions of the mESC dataset as well. The incomplete mESC datasets include random subsets of experiments with all genotype frequency values present in each experiment, truncations of the complete dataset using only the top highest genotype frequencies in each experiment, truncations of the complete dataset using only a number of random genotype frequency values from each experiment, and assigning an equal frequency to each of the five highest genotype frequencies in each experiment. We may note that we need only to apply these changes to genotype frequency values, since deletion length frequency values are calculated from genotype frequency values and will subsequently change automatically. It can be observed that the first three incomplete mESC datasets listed represent occurrences where there is no missing data, but the size of the completed dataset is truncated in various significant ways. On the other hand, the fourth dataset listed represents more closely the case when genotype frequency values are not recorded by researchers (it is assumed that higher frequency observations are more likely to be observed by researchers; hence we use the highest five genotype frequencies). In this case, we assign an arbitrary frequency value that is the same for each of the 5 genotype frequencies. The results of both the complete mESC dataset and the incomplete versions are detailed in the next sections in order to observe, interpret,

and compare the difficulties that occur in a supervised neural network when using various forms of incomplete data as opposed to complete data.

2.3 Using inDelphi on the complete mESC data

Using the complete mESC dataset, inDelphi predicted a Pearson $r = 0.9518$ for the test genotype frequency distributions and Pearson $r = 0.9496$ for the test deletion length frequency distributions at the final epoch (Table 2.1). Clearly, the table shows that the Pearson r value for both the test genotype and deletion length frequency distributions are very close to 1, indicating almost complete coincidence of the predicted and actual frequency values in the testing dataset. We may also observe in Figure 2.1 that the Pearson r increases quickly and levels out to approach a value near 1 for both the genotype and frequency correlations in the testing and training datasets, as seen in the last four graphs in the figure. Similarly, training and testing losses are minimized quickly and stabilize at around -1.65. This stabilizing minimum indicates that the model prediction error has reached a minimum. Each of the three colored lines in the figure indicates a different run through the neural network. Runs may differ mostly from random variation in data and initialization values. It is most desirable for the neural network to give runs that are as identical as possible, so that accurate results may be anticipated reliably despite random variation. Figure 2.1 shows that runs for this complete mESC dataset are generally very similar and close together. Hence, we use the complete mESC dataset and results in the series of graphs in Figure 2.1 as a reference of an ideal and successful neural network.

Table 2.1: Average loss and correlation values on complete mESC data.

Epoch	Train Loss	Train Gen Corr	Train Del Corr	Test Loss	Test Gen Corr	Test Del Corr
1	-0.9167	0.8088	0.7928	-1.1111	0.8528	0.8399
5	-1.5902	0.9467	0.9341	-1.6108	0.9469	0.9392
10	-1.6269	0.9517	0.9412	-1.6406	0.9503	0.9459
15	-1.6339	0.9524	0.9429	-1.6414	0.9505	0.9462
20	-1.6334	0.9518	0.9434	-1.6473	0.9507	0.9477
25	-1.6366	0.953	0.9434	-1.6481	0.951	0.9476
30	-1.6471	0.9537	0.9456	-1.6572	0.9518	0.9496

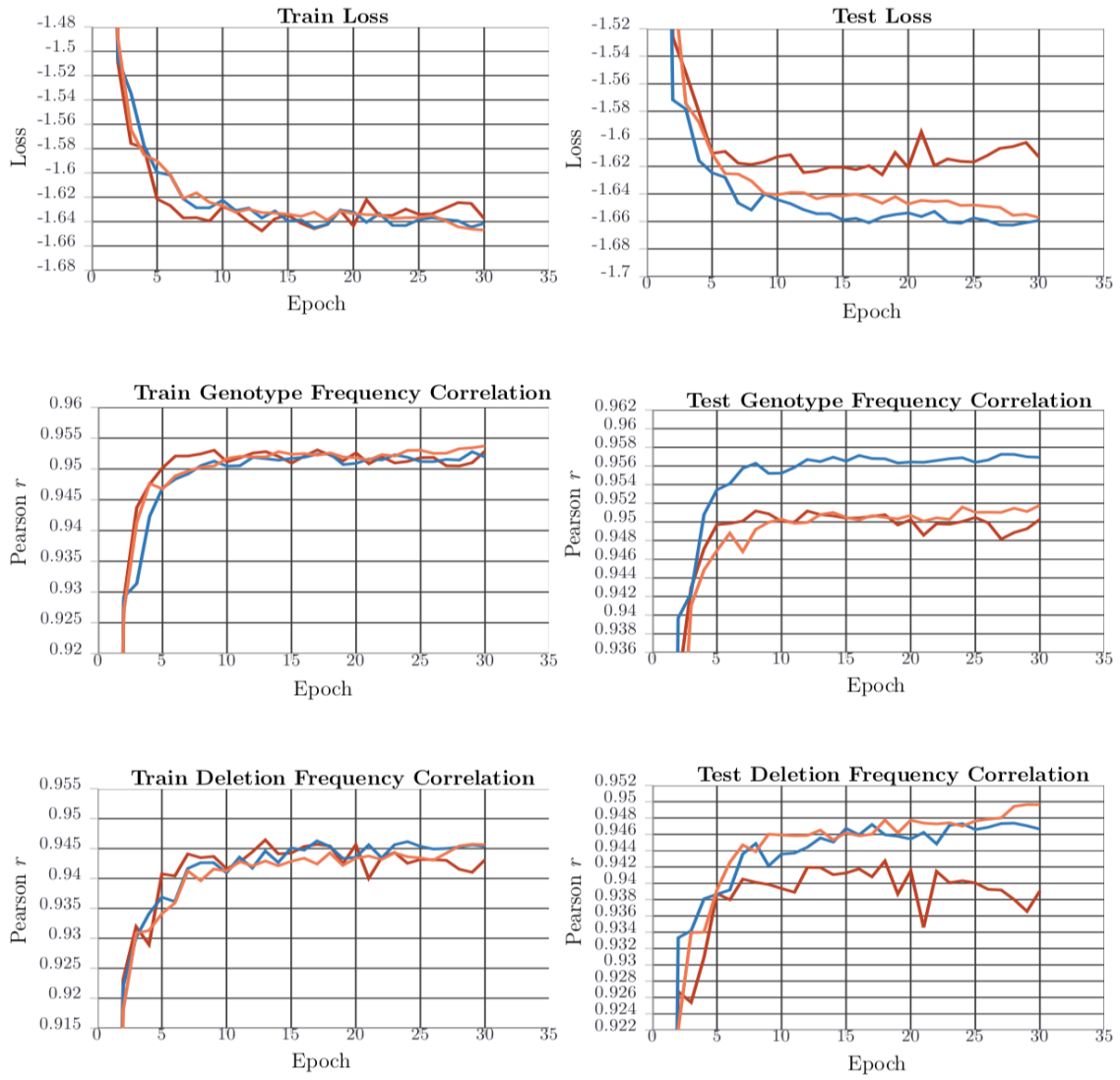


Figure 2.1: Loss and correlation values on complete mESC data. Each colored line is a different run.

2.4 Using inDelphi on incomplete mESC data

While inDelphi is trained on extensive and complete data, such sufficiency of data is not always feasible. It is quite likely that due to time, cost, and resource restraints, or a generalization to a modified scenario, training and testing datasets may become compromised or deficient in some way–that is, incomplete. Given the large amount of data in the mESC dataset, it is reasonable to

modify this dataset to incomplete versions of itself, run the inDelphi model on such incomplete versions, and observe similarities and differences in comparison to the complete dataset.

2.4.1 Random subsets

One way sequencing data may be deficient is the lack extensive observations. That is, only a limited amount of experiments may have been performed. To simulate such a scenario, random subsets of 50%, 25%, 10%, and 5% of the complete mESC dataset were taken–in effect simulating a reduced number of experiments performed. inDelphi was tested on these four reduced amounts of data. As a result, correlation values remained fairly stable among all reduced amounts. For 50% of the complete data, a Pearson $r = 0.9507$ was observed for test deletion genotype frequency distributions and a Pearson $r = 0.9414$ was observed for test deletion length distributions (Table 2.2). For 5% of the complete data, a Pearson $r = 0.9472$ was observed for test deletion genotype frequency distributions and a Pearson $r = 0.924$ was observed for test deletion length distributions (Table 2.3). This compares quite similarly to the Pearson $r = 0.9518$ for the test deletion genotype frequency distributions and Pearson $r = 0.9496$ for the test deletion length frequency distributions in the complete dataset (Table 2.1). It is, however, noticeable that with each successively smaller percentage of the data taken, the similarity of individual runs through the neural network becomes more stochastic (Figures 2.2, 2.3). This is likely attributed to the randomness in selecting different percentage subsets of the data.

Table 2.2: Average loss and correlation values on 50% mESC data.

Epoch	Train Loss	Train Gen Corr	Train Del Corr	Test Loss	Test Gen Corr	Test Del Corr
1	-0.8014	0.7788	0.766	-0.9661	0.8209	0.8101
5	-1.5915	0.9466	0.9348	-1.5892	0.947	0.9362
10	-1.6146	0.9497	0.9392	-1.6078	0.9499	0.9392
15	-1.6286	0.9515	0.942	-1.6172	0.9514	0.9408
20	-1.626	0.9516	0.941	-1.6121	0.9502	0.9402
25	-1.6295	0.9515	0.9421	-1.6153	0.9505	0.9409
30	-1.6318	0.952	0.9425	-1.617	0.9507	0.9414

Table 2.3: Average loss and correlation values on 5% mESC data.

Epoch	Train Loss	Train Gen Corr	Train Del Corr	Test Loss	Test Gen Corr	Test Del Corr
1	-0.843	0.7682	0.7933	-1.0189	0.8315	0.8233
5	-1.5929	0.9527	0.9299	-1.5613	0.945	0.9259
10	-1.5681	0.9455	0.9287	-1.5338	0.9395	0.9226
15	-1.6266	0.9568	0.9351	-1.5427	0.9447	0.9201
20	-1.6294	0.9581	0.9353	-1.5543	0.9465	0.9218
25	-1.6569	0.9608	0.9423	-1.5508	0.9461	0.9212
30	-1.6319	0.9585	0.9356	-1.5642	0.9472	0.924

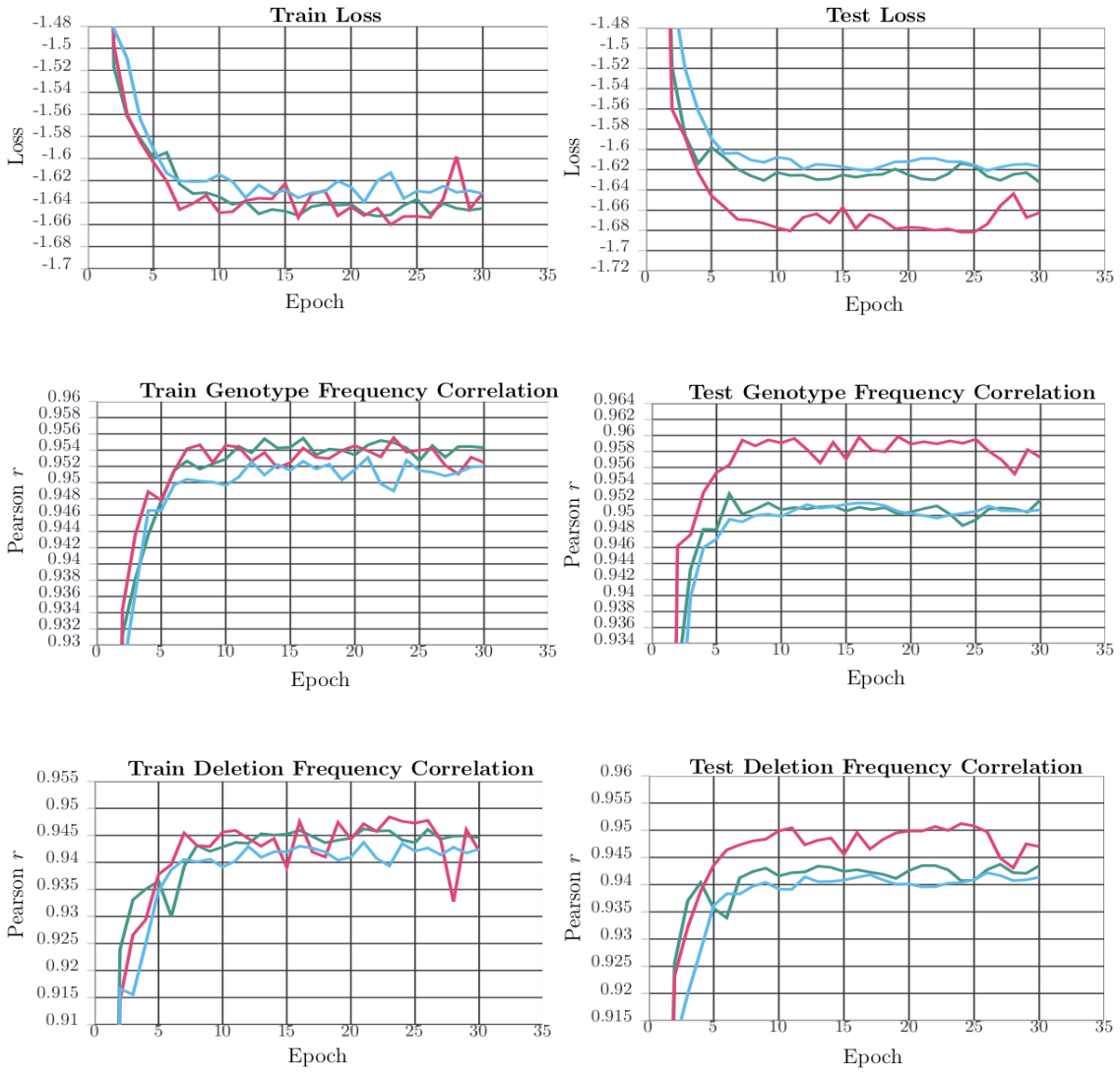


Figure 2.2: Loss and correlation values on 50% mESC data. Each colored line is a different run.

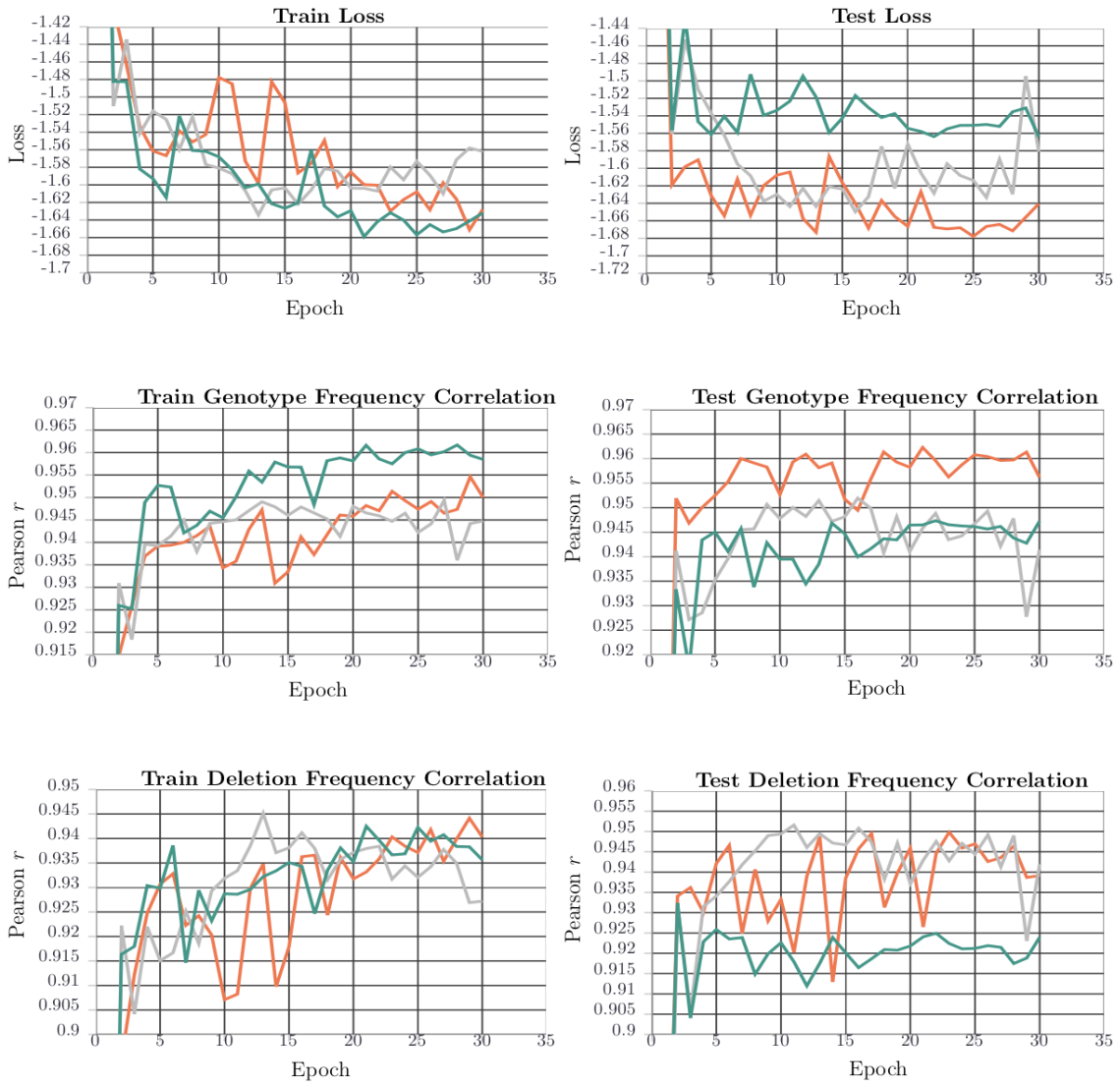


Figure 2.3: Loss and correlation values on 5% mESC data. Each colored line is a different run.

2.4.2 Top Frequencies

Another way sequencing data may be deficient is the lack of complete frequencies per experiment. That is, only the highest frequencies may have been obtained in the experiments. To simulate such a scenario, the top 10 highest genotype frequencies, top 5 highest genotype frequencies, and top 3 highest genotype frequencies for each experiment were taken from the complete mESC dataset. inDelphi was tested on these three datasets with varying high frequencies. Once again, correlation values remained fairly stable among all three datasets, although decreasing slightly at

top 3 highest frequencies. For top 10 highest frequencies, a Pearson $r = 0.929$ was observed for test deletion genotype frequency distributions and a Pearson $r = 0.9614$ was observed for test deletion length distributions (Table 2.4). For top 3 highest frequencies, a Pearson $r = 0.9028$ was observed for test deletion genotype frequency distributions and a Pearson $r = 0.9667$ was observed for test deletion length distributions (Table 2.5). This still compares quite similarly to the Pearson $r = 0.9518$ for the test deletion genotype frequency distributions and Pearson $r = 0.9496$ for the test deletion length frequency distributions in the complete dataset (Table 2.1). It can be observed in this scenario that the lack of randomness, in contrast to the random subsets previously discussed, allows the runs within each dataset to be more similar to each other (Figures 2.4, 2.5).

Table 2.4: Average loss and correlation values on top 10 frequencies of mESC data.

Epoch	Train Loss	Train Gen Corr	Train Del Corr	Test Loss	Test Gen Corr	Test Del Corr
1	-0.7241	0.6976	0.7708	-0.9043	0.7512	0.8175
5	-1.6084	0.9223	0.9577	-1.601	0.9211	0.9573
10	-1.6354	0.9284	0.9619	-1.6363	0.9294	0.9618
15	-1.6358	0.9282	0.9618	-1.6278	0.9273	0.9608
20	-1.6385	0.9288	0.9623	-1.6278	0.9275	0.9607
25	-1.6288	0.9267	0.961	-1.6316	0.9286	0.9611
30	-1.6412	0.9294	0.9626	-1.6337	0.929	0.9614

Table 2.5: Average loss and correlation values on top 3 frequencies of mESC data.

Epoch	Train Loss	Train Gen Corr	Train Del Corr	Test Loss	Test Gen Corr	Test Del Corr
1	-1.0071	0.8082	0.8026	-1.1688	0.8267	0.8634
5	-1.6178	0.9022	0.968	-1.6096	0.906	0.9655
10	-1.6599	0.9115	0.9748	-1.6538	0.9134	0.9717
15	-1.6772	0.9146	0.9765	-1.6647	0.9163	0.9727
20	-1.6137	0.9058	0.9636	-1.5908	0.9029	0.9597
25	-1.5768	0.905	0.9523	-1.5146	0.887	0.9447
30	-1.6779	0.9164	0.9748	-1.6198	0.9028	0.9667

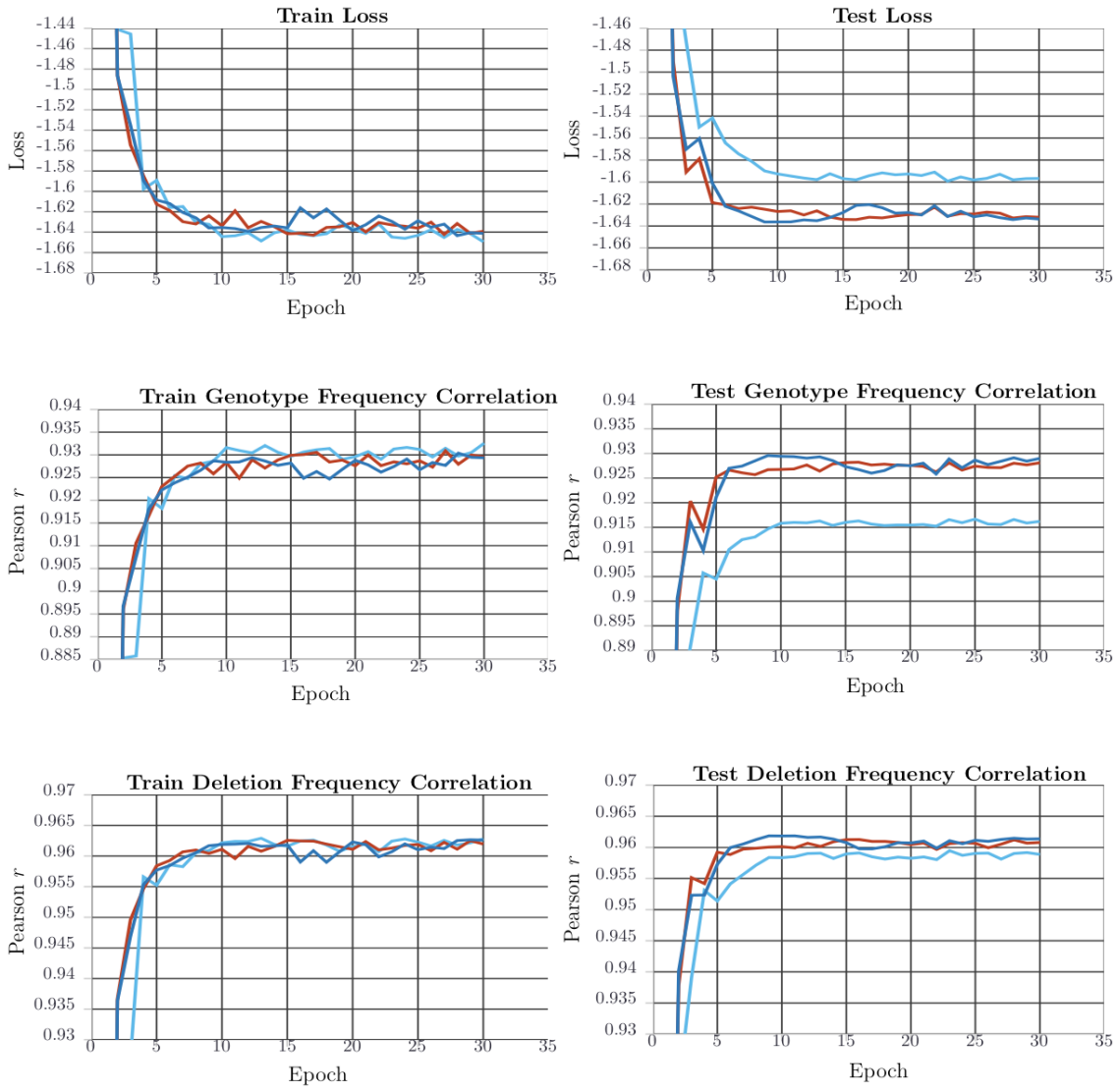


Figure 2.4: Loss and correlation values on top 10 frequencies of mESC data. Each colored line is a different run.

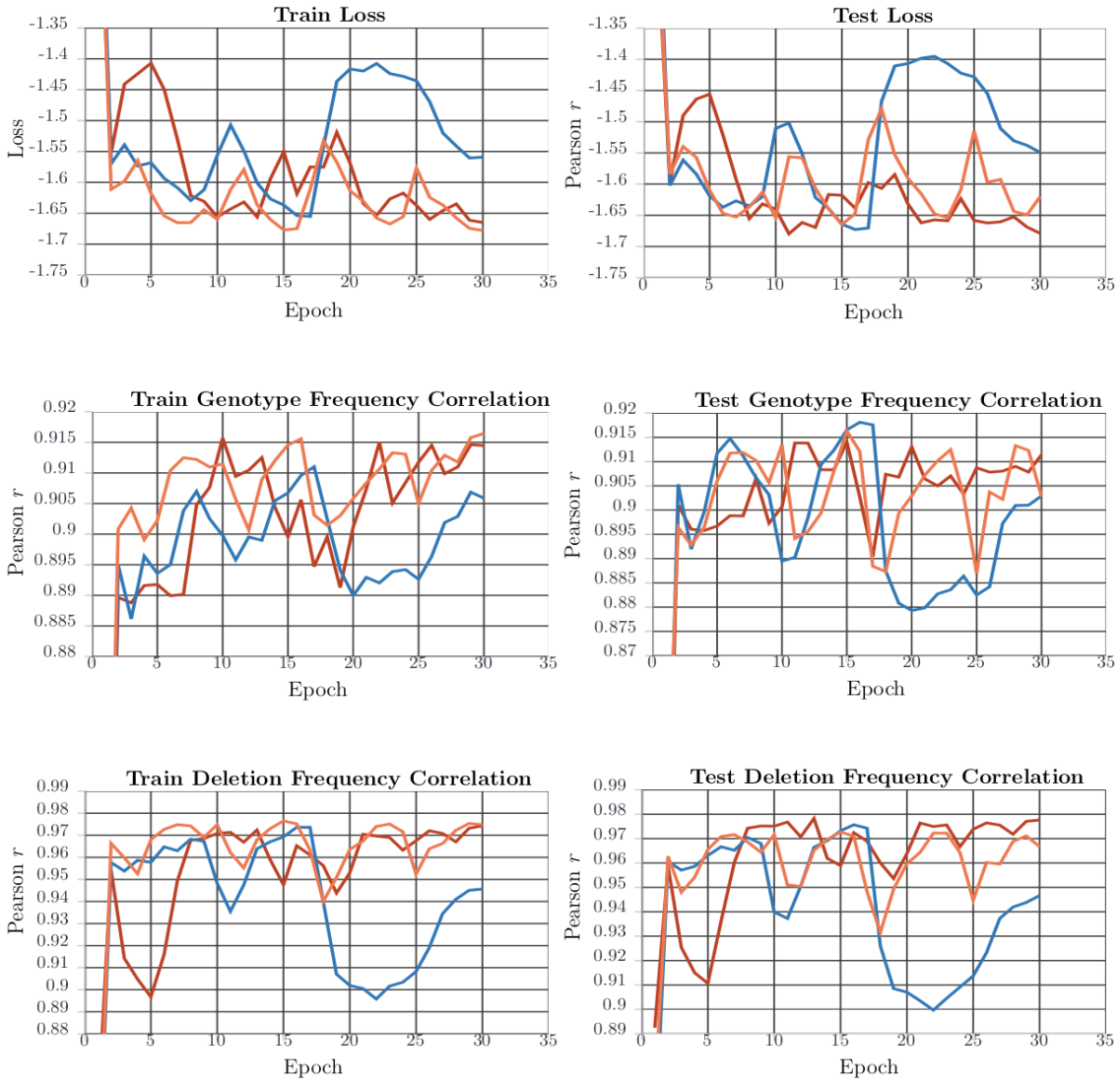


Figure 2.5: Loss and correlation values on top 3 frequencies of mESC data. Each colored line is a different run.

2.4.3 Random Frequencies

An additional way to modify the complete mESC data is to take random frequencies from each experiment. Situations like this may occur when data collection is sparse and incomplete. To simulate such a scenario, 5 random nonzero genotype frequencies from each experiment were taken from the complete mESC dataset. inDelphi was tested on this dataset and results showed that correlation values did become lower in comparison to the complete mESC dataset. A Pearson $r =$

0.8472 was observed for test deletion genotype frequency distributions and a Pearson $r = 0.9002$ was observed for test deletion length distributions (Table 2.6). This compares less favorably to the Pearson $r = 0.9518$ for the test deletion genotype frequency distributions and Pearson $r = 0.9496$ for the test deletion length frequency distributions in the complete dataset (Table 2.1). Although 5 random frequencies have lower correlation values than those from the complete dataset, the correlation values are still high and do show a significant amount of correlation still. Likely, there are similarities across all experiments. However, once again, each run in the 5 random frequencies dataset is quite different due to the inherent randomness in frequency selection (Figure 2.6).

Table 2.6: Average loss and correlation values on 5 random frequencies of mESC data.

Epoch	Train Loss	Train Gen Corr	Train Del Corr	Test Loss	Test Gen Corr	Test Del Corr
1	-0.9736	0.8011	0.7639	-1.0293	0.7897	0.8125
5	-1.3592	0.8559	0.9113	-1.3082	0.8452	0.8984
10	-1.3612	0.8562	0.911	-1.2937	0.8408	0.896
15	-1.3663	0.8585	0.9125	-1.31	0.8446	0.8996
20	-1.3673	0.8589	0.912	-1.3122	0.8454	0.8996
25	-1.3566	0.8547	0.9105	-1.307	0.8433	0.8996
30	-1.3672	0.8573	0.9133	-1.3176	0.8472	0.9002

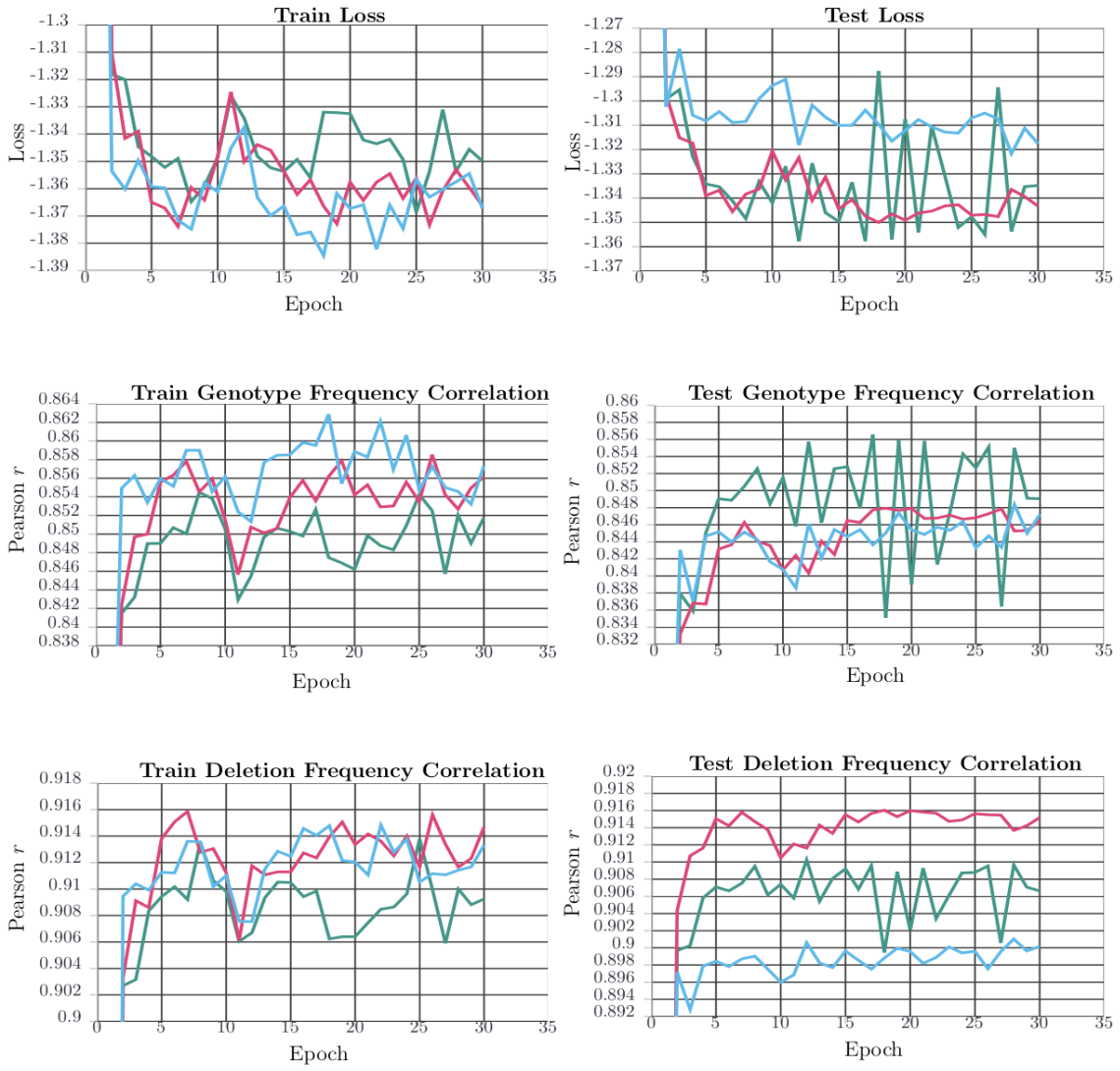


Figure 2.6: Loss and correlation values on 5 random frequencies of mESC data. Each colored line is a different run.

2.4.4 Equal Frequencies

The complete mESC data can once again be modified to create a dataset with top frequencies set to equal values. Situations like this may occur in scenarios where labels are not explicitly provided. To simulate such a scenario, the top 5 highest genotype frequencies from each experiment in the mESC dataset were all set to approximately 0.16. inDelphi was tested on this dataset and results showed that correlation values became significantly lower in comparison to those in the

complete mESC dataset. A Pearson $r = 0.6618$ was observed for test deletion genotype frequency distributions and a Pearson $r = 0.839$ was observed for test deletion length distributions (Table 2.7). This compares much less favorably to the Pearson $r = 0.9518$ for the test deletion genotype frequency distributions and Pearson $r = 0.9496$ for the test deletion length frequency distributions in the complete dataset (Table 2.1). From this dataset, it appears that genotype correlation suffers more than deletion correlation. However, in any case, it is obvious that learning in general suffers significantly without the true frequency information. Perhaps setting equal frequencies is too large an assumption to make.

Table 2.7: Average loss and correlation values on 5 equal frequencies of mESC data.

Epoch	Train Loss	Train Gen Corr	Train Del Corr	Test Loss	Test Gen Corr	Test Del Corr
1	-0.5841	0.6559	0.7406	-0.6134	0.6621	0.7543
5	-0.7674	0.652	0.8434	-0.7934	0.6596	0.8493
10	-0.8009	0.6523	0.8575	-0.8196	0.6596	0.8602
15	-0.7966	0.6536	0.8554	-0.8161	0.6594	0.8582
20	-0.7667	0.6573	0.8411	-0.7887	0.6596	0.8438
25	-0.7541	0.6621	0.8309	-0.7817	0.6628	0.8381
30	-0.7562	0.6613	0.8326	-0.7828	0.6618	0.839

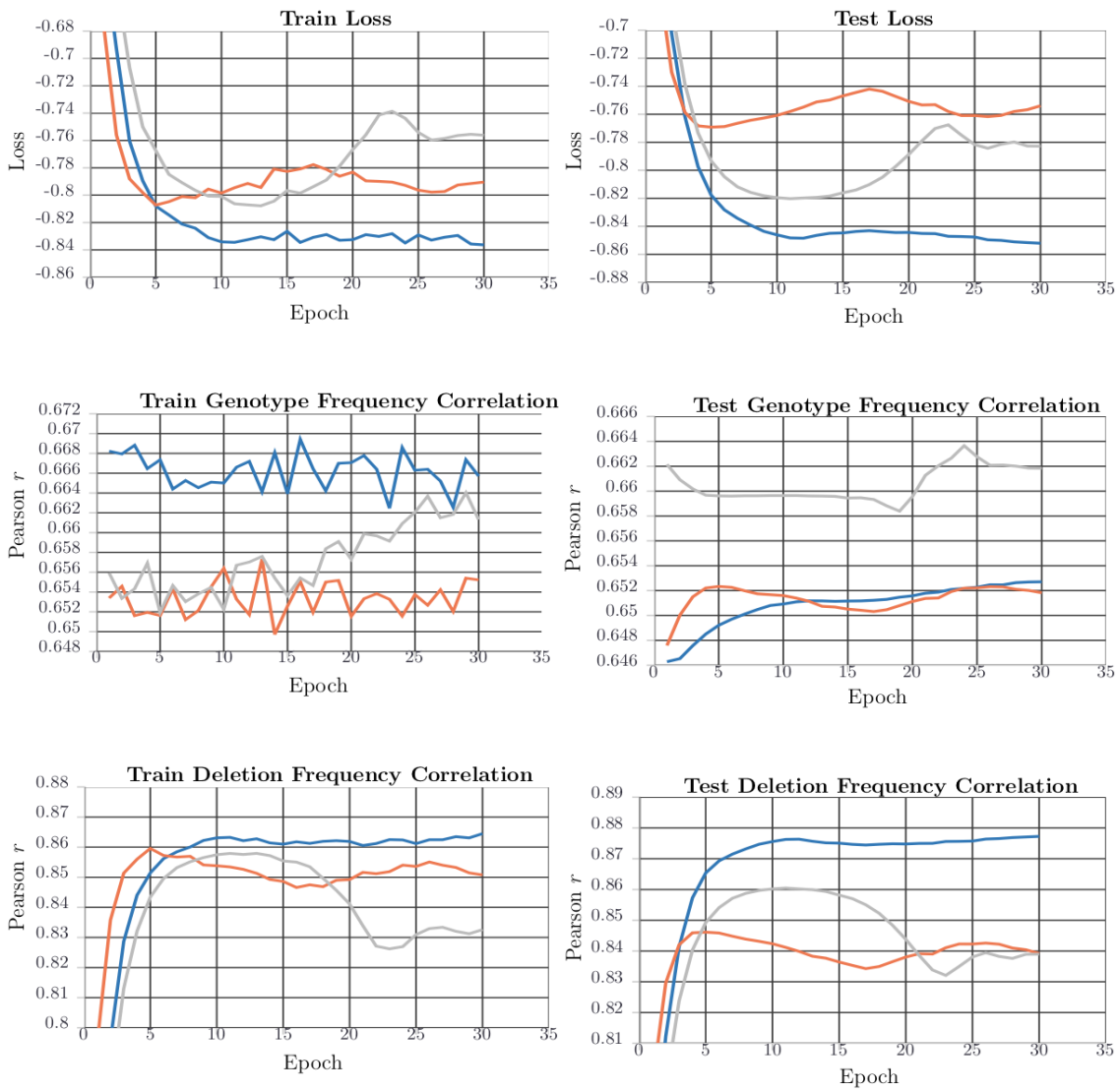


Figure 2.7: Loss and correlation values on 5 equal frequencies of mESC data. Each colored line is a different run.

2.5 Conclusions about the Neural Network

From the various datasets, it is obvious that random subsets and top frequencies of complete data do not affect correlation values significantly. Random selection of frequencies does result in some suffering of learning while equal frequencies result in significant suffering of learning. These results suggest that frequency plays an important role in the prediction of CRISPR products. To

remedy these negative effects, data imputation with machine learning methods will be implemented to create more complete datasets that can be successfully passed through the neural network with greater accuracy.

Chapter 3:

Methods and Results of Data Imputation

3.1 The Problem of Missing Labels

By the research presented in Chapter 2, it is quite evident that although DNA pathways in CRISPR-Cas9 mediated double strand breaks are highly stochastic and heterogeneous–comprising hundreds of repair genotypes, these events can still be predicted precisely and accurately. We have seen previously that a simple neural network structure as depicted in Fig.1.3 can predict editing outcomes with high accuracy and low error. Even with deficient data, such as a decreased number of experiments and a decreased number of outcomes per experiment, it is still possible to predict genome editing outcomes with sufficient accuracy and precision. However, such a feat is only possible through an extensive training of known outcomes. That is, we are able to predict a new distribution of genotype frequencies and deletion length frequencies by training on already known frequencies. In the context of machine learning, these distributions of genotype frequencies and deletion length frequencies are the desired labels. If a portion of these labels are removed, learning through the neural network suffers immensely. Up until now, the problem of predicting genotype and deletion length frequencies has been framed in a supervised learning framework. That is, our results are only accurate if sufficient training labels are provided. However, such a complete dataset is not always possible to obtain given the constraints of reality. In this way, we must resort to other solutions, namely improvement of the dataset itself, or more specifically, data imputation.

Although all labels may not be present in a given dataset, there is still valuable structure in the feature values of those missing labels. Therefore, it is far more sensible to use this unlabeled data in combination with various machine learning methods to give estimates of these missing labels, rather than discard all of these unlabeled observations altogether.

In this environment, various methods will be tested using the inDelphi mES dataset–modified to imitate circumstances of partially unlabeled data. Two main datasets will be used in the following methods. One dataset uses the mES data by randomly removing the frequency labels for half of the observations in each experiment. We call this *Dataset 1*. The other dataset further diminishes the labeled frequencies by first using only a random half of the mES data, and then for that half, removes all frequency values for a random 30 percent of the remaining experiments, and removes one third of the frequency values per experiment for another random 40 percent of the remaining experiments. We call this *Dataset 2*. In this way, we may simulate a more reasonably sized and realistic dataset.

3.2 Data Imputation Using Means

Neural networks are structures that work best with large and complete amounts of data. With such abundant missing labels, we must discard all such unlabeled training observations since we cannot minimize a loss without a label. We have previously seen that passing data without known frequency labels into our model in Fig. 1.3 results in a great suffering of learning. Thus, we must make use of the structure of such unlabeled data through a process of data imputation so that we may effectively pass the data through our neural network model in Fig. 1.3 appropriately. Data imputation refers to the process of replacing missing data with substituted values. As a simple, baseline comparison, for all missing frequency values, we take the mean of all known frequency values with the same feature values, and replace those missing values with the given mean.

3.2.1 Dataset 1

Using Dataset 1, imputation using mean values was performed. The mean squared error between the original and imputed values was 0.00123. Now, we compare performance in the neural network for Dataset 1 using imputed frequency values versus true frequency values. The progression of genotype and deletion frequency correlation values for the imputed frequencies and true frequencies are listed in Tables 3.1 and 3.2 respectively. Fig. 3.1 graphically represents the information presented in Tables 3.1 and 3.2, where the top blue line represents the performance of the original true frequency values on Dataset 1 and the bottom red line represents the performance of the imputed frequency values on Dataset 1. It is obvious that the performance is far better when the true frequencies are used. However, correlation is still present when using imputed values, suggesting that the neural network still can learn from a basic method of missing label substitution.

Table 3.1: Correlation values per epoch using mean imputation on Dataset 1.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.629	0.5531	0.6474	0.5758
1	10.0	0.7582	0.7667	0.7521	0.7651
2	20.0	0.7593	0.7688	0.753	0.7668
3	30.0	0.7584	0.7691	0.7514	0.7666

Table 3.2: Correlation values per epoch using true values on Dataset 1.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.7018	0.6282	0.72	0.6486
1	10.0	0.9464	0.9506	0.9461	0.949
2	20.0	0.9478	0.9518	0.9478	0.9505
3	30.0	0.9475	0.9513	0.9484	0.951

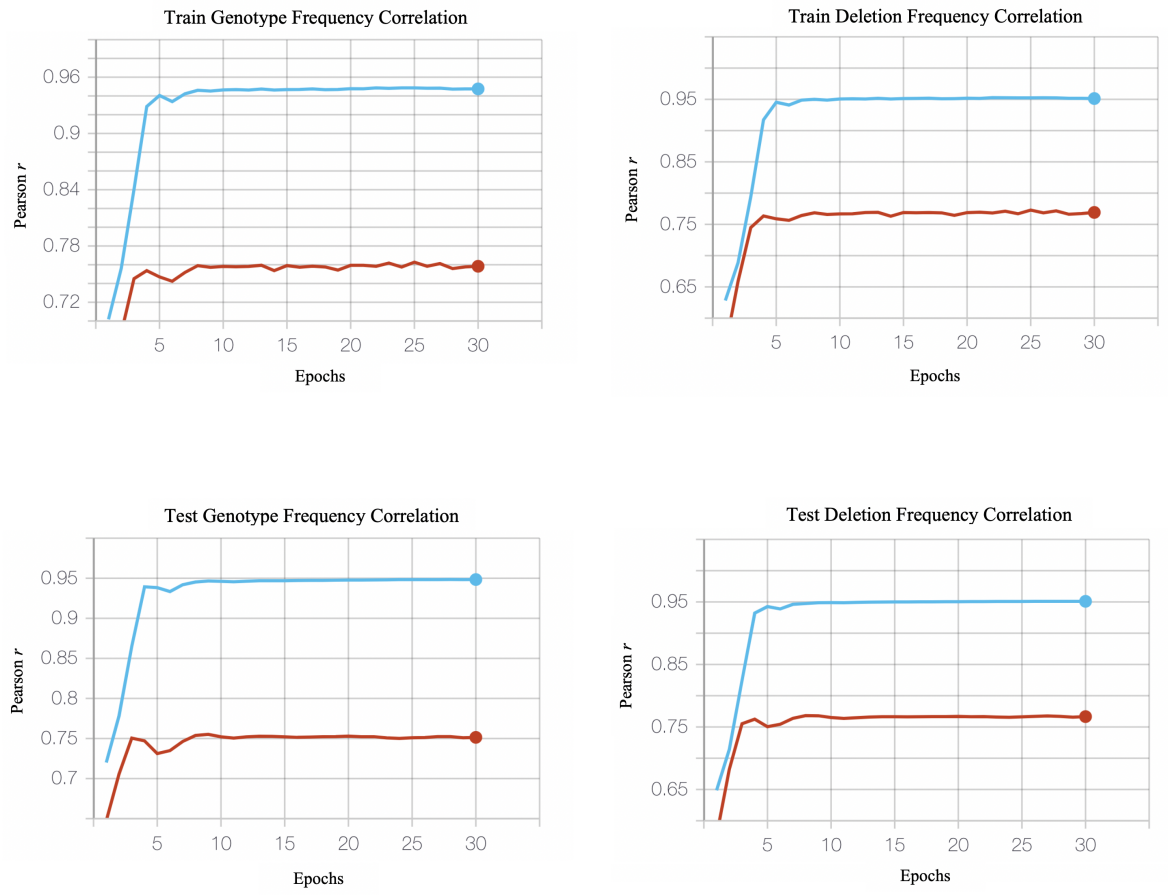


Figure 3.1: Genotype and deletion frequency correlation values for Dataset 1. The blue line represents the performance of the true frequency values while the red line represents the performance with mean imputed frequency values.

3.2.2 Dataset 2

Imputation using mean values was once again performed using Dataset 2. Dataset 2 is much smaller than Dataset 1. Essentially, Dataset 2 differs from Dataset 1 in the sense of having far fewer experiments with many more missing frequency labels. In order to impute accurate labels to Dataset 2 which are able to learn patterns that result in high genotype and deletion frequency correlation values when passed through the neural network, the imputation algorithm must be extremely robust. While using mean values to impute missing labels is a sensible option, it is still quite a basic algorithm that lacks in sophistication. For Dataset 2, when the missing labels were imputed using means, the mean squared error between the original and imputed values was

again 0.00123. However, when using these imputed values as data to pass through the neural network, gradients vanished, producing non-applicable frequency values. Thus, it is obvious that mean imputation does not perform as robustly as desired, especially on smaller datasets with many missing labels.

3.3 K-Nearest Neighbors Regression

Now we try imputing the missing labels using k-Nearest Neighbors Regression (kNN). kNN is a nonparametric method that can be used for regression. The neighbors are taken from a set of objects for which the object property value, or label, is known. For an unknown label, the value is the average of the k nearest neighbors, with ”near” being a distance metric of choice. In our case, $k = 3$ and the distance metric used is the mean squared difference on features for which two rows both have observed data. Using various k values result in roughly similar results.

3.3.1 Dataset 1

Dataset 1 is again used to impute missing frequency labels using kNN. The mean squared error between the original and imputed values is 0.00162. The progression of genotype and deletion frequency correlation values for the imputed frequencies and true frequencies are again listed in Tables 3.3 and 3.4 respectively. Fig. 3.2 graphically represents the information presented in Tables 3.3 and 3.4, where the top gray line represents the performance of the original true frequency values on Dataset 1 and the bottom green line represents the performance of the imputed frequency values on Dataset 1.

Table 3.3: Correlation values per epoch using kNN imputation on Dataset 1.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.5984	0.5493	0.6004	0.5554
1	10.0	0.7638	0.7682	0.7345	0.7415
2	20.0	0.7697	0.7723	0.7449	0.749
3	30.0	0.7704	0.7745	0.7463	0.7509

Table 3.4: Correlation values per epoch using true values on Dataset 1.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.7395	0.7093	0.7566	0.7328
1	10.0	0.9424	0.9465	0.9467	0.9509
2	20.0	0.9462	0.9503	0.9521	0.9557
3	30.0	0.945	0.9492	0.9535	0.9571

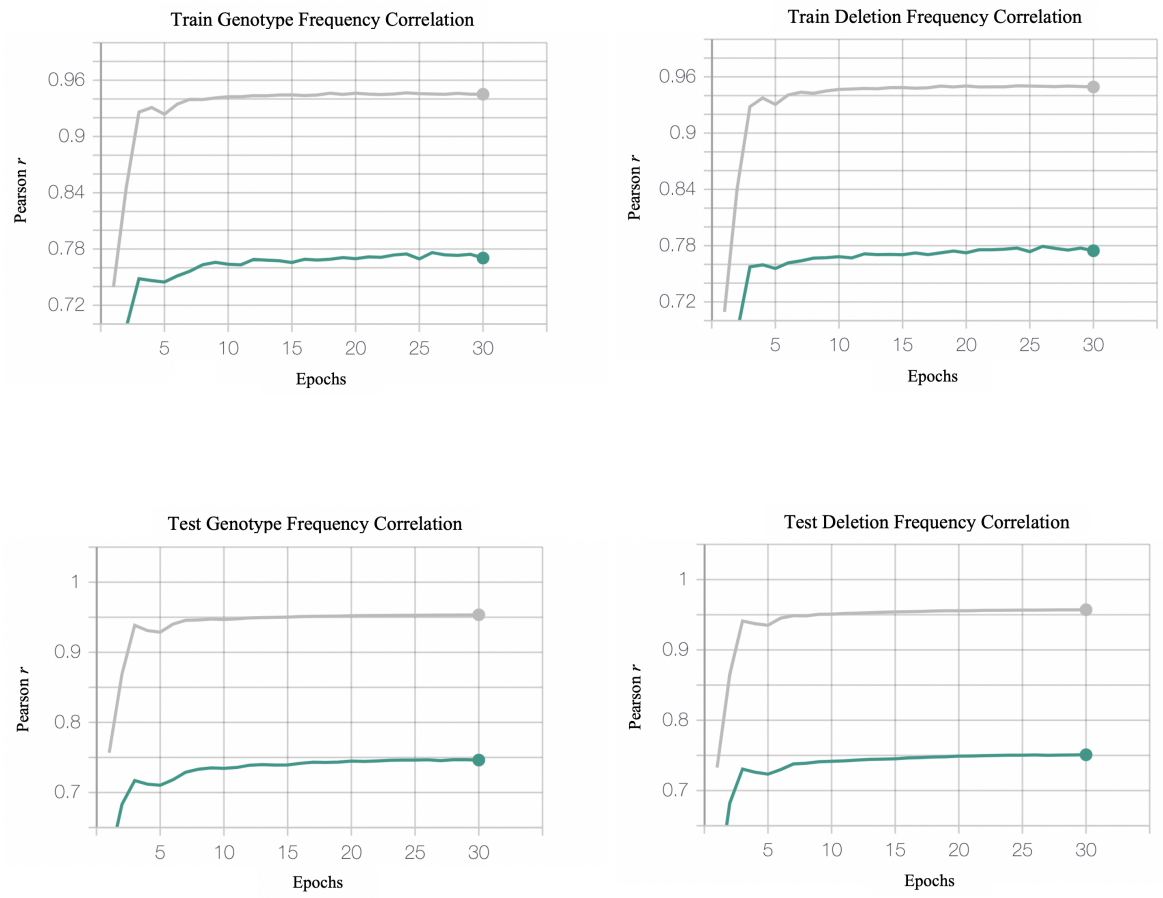


Figure 3.2: Genotype and deletion frequency correlation values for Dataset 1. The gray line represents the performance of the true frequency values while the green line represents the performance with kNN imputed frequency values.

3.3.2 Dataset 2

The results of a similar analysis using kNN imputation for Dataset 2 are shown in Tables 3.5 and 3.6 and Fig. 3.3 below. The mean squared error between the original and imputed values is 0.00154.

Table 3.5: Correlation values per epoch using kNN imputation on Dataset 2.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.5859	0.5139	0.5598	0.4754
1	10.0	0.7462	0.7267	0.7622	0.7356
2	20.0	0.7572	0.7672	0.7658	0.7713
3	30.0	0.7547	0.7657	0.7675	0.7744

Table 3.6: Correlation values per epoch using true values on Dataset 2.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.7209	0.667	0.7438	0.7012
1	10.0	0.9476	0.9445	0.9475	0.9433
2	20.0	0.9483	0.95	0.9474	0.9481
3	30.0	0.947	0.9505	0.9482	0.951

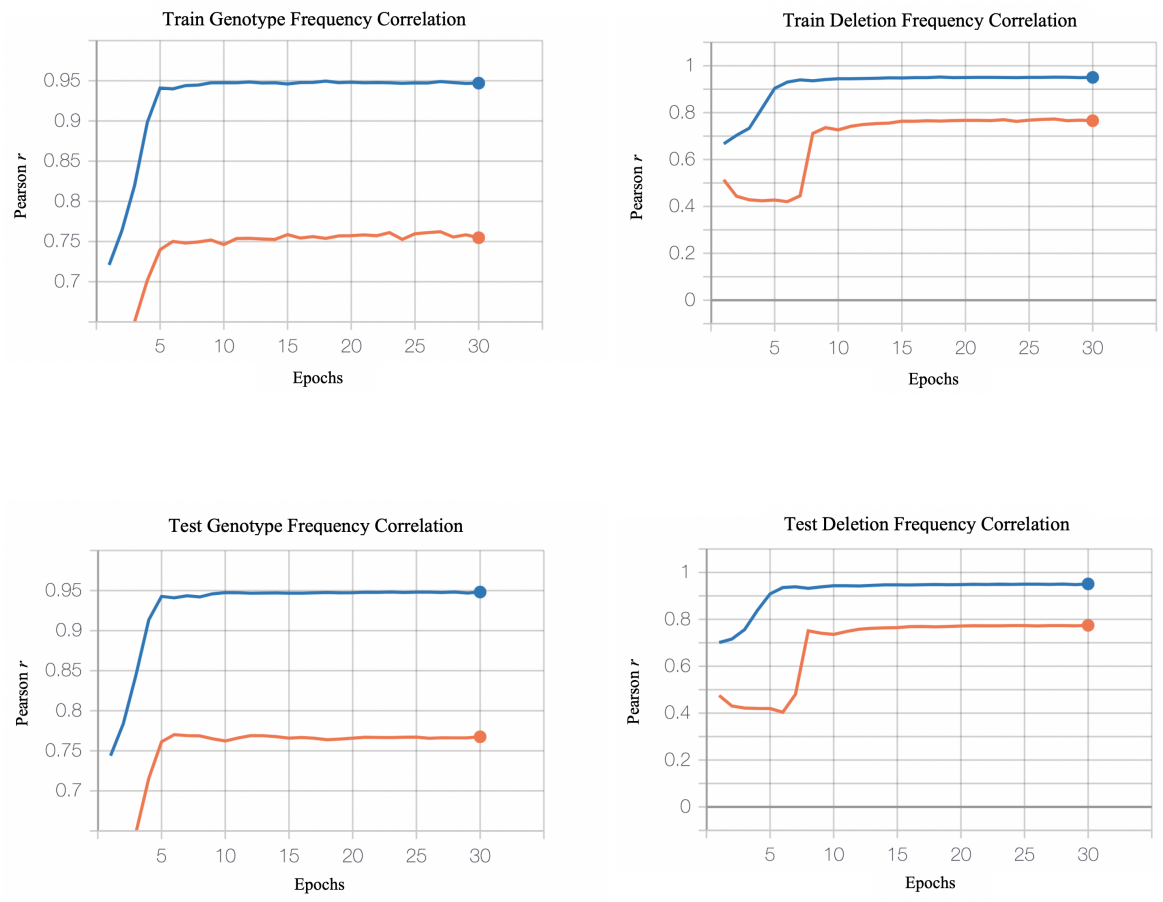


Figure 3.3: Genotype and deletion frequency correlation values for Dataset 2. The blue line represents the performance of the true frequency values while the orange line represents the performance with kNN imputed frequency values.

By observing the sets of tables and figures associated with kNN imputation, we can see that performance is quite similar to that of mean imputation. The results still display clear correlation, but not to an extremely high degree. This is likely due to the low amount of features that describe our data. If we recall, genotype frequency correlation is only described by two features– microhomology length and GC fraction. Thus, there is bound to be some feature value overlaps, which may obscure some of the complex patterns in the data.

3.4 Multiple Imputation by Chained Equations (MICE)

Another method to impute missing labels is multiple imputation by chained equations (MICE). MICE operates under the assumption that the given variable(s) used in the imputation procedure, in our case, the frequency values, are missing at random–which means that the probability that a value is missing depends only on observed values and not on unobserved values. In MICE, a series of regression models are run where the variable with missing data is modeled conditional upon the other variables in the data. A brief outline of MICE is listed below.

Steps:

1.) A simple imputation, such as imputing the mean, is performed for every missing value in the dataset. These mean imputations can be thought of as ”placeholders.”

2.) The ”placeholder” mean imputations for a dependent variable (in our case, we only have one dependent variable–genotype frequency) are set back to missing.

3.) The values from Step 2 are regressed on the other variables in the imputation model (microhomology length and GC fraction).

4.) The missing values for the variable in Step 2 are replaced with predictions from the regression (in our case, linear regression) model. If there are additional features with missing values, then the variable from Step 2 is subsequently used as an independent variable in the regression model for those features (in our case, we have no additional features with missing values).

5.) Steps 2-4 are repeated for a number of cycles, with the imputations being updated at each cycle. Note that in each iteration, the imputed values from the previous iteration are subsequently used as the ”placeholder.” The idea is that by the end of the cycles, the distribution of the parameters governing the imputations (e.g., the coefficients in the regression models) should have converged in the sense of becoming stable. This will, for example, avoid dependence on the order in which the values were imputed.

3.4.1 Dataset 1

The results of using MICE for Dataset 1 are shown in Tables 3.7 and 3.8 and Fig. 3.4 below. The mean squared error between the original data and MICE data values is 0.00101. We can observe that final genotype and deletion correlation values are quite high compared to previous methods–around 0.85 for both correlation frequency types. We even observed that for the first ten or so epochs, deletion frequency correlation was higher for the MICE frequency values in comparison to the true frequency values–although correlations did eventually stabilize, with the true frequency values still reaching a higher correlation.

Table 3.7: Correlation values per epoch using MICE on Dataset 1.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.556	0.4626	0.5805	0.4915
1	10.0	0.841	0.8356	0.8381	0.839
2	20.0	0.8457	0.8397	0.8456	0.8465
3	30.0	0.8471	0.8413	0.846	0.8475

Table 3.8: Correlation values per epoch using true values on Dataset 1.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.7414	0.5473	0.7467	0.5404
1	10.0	0.9479	0.4386	0.9449	0.4425
2	20.0	0.9495	0.9506	0.9452	0.9472
3	30.0	0.9485	0.952	0.9444	0.949

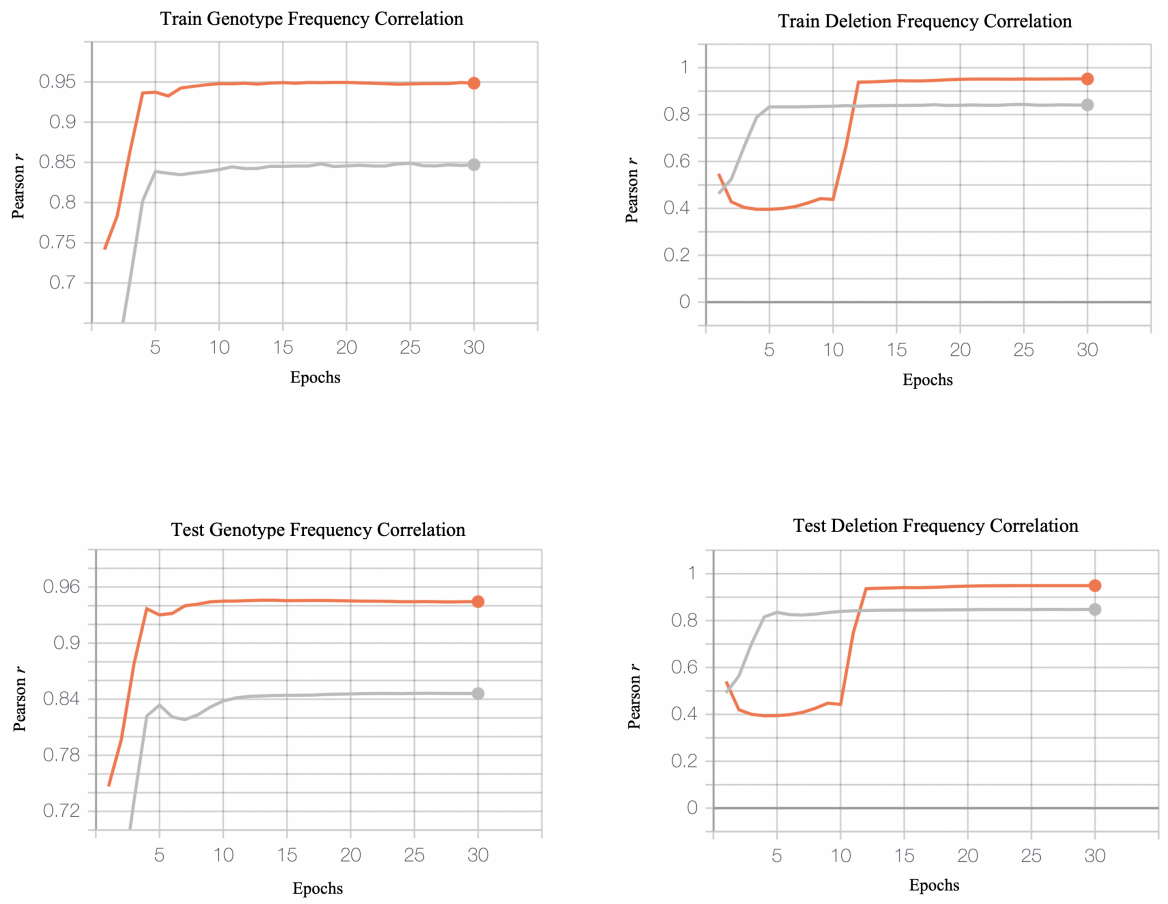


Figure 3.4: Genotype and deletion frequency correlation values for Dataset 1. The orange line represents the performance of the true frequency values while the gray line represents the performance with MICE imputed frequency values.

3.4.2 Dataset 2

The results of using MICE for Dataset 2 are shown in Tables 3.9 and 3.10 and Fig. 3.5 below. The mean squared error between the original and MICE frequency values is 0.00101 as well.The genotype and deletion frequency final correlation values are still quite high for Dataset 2, although not quite as high as for Dataset 1. However, this slight decrease in correlation values is certainly expected given the large decrease in observation size and extreme increase in the lack of frequency labels in the dataset. All frequency correlation values stabilize at around 0.83, which still represents quite a high degree of correlation.

Table 3.9: Correlation values per epoch using MICE on Dataset 2.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.6304	0.6234	0.6916	0.6778
1	10.0	0.8305	0.8249	0.837	0.8325
2	20.0	0.8391	0.8312	0.8392	0.8307
3	30.0	0.8337	0.8224	0.8391	0.8313

Table 3.10: Correlation values per epoch using true values on Dataset 2.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.707	0.6255	0.7147	0.6364
1	10.0	0.9432	0.9465	0.9432	0.9433
2	20.0	0.9455	0.9506	0.9444	0.9461
3	30.0	0.9465	0.9508	0.9443	0.9461

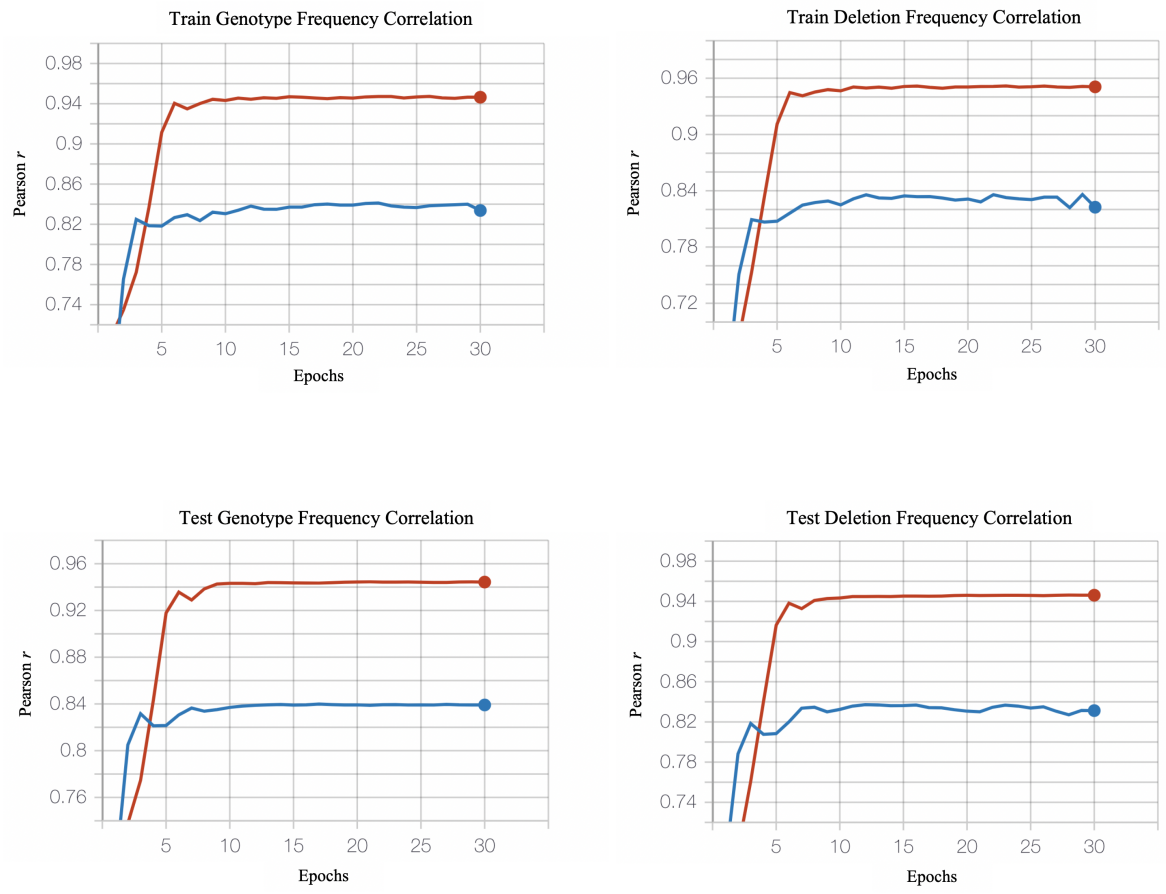


Figure 3.5: Genotype and deletion frequency correlation values for Dataset 2. The red line represents the performance of the true frequency values while the blue line represents the performance with MICE imputed frequency values.

It is important to note that MICE does operate under the assumption that frequency values are missing at random. Given that incompletely labeled datasets were simulated from completely labeled datasets with frequency labels removed at random, the datasets mentioned above do follow this assumption. However, given datasets with labels missing in a nonrandom manner, then it is possible that MICE may perform at a slightly compromised level.

3.5 Matrix Completion

We now introduce the matrix completion problem, where the aim is to estimate a large data matrix for which only a subset of its entries is observed. Given the amount of missing entries in

our datasets, our problem fits well into the scope of this problem.

In order to find a solution to the matrix completion problem, a low rank-matrix must be recovered from the given matrix with incomplete values. Intuitively, we want to come up with concise vector representations of our data such that if we want to find the value of an incomplete entry (frequencies in our case), we may take a dot product of these concise vectors. Since these computations must be handled efficiently, these concise vectors that represent our data must have as few dimensions as possible–hence having low rank. So ultimately, our incomplete matrix must be decomposed into a low rank matrix factorization representing the mentioned concise vectors such that we can perform simple dot products of these vectors to obtain missing values and hence recover a completed matrix also of resulting low rank. This process is called rank minimization.

In practice, rank minimization is not a simple task to perform. Therefore, alternative functions are often minimized instead of the rank, which still recover a low-rank data matrix. One such alternative is the minimization of the nuclear norm regularized least-squares equation. Mathematically, we must solve the equation

$$\min_{\boldsymbol{X}} \frac{1}{2}||\mathcal{A}(\boldsymbol{X}) - \boldsymbol{b}||_2^2 + \lambda||\boldsymbol{X}||_*$$

where λ is a positive scalar (given parameter), $\mathbf{X}$ is our incomplete data matrix, $\mathcal{A}$ is a linear mapping, $\boldsymbol{b}$ is a vector of measurements in the range of the linear map, and $||\boldsymbol{X}||_*$ is the nuclear norm defined as $||\boldsymbol{X}||_* = \sum_i \sigma_i(\boldsymbol{X})$, where $\sigma_i(\boldsymbol{X})$'s denote the singular values of $\boldsymbol{X}$. The nuclear norm can be thought of intuitively as the best convex approximation of the rank function over the unit ball of matrices.

This leads us to a sophisticated algorithm to solve the matrix completion problem called iterative soft thresholding of singular value decompositions (SVDs). Such an algorithm works by producing a sequence of solutions that converges to a solution of the nuclear norm regularized least-squares minimization problem when the number of iterations approaches infinity. It can be noted that iterative thresholding algorithms are extremely beneficial for their efficiency and scalability to large matrices.

3.5.1 Dataset 1

The results of this matrix completion method performed on Dataset 1 are shown in Tables 3.11 and 3.12 and Fig. 3.6. The mean squared error between the original frequency values and the matrix completed frequency values is 0.00105. We can see that this matrix completion method produces favorable results similar to that of MICE when performed on Dataset 1. Both have final correlation values all around 0.85. However, since this matrix completion method does still operate without the assumption that the frequency values are missing at random, such a method may be favorable in these cases.

Table 3.11: Correlation values per epoch using Matrix Completion on Dataset 1.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.6079	0.5453	0.6251	0.5825
1	10.0	0.8277	0.8159	0.8513	0.8457
2	20.0	0.8312	0.8204	0.853	0.8479
3	30.0	0.8293	0.8195	0.8535	0.8487

Table 3.12: Correlation values per epoch using true values on Dataset 1.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.7216	0.6722	0.7371	0.6974
1	10.0	0.9441	0.9478	0.9453	0.9525
2	20.0	0.9464	0.9497	0.9457	0.9536
3	30.0	0.9474	0.951	0.9462	0.9541

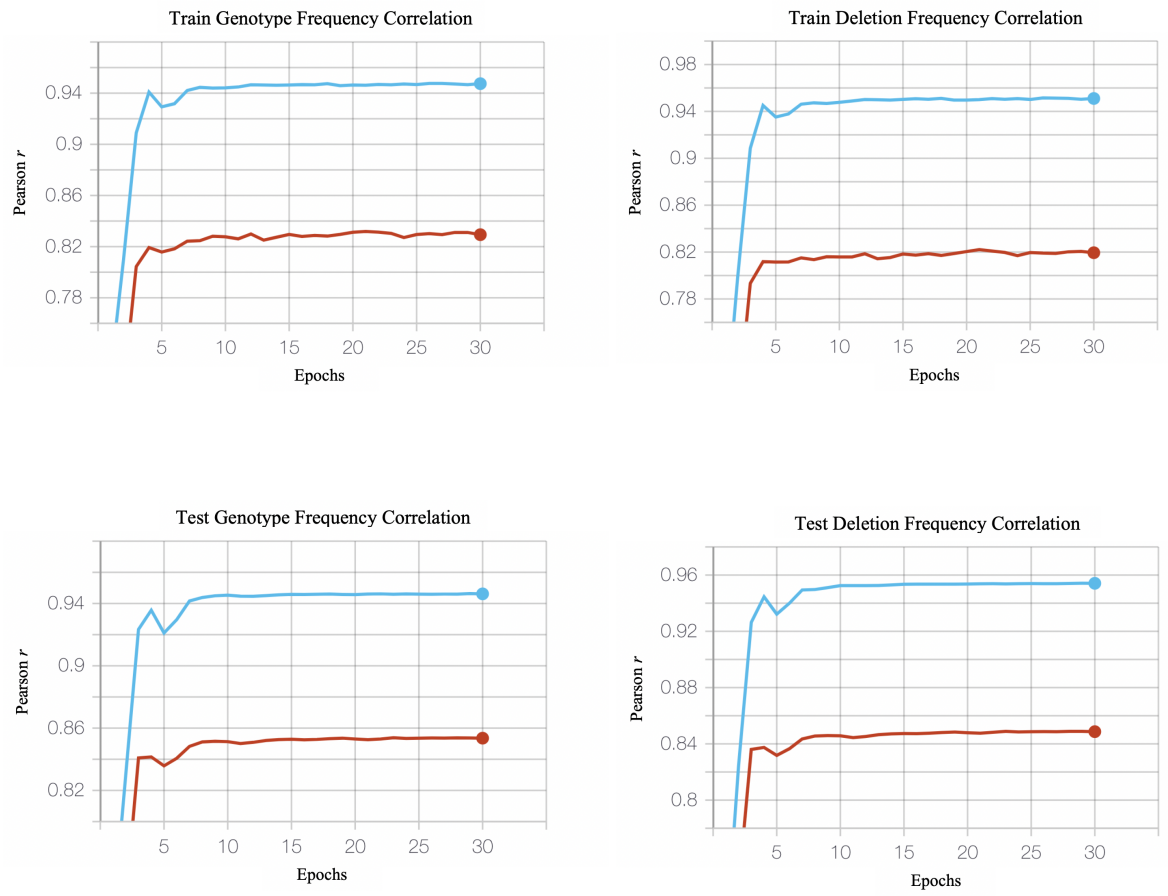


Figure 3.6: Genotype and deletion frequency correlation values for Dataset 1. The blue line represents the performance of the true frequency values while the red line represents the performance with matrix completed frequency values.

3.5.2 Dataset 2

The results of this matrix completion method performed on Dataset 2 are shown in Tables 3.13 and 3.14 and Fig. 3.7. The mean squared error between the original frequency values and the matrix completed frequency values is again 0.00105. This time, however, final correlation values are slightly lower. The test genotype and deletion frequency correlations are approximately 0.82 and 0.79 respectively. Certainly these results are still better than those of mean imputation and kNN imputation, but choosing between matrix completion and MICE for datasets similar to Dataset 2 will require further examination of the underlying data assumptions.

Table 3.13: Correlation values per epoch using Matrix Completion on Dataset 2.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.5837	0.5479	0.5712	0.5518
1	10.0	0.8175	0.7954	0.8067	0.7767
2	20.0	0.8237	0.7999	0.8196	0.7889
3	30.0	0.8251	0.8007	0.8185	0.7872

Table 3.14: Correlation values per epoch using true values on Dataset 2.

	Epoch	Train Gen Corr	Train Del Corr	Test Gen Corr	Test Del Corr
0	1.0	0.7643	0.6842	0.7929	0.7147
1	10.0	0.9453	0.9495	0.9444	0.9484
2	20.0	0.9468	0.9506	0.9485	0.952
3	30.0	0.9467	0.9502	0.9484	0.9521

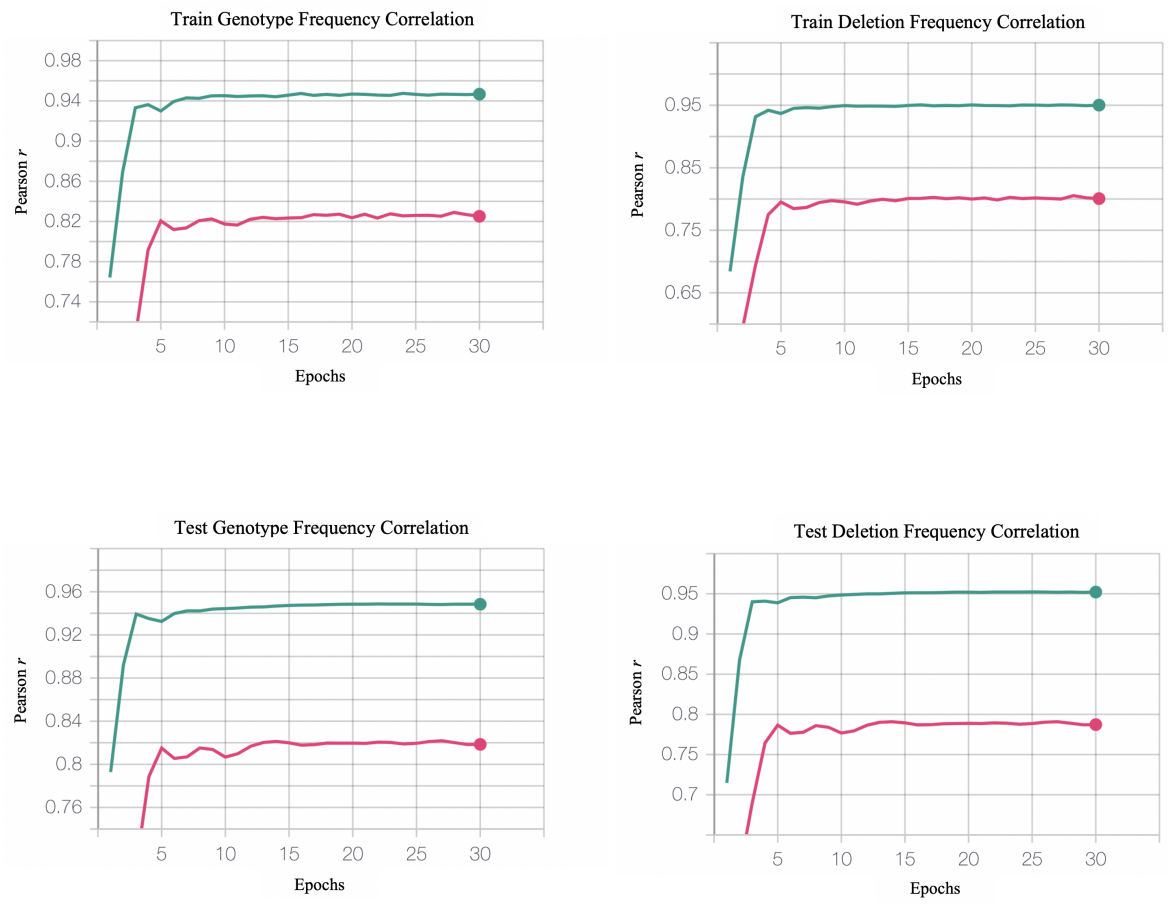


Figure 3.7: Genotype and deletion frequency correlation values for Dataset 2. The green line represents the performance of the true frequency values while the pink line represents the performance with matrix completed frequency values.

3.6 Conclusions about Data Imputation

Listed below is a summary of the final test genotype and deletion frequency correlation values for all mentioned label estimation techniques after being passed through the supervised neural network.

Table 3.15: Final correlation values using Dataset 1.

	Final Test Gen Freq Corr	Final Test Del Freq Corr
Mean Imputation	0.7514	0.7666
kNN Imputation	0.7463	0.7509
MICE	0.846	0.8475
Matrix Completion	0.8535	0.8487

Table 3.16: Final correlation values using Dataset 2.

	Final Test Gen Freq Corr	Final Test Del Freq Corr
Mean Imputation	n/a	n/a
kNN Imputation	0.7675	0.7744
MICE	0.8391	0.8313
Matrix Completion	0.8185	0.7872

From the tables above, we can see that mean and kNN imputation are less desirable methods of label estimation before passing the data through a supervised neural network. While kNN may have possible improvements with additional features added to better describe the data in higher dimension, it is unlikely that mean imputation will see much improvement with a change in dataset structure. Clearly, MICE and Matrix Completion are the most promising methods. MICE gives good results under the assumption that labels are missing at random. MICE also generalizes quite well to datasets of smaller size with an increased amount of missing labels. Matrix Completion works well for large matrices especially and is desirable for its efficiency in computation and relaxed underlying data assumptions. Matrix Completion does however generalize slightly less well to datasets of smaller size. Both MICE and Matrix Completion also allow the capacity for missing values in multiple features. Thus, we can conclude that MICE and Matrix Completion are both robust methods that generate promising results.

We see through the various methods presented in this paper that although frequency information is vital to producing high genotype and frequency correlation values by passage through the supervised neural network, even if a large amount of labels are missing, then those values can still be recovered accurately enough to produce low mean squared errors and sufficiently high genotype

and deletion frequency correlations. Therefore, if restraints are put upon the completeness of data we may obtain, then it is still possible to recover this data in a meaningful and accurate way given that the data is well understood. We may indeed still obtain promising results without frequency information present in full.

Chapter 4:

Conclusions

Ultimately, the neural network proves to be a powerful machine learning model in predicting complex patterns in genome editing when complete, fully labeled datasets are provided. Also, the neural network is still robust to operations on subsets of complete data. That is, taking small random subsets or using only high frequency observations still result in accurate prediction values. However, its degraded performance on unlabeled, missing data suggests that the neural network operates best as a supervised model using complete datasets.

Given the time, money, and resource constraints in the real world, such complete datasets are often hard achieve. Most datasets are unlabeled and deficient to some extent. This leads us to the conclusion that machine learning methods are often most beneficial when used together. In this case, we use the neural network and data imputation methods to remedy the negative effects in sparse and incomplete data. Through the most optimal data imputation methods previously mentioned, missing labels can be recovered accurately enough to produce low mean squared errors and sufficiently high genotype and deletion frequency correlations. That is, when data imputation methods and the neural network are used together, we obtain a powerful method in predicting genome editing outcomes on a wide variety of datasets.

Bibliography

[1] Brad Plumer, Eliza Barclay, Julia Belluz, and Umair Irfan. A simple guide to crispr, one of the biggest science stories of the decade, Dec 2018.

[2] Mary Gearing. Pitching mmej as an alternative route for gene editing, Feb 2016.

[3] Max W Shen, Mandana Arbab, Jonathan Y Hsu, Daniel Worstell, Sannie J Culbertson, Olga Krabbe, Christopher A Cassa, David R Liu, David K Gifford, and Richard I Sherwood. Predictable and precise template-free crispr editing of pathogenic variants. *Nature*, 563(7733):646–651, 2018.

[4] F Ann Ran, Patrick D Hsu, Jason Wright, Vineeta Agarwala, David A Scott, and Feng Zhang. Genome engineering using the crispr-cas9 system. *Nature protocols*, 8(11):2281, 2013.

[5] Howard HY Chang, Nicholas R Pannunzio, Noritaka Adachi, and Michael R Lieber. Nonhomologous dna end joining and alternative pathways to double-strand break repair. *Nature reviews Molecular cell biology*, 18(8):495, 2017.

[6] Ryan T Leenay, Amirali Aghazadeh, Joseph Hiatt, David Tse, Judd Hulquist, Nevan Krogan, Zhenqin Wu, Alexander Marson, Andrew P May, and James Zou. Systematic characterization of genome editing in primary t cells reveals proximal genomic insertions and enables machine learning prediction of crispr-cas9 dna repair outcomes. *bioRxiv*, page 404947, 2018.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher,

M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Felicity Allen, Luca Crepaldi, Clara Alsinet, Alexander J Strong, Vitalii Kleshchevnikov, Pietro De Angeli, Petra Páleníková, Anton Khodak, Vladimir Kiselev, Michael Kosicki, et al. Predicting the mutations generated by repair of cas9-induced double-strand breaks. *Nature biotechnology*, 37(1):64–72, 2019.

[9] J. Bruin. Multinomial logistic regression, February 2011.

[10] Chris Nicholson. A beginner's guide to neural networks and deep learning, 2019.

[11] Max Shen. indelphi-liba+libb-processed-indel-data. *Nature*, December 2018.

[12] Sangsu Bae, Jiyeon Kweon, Heon Seok Kim, and Jin-Soo Kim. Microhomology-based choice of cas9 nuclease target sites. *Nature methods*, 11(7):705–706, 2014.

Academic Vita

ANGELA TING

act5290@psu.edu

EDUCATION

The Pennsylvania State University | Schreyer Honors College ***Graduation: May 2020***
Bachelor of Science, Mathematics and Statistics

- Thesis: *Predicting CRISPR-Cas9 genome editing outcomes with machine learning methods.*
- Supervisor: Qunhua Li.

RESEARCH EXPERIENCE

Research Assistant, Pennsylvania State University ***August 2018 – May 2020***

- Supervisor: Qunhua Li.
- Used neural networks to increase accuracy and efficiency in methods predicting CRISPR-Cas9 genome editing outcomes in humans.
- Tested various methods to deal with missing data.
- Developing new semi-supervised methods to effectively predict genome editing outcomes in plants.

Research Assistant, Pennsylvania State University ***May 2018 – January 2019***

- Supervisor: Dennis Pearl.
- Developed web applications using R, JavaScript, and HTML/CSS to estimate, model, and visualize statistical data while establishing education research protocols.

PRESENTATIONS

Using neural networks to predict CRISPR-Cas9 genome editing outcomes. Undergraduate Research Poster Exhibition, Pennsylvania State University, September 2019.

Book of Shiny Apps for Statistics Teaching. Undergraduate Research Poster Exhibition, Pennsylvania State University, September 2018.

AWARDS

Matthew Rosenshine Fund for Excellence in the Department of Statistics	2019-2020
Academic Excellence Scholarship	2016-2020
Eberly College of Science Undergraduate Research Grant	2018-2019

ACTIVITIES

Mu Sigma Rho Honorary Society	2019-2020
Statistics Club	2016-2020
Mathematics Club	2016-2020
Schreyer Honors College	2016-2020

SKILLS

- Python
- R
- SAS
- Tensorflow
- PyTorch
- JavaScript
- HTML/CSS
- LaTex