EVALUATION OF KNOWLEDGE ENCODED IN DEEP NEURAL NETWORK IN THE IMPUTATION OF GENE EXPRESSION

BY

CHOW JENN PANG

A PROPOSAL

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology
(Perak Campus)

MAY 2018

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: _____
_____
_____

Academic Session: _____

I _____
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

_____ _____
(Author's signature) (Supervisor's signature)

Address:
_____
_____ _____
_____ Supervisor's name

Date: _____ **Date**: _____

EVALUATION OF KNOWLEDGE ENCODED IN DEEP NEURAL NETWORK IN THE IMPUTATION OF GENE EXPRESSION

BY

CHOW JENN PANG

A PROPOSAL

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology
(Perak Campus)

MAY 2018

DECLARATION OF ORIGINALITY

I declare that this report entitled "**EVALUATION OF KNOWLEDGE ENCODED IN DEEP NEURAL NETWORK IN THE IMPUTATION OF GENE EXPRESSION**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr. Ng Yen Kaow for giving me this opportunity to take up this project. I feel grateful that even I have lots of misunderstanding in the project, his still willing to lead me along the way and inspire me for new ideas to complete the project. Without his guidance, I would not able to complete this project.

Besides, I would also like to thanks my friend, Wing Khang, Lai, that have been cooperate with me all the way long in this project. I would not have complete this project without him. Not to forget Ling Xi, Cheng from City University of Hong Kong, who has been consistently providing opinion and feedback on this project, thanks for being so helpful and providing useful feedback.

Finally, I would like to express my gratitude to my family for their love and supports.

ABSTRACT

As deep learning technology are getting advanced and advanced, researchers have start to proposed to solve gene expression imputation problem with deep neural network. However, most of the time deep neural network remain a black box algorithm despite it may achieve outstanding result. Thus, in this project a probability model will be proposed and developed to investigate the features that have been learn by deep neural network that use for gene expression imputation.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

CNN	Convolutional Neural Network
DNA	Deoxyribonucleic Acid
GMC	Gaussian Mixture Clustering
KNN	K-nearest neighbor
NGS	Next Generation Sequencing
RNA	Ribonucleic Acid
RNN	Recurrent Neural Network
SVD	Single Value Decomposition
GANs	Generative Adversarial Network
ReLU	Rectified Linear Unit
MLP-SAE	Multilayer Perceptron Stacked Auto-Encoder
GO	Gene Ontology

CHAPTER 1. INTRODUCTION

1.1. Problem Statement

RNA (Ribonucleic Acid) is a very important component in living organism. One of its main role is to transcribe parts of DNA sequences into functional protein within the cells. Therefore, analysis of RNA would help us understand organism cell activity and for this reason, RNA expression profiling is under very active research.

In daily life, when an organism encounters a different environment or situation, its respective cell will produce corresponding protein to respond to the current situation. For example, human cell will actively produce alcohol dehydrogenase when ethanol concentration is increased in the human inner-body environment (Edenburg, 2007). Hence an expression profile that demonstrates an elevated RNA expression that corresponding to alcohol dehydrogenase may indicate that the subject was consuming alcoholic drink when the sample was taken. Therefore, RNA expression profiling is a useful technique to study the activity of gene under different conditions or environments. Some useful application of RNA expression profiling will be cancer classification, identification of gene that is useful in disease diagnosis and therapy, *etc*. (Liew *et al.*, 14 December 2010).

Researchers have developed many different methods to identify gene sequences, e.g. Sanger Sequencing, Shotgun Sequencing, Illumina (Solexa) sequencing, SOLiD Sequencing, *etc*. However, RNA sequence data retrieved from gene sequencing might sometimes suffer from missing value. When dealing with gene expression profiling, missing values are always a big challenge to solve, as many known methods to analyze RNA sequence data require a complete RNA sequence. Missing values in RNA sequence could adversely affect gene expression profiling output. Therefore, a good method to impute the missing gene expression values can be tremendously helpful to ensure the gene expression profiling analysis are accurate and precise, to allow better analysis. On the other hand, deep learning techniques are achieving outstanding results in several real world problem such as object detection and classification, computer vision and etc. Researchers have started to propose deep learning neural network to solve the problem of missing and incorrect gene expression value, where the proposed deep learning neural network will take in gene expression data from input, and produce imputed gene expression data as the output. However, even though these techniques

are getting advanced, deep learning neural network remain a black box algorithm to most researchers. Even through neural network may show outstanding results when it tried to solve missing and incorrect gene expression value problem, researchers might not understand how actually neural network impute them. Therefore in this project, we would like to investigate and analyze the performance of deep neural network used for gene expression imputation and uncover the underlying feature that deep neural network used for gene expression imputation.

1.2. Project Scope

In this project, a probability model will be proposed and developed to investigate the features that deep neural network learned when the deep neural network is trained to impute missing and incorrect gene expression value problem.

1.3. Project Objective and Contribution

This project aims to study the behavior of deep neural network in gene expression imputation. A probability model will be proposed and developed in order to investigate the features that deep neural network utilize in producing its outcome. The probability model will allow researchers to gain more understanding on how the neural network performs the gene expression imputation, and may help in improve future training of such networks.

1.5. Project Background

In living organism, the most important component inside living cell is the DNA. DNA resides inside cell nucleus, and is made up of four base, namely Adenine, Thymine, Cytosine, and Guanine. DNA stores all the information about the living organism, which makes them in charge in almost everything about living organisms, such as growth, development, reproduction, functioning, *etc*. One of the most important behaviors of DNA is to produce functional proteins to assist in cell activities. Whenever the cell is required to produce a new protein, the DNA inside the cell nucleus will be responsible in transforming the information in the cell into the required protein, through the processes of transcription and translation.

According to the Central Dogma of cell biology, the information required to produce new protein are passed from DNA to RNA, and RNA will move out of nucleus and assembly into new protein according to the information passed from the DNA. The process of producing

new protein from DNA to protein involves the transcription and translation of genes. Generally, in the process of producing a new protein, the first step is to transfer the information required to RNA. This process is known as transcription. In transcription, an enzyme named RNA polymerase will bind to the correct DNA, and it will moving from the beginning of the gene toward the end of gene. During the transcription process, a chain of RNA will be formed, as an RNA polymerase make a single step on the DNA nucleotide, a complement RNA base will be added on to the RNA sequence. The process will continue until the RNA polymerase reach the end part of the DNA. Then, the RNA polymerase will detach from the DNA, and the formed RNA sequence will be moved out from the nucleus and become ready to translate into a protein. After the RNA sequence moves out from the nucleus, the RNA sequence will be used to form a new protein, through the process of translation. In the process of translation, the nucleotide on RNA will be grouped into a group of three nucleotide, also known as triplet. The three nucleotide in a single group will be responsible to match with an amino acid. And traverse through down the RNA sequence, a sequence of amino acid will be formed up, and finally it will formed a new protein. At the last stage in translation, the RNA sequence is translated into a new protein. The process occur in the Central Dogma that produce protein also known as gene expression (Peter, et al., 2013).

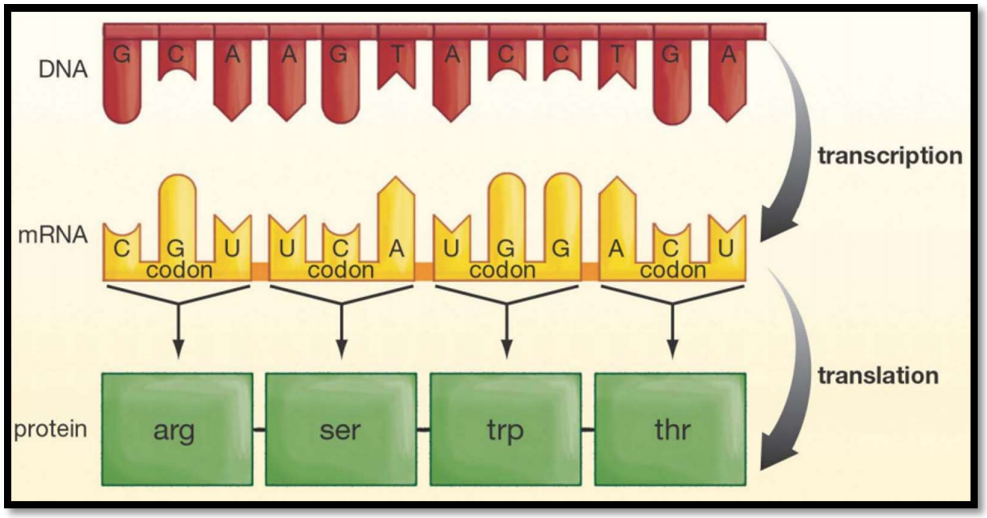


Figure 1: DNA transcription and translation. (Austin Community College, n.d.)

After understand the process of transcription and translation in organism cell, researchers came out with a new method to analyze cell activity using DNA and RNA, known as gene expression profiling. In gene expression profiling, several processes will be conducted

to analyze the gene expression. Firstly, the desired RNA will first be preprocessed through preparation, fragmentation, purification, amplification, and lastly sequencing to obtain the RNA sequence. Also, after obtaining the RNA sequence data, it may undergo further processing to enhance its quality before the next step. At present, the most often used sequencing technique is high throughput sequencing, also known as Next Generation Sequencing (NGS). In next generation sequencing, the long RNA will cleaved into multiple short sequence, and each sequence will then pass through machine to identify the RNA sequence independently. By doing so, it allows the sequencing of RNA to be parallelized, thus achieving fast, effective sequencing, and at the same time reducing the cost of sequencing. For example, in Illumina sequencing the long RNA will be cleaved into 100-150 base pair per read, and each read will be passed to a machine to identify the sequence (EMBL-EBI, 2018).

After the RNA sequence data is prepared well, the next step is to align and map the short read onto a reference genome. In this step, each short read obtained is aligned back onto reference genome in multiple location in the genome if the short read sequence is the same as genome sequence in the genome location.

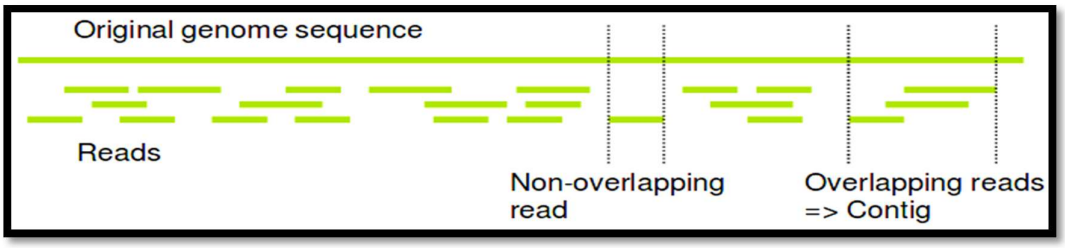


Figure 2: Illustration of RNA read alignment. (University of Helsinki, 2009)

Moving on, after the alignment has been done, the next step will be to count the number of RNA reads that are mapped to a particular gene. This process is also known as gene expression profiling. A gene expression value is basically just the number of RNA read on a particular gene. When a gene is expressed actively, more RNA sequence will be produced from that gene, hence the number of reads would increase. This is how researchers can use gene expression profiling to detect the gene activity inside a cell. After obtaining the count value, normalization could be done on the gene expression value. Normalization is usually performed to modify the gene expressions into values relative to each other's.

After the gene expression values are obtain, others analysis could take place to analyze the RNA sequence. For example, one of the most common analyses will be differential gene

expression. In this analysis, the gene expression value from the same genomic sequence are collected under different condition, such as under normal condition and under therapy condition. After the gene expression value are collected from different conditions, the difference between the gene expression value could be determine, so that researchers could analyze which gene are relatively active when the patient in under therapy, and researchers could extract the active gene and analyze the functionality of that gene toward patient disease (Han *et al.*, 2015).

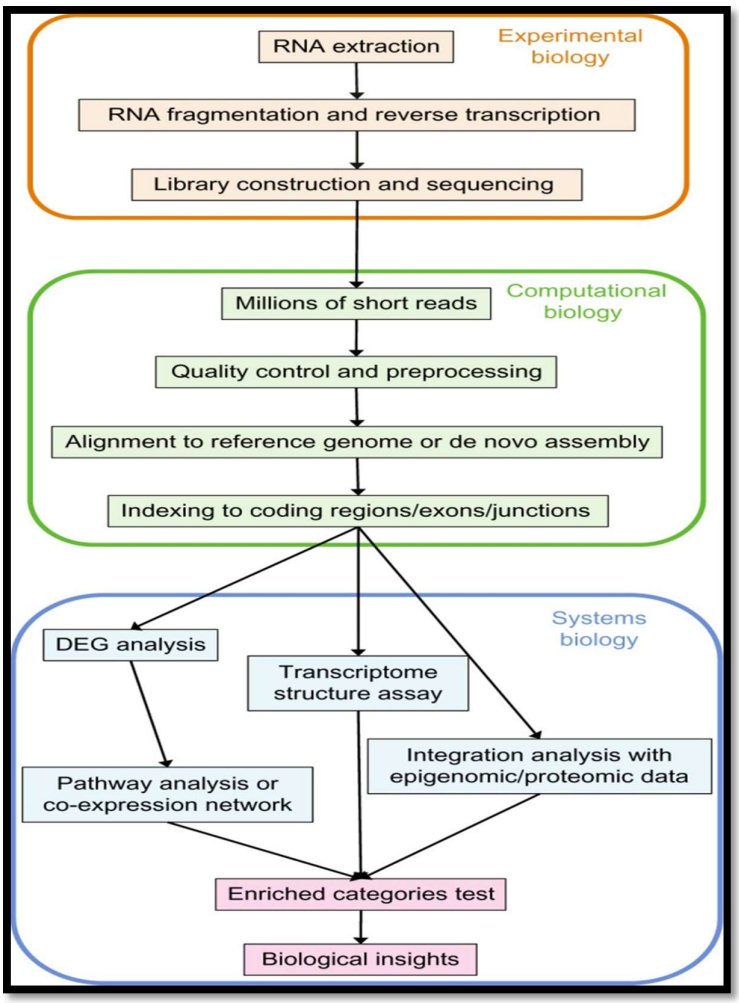


Figure 3: Sample workflow of RNA sequencing and RNA expression profiling. (Han et al.)

In this project I use a classification of genes known as GO terms. A go term, or gene ontology term, is an initiative developed by biologists that intent to integrate the representation of all gene and gene product characteristic across all species. In gene ontology, biologists try

to assign different labels to each gene according to their functionalities, characteristic, cell activities and *etc*. Go terms are mainly assigned based on three different domain, which is cellular component, molecular function and also biological process. For example, one can imagine go term as label and genre for video such as funny, horror, action and etc. A gene may have one or multiple go terms, or no go term at all. In fact, in human gene, more than half of the genes do not contain go term. Gene with the same pathway are like videos of the same genre; they will be assigned the same go term. Some examples of go terms are GO:0022900 for electron transport chain, and GO:0020037 for heme binding.

Rapid development in artificial intelligence, especially deep learning technique introduce a new opportunity in developing effective alternative method to solve a lot of real world problem. Prior to deep learning it was machine learning that first appear in the field of artificial intelligence. Machine learning is subset of artificial intelligence, which defined as the method to identify some pattern from input data, and could be used to predict future value and allow decision making under certain circumstances (Lee *et al.*, 2017). Machine learning is a data driven approach of analyzing past data to obtain some model which can be utilized to predict future data. They can be roughly categorized into two categories, namely supervised learning and unsupervised learning. In supervised learning, the data used is labeled. Examples of supervised learning are, for instance, prediction of house prices given the house size, or classification of data into categories. In supervised learning, each input data is prepared with a corresponding label, which the learning algorithm is expected to produce. If the output label is continuous number, then the supervised learning is known as regression; the learning of category output label such as “yes” and “no” is known as classification. On the other hand, unsupervised learning did not have output label for the input data, but the learning algorithm is used to find the hidden structure or patterns that exist in the input data. For example, the problem of clustering requires the machine to group the input data into several groups without any associated label (Lee *et al.*, 2017).

Many machine learning method suffer from the problem of feature selection. This problem is as follows. The process of machine learning is separated into two phases: training phase and testing phase. In the training phase of machine learning, the pattern inside the input data will be identified, and a model will be built based on the input data pattern; while in the testing phase, the trained model will be used to predict the output value based on the input.

Many machine learning algorithms require the input features to be pre-determined before training the data model. In another words, the data input into the machine learning algorithm must be extracted, pre-processed and filtered manually from the raw input. If the input feature was not pre-processed well, it might affect the performance of the machine learning algorithm. It typically requires a very long time to extract useful features from raw data before input into the learning algorithm, especially when dealing with complicated input such as images (Dong *et al.*, n. d.). This problem is not present in the learning algorithm known as deep learning.

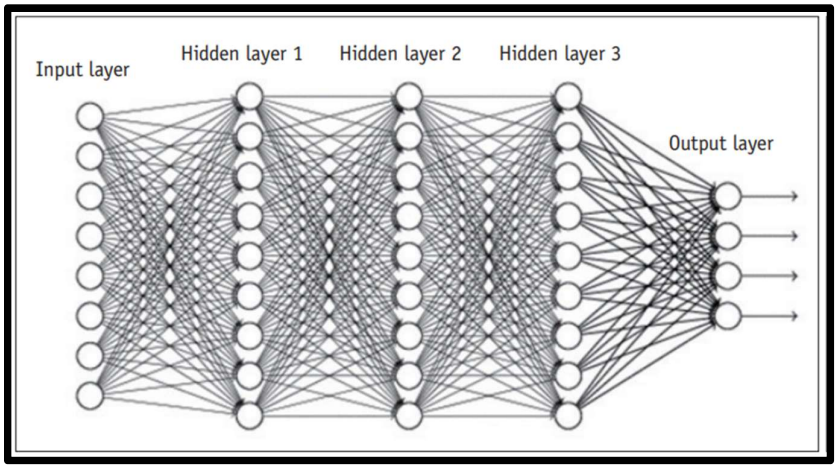


Figure 4. Typical neural network with 3 hidden layer. (Lee, et al., 2017)

Deep learning, or neural network, was inspired by the processes in the human neural system. While the human brain consists of neurons, a neural network is made up of artificial neurons, each which consists of a set of weights and biases which are multiplied to the inputs, and each generates an activation value. While a single neuron itself have limited function, the combination of multiple neurons into a single neural network can form a very complex model. In deep learning, most of the neural networks consist of multiple layers: the input layer, intermediate hidden layers, and the output layer. Each layer of neural network contains a list of neurons, and the neurons are connected to neurons in the next layer. The input layer are the features used to train the neural network, and the output layer output the desired result of the neural network. In between, the hidden layer receives sets of input from the previous layer, and propagate the activation value through the equation:

$$\alpha = \sigma(w^T x + b)$$

where **w** is the weight, **x** is the input from previous layer neuron, **b** is the bias, and **σ** as the transfer function. The transfer function **σ** in a deep neural network is typically a ReLU function

or a Leaky ReLU function. Besides that, maximum likelihood with stochastic gradient descent is the most popular method to fit the parameter **w** and **b** to the training data. By combining neurons through multiple layers, the neural network can perform well and learn complex data model (Litjens *et al*., 2017).

$$ReLU\ , S(x) = max(0, x)$$
$$Leaky\ ReLU = max(\alpha x, x)$$

CHAPTER 2. LITERATURE REVIEW

Convolutional Neural Network (CNN) is a famous deep learning architecture used for image recognition and classification problem. In the past few years, CNNs have successfully outperformed traditional image processing technique in image classification.

Although neural networks perform well in general, their content are often difficult to interpret, resulting in the trained networks being treated as black box algorithms to researchers. In 2014, Simonyan, Vedaldi and Zisserman proposed two methods to visualize the information captured inside the convolutional neural network. Firstly, the first method proposed in the paper is class model visualization. This visualization method is achieve by generating an image that maximizes the output score of neural network. In this method, the weight and parameter in neural network is fixed, while the output score of neural network is back-propagated with respect to the input image. By getting the gradient of output score with respect to input image, one can apply gradient descent on the input image to obtain a local maxima input image that maximizes the output score, hence the information that trigger the neural network the most can be visualized as an image, thus allowing researchers to understand more about the features that neural network learn.

Apart from that, the paper also suggested another visualization method through saliency map, known as image-specific class saliency visualization. Instead of determining the input image that maximizes the output score, this visualization method is trying to determine which feature in input image that contributes to the output score, and hence to determine the feature learned by the neural network. Hence in this method, the output score is differentiated with respect to the input image, and the features that contribute the most to the output score is can be determined by finding the features that have the highest absolute gradient. In other word, if an input feature contains high gradient when we differentiate the output score with respect to the input image, it indicates that this input feature have a huge impact to the output score. Saliency map is generated by obtaining the gradient of each input feature from the output score. By obtaining the saliency map, one can discover the features that actually contribute the most to the image classification and visualize the features that have been learned by neural network. From the saliency map obtained, it can be used for object localization as features with high gradient indicate are the ones that are important for the classification.

CHAPTER 2. LITERATURE REVIEW

The learning task considered in this project is that of imputation. More specifically, this project studies the imputation of gene expression values that are missing or incorrect. Several researchers have proposed different methods to impute correct gene expression value from the incorrect or missing gene expression value. In 2010, Liew, Law and Yan introduced several methods to impute missing value for micro-array technology. Although micro-array and RNA sequencing are different technology that try to achieve the same thing, the method used to impute missing value in micro-array could be used as a references in RNA sequencing data.

In gene expression profiling, the result from the experiment would be a large 2D matrix that keeps the expression values under each condition as a row in the matrix, and the expression values of each gene as a column. Hence each matrix element $\boldsymbol{Y_{ij}}$ denotes the gene expression value for gene $\boldsymbol{j}$ under condition $\boldsymbol{i}$. As proposed by Liew *et al.*, there are useful information available inside the gene expression matrix. The useful information available is the correlation structure in the entries of the gene expression matrix. Correlation exists in the matrix in terms of rows and columns. As the gene expressions are obtained from the same gene sequence under different conditions, the cellular activity should retain a minimal relationship that forms a correlation between each row in the matrix. Similarly, a correlation should also exist between columns in the matrix as the gene are expected to display similar characteristics under similar condition. Hence, it is possible to impute the missing gene expression values from neighboring gene expression value as correlation exist between row and column in the matrix.

As discussed by Liew et al, there are four approaches that can be used to impute missing values in gene expression matrix, which is global, local, hybrid and knowledge assisted approach. Firstly, global approach make use of correlation information from the global matrix. Global approach algorithm derive correlation from the whole matrix, and impute the missing value using the global relation derived. Known algorithm that categorized as global approach are SVD imputation and Bayesian principal component analysis. SVD imputation will make use of eigengenes which is a set of mutually orthogonal expression pattern that can be linearly combined to predict the expression of all gene in the RNA sequence. By obtaining the eigengenes, SVD imputation imputes the missing gene expression value by regressing the gene against k most significant eigengenes and linearly combine the k eigengenes through the coefficient of the regression. Besides SVD impute, Bayesian principal component analysis expressed the N-dimensional matrix into a linear combination of K principal axis vector as:

$$\gamma = \sum_{l=1}^{K} w_l v_l + \varepsilon$$

where w_l is the score factor and ε is the residual error. The score factor and residual error are then treated as normally distributed random variables in the PCA probabilistic model. After that, an Expectation Maximization-like algorithm will be used to predict he model parameter and impute the missing values. Although global approach works in several situations, the authors of the paper mentioned that global approach will likely have poor performance when the expression matrix does not contain a global correlation structure among the elements in the matrix, or when the correlation information only exhibits locally within neighboring elements. Therefore, another approach, the local approach is developed to impute the missing expression value through the neighboring information.

In the local approach to impute expression value, the algorithm only makes use of locally similarity in the matrix as model. Some well-known algorithms that are categorized in this category will be K-nearest neighbor imputation, least square imputation and local least square imputation. Firstly, K-nearest neighbor imputation (KNN impute) make use of pair-wise information between the missing value gene and the ***K*** nearest references gene to determine the missing value. In KNN impute, the missing value in the target gene is imputed as the weighted average of the ***K*** reference ***jth*** component, where the weights for the ***j*** component are set inversely proportional to the Euclidean distance between the *j* component and the missing value gene. According to Liew *et al.*, the performance of KNN impute produce very good results when the local correlation between the cells are very strong. Besides KNN impute, several local approach imputation algorithms make use of the idea of least square regression to impute the missing gene expression. For example, in the least square imputation method, a linear regression model are used to model the missing value gene with other references gene.

$$\gamma = \alpha + \beta x + \varepsilon$$

Firstly, least square imputation will first select the K most correlated gene according to Pearson correlation values, and for each of the K most correlated gene, a missing values is estimated from it through regression. After K missing values are obtained from regression, the K missing value is then linearly combined to form the final estimation value. In least square imputation, the row and column correlation structure are also considered to increase the

accuracy. Another method, local least square imputation uses multiple regression model to impute the missing value from reference gene.

In gene expression matrix, if the data set was homogeneous, then the correlation information should exist globally in the matrix hence global approach algorithms would perform better. If the data set was heterogeneous, local approaches would perform better than global approach as correlation only exist locally in the matrix entries. Therefore, as both global approach and local approach illustrate that both of them have their own advantage and disadvantage on different kind of gene data, hybrid approach has been suggested so that the hybrid method would work well under different data set. LinCmb is such a hybrid method that capture the global and local correlation information and used it to impute the missing expression value. In LinCmb, five different imputation method are combined together to impute the missing value, which is row average, KNN impute, SVD impute, Bayesian principle component analysis and GMC impute. Among these five imputation methods, row average, KNN impute and GMC impute make use of local correlation information to output the missing value, whereas the other two methods, SVD impute and Bayesian principal component analysis make use of global correlation information to impute the missing values. After obtaining each missing value from the five methods, LinCmb convexly combines the missing value imputed through a set of weights. For the weight used in combining the imputed value from five different method, LinCmb first generates fake entries in the matrix where the value is not missing. The five imputation methods are used to impute the values at some known entries in the matrix, then least square regression is performed on the predicted values and the actual values. The optimal weight is obtain by averaging all the weights in 30 iterations. After the optimal weights have been obtain, LinCmb will be run on actual missing expression values, and the imputed output from all five methods will be convexly combine using the optimal weights.

Lastly, Liew *et al.* also suggested that knowledge assisted approach in imputing missing expression value, where this approach suggested that domain knowledge and external information about the gene expression experiment could be integrated into the process of imputation such that the output of the imputation will be much more accurate. For example, knowledge about the biomolecular process, regulatory mechanism inside cell activity, others external knowledge for the data sets, can all be used to give a hint on the process of imputation

such that approximation about the missing expression value, and the exact genes inside the expression matrix are highly correlated with the missing value gene and *etc*. Therefore, by involving domain knowledge and external information into the imputation process, the accuracy of imputation would be much more promising. (Liew *et al.*, 2010)

In 2013, Baghfalaki *et al.* proposed their study on missing value imputation on RNA sequence data by using statistical models. In this study, several statistical models were used to impute the missing expression value in RNA sequence data. Firstly, Baghfalaki *et al*. proposed the use of regression model. In the regression model, the missing values are imputed with the predicted score from a regression model. The method would first find K most correlated gene with the missing value gene, the K most correlated gene would then be fitted into a regression model, and the fitted regression model will then be used to impute the missing value from the expression value. In this regression model, expectation maximization approach will be used to determine the parameter of the regression model. Another statistical approach discussed is based on Poisson mixture. In this statistical model, the Poisson mixture model is used to cluster the RNA sequence data and predict the missing expression values. Similar to the regression model, Poisson mixture models also used expectation maximization approach to estimate the parameter and the clustering of each gene. After parameter estimation, one can use Integrated Complete Likelihood and Bayesian information criterion to determine the number of clusters for the RNA sequence data. In order to use the Poisson mixture model to impute missing expression values, the RNA sequence data will first be clustered based on the parameter determine above, then for each missing value, it will be imputed as the mean value of all available gene in the same cluster.

Another regression model discussed is the Bayesian Poisson regression model. In this model, a Bayesian approach will first be used to estimate the parameter for the model, and then the Poisson imputation approach will fit the Poisson model and calculate the posterior mean and posterior variance to determine the parameter for the model. After the parameter has been determine, the Bayesian Poisson model will then be used to impute the missing value. Other approach such as quasi-Poisson approach used quasi-Poisson family instead of Poisson family to estimate the parameter.

Lastly, Baghfalaki *et al.* also discussed using Bayesian generalized linear model to impute missing expression value. In this model, a generalized linear model with independent normal *t* or Cauchy prior distribution for the coefficient will be used to impute the missing value in the expression matrix. To apply this model, the model should first generate an impute matrix, then iteratively and randomly impute expression value using regression model to the missing value entries, until it reaches the approximate convergence in the expression matrix. In addition, the concept of multiple imputation were also discussed in the paper, where it suggested that instead of imputing the expression value one by one, the imputation method may impute multiple missing values at the same time (Liew, et al., 2013).

In 2016, Xie *et al.* proposed the use of deep auto-encoder in gene expression prediction. In the paper, they developed a multilayer perceptron stacked auto-encoder to predict the gene expression value. The multilayer perceptron stacked auto-encoder (MLP-SAE) model consists of four layer, an input layer, two hidden auto-encoder layer, and lastly the output layer, as shown in Figure below.

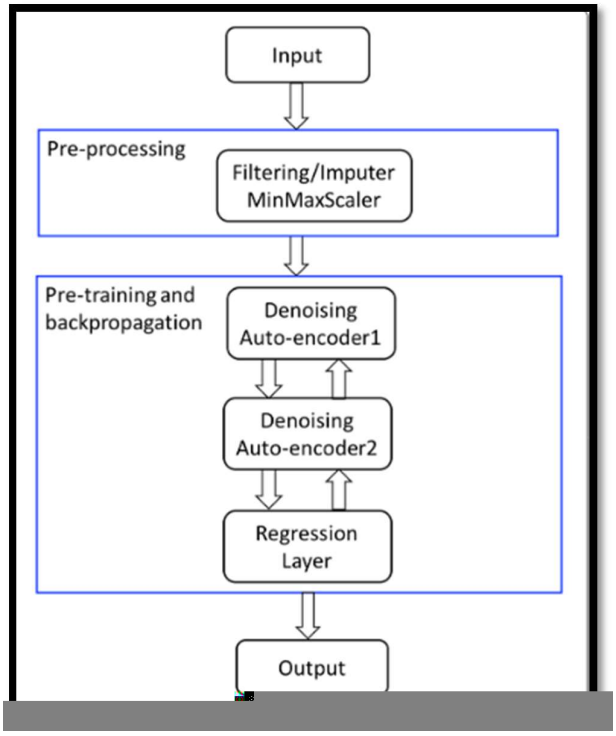


Figure 5. Overall workflow MLP-SAE model. (Xie, et al., 2016)

In the model, the input was preprocessed before being fed into the neural network. For the hidden layer stacked denoising auto-encoder is used. Also, since this neural network is required to predict the gene expression value which is quantitative, a regression layer is used as the output layer. Dropout was used in the model to ensure that no over-fitting will occur. That is, in every training iteration some of the functional units in the neural network are randomly chosen to be temporally disabled for training. Neural networks that apply dropout often achieve higher model performance and are also able to avoid over-fitting. Xie *et al.* showed that their model works better with dropout applied. By using their model with dropout technique, Xie *et al.* used the model on predicting gene expression, and showed that their model was able to predict the gene expression values close to the true values, as shown in Figure 8. Therefore, this model might be able to become a starting point in constructing a model to impute the RNA gene expression value.

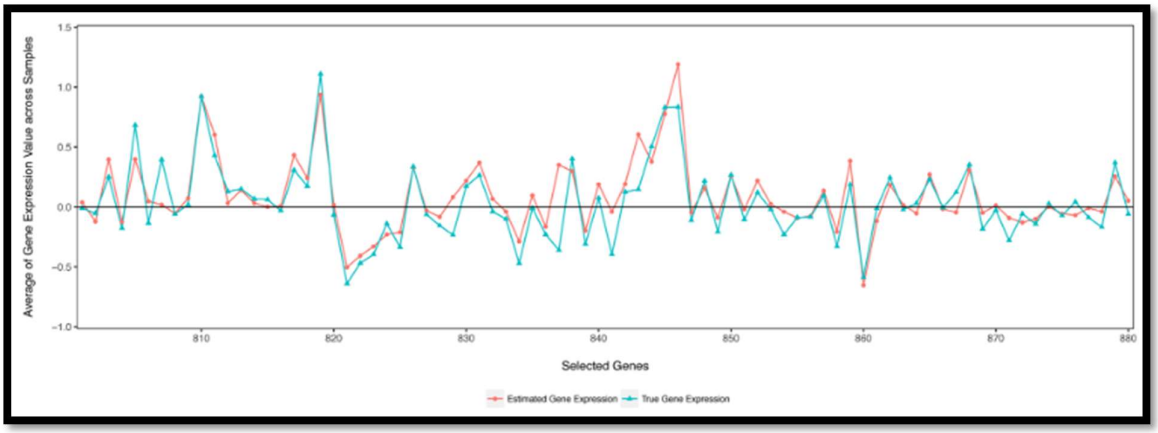


Figure 6. True Expression and Predicted Expression of All Genes Using MLP-SAE with Dropout (Xie, et al., 2016)

It is worth mentioning that some researchers have proposed a different kind of auto-encoder, known as the adversarial auto-encoder, which adapted the characteristic of Generative Adversarial Network (GANs) neural network. As proposed by Goodfellow *et al.* in 2014, GANs is a special kind of neural network that consists of two competitive components in the neural network: a generative model and a discriminative model. The discriminative model is a neural network that computes the probability that the incoming sample belongs to the true sample, but not from the sample generated by the generator neural network. In contrast, the generator model is another neural network that is trained to produce sample that the discriminator model couldn't differentiate from the true sample. By having two competitive components in the neural network, the performance of both discriminator and generator will

keep improving, until finally a generator model that is able to produce highly likely sample from the input sample is obtained. As proposed in the paper from Makhzani *et al.*, adversarial auto-encoder can be achieved by using auto-encoder in GANs where the auto-encoder is the generator that tries to produce input sample that are difficult to be distinguished from the true sample, and another adversarial neural network that act as the discriminator to differentiate the sample between true sample and sample from generator.

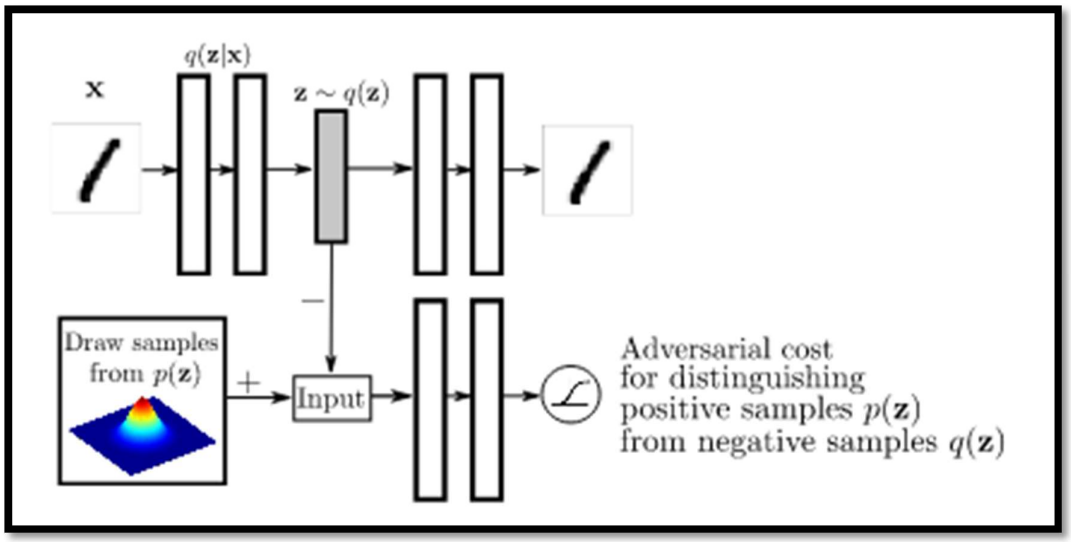


Figure 7. Architecture of an adversarial autoencoder (Makhzani et al, n.d.)

CHAPTER 3. SYSTEM DESIGN

It is natural to expect that an input gene expression feature that is highly related to an output gene in terms of some biological process should demonstrate higher influence over that output expression values in the learned model. In this chapter, I will develop a model to test this hypothesis. To investigate this possibility, one needs to address the following two relationships:

(I) Biological relation between the input gene expression and the output gene expression

(II) The influence that the input gene expression has on the output gene expression in the learned model.

I use the number of go terms shared by the gene of the input expression and that of the output expression to capture the relationship required in (I). To capture (II), the gradient between the gene of the input expression and the gene of the output expression can be used. To show the hypothesis, it suffices that one shows that the learned model demonstrate a correlation between (I) and (II) that is above chance. Therefore, a hypothesis testing will be conducted, with the null hypothesis and alternative hypothesis defined as

H_0: The model learned demonstrate a relationship between (I) and (II) within chance.

H_1: The model learned shows high correlation between (I) and (II) beyond the probability dictated by chance alone.

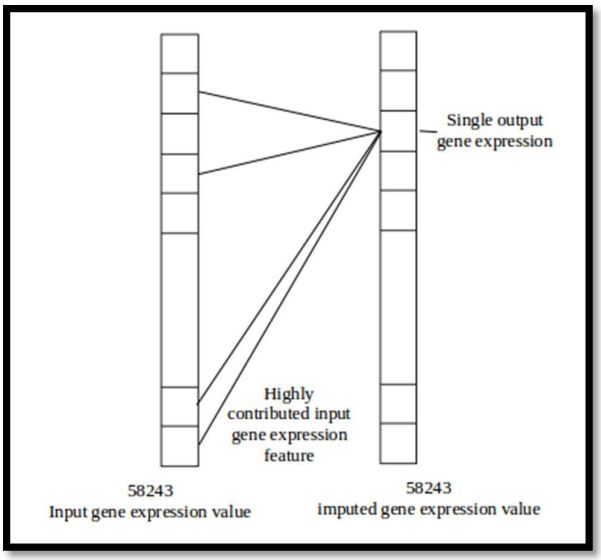


Figure 8. Illustration of output gene expression feature and highly contributed input gene expression features

I first establish some basic probabilities for hypothesis testing. Consider the probability that at least q out of k gene expression features randomly chosen output of n gene expression features share at least x go terms with the output gene expression. To simplify this calculation we further assume that m out of the n genes share at least x go terms with the output gene expression features. That is,

n = the total number of gene expression feature

k = the number of gene expression features

q = the number of gene expression features out of the chosen k gene expression features that share at least x common go term with the output gene ($q \leq k$)

x = the number of common go terms between input gene expression feature and output gene expression feature

m = the number of gene expression features from all input gene expression features that have at least x common go terms with gene of the output expression

The case where exactly q out of k gene expression features share at least x go terms with the output can is given by $\frac{\binom{m}{q}\binom{n-m}{k-q}}{\binom{n}{k}}$, as demonstrated in the figure below.

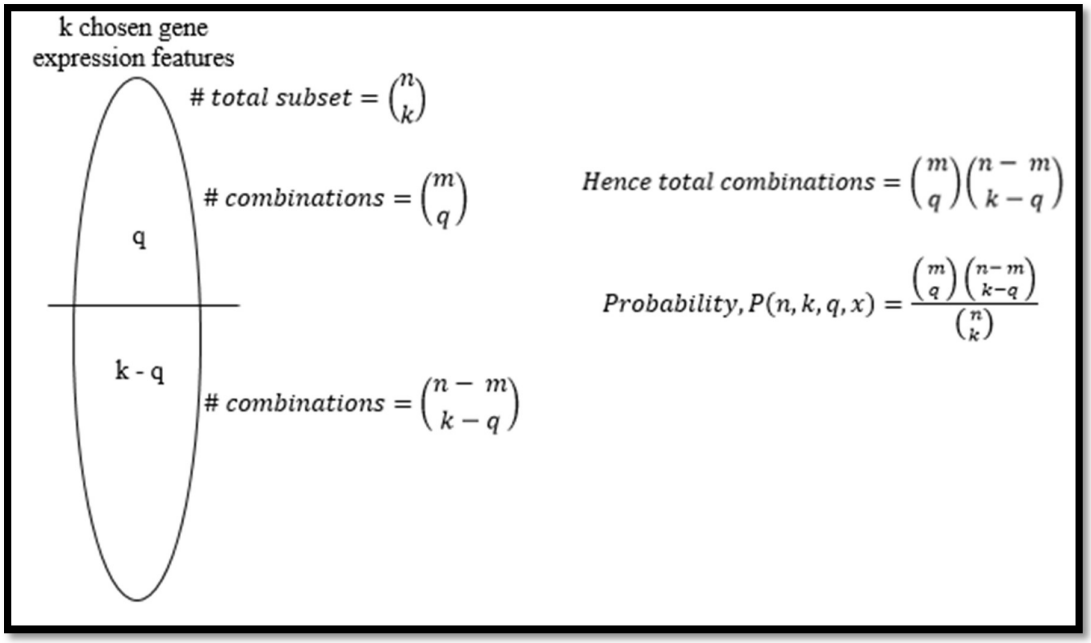


Figure 9. Proposed base probability model

Since we want at least q gene expression features, the probability will range over j from q to k, giving us $p = \sum_{j=q}^{k} \frac{\binom{m}{j}\binom{n-m}{k-j}}{\binom{n}{k}}$.

Our aim is to compare this probability, where the k genes are chosen randomly, with the case where we choose the top k genes according to the gradients encoded in the learned model. (To simplify this comparison we fix q and x to 1, and let k =10. More on this in the Results section.)

However, one parameter remains variable across different output genes, namely, m. To remedy this variation we choose output genes that has the same m and only compare the probabilities of these genes. For this set of genes, it clear that they yield the same p. Assuming that there are a total r such genes, the probability that s out of these r genes used gene expression features that share go terms with their input genes would be $\binom{r}{s}(p)^s(1-p)^{r-s}$, a binomial distribution. This gives us the distribution of the null hypothesis.

We now describe how we determine the k genes from the learned model. These k genes are the ones which has the most influence on the output gene expression according to the learned model. In order to determine these highly influential input gene expression features for each output gene expression feature, the concept of saliency map is applied where for each output gene expression feature, back-propagation is done on every single output expression feature to input gene expression feature to obtain the gradient of each input gene expression feature toward the single output gene expression feature. Hence, the highly contributed input gene expression can be selected as the input gene expression feature that have the highest absolute gradient value, also known as top absolute gradient gene expression feature. By obtaining the top absolute gradient gene expression features, the top absolute gradient gene expression features can be used to determine whether the output gene expression features used gene expression features with same go term for imputation. Finally, by counting the number of output gene expression features used gene expression with common go term for imputation as test-statistic, the test-statistic can be applied in the formula to obtain the p-value used for hypothesis testing.

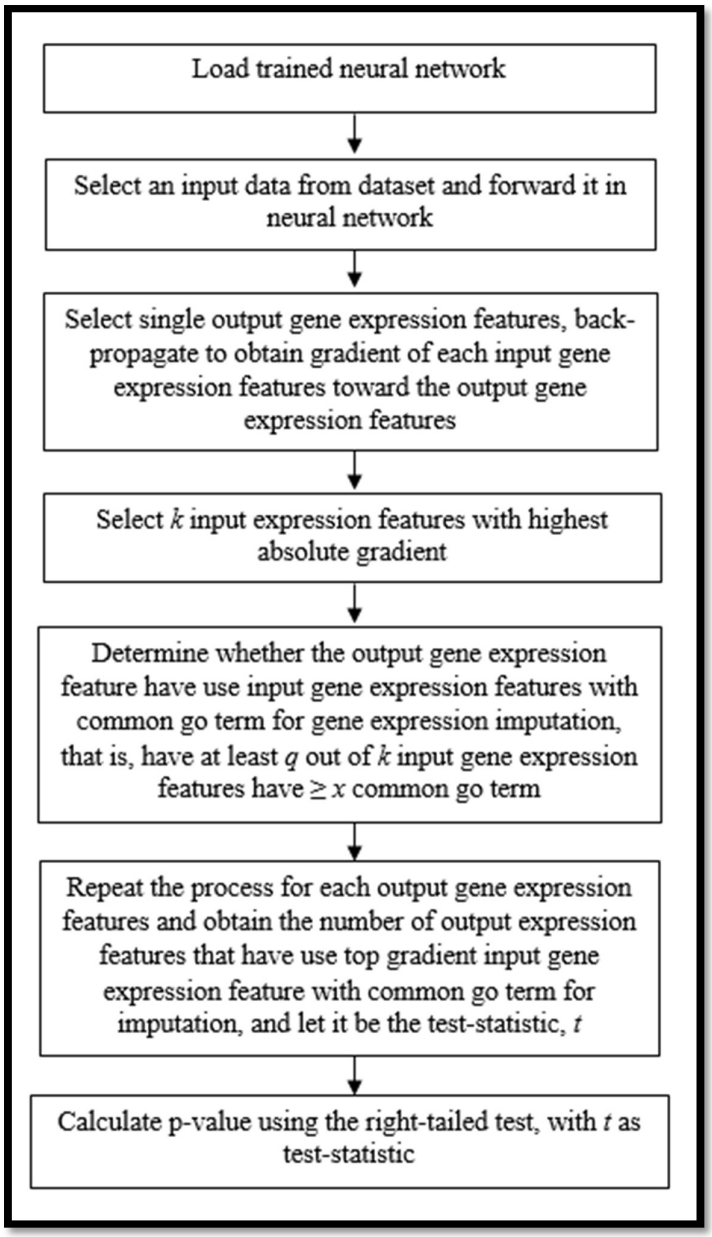


Figure 10. Flowchart for evaluating neural network using proposed probability model

CHAPTER 4. METHODOLOGY, TOOLS AND IMPLEMENTATION

4.1. Methodology

In order to achieve this project in a reasonable time, a methodology has been proposed to accomplish the project.

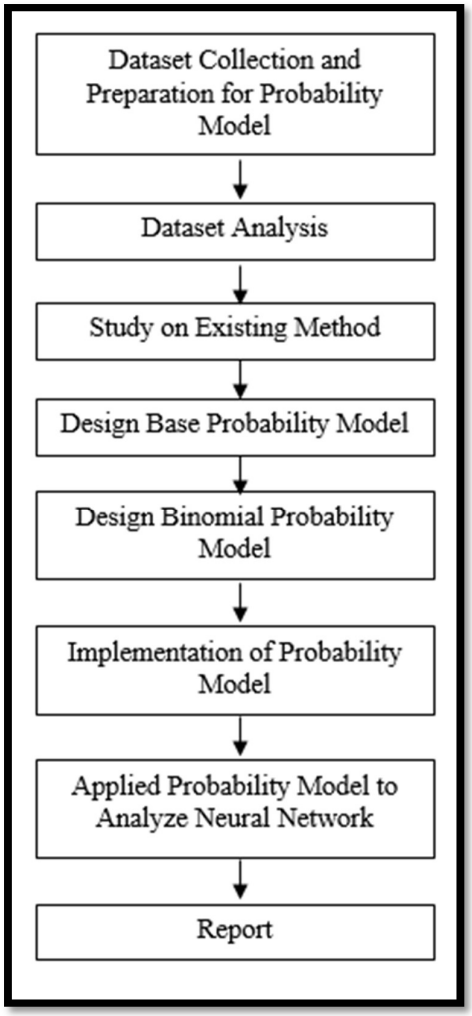


Figure 11. Methodology used for the project

Firstly, dataset like go terms are collect and download from the Internet, and analysis have been conducted on the data. Next, existing method such as saliency map, neural network and etc. have been study for project purpose. Moving on, the base probability model and the binomial distribution model have been designed and developed. The probability model is implemented and it is used to analyze neural network. Lastly, a report have been written.

4.2. Tools

Throughout the project, R programming language is used to prepare dataset such as download go terms. R library biomaRt is used to download go term for each gene expression feature from Internet. Besides, Python programming language is used to implement the probability model. PyTorch Python library is used for neural network implementation.

4.3. Implementation

In this project, several Python script is used in this project. Firstly, base_probability_model.py is used to implement and generate the base probability model, providing the parameters *n* and *k*. The generated probability model will be store in a file for future use using Python pickle library. The generated probability model is store in the following format.

- Loading the file into Python return a Python dictionary that store base probability model for each gene expression features.
- Gene id such as ENSG00000223972 is used as the key to access the base probability model for each gene, and each base probability model is a Python dictionary that store the information as

    - **common_go_term_count**

      A list that store the number of gene expression features sharing $\geq x$ common go term with the gene expression feature. For example, data['ENSG00000223972']['common_go_term_count'][*x*] return a integer that tells the number of gene expression features that shares *x* or more go term with gene ENSG00000223972

    - **pdf**

      A two-dimensional list that store the base probability model P(n, k, q, x). In the two-dimensional list, the row corresponding to value of *q* and column corresponding to value of *x* in base probability model P(n, k, q, x). For example, data['ENSG00000223972']['pdf'][q][x] tells the value of P(n, k, q, x).

    - **cdf**

a two-dimensional list that store the cumulative base probability model, $\sum_{i=q}^{k} P(n, k, i, x)$. In the two-dimensional list, the row corresponding to value of q and column corresponding to value of x in base probability model P(n, k, q, x). For example, data['ENSG00000223972']['cdf'][q][x] tells the value of $\sum_{i=q}^{k} P(n, k, i, x)$.

Next, saliency.py is used to generate gradient and determine top *k* absolute gradient gene expression feature, providing neural network model file path and value of *k*. The top *k* absolute gradient gene expression feature will be store in a file for future use using Python pickle library, and having format as follow.

- Loading the file into Python return a Python list that store top absolute gradient gene features for each gene expression features.
- Each element in the list will be a Python dictionary that store the information of top absolute gradient gene expression features as
    - **min** : the minimum value for all the gene expression features gradient generated
    - **max** : the maximum value for all the gene expression features gradient generated
    - **mean** : the mean value for all the gene expression features gradient generated
    - **sum** : the sum for all the gene expression features gradient generated
    - **idx** : the index of gene id that have top *k* highest absolute gradient value, in ascending order
    - **val** : the gradient value corresponding to the gene expression features index in 'idx'

Lastly, Python script evaluate_nn.py is used to evaluate neural network and determine the test-statistic and p-value for a given neural network, providing the file generated from saliency.py and base_probability_model.py, together with the value of *k*, *q* and *x* for P(n, k, q, x). The script will load the file and calculate the test-statistic and p-value for the neural network.

CHAPTER 5. RESULTS

This chapter shows the probability calculated from the learned model, and demonstrate how the values suggest that the relationship between the input and output gene expressions encoded in the learned model to be non-random.

There are a total of n = 58243 gene expression features in the dataset. To simplify calculation, I let k=10, indicate that the top 10 absolute gradient input gene expression features will be used to analyze the go term relationship with the output gene expression feature. Furthermore, q is set to 1, indicating that at least 1 out of 10 of the top absolute gradient input gene expression features will have to have at least x common go terms to be considered as the output gene expression features used input gene expression features with common go term for imputation. Finally, x is also set to 1 as it is to determine whether the top absolute gradient input gene expression feature have common go term with the output gene expression feature. Therefore, the base probability derived will be

$$p = \sum_{j=1}^{10} \frac{\binom{m}{j}\binom{58243-m}{10-j}}{\binom{58243}{10}}$$

At this point, the calculation can be simplified if m is fixed. However, the value of each m depends on the output gene, and these m values is not fixed in general. Hence, in order to simplify the problem, instead of using all gene expression features, I choose only the genes of the same value of m. To find a suitable value of m, I calculated the number of genes of different values of m and identified the m value with the most number of genes, that is, the group with the largest number of gene expression features. This corresponds to a value of m = 8340, which includes 352 gene expression features. This yields the probability

$$p = \sum_{j=1}^{10} \frac{\binom{8340}{j}\binom{58243-8340}{10-j}}{\binom{58243}{10}} = 0.7868$$

Hence, the binomial distribution probability for hypothesis testing is

$$\binom{352}{x} 0.7868^{x}(0.2132)^{352-x}$$

giving the cumulative density function

$$\sum_{x=t}^{352} \binom{352}{x} 0.7868^{x} (0.2132)^{352}$$

To evaluate each deep neural network, each output gene expression features is back-propagated to obtain the top 10 absolute gradient gene expression features. Then, the output gene expression features is determine to have used highly contributed input gene expression features with common go term for imputation. Finally, the number of output gene expression features that used highly contributed input gene expression features with common go term for imputation is counted and is used as test-statistic to determine the *p*-value.

Model	Loss	Test-statistic (t)	*p*-value
FC_1_1000	1.42	310	3.291e-06
FC_1_2000	1.39	328	6.998e-14
FC_1_2500	1.38	324	9.326e-12
CONV_c3_h1000_4_retrain	1.27	352	2.216e-37
CONV_c3_h1000_5_retrain	1.31	350	1.026e-33

Table 1. Results for deep neural network

From the results, it shows that neural network with the least loss obtain relatively smaller p-value, compare to neural network with higher loss.

The *p*-values for all the models are significantly small for us to reject the null hypothesis. Therefore, this lends evidence to the hypothesis that the learned model is not random in terms of the go terms. That is, the gene expression features have relevant common go terms with highly contributed input gene expression feature, indicating that the neural network does utilize input gene expression feature from same go term to impute output gene expression value.

CHAPTER 6. CONCLUSION

In conclusion, a probability model have been proposed and developed to investigate the features learn by neural network for gene expression imputation. Through the probability model, we have successfully shown that the neural network does utilize input gene expression feature with same go term to impute the gene expression value. Hence, by developing this probability model, researchers in the future may make use of this probability model to gain more understanding about the neural network and how the neural network learn features to perform gene imputation. Besides, gaining such understanding on features learn by neural network allow researchers to train deep neural network more effectively, such as drop-out can be applied on input gene expression features that do not have common go terms to further enhance the results of neural network. Moreover, as this project only consider a small subset of gene expression feature to investigate the features learn by neural network, future improvement can be done by develop a more robust and comprehensive probability model to include all gene expression feature for evaluation. Lastly, with this probability model, it is hope that it can enhance the result from neural network for gene expression imputation, and allow analysis on gene expression profiling to be much more accurate and precise, thus benefit different aspect in related field.

BIBLIOGRAPHY

1. Allen, A., Li, W., n.d. Generative Adversarial Denoising Autoencoder for Face Completion. [Online] Available at https://www.cc.gatech.edu/~hays/7476/projects/Avery_Wenchen/ [Accessed 1 March 2018]
2. Austin Community College, n.d. Molecular Genetics - From DNA to Trait. [Online] Available at http://www.austincc.edu/mlt/mdfund/mdfund_unit4notesTranscription%20and%20Translation%2006.pdf [Accessed 1 March 2018].
3. Baghfalaki, T., Ganjali, M., Berridge, D., 2013. Missing Value Imputation for RNA-Sequencing Data Using Statistical Models: A Comparative Study. [Online] Available at https://www.atlantis-press.com/php/download_paper.php?id=25862105 [Accessed 1 March 2018].
4. Dong, Y., Pan, Y., Zhang, J. & Xu, W., n.d. Learning to Read Chest X-Ray Images from 16000+ Examples using CNN. [Online] Available at: http://iiis.tsinghua.edu.cn/~weixu/files/bigdata4health2017_pan.pdf [Accessed 15 August 2017].
5. Edenburg, H.J., 2007. The Genetics of Alcohol Metabolism: Role of Alcohol Dehydrogenase and Aldehyde Dehydrogenase Variants. [Online] Available at https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3860432/pdf/arh-30-1-5-13.pdf [Accessed 1 March, 2018].
6. EMBL-ABI, 2018. Illumina sequencing. [Online] Available at https://www.ebi.ac.uk/training/online/course/ebi-next-generation-sequencing-practical-course/what-next-generation-dna-sequencing/illumina- [Accessed 1 March 2018].
7. Goodfellow, I., Bengio, Y., Courville, A., 2016. Chapter 14: Autoencoders. [Online] Available at https://www.deeplearningbook.org/contents/autoencoders.html [Accessed 1 March 2018]
8. Han, Y.X., Gao, S.G., Muegge, K., Zhang, W., Zhou, B., 2015. Advanced Application of RNA Sequencing and Challenges. [Online] Available at https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4648566/. [Accessed 1 March 2018].

BIBLIOGRAPHY

9. Lee, J.-G.et al., 2017. Deep Learning in Medical Imaging: General Overview. [Online] Available at: https://synapse.koreamed.org/Synapse/Data/PDFData/0068KJR/kjr-18-570.pdf [Accessed 15 August 2017].
10. Liew, W.C., Law, N.F., Yan, H, 14 December 2010. Missing value imputation for gene expression data: computational techniques to recover missing data from available information. [Online] Available at https://academic.oup.com/bib/article/12/5/498/268546 [Accessed 1 March 2018].
11. Litjens, G. et al., 2017. A Survey on Deep Learning in Medical Image Analysis. [Online] Available at: http://ac.els-cdn.com/S1361841517301135/1-s2.0-S1361841517301135-main.pdf?_tid=ef8c3446-81fd-11e7-8bbe-00000aacb35f&acdnat=1502831511_1068f92fac2d7e69262e56b2c759595e [Accessed 15 August 2017].
12. Makhzani A., Goodfellow, I., Shlens, J., Jaitly, N., Frey, B., n.d. Adversarial Autoencoders. [Online] Available at https://arxiv.org/pdf/1511.05644.pdf. [Accessed 1 March 2018]
13. Raven, P., Johnson, G., Mason, K., Losos, J., Singer, S., 2013. Biology. [Online] Available at http://biology.org.ua/files/lib/Raven_Johnson_McGraw-Hill_Biology.pdf [Accessed 1 March 2018].
14. Simonyan, K., Vedaldi, A., Zisserman, A., 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. [Online] Available at https://arxiv.org/pdf/1312.6034.pdf. [Accessed 1 Jun 2018]
15. Stanford University, n.d. Autoencoders. [Online] Available at http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/ [Accessed 1 March 2018]
16. University of Helsinki, 2009. Basics on Molecular Biology. [Online] Available at https://www.cs.helsinki.fi/bioinformatiikka/mbi/courses/09-10/itb/Lectures_1509_and_1709.pdf [Accessed 1 March 2018]
17. Xie, R., Wen, J., Quitadamo, A., Cheng, J., Shi, X., 2016. A deep auto-encoder model for gene expression prediction. [Online] Available at https://link.springer.com/content/pdf/10.1186%2Fs12864-017-4226-0.pdf. [Accessed 1 March 2018]

APPENDIX A

Source Codes

A-1. saliency.py

```
import pickle
import numpy as np
import torch
from torch import tensor
import argparse

import utils

############################################################################################
#########################

def generate_gradient(model_file, low_set = None, k = 10, write_to_file = True,
out_file = 'gradient.grad', verbose = True):
	if low_set is None:
		_, _, low_set = utils.get_dataset()
		low_set = low_set[0].float()

	low_set = low_set.unsqueeze(0).float().cuda()
	low_set.requires_grad = True
	low_set.retain_grad()

	net = torch.load(model_file).cuda() if (type(model_file) is str) else
model_file.cuda()
	net.eval()

	for param in net.parameters():
		param.requires_grad = False

	output = net(low_set)
	grad = []

	for idx in range(low_set.shape[-1]):
		final_grad = torch.zeros(low_set.shape[-1], requires_grad =
True).cuda()
		final_grad[idx] = 1

		output.backward(final_grad, retain_graph = True)
		sort_grad_abs = low_set.grad[0].abs().sort()[1]

		grad.append({
			'min': low_set.grad[0].min().item(),
			'max': low_set.grad[0].max().item(),
			'mean': low_set.grad[0].mean().item(),
			'sum': low_set.grad[0].sum().item(),
			'idx': sort_grad_abs[-k:].tolist(),
			'val': low_set.grad[0][sort_grad_abs[-k:]].tolist()
		})

		low_set.grad.zero_()

		if verbose:
			print (utils.progress_str(idx + 1, low_set.shape[-1]), end =
'')
```

APPENDIX A

```
	if verbose:
		print()

	if write_to_file:
		with open(out_file, 'wb') as f:
			pickle.dump(grad, f)

	for param in net.parameters():
		param.requires_grad = True

	if not write_to_file:
		return grad

################################################################################
#########################

if __name__ == '__main__':
	parser = argparse.ArgumentParser(description = 'Generate gradient for
neural network')
	parser.add_argument('nn_path', help = 'neural network model file')
	parser.add_argument('output_file', help = 'output file to store gradient
generated')
	parser.add_argument('-k', type = int, default = 10, help = 'number of top
absolute gradient feature to generate')
	parser.add_argument('-v', action = 'store_true', help = 'print progress
message')

	args = parser.parse_args()
	generate_gradient(args.nn_path, k = args.k, out_file = args.output_file,
verbose = args.v)
```

A-2. base_probability_model.py

```
import numpy as np
import pickle
import argparse

import utils
from utils import C

################################################################################
###########

def single_q_prob(n, k, q, m):
	if k > n or q > m or (k - q) > (n - m):
		return 0.0

	return C(m, q) * C(n - m, k - q) / C(n, k)


def generate_base_probability(k = 10, write_to_file = True, out_file = '10.prob',
verbose = True):
	prob_model = {}
	gene_id = utils.get_gene_id()
	go_term = utils.get_go_term()
	n = len(gene_id)

	for gene_idx, gene in enumerate(gene_id):
		common_go_term_count = [0] * (len(go_term[gene]) + 1)

		for gene_2 in gene_id:
			for idx in
range(len(set(go_term[gene]).intersection(set(go_term[gene_2]))) + 1):
				common_go_term_count[idx] += 1

		pdf = np.zeros((k + 1, len(common_go_term_count)))
		cdf = np.zeros((k + 1, len(common_go_term_count)))

		for q in range(k + 1):
			for idx_x, m in enumerate(common_go_term_count):
				pdf[q][idx_x] = single_q_prob(n, k, q, m)

		for x in range(pdf.shape[1]):
			for p in range(pdf.shape[0]):
				cdf[p][x] = pdf[p:, x].sum()

		prob_model[gene] = {
					'common_go_term_count':
common_go_term_count,
					'pdf': pdf.tolist(),
					'cdf': cdf.tolist()
				}

		if verbose:
			print (utils.progress_str(gene_idx + 1, len(gene_id)), end =
'')

	if verbose:
```

APPENDIX A

```
            print()

      if write_to_file:
            with open(out_file, 'wb') as f:
                  pickle.dump(prob_model, f)
      else:
            return prob_model

################################################################################
#########################

if __name__ == '__main__':
      parser = argparse.ArgumentParser(description = 'Generate base probability
model')
      parser.add_argument('output_file', help = 'output file to store base
probability model generated')
      parser.add_argument('-k', type = int, default = 10, help = 'number of
feature to choose')
      parser.add_argument('-v', action = 'store_true', help = 'print progress
message')

      args = parser.parse_args()
      generate_base_probability(k = args.k, out_file = args.output_file, verbose
= args.v)
```

A-3. evaluate_nn.py

```
import os, argparse
import numpy as np
import matplotlib.pyplot as plt
import utils
from utils import C

################################################################################
#########

def binomial_probability_model(r, p, t):
	p_value = 0

	for i in range(t, r + 1):
		p_value += C(r, i) * (p ** i) * ((1 - p) ** (r - i))

	return p_value

def evaluate_nn(grad_file, prob_file, k = 10, q = 1, x = 1):
	gene_id = utils.get_gene_id()
	go_term = utils.get_go_term()
	grad = utils.load_pickle(grad_file) if (type(grad_file) is str) else
grad_file
	prob = utils.load_pickle(prob_file) if (type(prob_file) is str) else
prob_file
	common_go_term_count_gene = [gene for gene in prob if
len(prob[gene]['common_go_term_count']) > x]
	count = {}

	for gene in common_go_term_count_gene:
		if prob[gene]['common_go_term_count'][x] in count:
			count[prob[gene]['common_go_term_count'][x]].append(gene)
		else:
			count[prob[gene]['common_go_term_count'][x]] = [gene]

	p_value = {x: None for x in list(count)}

	for m, same_m_gene in count.items():
		base_prob = prob[same_m_gene[0]]['cdf'][q][x]
		test_statistic = 0

		for g in same_m_gene:
			cgtc = [0] * (len(go_term[g]) + 1)

			for g2 in grad[gene_id.index(g)]['idx'][-k:]:
				for idx in
range(len(set(go_term[g]).intersection(set(go_term[gene_id[g2]]))) + 1):
					cgtc[idx] += 1

			if cgtc[x] >= q:
				test_statistic += 1

		p_value[m] = [len(same_m_gene), test_statistic,
binomial_probability_model(len(same_m_gene), base_prob, test_statistic)]

	return p_value;
```

APPENDIX A

```
################################################################################
#########

if __name__ == '__main__':
	parser = argparse.ArgumentParser(description = 'Generate gradient for neural
network')
	parser.add_argument('gradient_file_path', help = 'path to neural network
gradient file')
	parser.add_argument('base_probability_model_file', help = 'path to base
probability model file')
	parser.add_argument('output_file', help = 'output file to store gradient
generated')
	parser.add_argument('boxplot_path', help = 'output file to p-value box-plot')
	parser.add_argument('-k', type = int, default = 10, help = 'number of top
absolute gradient feature to used for evaluation')
	parser.add_argument('-q', type = int, default = 1, help = 'number of top
absolute gradient feature to have at least x common go term')
	parser.add_argument('-x', type = int, default = 1, help = 'minimum number of
common go term to be shared (>=)')

	args = parser.parse_args()
	p_value = evaluate_nn(args.gradient_file_path,
args.base_probability_model_file, k = args.k, q = args.q, x = args.x)

	with open(args.output_file, 'w') as f:
		f.write('final p-value = {}\n'.format(np.prod([p_value[x][-1] for x in
p_value])))
		f.write('number of gene have p-value <= 0.05 =
{}\n'.format(sum([p_value[x][0] for x in p_value if p_value[x][-1] <= 0.05])))

		f.write('\n')
		f.write('\t'.join(['m', 'number of gene have m', 'test statistic', 'p-
value', 'p-value <= 0.05?']) + '\n')

		for m in sorted(p_value.keys()):
			f.write(str(m) + '\t' + '\t'.join([str(x) for x in p_value[m]])
+ '\t' + ('1' if p_value[m][-1] <= 0.05 else ' ') + '\n')

	plt.title('boxplot for p-value
({})'.format(os.path.basename(args.gradient_file_path)))
	plt.ylabel('p-value')
	plt.boxplot([p_value[x][-1] for x in p_value])
	plt.savefig(args.boxplot_path)
```

APPENDIX A

A-4. utils.py

```
import os, sys
import pickle
import numpy as np
import pandas as pd
from torch import tensor

################################################################################
##############

DATASET_DIR = '/home/wingkhang/gene-expression/skeletal_muscle/data/'
NN_DIR = '/home/wingkhang/gene-expression/skeletal_muscle/trained-models/'
HIGH_COVERAGE_DATASET = os.path.join(DATASET_DIR, 'high_coverage_data_lg_n200000')
LOW_COVERAGE_DATASET = os.path.join(DATASET_DIR, 'low_coverage_data_lg_n200000')

GENE_ID_PATH = 'data/etc/gene_id'
GO_TERM_PATH = 'data/etc/go_term'


def get_dataset(high = HIGH_COVERAGE_DATASET, low = LOW_COVERAGE_DATASET):
	high_set = pd.read_csv(high, sep = '\t')
	low_set = pd.read_csv(low, sep = '\t')

	gene_id = high_set['Geneid'].tolist()
	high_set = tensor(high_set.drop(['Geneid'], axis = 1).as_matrix().T)
	low_set = tensor(low_set.drop(['Geneid'], axis = 1).as_matrix().T)

	return gene_id, high_set, low_set


def get_gene_id(path = GENE_ID_PATH, from_dataset = False):
	if from_dataset is False:
		with open(path, 'rb') as f:
			return pickle.load(f)

	high_set = pd.read_csv(HIGH_COVERAGE_DATASET, sep = '\t')

	return high_set['Geneid'].tolist()


def get_go_term(path = GO_TERM_PATH, filtered = True):
	with open(path, 'rb') as f:
		go_term = pickle.load(f)

	if filtered:
		for key in go_term:
			if '' in go_term[key]:
				go_term[key].remove('')

	return go_term


def load_pickle(file):
	with open(file, 'rb') as f:
		return pickle.load(f)
```

APPENDIX A

```
def save_pickle(file, data):
	with open(file, 'wb') as f:
		pickle.dump(data, f)

################################################################################
##############

def C(n, r):
	while len(C.fac) - 1 < n:
		C.fac.append(C.fac[-1] * len(C.fac))

	return C.fac[n] // C.fac[r] // C.fac[n - r]

C.fac = [1, 1]


def progress_str(k, total):
	return "\r[ {} % ] ( {} / {} )".format(int(k / total * 100), k, total)
```

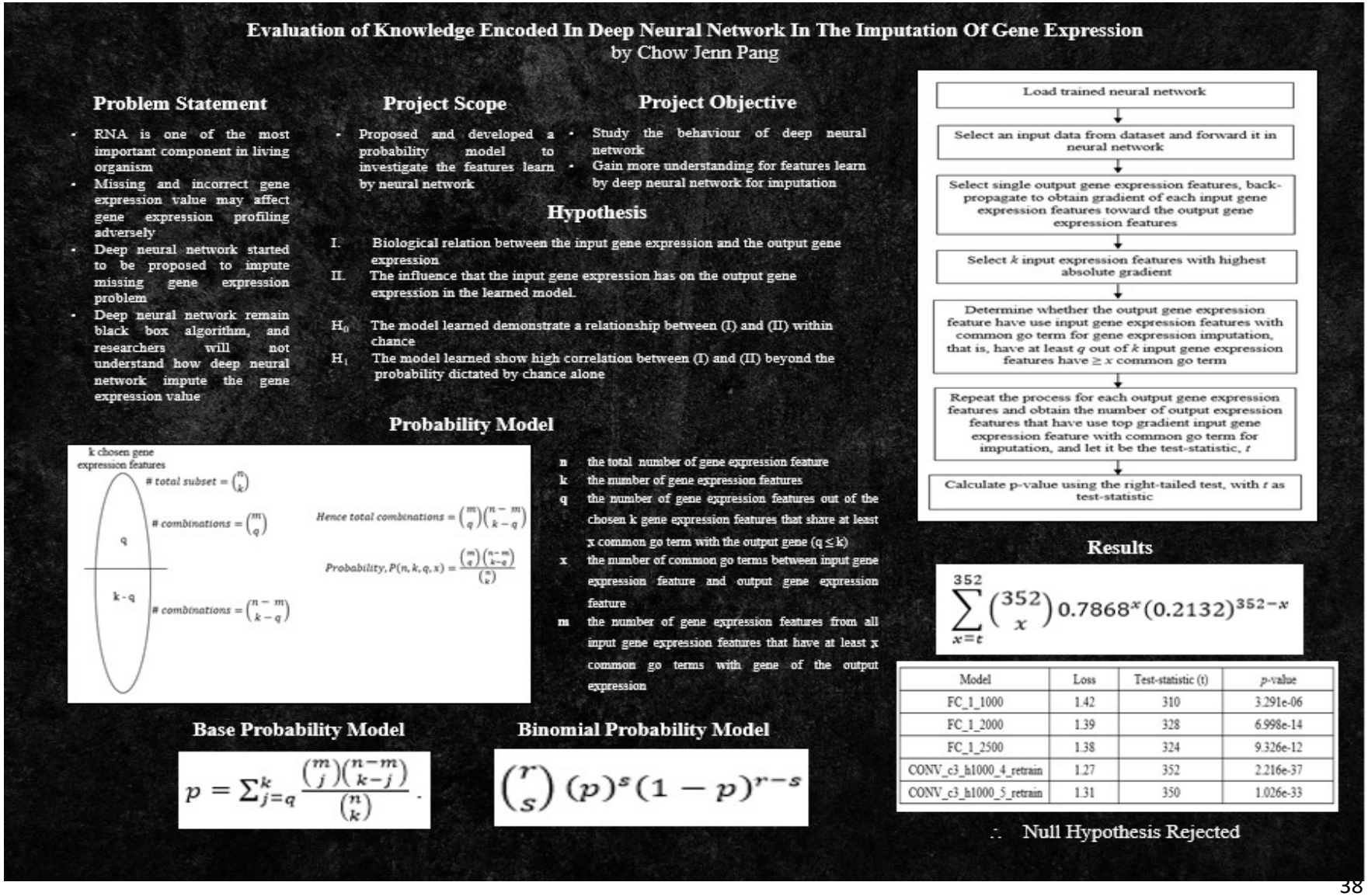
A-5. download_go_term.R

```
library('biomaRt')

mart <- useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")
gene <- strsplit(readLines(file('gene_id', 'r')), ',')[[1]]

for (g in gene)
{
	print (g)
	results <- getBM(attributes = c("go_id"), filters = "ensembl_gene_id", values
= (g), mart = mart)
	writeLines(unlist(results), file(paste('full_go_term/', g, sep = '')))
}
```

APPENDIX B


Evaluation of Knowledge Encoded In Deep Neural Network In The Imputation Of Gene Expression
by Chow Jenn Pang
Problem Statement
RNA is one of the most important component in living organism
Missing and incorrect gene expression value may affect gene expression profiling adversely
Deep neural network started to be proposed to impute missing gene expression problem
Deep neural network remain black box algorithm, and researchers will not understand how deep neural network impute the gene expression value
Project Scope
Proposed and developed a probability model to investigate the features learn by neural network
Project Objective
Study the behaviour of deep neural network
Gain more understanding for features learn by deep neural network for imputation
Hypothesis
I. Biological relation between the input gene expression and the output gene expression
II. The influence that the input gene expression has on the output gene expression in the learned model.
H0 The model learned demonstrate a relationship between (I) and (II) within chance
H1 The model learned show high correlation between (I) and (II) beyond the probability dictated by chance alone
Probability Model
k chosen gene expression features
total subset = (n k)
combinations = (m q)
q
k - q
combinations = (n - m k - q)
Hence total combinations = (m q)(n - m k - q)
Probability, P(n, k, q, x) = (m q)(n - m k - q) / (n k)
n the total number of gene expression feature
k the number of gene expression features
q the number of gene expression features out of the chosen k gene expression features that share at least x common go term with the output gene (q ≤ k)
x the number of common go terms between input gene expression feature and output gene expression feature
m the number of gene expression features from all input gene expression features that have at least x common go terms with gene of the output expression
Base Probability Model
p = Σ_{j=q}^{k} (m j)(n-m k-j) / (n k)
Binomial Probability Model
(r s) (p)^s (1 − p)^{r−s}
Load trained neural network
Select an input data from dataset and forward it in neural network
Select single output gene expression features, back-propagate to obtain gradient of each input gene expression features toward the output gene expression features
Select k input expression features with highest absolute gradient
Determine whether the output gene expression feature have use input gene expression features with common go term for gene expression imputation, that is, have at least q out of k input gene expression features have ≥ x common go term
Repeat the process for each output gene expression features and obtain the number of output expression features that have use top gradient input gene expression feature with common go term for imputation, and let it be the test-statistic, t
Calculate p-value using the right-tailed test, with t as test-statistic
Results
Σ_{x=t}^{352} (352 x) 0.7868^x (0.2132)^{352−x}
Model | Loss | Test-statistic (t) | p-value
FC_1_1000 | 1.42 | 310 | 3.291e-06
FC_1_2000 | 1.39 | 328 | 6.998e-14
FC_1_2500 | 1.38 | 324 | 9.326e-12
CONV_c3_h1000_4_retrain | 1.27 | 352 | 2.216e-37
CONV_c3_h1000_5_retrain | 1.31 | 350 | 1.026e-33
∴ Null Hypothesis Rejected

FYP 2 Report

ORIGINALITY REPORT

5%	1%	5%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Liew, A. W.-C., N.-F. Law, and H. Yan. "Missing value imputation for gene expression data: computational techniques to recover missing data from available information", Briefings in Bioinformatics, 2011. Publication	1%
2	Amit Paul, Jaya Sil, Chitrangada Das Mukhopadhyay. "Gene selection for designing optimal fuzzy rule base classifier by estimating missing value", Applied Soft Computing, 2017 Publication	1%
3	Rui Xie, Jia Wen, Andrew Quitadamo, Jianlin Cheng, Xinghua Shi. "A deep auto-encoder model for gene expression prediction", BMC Genomics, 2017 Publication	<1%
4	Dino Kečo, Abdulhamit Subasi, Jasmin Kevric. "Cloud computing-based parallel genetic algorithm for gene selection in cancer classification", Neural Computing and Applications, 2016	<1%

	Publication	
5	Barbara Calabrese. "Data Cleaning", Elsevier BV, 2018 Publication	<1%
6	www.jstage.jst.go.jp Internet Source	<1%
7	Priyadarshini Panda, Abhronil Sengupta, Kaushik Roy. "Energy-Efficient and Improved Image Recognition with Conditional Deep Learning", ACM Journal on Emerging Technologies in Computing Systems, 2017 Publication	<1%
8	Erzin, Y.. "Artificial neural network models for predicting soil thermal resistivity", International Journal of Thermal Sciences, 200810 Publication	<1%
9	www.denizyuret.com Internet Source	<1%
10	M. Pittore, M. Cappello, M. Ancona, N. Scagliola. "Role of image recognition in defining the user's in 3G phone applications: the AGAMEMNON experience", IEEE International Conference on Image Processing 2005, 2005 Publication	<1%
11	"Statistical Analysis of Next Generation	

	Sequencing Data", Springer Nature America, Inc, 2014 Publication	<1%
12	journalofinequalitiesandapplications.springeropen.com Internet Source	<1%
13	Yi Shi. "Classification Accuracy Based Microarray Missing Value Imputation", Bioinformatics Algorithms Techniques and Applications, 08/09/2007 Publication	<1%
14	Yulan Liang. "Bayesian Finite Markov Mixture Model for Temporal Multi-Tissue Polygenic Patterns", Biometrical Journal, 02/2009 Publication	<1%
15	Kapur, Arnav, Kshitij Marwah, and Gil Alterovitz. "Gene expression prediction using low-rank matrix completion", BMC Bioinformatics, 2016. Publication	<1%
16	Laura L. Elo. "Predicting Gene Expression from Combined Expression and Promoter Profile Similarity with Application to Missing Value Imputation", Modeling and Simulation in Science Engineering and Technology, 2007 Publication	<1%
17	Kshirsagar, M., J. Carbonell, and J. Klein-Seetharaman. "Techniques to cope with	<1%

missing data in host-pathogen protein interaction prediction", Bioinformatics, 2012.
Publication

---

18 D. Menétrey. "Retrograde tracing of neural pathways with a protein gold complex", Histochemistry, 1985 <1%
Publication

Exclude quotes	On	Exclude matches	< 8 words
Exclude bibliography	On		

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF _____

Full Name(s) of Candidate(s)	
ID Number(s)	
Programme / Course	
Title of Final Year Project	

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: _____ % **Similarity by source** Internet Sources: _____ % Publications: _____ % Student Papers: _____ %	
Number of individual sources listed of more than 3% similarity: _____	
Parameters of originality required and limits approved by UTAR are as follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

_____
Signature of Supervisor
Name: _____

Date: _____

_____
Signature of Co-Supervisor
Name: _____

Date: _____

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	
Student Name	
Supervisor Name	

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Cover
	Signed Report Status Declaration Form
	Title Page
	Signed form of the Declaration of Originality
	Acknowledgement
	Abstract
	Table of Contents
	List of Figures (if applicable)
	List of Tables (if applicable)
	List of Symbols (if applicable)
	List of Abbreviations (if applicable)
	Chapters / Content
	Bibliography (or References)
	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
	Poster
	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report. _____ (Signature of Student) Date:	Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction. _____ (Signature of Supervisor) Date: