A Deep Learning Approach for Classifying Pathology of Cancer Cell Lines using Vector Representations of Gene Expression and Attribute Data

by Michael George Hirezi

B.S. in Biochemistry, August 2016, Louisiana State University

A Thesis submitted to

The Faculty of
The Columbian College of Arts and Sciences
of The George Washington University
in partial fulfillment of the requirements
for the degree of Master of Science

May 19, 2019

Thesis directed by

Qing Zeng
Professor of Clinical Research and Leadership

© Copyright 2019 by Michael George Hirezi
All rights reserved

Abstract

A Deep Learning Approach for Classifying Pathology of Cancer Cell Lines using Vector Representations of Gene Expression and Attribute Data

Machine Learning and Big Data are two burning topics which have not avoided the contemporary surge in accession of large-scale biomedical data. A variety of technologies and research platforms, such as the efficient employment of next-generation sequencing technologies, ChIP sequencing, or mass spectrometry pipelines, have allowed for the gathering of biochemical information capable of forming genetic, epigenetic, or proteomic answers to crucial research questions. The challenge, however, is with the continued and effective use of these technologies there will come the need to better make sense of and manage the data that is produced. The concept of machine learning aims to address this challenge, and its applications in biomedicine will no doubt be formulated as these advanced computational methods are refined by the wave of data scientists moving into related fields such as Bioinformatics or Genomics. In this paper, a use-case of deep learning - a subset of machine learning - is explored onto cancer cell line gene expression and other genetic data to perform a simple classification task: is the sample ‘primary’ or ‘metastatic’ in pathology? The genetic data is transformed into a word-embedding using the popular word2vec set of algorithms which is subsequently used to convert sample information into images recognized by a Convolutional Neural Network (CNN). A classification accuracy of 98% is reported and a forward-thinking discussion held on the importance and advantages that deep learning contains when compared to traditional machine learning and statistical applications in Bioinformatics.

Table of Contents

Abstract ……..... iii

List of Figures ……..... v

Chapter I: Introduction ……..……..... 1

Chapter II: Background …….……..... 7

Chapter III: Methods ……..... 16

Chapter IV: Results ……..... 38

Chapter V: Discussion ……..... 47

Chapter VI: Conclusion ……..... 51

Bibliography or References 53

List of Figures

Fig 1: The Artificial Neuron and Network.…….…… 10

Fig 2: The word2vec word-embedding……..…..... 14

Fig 3: Significance Analysis of Microarrays Plot……...……..…….. 22

Fig 4: Sample of the text corpus input ……..……. 28

Fig 5: A visual representation of the word-embedding produced for the final model… 38

Fig 6: Visual representations of cancer cell line genetic expression and attribute data.. 40

Fig 7: A comparison between a single versus double CNN architecture…….……. 42

Fig 8: A comparison of Keras optimizers…….. 44

Fig 9: Classification accuracy of the final model…….. 46

Chapter I: Introduction

The topic of machine learning can be fundamentally understood as the study or implementation of computational algorithms and models that are effectively capable of improving their performance through experience and reiteration. When the performance of a given machine learning method is considered, it is within the scope of some sort of task or function such as prediction or classification. The performance is also often evaluated in the context of exceptionally large datasets – now typically identified under the umbrella term 'Big Data' - and the motivation is typically to circumvent the requirement of human readjustment to prospective models in accordance to the challenges related to the complexity or sparsity of these types of data [1]. It supposes that the underlying trends in the data cannot be feasibly evaluated using human expertise or annotation alone, making machine learning tasks appealing as a supplement to research, and eventually practical application. Regular and commonly referenced tasks that are used to introduce the concept include logistic regression analysis, k-Nearest neighbors classification, and the splitting of datasets using decision trees [2]. Due to its ability to handle data in a manner that is now possible thanks to its computational methods, the field of machine learning is no doubt the fastest growing within the Data Sciences, and machine learning tasks such as prediction or classification have been compounded to form the framework for some of the most promising, breakthrough technologies of this era. For example, the concept pervades our daily lives, and machine learning methods form the basis for speech recognition through mobile device or home-connected assistants like 'Siri' or 'Alexa' [3], or even the recent

advent of the practical autonomous driving vehicles being developed by competitors Uber and Lyft [4].

Machine learning's applications are far-reaching, and the concept is in no way confined to an exclusive domain. It has found relevant uses in a diversity of fields such as Economics [5], Medicine [6], Marketing [7], Linguistics [8], Video Game Development [9] or even the Arts [10], for the purpose of this project, however, the applicability of machine learning will be focused on Bioinformatics. Interest concerning the potential usefulness of machine learning within Bioinformatics is attributed in large part to the exponential growth in both the retrieval and organization of biologically relevant data such as proteomic, genomic, or epigenomic data. Examples of the sort of data that have become recently available include gene / protein sequencing or arrays to measure abundance or expression, ChIP sequencing to characterize protein interactions with DNA, and methylation or acetylation profiles to better understand the epigenetic basis of gene expression control [11]. This growth can be owed to advances in, as well as the efficient employment of, technologies such as mass spectrometry, immunoprecipitation, ionization technologies, and next-generation sequencing technologies [12]. The vast amount of heterogenous data that are bound to be produced from the unremitting use of (and not exclusive to) the aforementioned methods will continue to amplify the need to compile and make sense of them through capable techniques and methods. The idea should be that all of the data be used in tandem to form a holistic understanding of the underlying biochemical systems that might define a particular research question of interest. The prospect of machine learning aims to begin to fill that gap, and it is plausible to think that with more investment, particularly and

recently vocally from influential entities like Amazon and Google [13], the support to handle and make sense of genomic datasets will become more prominent as well as the task become more manageable at an industrial scale. From an academic perspective, and throughout the course of years, there have been a variety of attempts to expand upon the ways in which machine learning tasks could be applied in the context of the Bioinformatics discipline [14][15]. Some varieties of the spectrum of applications include prediction of protein secondary structure using near-neighbor methods, clustering of genes using expression data, text-mining protein/gene information, and detection of early stage cancer using mass spectroscopy data. The possibilities of use at this time appear to be as endless as its canvas implies, and the field continues to be explored for its essential usefulness in Bioinformatics and its related disciplines.

Of particular interest in this study is the area that explores alternative uses and analyses of gene expression microarray data. Microarrays are glass slides onto which single-stranded DNA are ligated at specific, predetermined sites [16]. Each site should designate a relationship to a target gene through apparent sequence identity. This entails that the technology takes advantage of the nature of complimentary binding of nucleic acids, where what is often designated as a ‘probe sequence’ is used to capture a complimentary ‘target sequence’. These probe-target pairs are what are demarcated by the different sites on the array. Perhaps the most common measure derived from this technology is the determination of relative mRNA abundance between two experimental groups such as a disease state versus a non-disease state [17]. The two distinct groups, for example, are typically labeled with different fluorescent dyes, so that when the array is excited by a laser, the population at each probe site that is of higher abundance should be

qualified given the color that is emitted. The values of expression for the previous method are determined by measuring light intensities, and there are other variations of the technology described above which are capable of providing gene expression data in a different form. For the evaluation of transcriptome data, RNA-seq is often appointed in preference to other methods due to its demonstrated advantages in accuracy [18]; the method does not measure abundance with respect to light intensity, but rather it counts the number of reads which correspond in identity to a reference after a step called alignment. The measure is reported in terms of counts of aligned reads to a reference. Regardless of the method used however, the data that is produced is rich in information, often profiling more than 5000 gene loci per sample chipped, and can be used to investigate meaningful biochemical and computational/statistical relationships between genes or networks of genes. Given the premise of this project, it is argued that machine learning techniques are one prospect of use for the type of data produced from gene expression microarrays. This is in no way one of the first explorations into the subject. Most early demonstrations and evaluations of the efficacy of a machine learning approach applied to gene expression data tend to consist of supervised classification tasks or feature selection methods. For example, in [19] a method to classify genes by function is presented using a supervised machine learning model called a Support Vector Machine (SVM) where classes are delineated in two-dimensional space via a hyperplane. Classes should lie on different sides of the hyperplane representing genes of different functionality. More recently [20] the efficiency of a variety of classification methods were compared, including SVMs, when applied to eight sets of two-class microarray datasets.

In this summary of research, a novel representation and use of gene expression data is presented for a deep learning classification task. Normalized Affymetrix microarray gene expression data for cancerous cell lines, as provided by the Broad Institute and through the Cancer Cell Line Encyclopedia (CCLE), are first transformed into word-embeddings, or vector representations of words, using a well-documented class of Natural Language Processing algorithms termed word2vec. The data structure for the embedding produced by word2vec consists of a list of words or concepts represented in the model as well as their associated vectors which should describe the semantic relationship between each word based off of a measure called cosine similarity. The dimensionality of these vectors can be reduced to two dimensions using a technique commonly applied to high-dimensionality data called t-Distributed Stochastic Neighbor Embedding (t-SNE). The two-dimensional version of each concept's vector can be treated as an x-coordinate value and a y-coordinate value on a coordinate plane and can be afterwards plotted into a scatterplot image which, given the premise of the word-embedding, should contain specific spatial information regarding the semantic relationships of the concepts captured through the word2vec output and the unique locations for each concept on the plane. The images produced in the previous step are then placed into a Convolutional Neural Network (CNN), a type of artificial neural network whose function is suitable to perform visual recognition or classification tasks on data whose format are images. A commonly referenced application of the CNN, particularly in Biomedical applications, is the implementation of this form of network for pattern recognition on medical imaging data such as Computed tomography (CT) images or X-ray images. For example, in [21] it is used to distinguish lung patterns on CT scans

obtained to investigate and presumptively diagnose interstitial lung disease. In this case, The Convolutional Neural Network is asked to distinguish between two types of images given each image produced for each cell line. As mentioned previously, the image that is presented to the network is a product of natural language processing and is meant to be an abstraction - perhaps a signature - of the meaningful genetic attributes that describes the cell line under question. It should be mentioned that there are a variety of ways that each cell line can be labelled, such as by tissue type or primary site, however, it was chosen to use the labels ‘primary’ and ‘metastasis’ which refers to the pathology of the sample prior to immortalization of that line. The distinction works as an appropriate and basic measure of the usefulness of the following deep learning classification task. The accuracy of classification of the method is presented and other nuances encountered throughout the process are touched on throughout the course of the project. The subject of machine learning is vast and still exploratory, and it is understood that this project could have taken a variety of courses of analyses given the same methodological setup. This report seeks to highlight a use-case for the application of deep learning onto large-scale genetic information; the context applied is within Oncology, however this methodology is in no way restricted to the classification of cancer or pathology of cancer.

Chapter II: Background

Deep Learning

The proposed method employed in this study can be qualified as a Deep Learning approach; and a study of deep learning cannot be initiated without an understanding of Artificial Intelligence (AI). AI is the term used to describe the prospect of machines to be able to function in ways other than explicitly programmed. Machine Learning, a subset of AI, can be viewed as one case of implementation of this computational model, where performance is improved quasi-autonomously with experience. As for Deep Learning, any reference to it is a reference to a more specific description of Machine Learning. In one way, Deep Learning can be viewed as a subset of machine learning that learns the basis of abstract or lower-level features that are assumed to come together to form higher-level features [22]. This method takes advantage of the observation that most natural measures tend to follow this trend, particularly if thought about in terms of the way people process the same types, but perhaps simpler or more abstract forms, of information. The concept is often demonstrated by using image recognition as an example [23], where combinations of motifs in an image can be learned and reassembled into larger hierarchal pieces. For example, if we wanted to distinguish between the faces of two similar-looking individuals, we might be inclined to first look at the outlines of some of their features, such as their nose, ears, and eyes. It might be noticed that the bridge of one individual's nose is curved where the other is more edged; this observation would qualify as an example of an abstract or lower-level feature and would be subsumed under some hierarchy of higher-level features such as the bridge itself, and then the nose, and then our final object of interest, the face. The logic is intuitive and there have been

prior successful explorations into the applications of Deep Learning for facial recognition, using bootstrapped images of celebrities that were identified with up to 98.95% accuracy through Convolutional Neural Networks, the same final class of algorithm employed in this study [24].

Deep Learning can be viewed as distinguishing itself from other machine learning methods by its design or architecture, which also instinctually finds its inspiration from the workings of logical human cognition. Particularly, deep learning methods are comprised of what are called Artificial Neural Networks (ANNs). These ANNs are formulated from their basic unit, the *artificial neuron* (Figure 1A), similar to the biological neurons that are observed naturally in Neurology [25]. The anatomy of an artificial neuron can be observed as first taking some values as inputs, where each input should represent some feature from the data or previous layer in the model. Each input is assigned a synaptic weight which transforms the value of the input based off of its perceived strength of connection to the prospective output. These weights are initially randomly assigned and they are adjusted to reflect the true strength of that input as training continues through iteration. At a particular neuron the sum of all of its inputs after being modified by its weight are calculated and passed through an activation function which confines the output to some value which either meets or does not meet a threshold that would decide whether the neuron be fired and be used as input for the next layer in the model. The activation layer also works to induce non-linearity to the model. It can be thought of as representing how a sufficient action potential is required to initiate transmission of a neurological signal down an axon. If the threshold for activation is not

met, then the signal is not induced, supposedly because the information that would have been sent does not warrant sensory attention.

Neurons are aligned together in layers, and information moves from one layer to the next given the premise of what is described above. Figure 1B illustrates how neurons are organized into layers and how the functioning of each of these units are aggregated together to form the final architecture of the artificial neural network. Typically, there are three component layers that must exist to form a complete ANN: an input layer, a hidden layer, and an output layer. The purpose of the input node is to receive information from outside of the confines of the model. No computation is performed in this layer and its function is to pass on the information onto the first hidden layer and there forth. The hidden layers should act in accordance to the properties of an artificial neuron as described above, and there are variations to the sort of functions that each hidden layer perform which are responsible for the bulk of customizability and exploration related to architecture design in Deep Learning model creation. Lastly and logically, the output layer specifies the final outputs of the model. The same basic foundations of design should form the crude architecture for the diverse classes of Neural Networks employed in machine learning practice. In the case of this report, two different Artificial Neural Networks are used to transform and analyze the data used. First a class of neural network models that, in most cases is dispatched for pure natural language processing tasks, is used in an unconventional way to handle gene expression and attribute data converted to discretized values. These models called collectively as word2vec are a component of the preprocessing step of this project. The output of the word2vec model is subsequently used as input into the second ANN which is of the class Convolutional Neural Networks

(CNNs). Convolutional Neural Networks are utilized to parse through image data as input. The genetic data is transformed throughout the process to resemble images so that they can be used as input in the CNN.

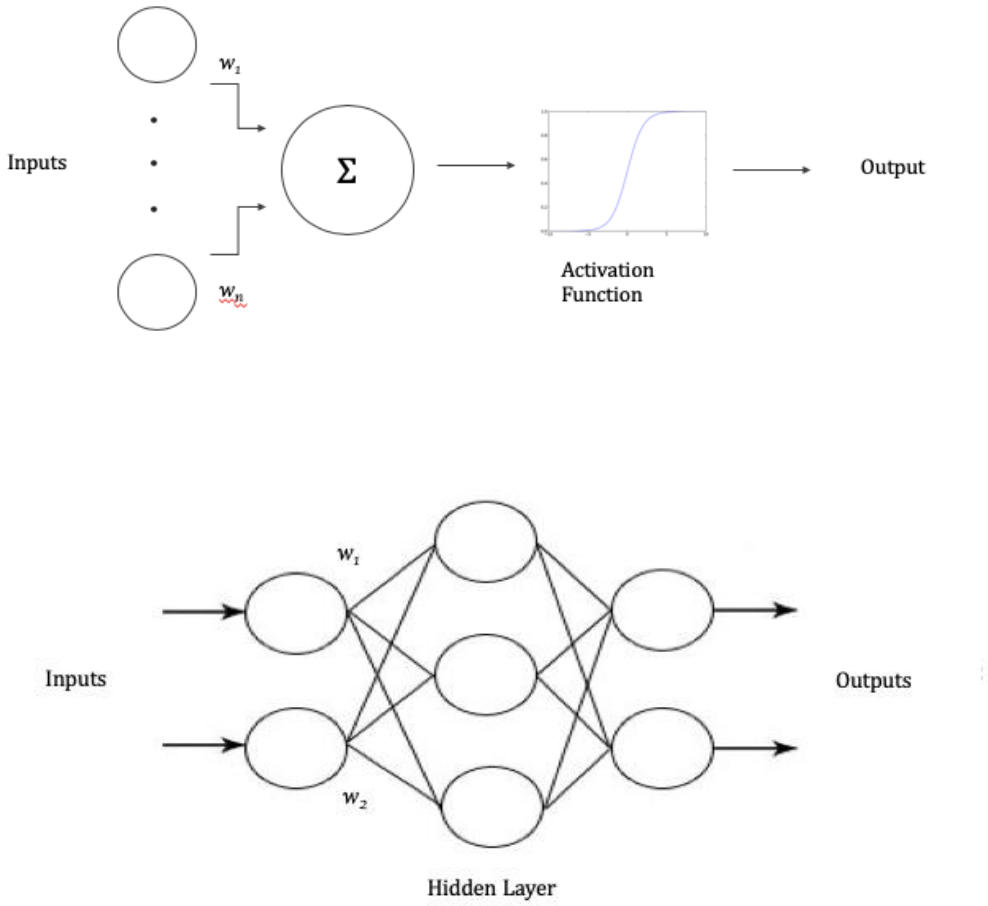


Figure 1

Deep learning models are composed of artificial neural networks. The basic component A.) is the artificial neuron which takes a weighted sum of its inputs which are transformed using an activation function. If a threshold is met, that neuron is fired and produces an output similar to the biological activation potential. B.) The neurons are stacked together in layers to create an artificial neural network.

Natural Language Processing and word2vec

Natural Language Processing (NLP) pertains to the division of data science that seeks the advanced comprehension of natural human language. Natural language processing tools take human language in its form as data and typically attempts to understand the semantic relationships between words in the same way a human would try and perform the same task. Just like with the aforementioned applications of machine learning, this sort of approach should be able to address the challenges that derive from exceptionally large amounts of data, or vast bodies of language in this case. For example, in a biomedical context, there is considerable interest in trying to utilize NLP for the curation of meaning from Electronic Health Record systems [26]. In this report, a natural language processing method is used to transform the data into a form called a word-embedding, or vector representation of words. These vector representations of words are imperative for the processing of natural language, where computational tasks are able to be performed on the unique vectors assigned to each word. The computations applied to these vectors can then be traced back to the word that they represent. This is particularly useful within the framework of a machine learning methodology.

There are a variety of natural language processing tools available and easily accessible from within a Python or Linux environment. A good overview on the diverse set of applications as well as the way in which these tools can be utilized is summarized in the book [27]. word2vec is implemented to convert a list of genetic information into a word-embedding, where each concept used is the natural language-converted form of some measure of genetic expression, or some genetic aberration, i.e. an amplification, mutation, or deletion. word2vec was first introduced by a group of data scientists from

Google Inc [28]. The preceding publication reported the efficacy in estimating semantic word representations in vector space using this neural network model that they devised. The model converts each word into a vector, where semantic meaning is implicated by the cosine distance between two vectors. Using this method, relationships like Athens / Greece or apparent / apparently where discovered and demonstrated as evidence for the potency of the model.

The word2vec class of algorithms are defined by their essential neural network architecture [29]. It works by assigning what is termed a ‘one-hot’ vector or encoding for each word. The one-hot encoding entails that each vector is n-dimensions, where n is the number of words present in the vocabulary. The values of each distinct vector are all-zero except for a single numerical “1” value which designates the word in the context of its index within the vocabulary of the text body that is being processed. This ensures that each vector represents one word and that each vector is unique within the model’s framework. These vectors are used as input into an artificial neural network consisting of an input layer, a hidden layer, and an output layer.

The network can be conceptually thought of as being designed to perform one of two prediction tasks; that is, either to predict a word given a context, or predict a context given a word. These variations of the model are referred to either as the Continuous Bag of Words (CBOW) or Skip-Gram models, respectively. The model should produce a new vector for each word, where similar vectors in terms of cosine similarity should represent words closely related given the prior questions asked. What that means is that vectors of similar values should correspond to vectors likely to be in the same context. The same assumption is made in this project, where the natural language text that is used is

obtained by the transformation of numerical gene expression and other genomic data into a natural language format. It is hypothesized, given the prior success of the word2vec-related algorithms in performing similar natural language processing tasks, that an application of word2vec onto the genetic data should preserve the relationship inherent between concepts when converted into vectors. That is, the purpose of converting each word into a vector is to preserve some relationship of context. Those relationships are captured by plotting each concept onto a plane using its vector assigned from the output. Each concept's position on this plot is achieved by the reduction of the multidimensional vector output into two-dimensional vectors using a technique called t-distributed stochastic neighbor embedding (t-SNE). The two dimensions of the final vector are used as de facto x-coordinates and y-coordinates on the plane. Those x-coordinate and y-coordinate values are formed into an index, so that a list of genetic attributes for each cell line can be paired to their appropriate positions on the plot. Given these premises, it can be presumed that clusters of concepts in one spot on the final image should represent clusters of related concepts. The assumption was demonstrated on a more a robust and human-interpretable dataset prior to initiation of this project (Figure 2) which consisted solely of medications given across different patient encounters. The image below reiterates the observations made through prior publications. To put it into perspective, the reason for creating these images is to be used as input into the Convolutional Neural Network, whose task is to identify motifs, such as perhaps the clusters of related concepts alluded to above, that might offer predictive power in classifying between our two labels 'primary' and 'metastasis'.

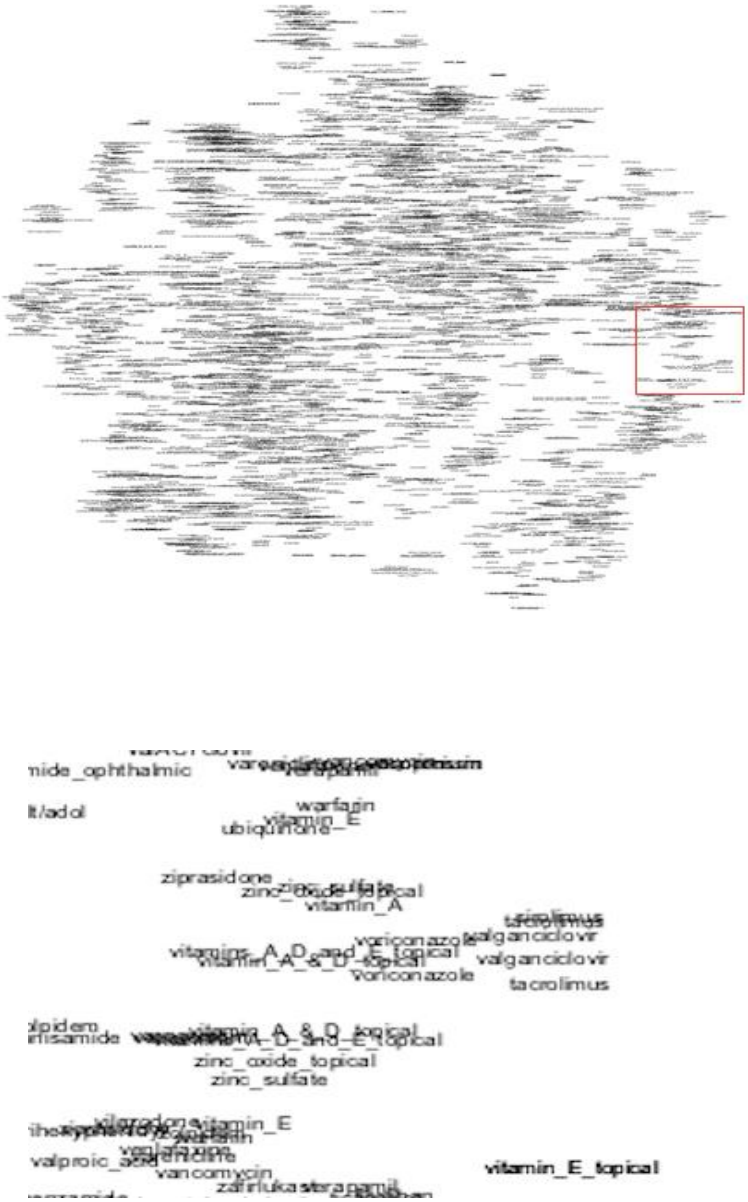


Figure 2

A.) a word-embedding produced using the word2vec algorithm is visualized using its two-dimensional vector values as prospective x- and y-coordinate values on a coordinate plane. B.) Concepts semantically related should appear close in distance on the plot, such as the different vitamins and dietary supplements taken from a section of the visualization above.

Convolutional Neural Networks

Convolutional neural networks are a class of Deep Learning algorithms that take data in the form of images, and assigns learnable weights to features of that image which are detected within the framework of an artificial neural network architecture [30]. The usefulness of this algorithm is that it is able to discern features of importance for segregating different classes or types of images based off of the weighting process. A CNN is fitted in this report to perform a classification task on images in the form of two-dimensional scatterplots - either labeled ‘primary’ or ‘metastasis’ – that are produced using an index derived from a word-embedding created in the prior step. The distinctive component of a convolutional neural network is the convolution layer, which applies what is called a filter or kernel across different portions of the image. A filter can be thought of as a small matrix area of observation over the image input. The filter’s function is to find patterns of significance across all three channels (RBG) of the portion of the image it is current observing. Patterns are matched by calculating the dot product of the filter matrix over the actual pixel values of the region it is observing. The result of this function is used as input for the proceeding hidden layer. Given this overview, different filters are weighted to identify different motifs present in the image. Through the workings of the artificial neural network, features of importance should be determined through training and reinforcement. This sort of neural network architecture is applied here to learn the subtle patterns distinct between our two labels which could help the model perform its eventual classification task.

Chapter III: Methods

Gathering and Management of Data

The core premise of this study is to explore the utility of deep learning methods when applied to aggregated microarray gene expression and other gene attribute data. This data is combined together from the same source but different experiments, preprocessed to resemble natural language, transformed into images, and then used to fit a CNN model that will perform a classification task. The labels used in the classification describe the pathology of disease, where a given sample is either labeled 'primary' or 'metastasis'. In order to achieve this goal, and given well-documented issues regarding the management of genetic data such as interoperability of experiment-types or a small ratio of samples per features [31][32], a relatively consistent and complete dataset was required. The data that was chosen for this project was obtained from the openly available Cancer Cell Line Encyclopedia (CCLE) resource that is managed through the Broad Institute. The CCLE which was launched in 2004 as a collaboration between MIT and Harvard with the stated goal of improving human health through leveraging genomics to understand the mechanisms and applied treatment of human diseases. It is a continuously updated resource funded by Novartis that offers both an online platform and a mirror with publicly available downloadable data. The CCLE project aims to characterize what is currently 1,457 commonly used cancer cell lines in biomedical research. The downloadable versions of the data are made publicly available for internal research purposes. Acknowledgement for use of the data is requested in the Terms of Access, as well as a declaration that the original collection and analysis of the data should not influence any subsequent conclusions such as those derived in this separate report.

Three separate datasets were downloaded from the resource for use in this project. They include Ensemble annotated, gene-centric, RMA-normalized mRNA expression data across 18,875 gene loci, miRNA expression data across 684 loci, and binary calls for amplification, deletion, and mutation data across the same loci present in the first set. All three sets were made available in Gene transfer format (.gtf) and were accessible via a text processor for preprocessing. The .gtf format is a standard for viewing gene structure data, and is discerned by a metadata section catered to include the appropriate gene description information. All expression data was retrieved from Affymetrix U133+2 arrays which are a standard chip technology developed by Thermo Fisher Scientific – US and present values of expression in terms of light intensity. It was judged reasonable to use data obtained from the same microarray platform with consideration to the known influence that batch effects, or technical / experimental means of error, raise due to the use of data obtained from different microarray technologies or different research centers or sources [33]. For example, there is still contemporary evidence that suggests that microarray data obtained using different technologies, and in compliment to experimental error, can contribute to results that are not comparable [34]. This interrogates the question of whether samples that are retrieved from diverse sources are truly interoperable within a research setting. To try and minimize any impact that batch effects might have on the conclusions of this paper, it was felt appropriate to both choose data obtained from the same source, and retrieved using Affymetrix chip arrays. This decision sacrificed a greater aggregation of data, which could have resulted in an experimental dataset with more samples.

Normalization of the data was also a topic considered given the prior concerns regarding comparability [35]. The gene-centric dataset was already prepared and made available using a method called Robust Multi-array Average normalization (RMA-Normalization), however, the miRNA expression data required similar manual normalization. The process was performed in accordance to the same normalization method as applied to the gene-centric data. RMA-normalization is a method designed particularly for the adjustment of Affymetrix gene expression data and its usefulness is demonstrated in [36]. With respect to the actual method, it can be methodically described as the application of quantile normalization for each gene and across all samples requiring normalization, followed by a log2 transformation. A log2 transformation is apparently used due its tendency to fit the fold-change patterns exhibited by most genes when examined for differential expression. RMA-normalization on the miRNA set was executed using the pandas and numpy modules in Python.

The .gtf metadata sections were removed using an open source text-processor and the sets were subsequently converted into a tab-delineated .csv file where the rows represent the values of expression for a particular annotated gene locus, the columns cancer cell line samples. The rows in the last dataset are given as binary calls for each gene position per sample, where a gene aberrations, defined in this case as either a deletion, amplification, or mutation, are called as either a “0” or “1”, representing the absence or presence of a specified aberration, respectively. To reiterate, this dataset only covers the gene loci present in the gene-centric dataset and there are no calls for the miRNA loci. Of the cell lines that shared features in all three sets there were 955 out of the 1457 lines annotated through the CCLE resource.

An index was obtained from the CCLE resource which contained supplementary information for each cell line, as well as a variety of descriptive labels such as the ethnicity, gender, pathology, and tissue-type. At this stage an appropriate label was chosen for the classification task that would be examined in this project. Due to the relatively small amount of sample sizes available compared to the number of features present as gene loci represented and their respective attributes, the binary label of pathology was chosen to be used in our deep learning approach for both its simplicity and its meaningful characterization within an Oncological setting. A more complex labelling structure would significantly diminish the number of samples represented per label and would either risk demonstrable overfitting or amplify the amount of noise in the model. Another binary label such as Gender would run the risk of not being able to be delineated by a classification task. 671 of the 955 samples noted above were of primary pathology and 284 were metastatic. As is explained later, feature-reduction via differential expression analysis and binning are also employed in part to reduce the number of features relative to our number of samples available, an important consideration when working with datasets that follow the aforementioned trend. A balance between the number of samples captured per label, as well as the features included in our model is sought out throughout the course of the method. The features that are kept for fitting the final model are those which are considered of having some importance. The magnitude of importance is not necessarily predetermined through the feature-reduction step given that our final model should examine motifs of feature in image format rather than single features at a time.

Feature Reduction: Selection of concepts of interest

In order to reduce the numerous feature space that is inherent to genomic data, there must be some criteria to select genetic attributes of probable importance to the model. The technique of reducing the feature space is properly termed feature reduction or feature selection. In this case, a common technique used for differential expression analysis was applied separately on both the gene-centric and miRNA expression data. The technique used employs a statistical technique designed particularly for use on microarray data and is called Significance Analysis of Microarrays (SAM). Introduced by [37], the technique reveals statistically significant changes in expression per gene by running through a specified amount of permutations that assign a score to each gene based on its change in gene expression relative to the standard deviation of repeated measurements. A cutoff or threshold, called the delta value, was chosen to select genes of interest and that value is predicted based off of minimizing the false positive rate. All genetic aberrations that covered the same gene loci determined through SAM were included.

The SAM technique has been converted to a tool via software by Michael Seo, Balasuubramanian Narsimhan, and Robert Tibshiani [38], and is accessible in a variety of media, including as an addon through Excel for more blunt analyses. For this project SAM was accessed using a web-based compilation of the common R-based version of the tool made available as a repository through GitHub by MikeJSeo. R was first setup in an Anaconda environment and the required dependencies were installed in that environment. The code used to develop the interface is made available through the interface and no alterations were made prior to use. Both the numerical forms of the

gene-centric expression data and miRNA data required reformatting in order to be recognized by the SAM analysis tool. The type of format is dependent on the type of analysis performed. In the case of this study, a two class unpaired analysis is being initiated, where the two classes are primary and metastasis. The unpaired descriptor refers to the fact that the samples which compose both groups are not the same. The first two columns are formatted so that the first provides an index g1...gx where x is the number of rows or genes made available in the analysis. The second column is simply the name of the gene or gene loci. A header must also be added where a binary label, in this case "0" or "1" will designate which group that current column or sample belongs to. SAM assumes that the data is already normalized, and a normalization which included a log2 transformation must be specified due to application of the RMA-normalization technique. All other parameters were kept the same and the analysis ran through 50 permutations to get the result.

For both datasets, the SAM tool was run without consideration for the false positive rate. The parameter which defines the threshold as described above is called the delta value, and the value was kept at 0 so that all genes are considered significant. A plot is produced to visualize the predicted versus actual score for each gene as seen in (Figure 3a). A table is provided which estimates the false positive rate based off of a given a delta value and the value was subsequently adjusted to reduce the false positive rate when determining genes of statistical importance (Figure 3b). There were 402 genes which met the final criteria. Their identities were saved into a .csv file and used later to filter out those genes from the primary tables as retrieved from CCLE.

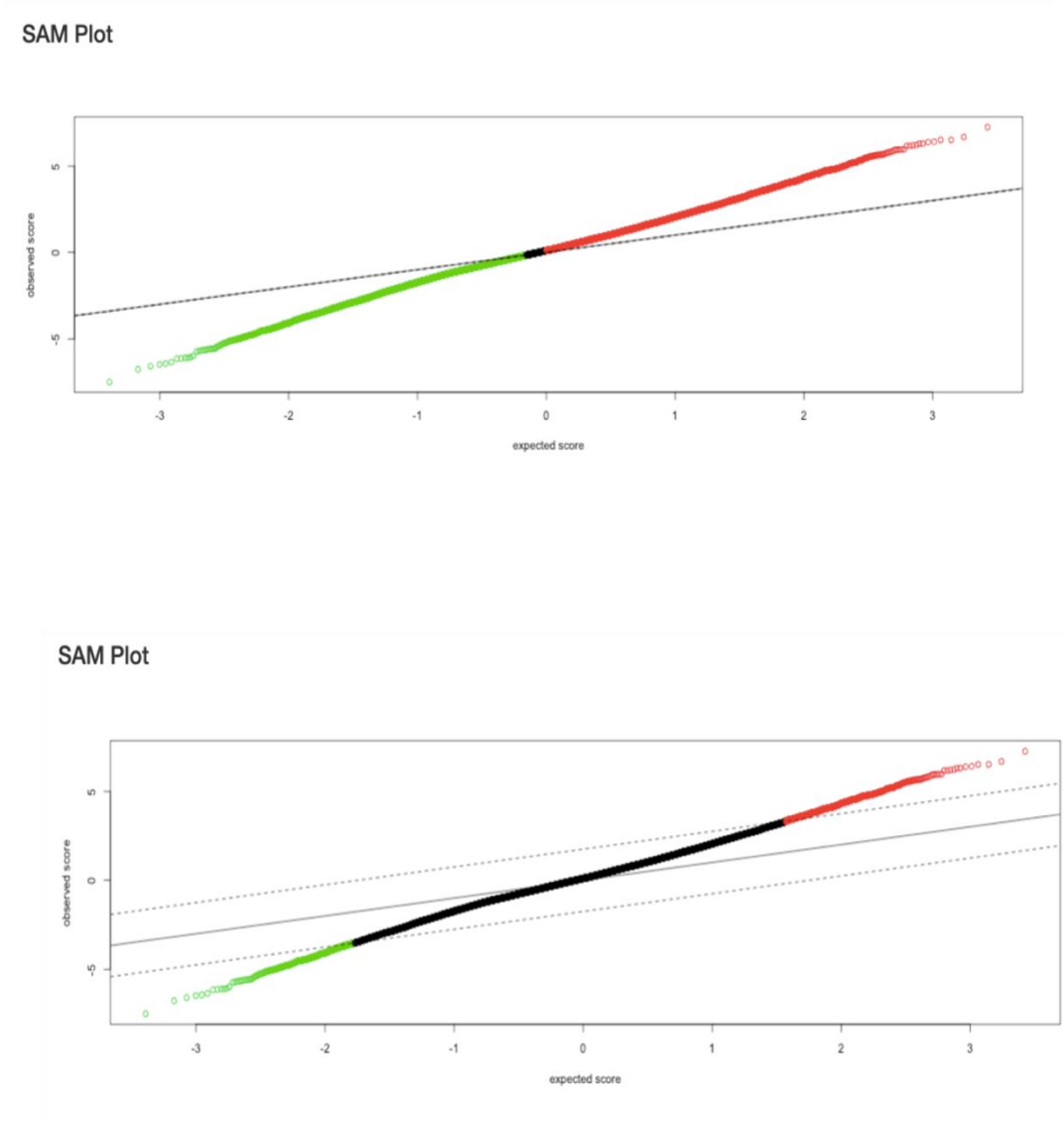


Figure 3

Genes of significance are plotted using SAM's observed score versus expected score all genes in the dataset. Positively differentially expressed (red) and negatively differentially expressed (green) genes are demarcated by a window established by the delta value. The delta value was initially set to zero A.) and then was readjusted so that the estimated false positive rate was 0 B.) 402 genes met the final criteria.

Converting the data to Natural Language

The class of word2vec algorithms process natural language presented as a text body or 'corpus' as input and produce an output of vectors, where each vector corresponds to one word or semantic concept as presented in the text. The obvious prerequisite to use this algorithm is to present the data in a form that resembles a body of natural language. This preprocessing step was required and executed for all three of the datasets downloaded from the CCLE resource.

The Affymetrix gene expression data was processed the same for both the gene-centric and miRNA sets after RMA-normalization was applied for the latter in the manner described in the previous section. In order to convert the numerical values of expression to natural language it was devised to discretize the values into three binned categories designated by the following patterns. First, the method of discretization refers to the categorization of numerical data into a form that generalizes but still attempts to retain the original meaning of that value in its context. It is commonly seen in the form of categorizing expression values as overexpressed, underexpressed, or normally expressed [39]. In this project, each gene is qualified in this same form. Consider in the example below, where 'gene' refers to the current row of the set or the current gene of interest: The values of expression were converted to either gene_HIGH, gene_LOW, or gene_MEAN dependent on whichever of the bins that value lies within. It is implied then that the concept of creating bins, or binning, must be introduced prior to discretization, and it is accomplished by deciding upon ranges of upper and lower limits in which numerical values can be categorized. Due to the premise of discretization, each value in the set must be put into a bin, in such a way that abstracts the inherent meaning of the

original numerical value while still converting it into a form that resembles natural language. It is not to presuppose that binning and discretization are used solely for natural language processing tasks, but the general idea, which is to put the data in a format more easily readable or accessible, is certainly related in this case. The reason why discretization is required is to make each numerical value accessible to a natural language processing tool. The discretization terminology used in this project is basic and intuitive where, for example, the term 'A1CF_HIGH' under the sample column 'HEPG2_LIVER' would imply that the A1CF gene was determined to be relatively overexpressed for this cell line as determined by the range qualified as being high for the gene through the binning step.

Given the nature of binning, it is a method that needs to be defined by the user. The topic of binning in the context of expression data, particularly for the sake of discretization, has been explored in detail and a variety of methods are introduced in Gallo et al (2015). Only one binning method was explored in this project and it is based off of a binning method postulated in accordance to the mean of a set of expression values for a gene plus or minus one unit of standard deviation as originally investigated in [40]. The reasoning derives from the Three Sigma Rule as illustrated in Pukelsheim (1994) [41]. The Three Sigma Rule is a statistical conclusion that supposes that the probability that some value X within a unimodal distribution will deviate away from its mean by more than three units of standard deviation should be less than 5% or 0.05. The observation is often translated to what is dubbed the 68-95-99.7 rule or the *empirical rule* for unimodal distributions. Where each numerical value in the term 68-95-99.7 refers to the probability that some value X lies within one, two, or three units of standard

deviation, respectively. For example, given the aforementioned proposition, there is a 68% chance that some value X will lie within one standard deviation from the mean. This fundamental inference was chosen as the binning method for its simplicity, and was used to bin and discretize the values for the genetic expression data used in this report. The only difference between that explained in Madiera and Oliviera (2008) and that used here is that two units of standard deviation from the mean – rather than one - were specified to qualify the outer limits of the normal or '_MEAN' bin as described above for each gene. This implies that the mean bin is predicted to contain 95% of the expression values. Any value above or below this bin is contained in the '_HIGH' and '_LOW' bins, respectively. These two alternative bins are estimated to carry the other 5% of outlier or extreme values given the definition of the Three Sigma Rule, and the method of binning just described. The reasoning to make the definition of an extreme value more stringent was to reduce the amount noise due to the large number of gene features represented in our data while still retaining variables of probable significance. The underlying hypothesis for what is significant is that gene expression values that fall within the extreme or outlying bins for that gene loci represent divergence from normal expression patterns due aberrations that ultimately might be a product of the genomic hallmarks of metastasis [42]. That is that genomic aberrations are responsible drivers of metastasis. This does not entail that all features represented are significant, and the proposition is that the final model should be able to discover and distinguish between features or groups of features of importance through a classification task.

The actual process of binning as described above was performed using the numpy and pandas modules in Python 3.7.1.

The same code was executed for both the RMA-normalized gene-centric and miRNA data and the output is a .csv file that forms a list of each gene and its corresponding upper and lower limit for the mean bin. This output is used as subsequent input for a script which takes the original numerical microarray data and converts each value in each row to its corresponding discretized values as designated by the bins created in the prior step.

Because the last set used for this project is comprised of only binary calls “0” or “1”, no binning step was required. Each concept per sample was simply converted to natural language depending on its value. A value of “1” is recognized in the code for each gene position within each column and was converted into gene_MUT, gene_AMP, gene_DEL if present, where ‘gene’ refers to the current row or identity of the gene of interest at that position. The reasoning for this processing step is the same as for the expression data; the data is being converted into a format that both abstracts the underlying semantic meaning of the information presented while displaying it in a matter that will mimic natural language when used as input for the word2vec model. Absence of any of the three types of features for a given sample were omitted from the final model so that only positively-called aberrations were captured. The reasoning for this is also the same as prior, where redundant features are removed and the number of features reduced so that features of probable significance to the model are retained. This point is emphasized, particularly because it is necessary to approach the project in this manner. The main purpose of the project is to demonstrate the efficacy of this method in being able to discern between primary versus metastatic pathology of cancer lines given the transformation of the data into an image representation through a word-embedding

intermediary. If the model is successfully demonstrated, then it can afterwards examine to obtain accessible information regarding the relationships of concepts that are determined to influence the classification function.

Using word2vec and t-SNE to create vector representations of words

Up to this point, gene expression and attribute data were converted to natural language based off of a binning method as described in the previous section. In order to be used as input into our word2vec algorithm, however, the data must be presented in the form of a text corpus, or otherwise a body of text as natural language. A typical example of a text corpus that can be used for word2vec applications would be a novel, where the algorithm would function to find semantic relationships between the words that comprise the vocabulary of that novel. It should be mentioned that the vocabulary is a variable determined by the model and does not necessarily include every word available in the corpus. Thresholds can be specified to define the inclusiveness of the vocabulary used. In order to create a proper corpus, the data prepared in the previous step is converted to a .txt file that is formatted so that each sentence in it serves as a description of all of the revised genetic features for a given a sample, e.g. gene_HIGH, gene_LOW, gene_AMP, gene_MUT, gene_DEL. For example, a section of the final text corpus is presented in (Figure 4).

Figure 4

A section of the final text corpus used as input to the word2vec algorithm. Discretized values for high or low expression, as well as positive calls for mutations, deletions, and amplifications are listed for each cancer cell line sample as a sentence which is evaluated as a context in the word2vec algorithm.

As seen in line or sentence ‘2’, the first two words in the list are nmiR00257.1_LOW and DCLK1_HIGH. This portion of the sentence can be interpreted as the current sample was found to have the miRNA nmiR00257.1 underexpressed as well as the gene DCLK1 overexpressed. The identity of the sample is left anonymous for each sentence at this stage, particularly because only the relationships between the concepts are meaningful for the creation of the word-embedding. Consider the gene DCLK1 from the example sentence now; for the purpose of creating the embedding, it does not matter which

sample has overexpression of this gene. What is important is that each sentence only contains information regarding one sample, so that the relationships between concepts in that sample can be determined by the underlying context-word question as described in the Background. Still considering that DCLK1 is the current concept of interest, what is being evaluated, rather, is how many times other concepts appear in the same context as overexpressed DCLK1 across all samples. So if underexpression of nmiR00257.1 is often in the same context as overexpressed DCLK1, it could be assumed that the cosine similarity between the two output vectors for DCLK1_HIGH and nmiR00257.1_LOW would be close, ultimately representing semantically similar concepts.

In order to create a text corpus in the format defined above, the csv and re (regex) modules in Python 3.7.1 were used to parse through the discretized expression and binary call .csv tables. Concepts of interest were recognized through regex pattern matching, and the concepts that passed the matching phase were written into the .txt file. A new line is established in the output .txt file after each iteration through a sample column as identified by the current_sample variable. The re module, which allows the user to establish regular expressions to match with strings of text is utilized to find matches of text that should not be included in the corpus. For example, all gene_MEAN terms were excluded from the corpus, as well as binary calls that were absent for a particular aberration such as a mutation, amplification, or deletion. What remains in the corpus are gene_HIGH, gene_LOW, gene_MUT, gene_DEL, and gene_AMP information for each sample column. It is in this step where the numerical values that represent the binary calls are identified and all values of "1" are converted to its corresponding natural language term as shown above. The final corpus should contain 955 sentences total, related to the

number of cell lines or samples that contain information from all three of the datasets used in this project. The sentences will vary in length and that length is dependent on the number of different genetic attributes of interest that are present for that cell line. Each sentence should be read as a list of these genetic attributes.

A customizable word2vec algorithm is available through the Gensim package, publicly available for use in Python. Gensim is licensed under the OSO-approved GNU LGPLv2.1 license and is available for both personal and commercial use unless any modifications are made to the original source code. No modifications were made in this study. The package contains a word2vec module which includes a variety of parameters and class that can be specified or called through script to fit the type of corpus or model prepare or desired. It was imperative that the class genism.models.word2vec.LineSentences was initialized in the code. This specifies that the algorithm will iterate over a text corpus that contains sentences, where one line is synonymous to one sentence. This entails that a contextual evaluation, which is the underlying function of word2vec, is being performed over each sentence.

The output of the genism word2vec algorithm are multi-dimensional vectors for each word, where vectors closer in distance should be words closer in semantic relatedness. This is in accordance to the derivation of the algorithm over the text corpus. The dimensions are reduced to a dimensionality of two – a presumptive x-coordinate value and y-coordinate value – using a technique called t-Distributed Stochastic Neighbor Embedding (t-SNE). This technique's intended use is to reduce high-dimensional data for the purpose of visualization. It is a variation of Stochastic Neighbor Embedding, a dimensionality reduction technique that works in a way that

preserves neighbor identities between vectors prior to reduction. T-SNE is available for application in Python through the commonly used Sci-Kit (sklearn) package and is called as 'from sklearn.manifold import TSNE' in the code. Following the conversion of each multi-dimensional vector to a two-dimensional vector, the values were normalized so that the range of both x and y values are restricted between -1 and 1. This will provide a cleaner and standardized visualization for the next step. The x- and y-coordinate values assigned to each vector or concept are combined to form an index which is used to create a 2D scatterplot visualization for each cell line.

Creating visualizations for each cell line

The index created in the previous step is used to match to the discretized data prepared in the first. It should be reiterated that each concept represented in the word-embedding should contain its own unique x-coordinate and y-coordinate value, where the position of that concept on the plot should be related (through cosine distance) to other semantically related concepts. That is that semantic relationships are ascertained by an exact location for each concept on the final plot. The underlying assumption is that the images produced in this step should capture the characteristics of each cell line which are given a label of two types: 'primary' or 'metastatic' pathology. An image is produced for each cell line as a two-dimensional scatterplot where markers are filled on the plot for concepts that are matched with the output from the previous step. That is, referring back to Figure 4, all of the concepts present in that list are matched to their respective x-coordinate and y-coordinate values and are subsequently plotted. All concepts that are not contained in that sentence are omitted from the scatterplot that is performed. That is that

these images serve as a de facto signature for that cell line, and given the expectations of the word-embedding produced by word2vec, the spatial orientation, as well as the patterning or distribution of concepts for each cell line, should provide the information required for our Convolutional Neural Network to perform a classification task capable of distinguishing between primary and metastatic pathology.

In order to practically create the images, sub-indices were created for each type of feature represented, derived from the normalized word2vec output. Distinct indices were created for concepts that define gene_HIGH expression, gene_LOW expression, _MUT, _AMP, and _DEL. This is done so that these features can be treated separately during the plotting stage, where differences in colors of the marker on our final plot add descriptive value as to what type of a feature is present at that position on the plot. It is not deliberately measured in this study as to whether the presence of color has an impact on model performance or not. It is hypothesized that there should be minimal improvement in performance given that the location of each concept on the plot should be unique and representative of that concept. The advantage of including color, however, is for visualization of the interrelated patterns and types of concepts present. The colors were designed so that the colors red / green represent over / underexpression, the color blue represents aberrations, and the colors pink / black represent features 95% likely to either occur in a primary /metastatic sample. The indices were created in Python 3.7.1 using pandas and regular expressions matching. Each concept should be present in two indices; one with the concept as the first column and its corresponding x-coordinate value as the second, and the other with the corresponding y-value in the second. This separation is used for clarity in coding given that the values are considered separately as distinct lists

when forming the graph. All indices were kept as pandas dataframes for the plotting step. Matplotlib, the commonly used 2D plotting library in Python, was imported for its pyplot module which contains easy-to-use functions, particularly for creating customizable scatterplots. The matplotlib.pyplot.scatter() function was used to create an image for each cell line with three parameters specified: the parameters x_cord and y_cord alludes to the list of all x-coordinate or y-coordinate values for the current image, and the color ('c') refers to the color_map which defines the color that each point should take. Each three of these lists are made simultaneously through the plotting function and just prior to creating the image. Because it is needed to produce 955 images and the data obtained from an extensible dataset, the script to create the images is designed to consider all attributes of the image including sample name, sample label, and output directory. The code is created so that all 955 images are produced in this step, titled properly, and sent to an appropriate folder as a final image. The code first grabs a sample cell line from an index which contains the cell line name, appointed to the variable sample_name, as well as all the data synonymous in identity to the concept names available through the normalized word2vec vector output. Indices which separate the cancer cell lines by pathology, dubbed 'ind_primary' and 'ind_metastasis' are also imported and the sample_name is matched so that a label of either 'primary' or 'metastasis' is assigned to this image to be created. The three required lists that will be used as parameters for the matplotlib.pyplot.scatter() function are initialized as blank and are filled in the next section of the code as the data is parsed through one concept at a time for the current sample of interest. The script is designed to look at one concept at a time and identify which index that concept belongs to. Recall that sub-indices were made in the previous step with the purpose of

distinguishing the type of concept under scrutiny. If there is a match to an index, it will append the appropriate x-coordinate value as specified by that index to the x_cord list, the y-coordinate to the y_cord list, and the corresponding color to the color_map. A figure size of 860 x 640 pixels is produced, and this value was determined due to its standard size. There is sufficient separation between points on the final plot which would presumptively otherwise hinder the model's ability to take advantage of the unique location each concept has on these plots. The axes were set to -1 and 1 to match the normalized vector values. Depending on the label designated at the beginning of the script, the image was saved to its appropriate folder with a title identical to its sample name. The script would then reiterate to capture the information of the next sample, replacing all of the aforementioned variables and lists with the new information for that sample, and ultimately, a new image. All 955 of the images were produced as a loop in this manner, and their form after production is the same as when used as input for the Convolutional Neural Network.

The Convolutional Neural Network architecture

The Convolutional Neural Network (CNN) is our last Deep Learning method applied in this project. Its function is to perform a classification task, where each of our images are given one of two labels with respect to their Pathology: 'primary' or 'metastasis'. The neural network was put together using Keras, an application programming interface written in Python and formulated for the easy implementation and modulation of neural networks such as CNNs. Keras requires to be run on a backend that supports GPU use, where a backend refers to a library of code that handles lower-level

functions required by a model-level library like Keras. With respect to the creation of deep neural networks through Keras, these lower-level functions include sorts like tensor products, or convolutions according to the documentation. There are three backend libraries compatible with Keras which include TensorFlow, Theano, and CNTK. TensorFlow was chosen as the backend for the Keras as a standard choice. Testing or evaluating performance comparisons between the three backends were not explored here. Keras and TensorFlow were integrated together within an Anaconda environment and executed using Python 3.6.8.

From keras.preprocessing.image the ImageDataGenerator class is first imported so that batches of images from the specified directory can be generated for the training and validation steps of our model-fitting step. None of the images were augmented using any of the parameters specifiable in the Keras documentation. Neither was there any exploration into the usefulness of resampling techniques on improving performance of the model. In order to construct a Convolutional Neural Network, the Conv2D layer must be imported. A MaxPooling2D layer is also used, which is common practice in CNN models to generalize on common features present in the image. It is not clear how a MaxPooling2D will affect the accuracy of the classification task, given that each concept should have a specific location on the plot, however the MaxPooling2D layers were kept nevertheless and without demonstrable hindrance to the performance of the model. Other types of layers imported and necessary for the creation of our model include Activation, Dropout, Flatten, and Dense layers.

The architecture of the neural network refers to the way that the layers are organized to form the entire model. There still remains a large element of

experimentation with respect architecture design and as well as being able to gauge the benefits of using one architecture versus the other. Two different architectures were implemented here. The first took on the form of three units of Conv2D-Activation-MaxPooling2D layers stacked, following a sequence of Flatten, Dense, Activation, Dropout, Dense, and then Output. The design was inspired by the popular ResNet architecture [43] which employs stacked units of Convolutional Layers prior to the model being introduced to alternative forms of layers such as Flattening or Dense layers. In contrast to the ResNet, however, there is no attempt to replicate the shortcut connection concept introduced by original publication.

The relu activation layer was used in all cases other than the output, which employs a conventional sigmoid function. The loss function used was binary_crossentropy, appropriate for a binary classification task. The second form of the architecture used six units followed by the same sequence of layers. They are dubbed the single and double architectures, respectively. The double showed better performance on images created using gene expression data only and was used for the final model. There are seven optimizer functions callable through Keras. They are RMSprop, SGD, Adagrad, Adadelta, Adam, Adamax, and Nadam. A performance test of each of these optimizers was done prior to fitting the final model. The RMSprop optimizer exhibited the best performance over 10 epochs and was chosen as the final optimizer for the model. 50 epochs are iterated in the fitting step with a batch size of 50 images. These hyperparameters were determined using random search across a range for each parameter, where changes in the training phase, e.g. learning rate, were observed by altering one hyperparameter at a time. The model was fitted using the

model.fit_generator() function, and the weights of the model can be saved and are recallable for further exploration into the output.

Chapter IV: Results

Vector Representation of Words

The overall pattern established through the final set of vectors can be visualized easily in Processing, a malleable language and software designed specifically for visual arts professionals who desire to develop a proficiency in coding. The visualization in (Figure 5) shows how each concept in the final embedding is spatially oriented in our coordinate plane so that the name of that concept appears in the same location as designated by its two vector values. Those two values are treated as presumptive x and y coordinates in this plane, which form the basis for creating our visualization. The output used to create this visualization is used as an index to create the two-dimensional images, where concepts matched in a cell line are plotted according to that index.

Figure 5 A visual representation of the word-embedding produced for the final model

Visualizing each cancer cell line

955 images were produced using the vector outputs for each concept as performed by the word2vec algorithm. The images are of 22 distinct tissue types. 671 of the images were saved into a folder which contained all primary samples, 284 of the images were saved into a folder which contained all metastatic samples. The directories were then split into training / testing datasets at a 70:30 ratio. Examples of how the images look can be seen below in Figure 6. Gene expression data is shown as red or green point to represent over or under expression of some concept. All other genetic attribute data is colored blue. In Figure 6, concepts are colored either black or pink for visualization of the clustering patterns of concepts 95% more likely to be present in a metastatic or primary sample. This is done to show the reader the adequate clustering of concepts that should be related in context. The image when used as input for the Convolutional Neural Network does not contain these additional colors. The word2vec model was coded so that all concepts represented in these images were included only if they appeared a minimum of 10 times throughout the corpus. This entails that the associations being derived between concepts is not determined due to relationships found in one sample or a pair of standout samples. Ideally with more future data, the threshold set at ten here can be further increased do determine strict relationships found broadly between the labels of choice for the model.

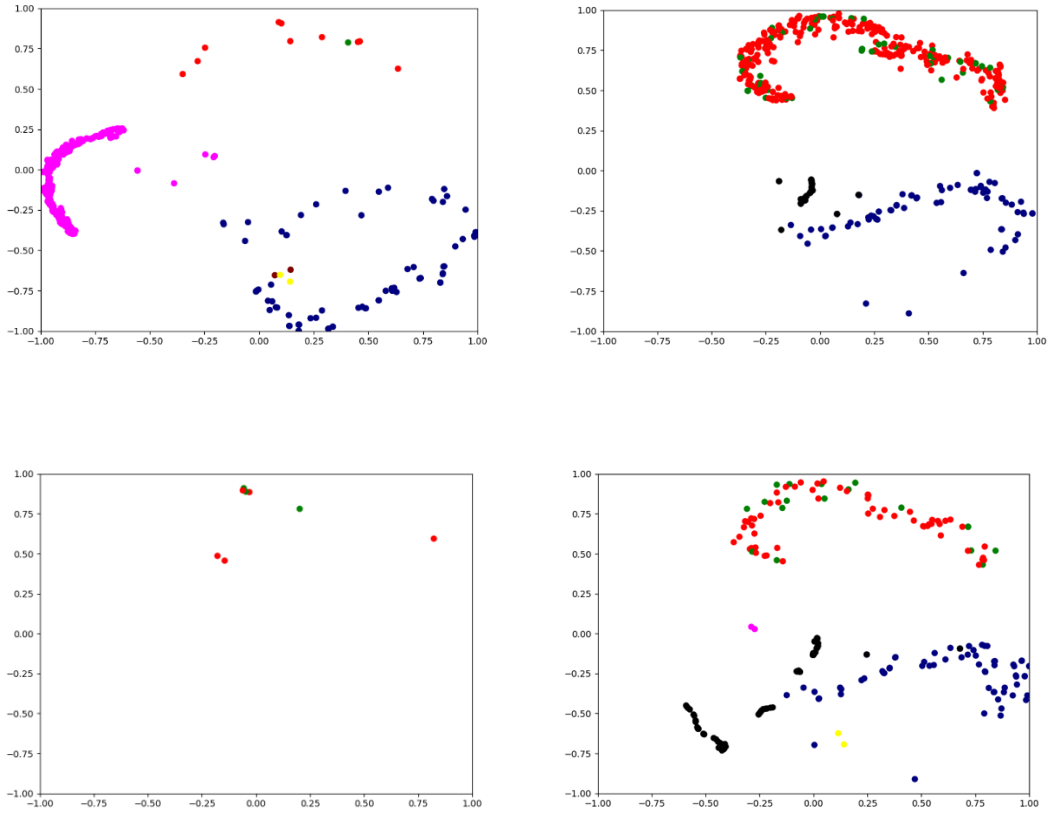


Figure 6. The two-dimensional vectors produced after the t-SNE reduction on the word2vec output are used as an index to produce a scatterplot. Gene expression data are colored red for overexpressed values and green for underexpressed values. All other genetic attribute data is colored blue. For visualization purposes concepts 95% more likely to be present in a metastatic sample or primary sample are colored black and pink for pairs of primary (left) and metastatic (right) A) lung tissue cancer cell line samples and B) osseous tissue samples.

Single versus Double Convolutional Neural Network Architecture

The single and double architectures as described in the previous section were tested for their performance against images produced using gene expression information only. The difference between both layer architectures is simply the number of Conv2D-Activation-MaxPooling2D stacks present. Other than the composition of these layer stacks, all parameters and settings were identical between both models. Ten epochs and a batch size of 50 were used. The loss function was binary cross-entropy, and the optimizer Adam which is the default. The comparison of performance of different optimizers are tested in a later step to choose the best optimizer for the final model.

As seen in Figure 7 the double architecture demonstrated a better learning curve, as well as a better final performance across 10 epochs. The learning rate seemed to cap off around the 20th epoch for the single architecture. The single architecture reached a classification accuracy of around 60% across 50 epochs where the double architecture performed markedly better at around 75%. It was chosen to pursue the double architecture for the final model because of these results.

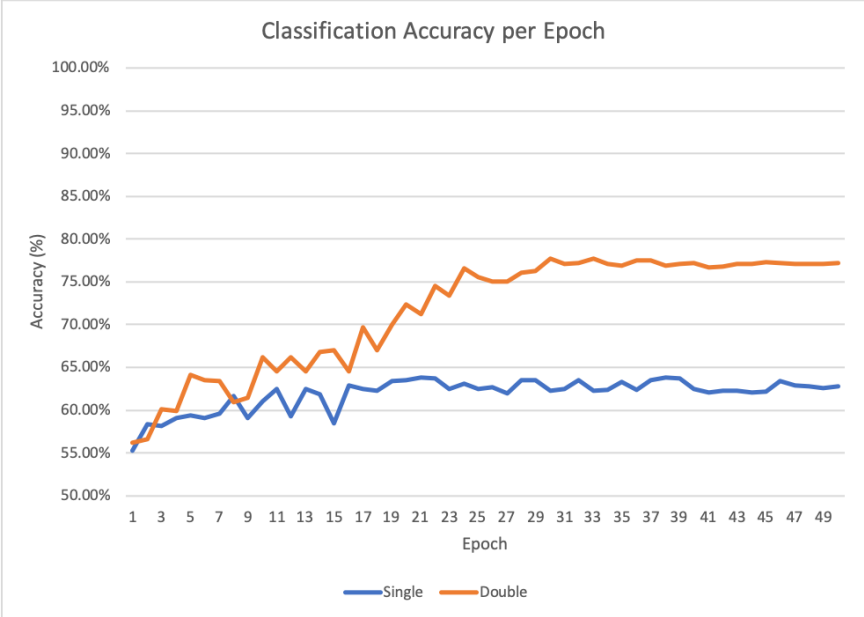


Figure 7

The classification accuracy is compared between a single versus double CNN architecture on a model produced using gene expression data only. The double architecture's performance was notably better over 50 epochs.

Hyperparameter search and comparison of optimizers

Hyperparameter search was used to find sufficient values for the epochs and batch size. This was done using a random search by testing each hyperparameter across a range of reasonable values. The default value for epochs was 10 and the default value for batch size was 50. The values 10, 30, and 50 were tested for epochs. The values 20, 50, and 100 were tested for batch size. The double architecture as defined in the previous section was used for the comparison. The optimizer is still Adam. Batch size was tested first, given the hypothesis that effects due to insufficient batch size are more likely to influence the general performance of the model when compared to epochs. This is because the number

of epochs is synonymous to the length of training. As a variable, length of training will only extend the model's performance given its inherent potential in the current learning phase. That is, given that all other variables are the same between two learning steps, 50 epochs should only be an extension of the same effective learning performance of 10 epochs. The worry is, however, that with an excessive number of epochs would follow the possibility of overfitting, one of the largest problems with highly complex computational models such as deep learning models.

Of the three values tested for batch size, a size of 50 demonstrated the best performance. Batch sizes of 20 or 100 showed poor performance in accuracy indicated that the proper number of images shown to the model per batch should lie somewhere in between.

As for epochs, the learning curve gradually improved through and after 30 epochs, however the change in rate started to decline shortly afterwards. Of the three values selected in the random search, it was determined then that 50 epochs were the best value for the final model.

The last parameter that needed to be considered was the optimizer to be used. All seven optimizers available through Keras were tested using the double architecture. A batch size of 50 was selected, and 10 epochs to observe a small window of learning. The learning curves and loss were compared between all seven trials (Figure 8). The RMSprop optimizer exhibited the best performance and was used in the final model architecture.

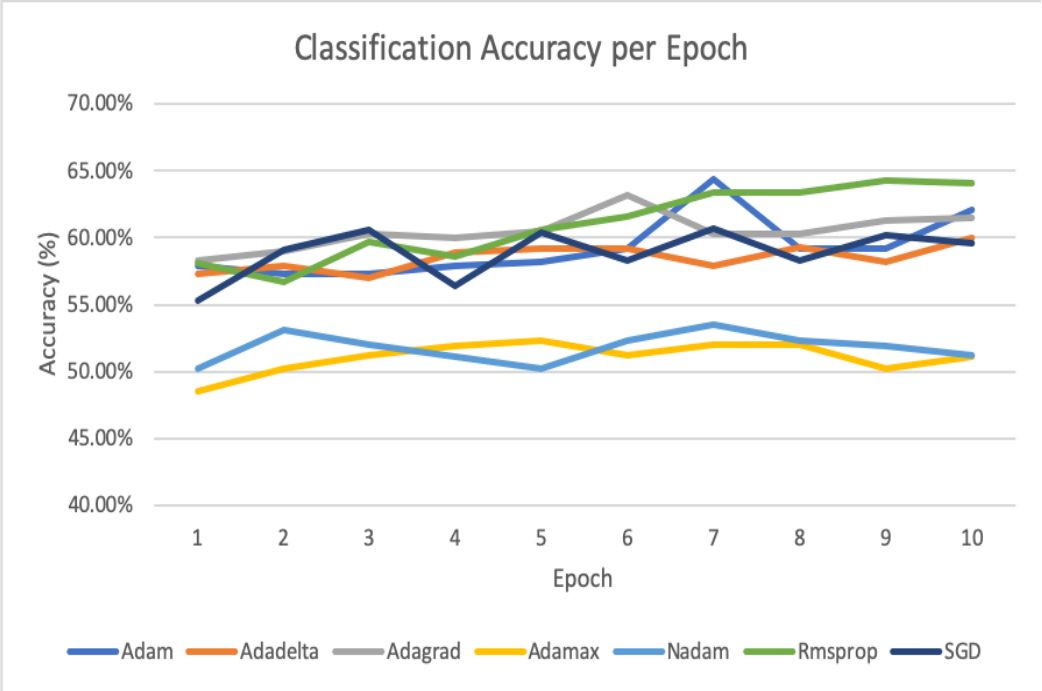


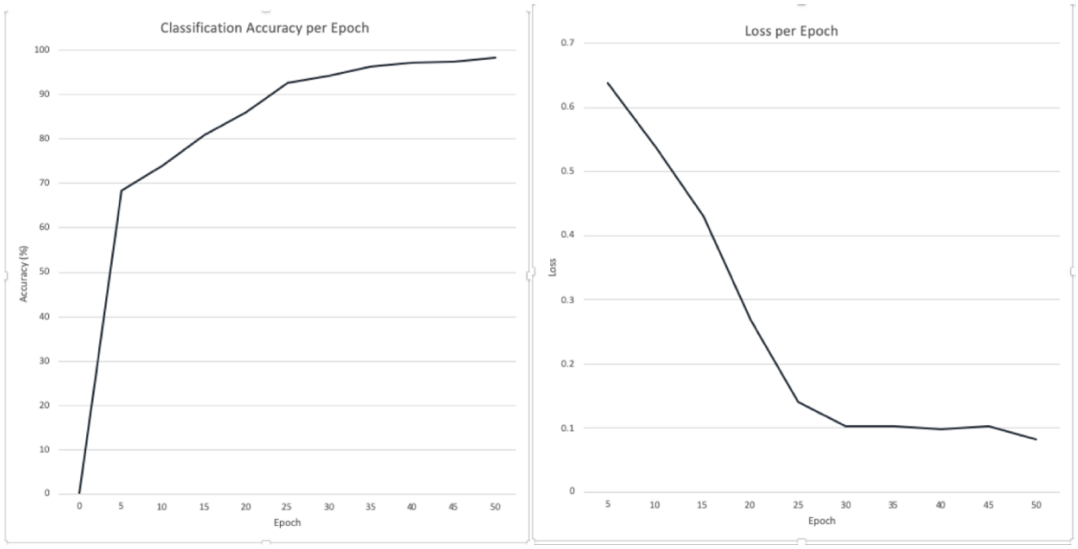
Figure 8

A comparison of optimizer performance using a double architecture. The classification performance was evaluated for a model consisting of gene expression data only across 10 epochs and a batch size of 10. The RMSprop optimizer as shown in green exhibited the best learning curve.

Classification Accuracy of Final Model

The final model took specifications according to the prior results. The double architecture as well as 50 epochs with a batch size of 50 were used. The optimizer function that was used was RMSprop. The final model included all gene expression, miRNA expression, as well as genetic attribute data defined in the methods section. This includes only the features determined as significant between both groups through the

SAM differential expression analysis tool. In performing the classification task, the final model reached an accuracy of 98% across the 50 epochs. This value is determined as an average between five separate runs initiated using different splits of the training and testing dataset. The learning curve as well as the loss over time can be seen in (Figure 9a). The performance for the final model was compared to the performance on two other sets of images produced using the same pipeline in the method section: one produced using only significant gene expression information, and one which used all features (Figure 9b). These two other models were selected for comparison for deliberate reasons. The former should compare the final model to the model used during the architecture design phase of the project. It is known that after 50 epochs the parameters of the model should achieve around 75% accuracy in classification if gene expression data is only known. This evaluates the importance of using the miRNA and other gene attribute data in supplement to gene-centric expression data. The latter proposes the inverse, where by including all features, a moderate amount of noise is being introduced. It is not expected that the final model should achieve good accuracy on these images, and this evaluation should provide a measure for the strength of the model. A high accuracy on the all-features image set would imply a high risk of overfitting. As seen below, there is a considerable improvement in the final model when compared to the other two. Learning improved at a considerable rate up until around the 30^{th} epoch. Interestingly, using all features tended to decrease performance of the model over time.



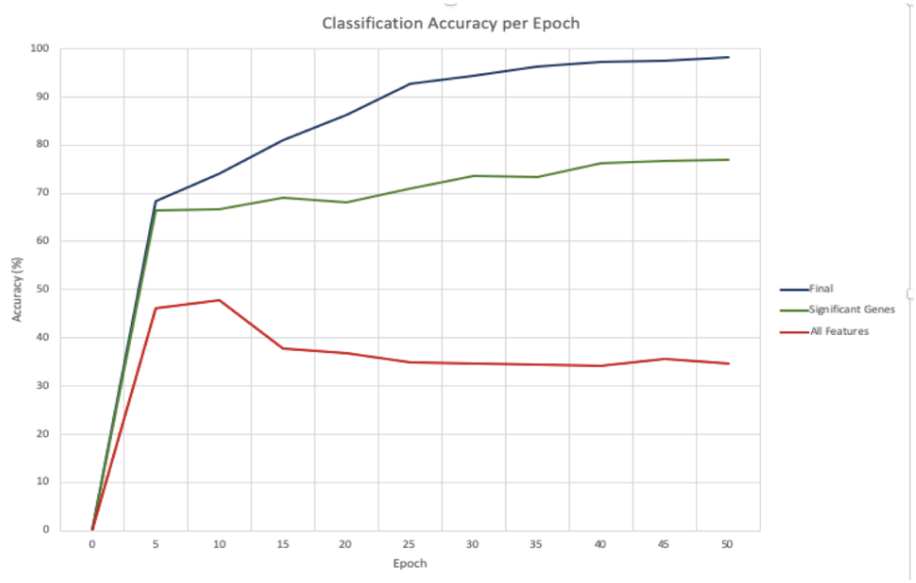


Figure 9

The learning curve of the final model is shown through classification accuracy as a percentage versus iterations through the training data (epochs). A.) The accuracy reached 98% after 50 epochs in the final model which included all significant genes and miRNA as well as their corresponding positive mutation, amplification or deletion calls. B.) The performance of this model is compared to sets of images using either expression data from significant genes only or all features.

Chapter V: Discussion

The results of the final model as demonstrated above exhibited remarkable performance as measured through classification accuracy on the testing set. However, the change in learning rate is shown to have capped off shortly after the 30th epoch of training. It can be argued that there is a clear demonstration of learning being performed by the Convolutional Neural Network. It would be supposed then, that the transformation of the data using the methods above, successfully handled the sparse and complex data in a way that produced features on the image that the CNN was able to weigh towards either a primary or metastatic label. Some of these features were evident as shown in (FIGURE A), where, for example, clearly demarcated clusters of concepts most likely to be present in either a metastatic or primary sample were produced. The learning curve showed a gradual uptrend across each iteration and follows a pattern indicative of a normal increase in performance through experience. The next question to ask, however, is due to the small sample size included in this study, particularly in relationship to the number of features present, is generalizability a possibility at this time? The innerworkings of neural networks, even more specifically CNNs, are not entirely well understood at this time, and it will require further analysis and insight in order to determine whether true learning is being performed with relationship to the underlying premise of the data transformation; that is, are more subtle clusters or points of interest being recognized according to the semantic relationship assumption explained previously? The first step to be able to perform these sorts of analyses would be to obtain a sufficient enough data so that all relationships between the meaningful gene concepts can be determined. It is hypothesized that due to the small sample size, and compounded due to the fact that a concept might

only appear at least 10 times across the dataset, and not necessarily always with other concepts it should be truly associated with, that there are meaningful, more subtle relationships that are not being captured by the model. That is that, perhaps even though the model might be able to generalize on broader more common features, it is not picking up on more rare relationships which could distinguish between a primary and metastatic state. This poses a complication not exclusive to this deep learning model, but gene expression data analyses in general.

One benefit of using a deep learning methodology, however, is that they are highly customizable and there are other components of the results that are accessible to the user for further analysis, such as the weights of the fitted model or the visualization of the word-embedding. In terms of customization, there are numerous considerations that the user must make when designing the architecture, or setting the hyperparameters of the model, and to be able to make these decisions warrants an explanation into how certain architectures might be better in certain environments or contexts versus others. In terms of being able to summarize the model, there should be a variety of avenues for retrospective analysis. For example, conventionally the user is easily able to gauge factors such as the weights of the final model in reviewing which features might have been important in influencing the accurate execution of the model's function. As the field of machine learning develops, it is predicted that there will be a subdiscipline specifically designed to explain the workings of deep neural networks for the sake continued analysis. The emerging field of *explainable artificial intelligence* holds this promise. There is currently research being performed through George Washington University's Biomedical Informatics Center which aims to introduce an

aspect of explanation over deep neural network models [44]. The research aims to formulate a measure of importance, dubbed the *impact score,* for each variable in the model by removing it from the samples and observing the effect on model performance. This same measure can be extrapolated to shed light on the gene concepts which are most meaningful for determining the classification outcomes of the model. Perhaps, given the specific nature of the word-embedding produced in this study, an impact score can be determined for specific genes, or clusters of supposedly interrelated genetic concepts as a novel method for gene discovery or gene network analysis and subsequent hypothesis testing. As the field of explainable artificial intelligence grows and the spectrum of applicable methods nurtures, the choice of a Deep Learning model as proposed here should offer a greater range of interpretability versus a more classic machine learning method.

Lastly, it cannot be avoided that the current classification task specified in this report has possibly minimal practical clinical application. That is, there are cheaper and more dependable methods to determine whether a lesion is of primary or metastatic pathology. As mentioned previously, one thing to bear in mind is that any label can be assigned to images. What might be of more importance is to classify cancers based off of available labels like tissue type or perhaps ethnicity. The binary label was chosen in part to keep the sample pools sufficiently large enough for fitting. What is of particular interest, however, is the application of this model onto a time-series context, and for the prediction (rather than classification) of metastasis given the sequential course of changes in patterns of the images over time. The underlying proposition, as stated numerously in this paper, is that the images should retain information about interrelated

concepts given the premise of word2vec algorithm and observed through unique points on the plot for each concept. If this holds true - and it is supposed that a larger sample set complimented with an explainable AI approach would be able to demonstrate the true efficacy of this method - then the subsequent application of the Convolutional Neural Network should be able to learn the broader patterns of features which describe primary versus metastatic pathology. By monitoring these plots over time, it is hypothesized that the time-series images should show one of two trends, a primary course or a metastatic course. Use of a deep learning model might offer the toolsets to make that sort of prediction given the image data. To emphasize this interest, the model employed in this study was tested on a more robust dataset prior to its application onto genetic / genomic data [45]. The dataset consisted of laboratory values for 304,573 clinical encounters retrieved from the Health Facts database. 101 laboratory tests were plotted for 2500 ICU and 2500 non-ICU patient encounters over slices of time; a three-dimensional image was produced in this case versus a two-dimensional image, where the third dimension corresponds to time. These temporal representations were placed into a CNN and performed a classification task similar to as described in this report. The model achieved 98% accuracy in determining whether the image it was viewing was an ICU or non-ICU encounter. Although still a classification task in this case, the usefulness of incorporating a time-series approach cannot be discounted, particularly given that the gathering of time-series gene expression data is on the rise and will require creative approaches to develop their analyses.

Chapter VI: Conclusion

In summary, what is demonstrated here is a novel deep learning approach for classifying pathology of cancer cell lines using vector representations of gene expression and attribute data. The performance of the model is demonstrated here with considerable performance, however more data, as well as better methods to explain how the model works are required to make a definitive conclusion regarding the efficacy of the model. What cannot be overlooked is the importance of the transformation of the data given the word2vec step. Through pre-processing, as well as the conversion of the data into a word-embedding, an easily-interpretable organization of the complexity and sparsity inherent to genetic data is performed. It must also always be asked why a deep learning model should be preferred over a more traditional machine learning or statistical method. Several reasons are highlighted above, including with an inspection of the usefulness of the word-embedding, as to why a deep learning approach offers an advantage, with the prospects of retrospective analysis and explainable artificial intelligence in mind.

There is a growing emphasis on the usefulness of potential integrations of machine learning applications within the Bioinformatics discipline. That interest is not exclusive to academia, and it can be currently observed as a critical talking point by leaders in a variety of fields and industries. The ultimate goal is the practical application of these methods for the betterment of their fields of use, and although a model like this might not see any immediate transition into practice, it provides a good framework for how genetic information can be utilized for both non-trivial tasks and information discovery. As the amount of available biological and omics data increases exponentially

in association to rising government and institutional intrigue, there will no doubt be a requirement to sift through the entirety of this information in an attempt to understand the interrelationship between the diverse set of features that will be tabulated. A study of deep learning illuminates this need, and the development of its methods is predicted to form a notable foundation to meet the future challenges that the field will face.

Bibliography

1. Chen, Y., Li, F. & Fan, J. Mining association rules in big data with NGEP (2015) 18: 577 https://doi.org/10.1007/s10586-014-0419-3

2. Harrington P. *Machine learning in action.* Shelter Island, NY: Manning Publications Co.; 2012.

3. Apple machine learning journal. https://machinelearning.apple.com/. Updated 2018.

4. Hermann, Jeremy and Del Balso, Mike. Meet michelangelo: Uber's machine learning platform. https://eng.uber.com/michelangelo/. Updated 2017.

5. Agrawal, A. Gans, J., & Goldfarb A. *The Economics of Artificial Intelligence: An Agenda*; (2019)

6. Rajkomar A, Dean J, Kohane I. Machine learning in medicine. *New England Journal of Medicine*. 2019;380(14):1347-1358.

7. Bose I, Mahapatra RK. Business data mining — a machine learning perspective. *Information & Management*. 2001;39(3):211-225.

8. Yogatama, D., d'Autume, C. de M., Connor, J., Kocisky, T., Chrzanowski, M., Kong, L., … Blunsom, P. (2019). Learning and Evaluating General Linguistic Intelligence. Retrieved from http://arxiv.org/abs/1901.11373

9. Dobrovsky A., Borghoff U.M., Hofmann M. (2019) Improving Adaptive Gameplay in Serious Games Through Interactive Deep Reinforcement Learning. In: Klempous R.,

Nikodem J., Baranyi P. (eds) Cognitive Infocommunications, Theory and Applications. Topics in Intelligent Engineering and Informatics, vol 13. Springer, Cham

10. Qiao T, Zhang W, Zhang M, Ma Z, Xu D. Ancient painting to natural image: A new solution for painting processing. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. January 2019:521-530.

11. Cho WCS. Proteomics technologies and challenges. *Genomics, Proteomics & Bioinformatics*. 2007;5(2):77-85.

12. Slatko BE, Gardner AF, Ausubel FM. Overview of next-generation sequencing technologies. *Current Protocols in Molecular Biology*. 2018;122(1):e59.

13. Sharon Begley. Amazon, google race to get your DNA into the cloud - Reuters. https://www.reuters.com/article/us-health-genomics-cloud-insight/amazon-google-race-to-get-your-dna-into-the-cloud-idUSKBN0OL0BG20150605.

14. Larrañaga P, Calvo B, Santana R, et al. Machine learning in bioinformatics. *Briefings in bioinformatics*. 2006;7(1):86-112. https://www.ncbi.nlm.nih.gov/pubmed/16761367. doi: 10.1093/bib/bbk007.

15. Baldi, P., & Brunak, S. *Bioinformatics : the machine learning approach*. MIT Press. 2001

16. Heller MJ. DNA microarray technology: Devices, systems, and applications. *Annual Review of Biomedical Engineering*. 2002;4(1):129-153.

17. Dermitzakis ET. From gene expression to disease risk. *Nature Genetics*. 2008;40(5):492-493.

18. Fu X, Fu N, Guo S, et al. Estimating accuracy of RNA-seq and microarrays with proteomics. *BMC genomics*. 2009;10(1):161. https://www.ncbi.nlm.nih.gov/pubmed/19371429. doi: 10.1186/1471-2164-10-161.

19. Michael P. S. Brown, William Noble Grundy, David Lin, et al. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences of the United States of America*. 2000;97(1):262-267. https://www.jstor.org/stable/121778. doi: 10.1073/pnas.97.1.262.

20. Pirooznia, M., Yang, J. Y., Yang, M. Q., & Deng, Y. (2008). A comparative study of different machine learning methods on microarray gene expression data. *BMC Genomics*, *9*(Suppl 1), S13. https://doi.org/10.1186/1471-2164-9-S1-S13

21. Anthimopoulos, M., Christodoulidis, S., Ebner, L., Christe, A., & Mougiakakou, S. (2016). Lung Pattern Classification for Interstitial Lung Diseases Using a Deep Convolutional Neural Network. *IEEE Transactions on Medical Imaging*, *35*(5), 1207–1216. https://doi.org/10.1109/TMI.2016.2535865

22. Yann LeCun, Yoshua Bengio, Geoffrey Hinton. Deep learning. *Nature*. 2015;521(7553):436-444. https://www.ncbi.nlm.nih.gov/pubmed/26017442. doi: 10.1038/nature14539.

23. Wu, R., Yan, S., Shan, Y., Dang, Q., & Research, S. B. (n.d.). *Deep Image: Scaling up Image Recognition*. Retrieved from https://arxiv.org/vc/arxiv/papers/1501/1501.02876v1.pdf

24. Parkhi, O. M., Vedaldi, A., & Zisserman, A. (n.d.). *Deep Face Recognition*. Retrieved from http://cis.csuohio.edu/~sschung/CIS660/DeepFaceRecognition_parkhi15.pdf

25. Shiffman D. *The nature of code;* 2012.

26. 1. Meystre S, Haug PJ. Natural language processing to extract medical problems from electronic clinical documents: Performance evaluation. *Journal of Biomedical Informatics*. 2006;39(6):589-599.

27. Navarre, S. W., & Steiman, H. R. (2002). *Natural Language Processing with Python*. *Journal of Endodontics* (Vol. 28). https://doi.org/10.1097/00004770-200204000-00018

28. Mikolov T, Chen K, Corrado G, Dean J. (2012) Efficient estimation of word representations in vector space

29. Meyer, D. (2016). How exactly does word2vec work?, retrieved from pdfs.semanticscholar.org

30. Rong, X. (2014). word2vec Parameter Learning Explained. Retrieved from http://arxiv.org/abs/1411.2738

31. Sansone S, Rocca-Serra P, Field D, et al. Toward interoperable bioscience data. *Nature Genetics*. 2012;44(2):121-126.

32. Brazma A, Vilo J. Gene expression data analysis. *FEBS Letters*. 2000;480(1):17-24.

33. Scherer, A. *Batch effects and noise in microarray experiments: sources and solutions*. J. Wiley; 2009.

34. Lazar, C., Taminau, J., Meganck, S., Steenhoff, D., Coletta, A., Molter, C., … Nowe, A. (2012). A Survey on Filter Techniques for Feature Selection in Gene Expression Microarray Analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *9*(4), 1106–1119. https://doi.org/10.1109/TCBB.2012.33

35. Quackenbush J. Microarray data normalization and transformation. *Nature Genetics*. 2002;32(S4):496-501.

36. Irizarry RA, Hobbs B, Collin F, et al. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*. 2003;4(2):249-264.

37. Hwang, B. J., Ford, J. M., Hanawalt, P. C., & Chu, G. (1999). Expression of the p48 xeroderma pigmentosum gene is p53-dependent and is involved in global genomic repair. Proceedings of the National Academy of Sciences, 96(2), 424–428. https://doi.org/10.1073/pnas.96.2.424

38. Chu, G., Tusher, V., & Tibshirani, R. (2001). Significance Analysis of Microarrays " Users guide and technical document. *Policy*, *98*(9), 1–41.

39. Gallo CA, Cecchini RL, Carballido JA, Micheletto S, Ponzoni I. Discretization of gene expression data revised. *Briefings in bioinformatics*. 2016;17(5):758-

40. Madeira SC, Teixeira MC, Sa-Correia I, Oliveira AL. Identification of regulatory modules in time series gene expression data using a linear time biclustering algorithm. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. 2010;7(1):153-165.

41. Pukelsheim. (1994). The three sigma rule. *American Statistician*, *48*(2), 88–91.

https://doi.org/10.1080/00031305.1994.10476030

42. Hanahan D, Weinberg R. Hallmarks of cancer: The next generation. *Cell*. 2011;144(5):646-674.

43. Li, S., Jiao, J., Han, Y., & Weissman, T. (2016). Demystifying ResNet. Retrieved from http://arxiv.org/abs/1611.01186

44. Shao, Y. et al (2019) Shedding Light on the Black Box: Explaining Deep Neural Network Prediction of Clinical Outcome

45. Workman, T. E., Hirezi, M., Trujillo-Rivera, E., Patel, A. K., Heneghan, J. A., Bost, J. E., … Pollack, M. (2018). A Novel Deep Learning Pipeline to Analyze Temporal Clinical Data. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 2879–2883). IEEE. https://doi.org/10.1109/BigData.2018.8622099