A Machine Learning Approach For The Classification Of Clinically Actionable Genetic Mutations Using Deep Learning

***Abstract*—The efficacy of cancer treatment has grown exponentially with the increased use of personalized medicine. Personalized, or precision medicine, tailors cancer care to the patient at hand. Oncologists perform molecular diagnostics and other forms of biological analysis to select which treatments would be suitable and optimal for the patient. One such technique employed is genomic sequencing, whereby examiners attempt to record all genetic abnormalities that occur in the malignant tissue, including mutations, atypical translation, and deletions. From there, they seek to distinguish between drivers, mutations that contribute to tumor growth, and passengers, which are neutral mutations. Distinguishing between the two is a manual process that requires clinical pathologists to review previous clinical literature to determine the nature of each and every genetic mutation sequenced. In this paper, a machine learning algorithm is devised and implemented to automatically classify genetic mutations based on textual literature. The recurrent neural network is proposed with a long-short term memory cells (LSTMs). The network is implemented with Theano in Python and trained with a GTX 760 2GB graphics card, designating 80% of the 3321 rows of data explicitly for training and the remaining 20% for the validation of the model. The model attained a validation accuracy of 79.7%. Further analysis revealed a Cohen kappa score of .73.**

I. Introduction

All forms of cancer come as a result of abnormal cell growth. More specifically, the genes which regulate the growth of tissue are affected by random mutations. Genes that are affected can be categorized into two classes by their function. Proto-oncogenes are genes which stimulate cell growth, reproduction and proliferation. Tumor suppressor genes are genes that decelerate cell division, restore damaged DNA, and initiate apoptosis. A series of mutations inactivate the tumor suppressor gene(s), thus leading to cellular proliferation, resulting in the mutation of proto-oncogenes into oncogenes. The byproduct of the aforementioned processes is a cancerous tumor [1].

A key component of combating against cancer and tumor growths is genetic sequencing. There can be a plethora of genetic mutations responsible for a single tumor. Some forms of cancerous tumors have thousands of genetic mutations. A drawback to this approach is the differentiation between mutations that trigger cancer growth, called drivers, and those neutral mutations, called passengers. The process whereby pathologists designate genetic mutations as being either drivers or passengers is a costly and inefficient process that involves a manual review of every sequenced genetic mutation. A classification is made based upon the conclusions of text-based clinical literature.

In this paper, an efficient machine learning algorithm is developed to automate this process. Requisite to exceeding the baseline performance of a clinical pathologist is a deep, recurrent neural network and lonsg short-term memory (LSTM) cells which treat sentences from the text-based clinical literature as a time series and feeds outputs from a preceding neuron into the consecutive neuron. The above deep learning models belongs to a class of machine learning algorithms known as supervised learning models whereby the features (input arrays) and labels (output vectors) are known. Although classification can likely be done through unsupervised techniques such as k-means and clustering, this paper will focus on a supervised deep neural network.

II. Model

A. Theoretical Model

The model employed for this task is a modified version of the recurrent neural network with the implementation of the long short term memory (LSTM) cell.

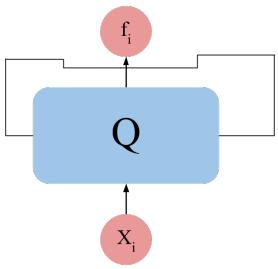

Fig. 1. A node in the recurrent neural network

In Figure 1, a node Q of the neural network takes input X_i and outputs f_i. The output, f_i, is then looped through the subsequent node in the network as the input X_i. Such a model allows information to be passed from one node to its successor. This algorithm is optimal for machine learning with data that comes in the form of a sequence of text, where the previous word may provide some indication of the following or the following may be dependent upon the previous [2].

B. Complications

One drawback of the recurrent neural network's sequential architecture is the information gap. If the algorithm at node n were to predict the character in a string of text, it may require information that occurs before $n-1$ to accurately make this prediction. An example would be the sentence "Her name is known. She went to the park." If the model were seeking to predict the relevant pronoun of the subject, it would encounter an issue as the preceding word is simply *known*, and the information needed to make the prediction occurs at the very beginning. In dealing with more copious amounts of data, the neural network may be rendered unable to make connections and draw conclusions as a result of the widening gap [3].

C. Long-Short Term Memory

Long Short-Term Memory cells (LSTMs) were proposed to solve the complexities arising from long-term dependencies [3]. As opposed to a traditional recurrent neural network which has node n depend on the output of $n-1$, the LSTM instead computes n with the operation $\Delta_n + n - 1$. LSTM's ability to handle long-term dependencies also resides within their architecture. A structure known as the memory cell constitutes the foundation of the LSTM model. The memory cell is comprised of an input gate layer, forget gate layer, internal cycling connection and an output gate layer [4].

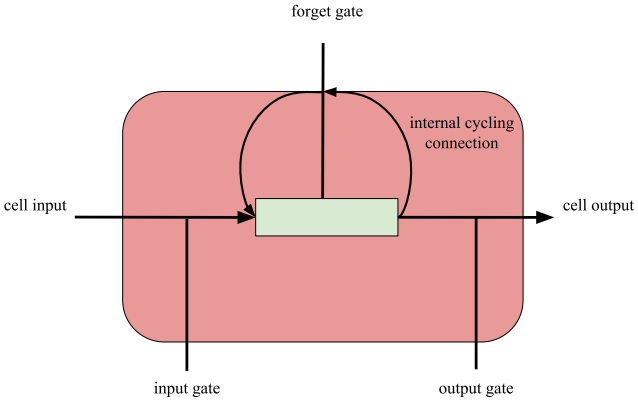

Fig. 2. The architecture of the LSTM model.

The first step in the long-short term memory model is computing the value for i_t, the input gate layer, and c_t, the candidate value for the state of the memory cell at time t. This is computed as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$c_t = tanh(W_c x_t + U_c h_{t-1} + b_c)$$

where σ is a sigmoid function, W_c, W_i, U_c and U_i are weight vectors and b_i and b_c are biases. Then, we compute f_t, the forget gate decision layer, which decides which information from the aforementioned gate should be retained and which should be discarded [5]:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

After computing the preceding, the former cell state C_{t-1} can be updated into cell state C_t. This requires a point-wise multiplication of our former cell state, C_{t-1} with f_t followed by adding it to the point-wise multiplication between i_t and c_t:

$$C_t = i_t * c_t + f_t * C_{t-1}$$

The output gate layer and output respectively can therefore be computed as:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t * tanh(C_t)$$

The output will depend on a sanitized version of our memory cell state. A sigmoid function, σ, with the above parameters determine which values of the cell state will be outputted. This becomes o_t. We then proceed to compute the point-wise multiplication between o_t and $tanh(C_t)$, to constrain our output values to only those that were designated to be outputted by the sigmoid function.

III. Data

Two different datasets are collected. One consists of 3321 rows and 2 columns. The first column is a numeric identifier. The second column contains text-based literature. The second dataset consists 3321 rows and 4 columns. The first column, a numeric identifier, corresponds to identifier row in the first dataset. The second and third rows are the gene and its respective amino acid variation. The fourth column is the numerical class of the gene-variation pair. In the second dataset, there are 9 distinct classes. The two datasets, as previously stated, are related by identifier column in each row. For example, in the first dataset, the first row contains identifier 0 and clinical text. The gene-variation pair in the second dataset with identifier 0 is the mutation described by the clinical text in the first row in the first dataset. Utilizing the matplotlib library along with techniques of exploratory data analysis, patterns and trends within the training data were uncovered.

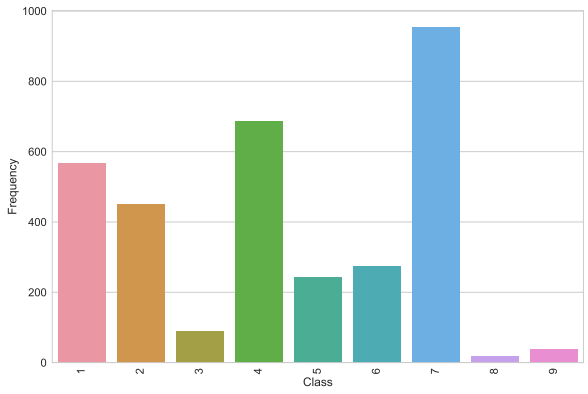

Fig. 3. Frequency Distribution of Classes

Insights into the frequency distribution of the classes revealed that a plurality of the mutations that occurred within the data belong to class 7. Class 1, 2, and 4 follow with a minuscule percentage of the mutations belonging to classes 8 and 9.

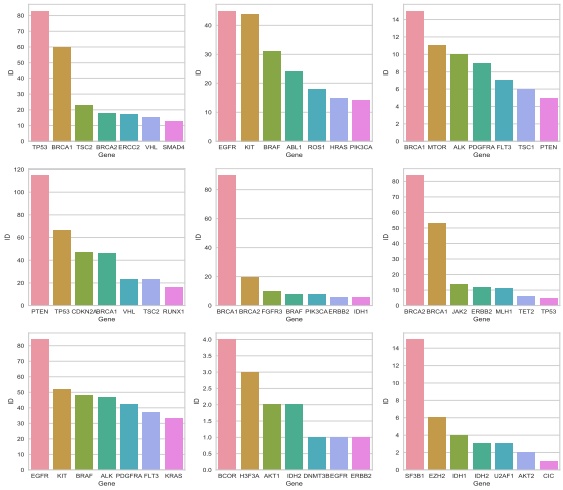

Fig. 4. Genes Per Class

With further data analysis, the distribution of genes per class provides insight into which genes dominate which classes. From the above figure, it can be concluded that gene BRCA1 highly dominates class 5, class 7 and class 6. Additionally, SF3B1 highly dominates class 9.

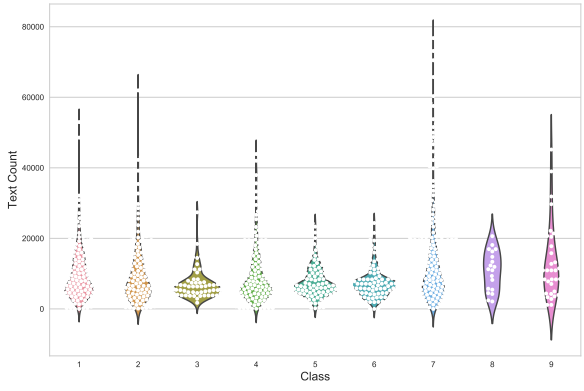

Fig. 5. Text Count Per Class

After merging the two data files together via their coordinating ID field, a violin plot is constructed to ascertain the text count per class. The mean count for many of the classes lies within the range of 0 to 20000. Class 7 is an statistical outlier, with text counts ranging from 0 to 80000, far exceeding the

rest of the classes. A total of 37 rows have a text count below 1000, 5 of which are rows with a null value for their text field (rendering them ineffectual when training the model).

Exploratory data analysis continues with a dive into the most frequently occurring terms for each class. Visualizing which terms occur the most often for which class will aid in feature selection and training of the algorithm. To complete such a task, training text will vectorized using tf-idf (term frequency-inverse document frequency) transformation. Tf-idf vectorization assigns each word in our training text a number proportional to its frequency in a class and inversely proportional to the number of classes in which it appears. For a term i in class j:

$$w_i, j = tf_i, j * log(\frac{N}{df_i})$$

where tf_i, j represents the number of occurrences of i in j, df_i represents the number of classes containing i and N represents the total number of classes [6].

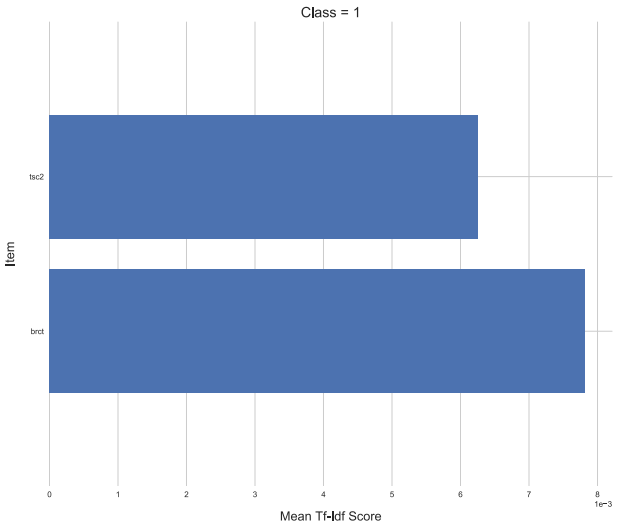

Fig. 6. Text Count Per Class

The above figure provides an example of the terms with the highest mean Tf-idf scores for a certain class. The terms with the highest Tf-idf scores for Class 1 include brct and tsc2, respectively.

IV. Results

A. Experimentation

The model was implemented in Python using the Theano library, scikit-learn, and Keras's native word tokenizer. Additionally, scikit-learn's feature extraction module was utilized for Tf-idf vectorization. The model was trained on 80% of the data, while 20% was allocated for validating the accuracy of our model (to ascertain the model's ability to generalize new data). The model required 10 epochs to train. An epoch is defined as one complete loop through the neural network which includes the feed-forward aspect and subsequent back-propagation. Training time was approximately 7 hours on a GTX 760 2GB GPU.

B. Performance Evaluation

Accuracy in differentiating between drivers and passengers is integral to the success of personalized medicine. Several measures were employed after training to gauge the efficacy of the model. The first statistic used was the Cohen kappa score, which measures the degree of agreement between two different batches of data. It is used to measure the reliability of the data collected and is defined as:

$$\kappa = \frac{\rho_o - \rho_e}{1 - \rho_e}$$

where ρ_o is the experimental agreement ratio and ρ_e is the hypothetical agreement ratio. A measure of accuracy is also employed in tandem with a confusion matrix, which is a matrix that provides insights into the predicted classifications made by the model and the actual class.

Performance Measure	RNN	RNN with LSTM
Cohen kappa score	.58	**.73**
Accuracy	.648	**.797**

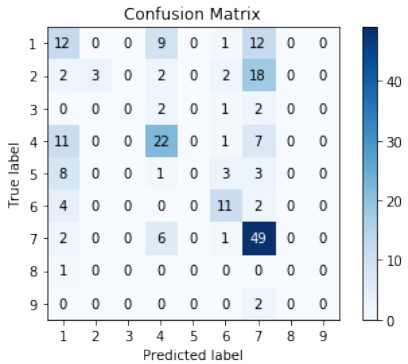

Fig. 7. Confusion matrix for RNN with LSTM

As evidenced by the above table and the confusion matrix, the recurrent neural network with LSTM outperforms a raw recurrent neural network. The raw recurrent neural network attains a Cohen kappa score of approximately .58 while the RNN with LSTM attains a score of .73. Additionally, the accuracy of the RNN with LSTM exceeds the raw RNN 79.7% to 64.8%. A confusion matrix for the recurrent neural network with long-short term memory cells is produced showing promising, balanced results with a clear and predictable bias towards class 7.

V. Conclusion

As personalized medicine becomes more preeminent in the field of oncology and disease treatment, automation and efficiency in classifying mutations will prove to be important. The work proposed in this paper is to classify mutations by previously published, text-based clinical literature. The recurrent neural network with long-short term memory cells was employed to achieve this task with a raw recurrent neural network used as a control. The two models are then scored based on a Cohen kappa score and accuracy. Additionally, a confusion matrix is used to view how well the model predicts the nature of the mutations. The greatest result attained was an accuracy of 79.7% with the recurrent neural network with LSTMs. The aforementioned results are promising and for future work can be implemented into a software application that will allow clinical pathologists to automate the process of labeling mutations.

References

[1] Levine, Arnold J., and Anna M. Puzio-Kuter. "The control of the metabolic switch in cancers by oncogenes and tumor suppressor genes." Science 330.6009 (2010): 1340-134

[2] Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever. "An empirical exploration of recurrent network architectures." Proceedings of the 32nd International Conference on Machine Learning (ICML-15). 2015.

[3] Hochreiter, Sepp. "The vanishing gradient problem during learning recurrent neural nets and problem solutions." International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6.02 (1998): 107-116.

[4] Hochreiter, Sepp, and Jrgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[5] Gers, Felix A., Jrgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." (1999): 850-855.

[6] Ramos, Juan. "Using tf-idf to determine word relevance in document queries." Proceedings of the first instructional conference on machine learning. Vol. 242. 2003