Cancer Lineage Subtype Classification Using Gene Hierarchy-Based "Visible" Neural Networks on Tumor Mutations

Benjamin Yang
Department of Computer Science and Engineering
University of California San Diego
San Diego, CA 92093
beyang@ucsd.edu

Nikhil Karnwal
Department of Electrical and Computer Engineering
University of California San Diego
San Diego, CA 92093
nkarnwal@ucsd.edu

Patrick Wall
Department of Bioengineering
University of California San Dieg
San Diego, CA 92093
pgwall@eng.ucsd.edu

Yang Li
Department of Cognitive Science
University of California San Dieg
San Diego, CA 92093
yal009@ucsd.edu

Yucheng Tu
Department of Mathematics
University of California San Diego
San Diego, CA 92093
y7tu@ucsd.edu

Abstract

Here we describe the construction of a "visible" neural network capable of predicting tumor type and subtype based solely on tumor sequencing data. This model uses the Cancer Dependency Map (DepMap) database, which contains all necessary genetic and cell line information. The model architecture is based upon previously described hierarchy-based networks that connect neurons based on cell process interactions described in the Gene Ontology (GO) Resource, a manually curated database mapping genetic interactions. An array of mutations is input at the top of the hierarchy, then the various neuron activations downstream indicate the cell processes that have been affected by the mutation input. This model structure is helpful for diseases like cancer, where cell processes are disrupted by mutations. Critical to this task is identifying biologically relevant numerical methods to input mutation information into the network. Three mutation scoring methods were used with various specificities for predicting how likely any mutation is to cause cancer: Binary, CHASMplus, and VEST4. Combinations of scores were also used, with each mutation being represented as a concatenation of CHASMplus/VEST4 scores. Additionally, four different model architectures were created to embed mutation information at various levels at the top of the hierarchy. Little difference was found between mutation input scoring methods using the shallowest network (Binary 16.67% accuracy, CHASMplus/VEST4 15.87% accuracy). However, deeper networks increased model accuracy and differentiated between the biologically relevant CHASMplus/VEST4 inputs and nonspecific binary inputs (Binary 12.27% accuracy, CHASMplus/VEST4 17.11% accuracy). This deeper framework indicates a new way for embedding disparate biologic information within a single input for physiologically relevant deep learning.

1. Introduction

Precision medicine has come to the forefront in clinical therapeutic development in nearly every clinical setting, especially with the rapid emergence of next generation sequencing (NGS) platforms and the opportunities and challenges that come with big data generation. Technologies such as single cell RNAseq or CHIP-seq give us an unprecedented view inside the cell and help peel back the seemingly stochastic nature of molecular biology. However, the sheer size of datasets needed to describe mechanistic relationships with any accuracy makes interpreting this data exceedingly difficult.

Deep learning offers the critical capability of being able to survey huge amounts of data and learn complex relationships, but the main criticism for biology and medicine-based deep learning models remains that the models are not readily interpretable. This poses a problem because models may produce unexpected predictions not previously described in the literature, but without any biological or physiological rationale there is no way of knowing if the model was able to learn and predict something not yet observed, or if the prediction was just erroneous noise. For example, autism spectrum disorders often arise due to a series of hundreds or thousands of mutations, and the mechanisms underlying these conditions remain unknown. Deep learning models may be able to learn what molecular signatures hiding within bioinformatic datasets give rise to autism spectrum phenotypes, but without a way to query how that decision was made, no conclusions can be drawn.

Recently, genetic interaction hierarchies have been proposed as suitable deep learning architectures to aid in model interpretation. This method embeds genetic interaction networks within the network's neuron connections, thus any prediction can be directly mapped back to real, described genetic processes or genes themselves. This allows researchers to make more informed decisions about a prediction's relevance with respect to cell function. Hierarchy-based networks have proven accurate in predicting yeast gene knockout effects and in predicting small molecule drug response, and efforts to implement them in the clinic are currently being pursued. Now, the critical problem arises: how do you represent genomic information so that the model learns and can understand perturbations of the normal cellular state?

Traditionally, models have used a binary mutation representation where a mutated gene is represented as 1 and a wildtype gene is a 0. Due to the complexity of protein interactions it is not always easy to predict a mutation's specific functional impact. For example, cancer driving mutations in oncogenes tend to be associated with "gain of function" mutations where the protein picks up a new phenotype that causes dysregulation of cellular processes to drive the cancer state. However, cancer driver mutations in tumor suppressor genes tend to be associated with loss of function mutations where the tumor suppressor loses its ability to prevent the cancer state. To this point, binary representation allows the model to only see if there is a mutation in a particular gene, but nothing about what the likely impact of that mutation will be. It is up to the model to learn whether a gene being mutated in the context of other mutated genes is important. This has shown to be effective but it is not entirely interpretable and can be greatly improved upon so that the network is truly interpretable.

Here, we propose a new methodology for hierarchical network learning that will use mutation effect prediction algorithms to identify and numerically characterize potential cancer driver mutations. Great progress has been made in the last decade to build reliable prediction algorithms, but no single method has shown consistent superiority in all queried datasets. To that end, we will use two of the currently best-performing prediction algorithms (CHASMplus and VEST4) to return separate scores for genetic mutations, then we will input both scores for each gene within the hierarchy. By continuing to layer additional bioinformatic "snapshots," the model should be able to better learn the physiological importance of mutations.

2. Motivation

As mentioned, deep learning provides an ideal framework for identifying complex relationships in often messy biologic datasets, however clinicians and biologists are reluctant to rely on neural networks because the predictions can sometimes contradict known dogmatic principles. Novel observations by a traditional deep network lack the essential interpretability needed to discern whether the decision is physiologically relevant or just noise. Frameworks have been created that embed physical cellular relationships within the network that are interpretable with respect to cell process activation states, but little progress has been made to introduce raw bioinformatic features in such a way as to be useful for deep learning problems. Our goal is to evaluate alternative methods for bioinformatic feature representation that increase model accuracy, interpretability, and show that these models can learn more sophisticated, relevant internal representations of complex biological phenomena based on the inputs

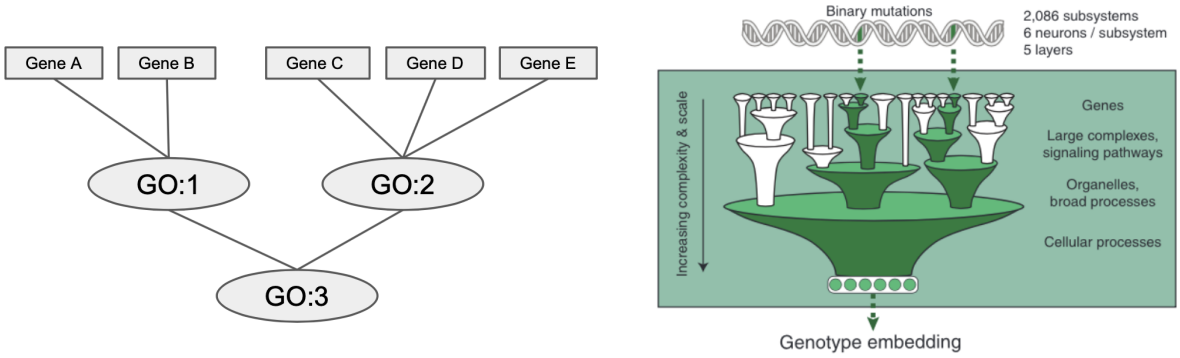
alone. This would help open the door for applying deep learning to medicine and biology, which could change how science is conducted both at the bench and at the bedside.

3. Related Work - Genetic Interaction Networks, and Mutation Scoring

Genetic interaction networks form a cornerstone of modern systems biology. The current estimate for the number of genes in the human genome is between 24,000 and 35,000, and it is estimated that there are over one trillion cells in a human body. Each cell carries a copy of these genes, but the ways in which these genes are regulated, expressed, and interact produce the multitude of cell types that are needed to result in a functional organism. Understanding the vastly complicated networks by which these genes interact provides a crucial roadmap for researchers to be able to make reasonable observations regarding normal cellular behavior, and begin to understand the subtle differences between cells that might lead to massive differences in biologic phenotype and function.

One such road map is the Gene Ontology Resource - a database curated by researchers that attempts to fully characterize each gene's interaction network, and organizes these networks into common functional pathways that ultimately lead to complex phenotypes. This database identifies each individual (known) gene and assigns it a Gene Ontology number (or GO number) unique to the database. Researchers then submit evidence supporting specific genetic interactions and the Gene Ontology Resource aggregates the data and establishes common interaction pathways that govern increasingly complex phenotypes. An example is shown in Figure 1:

Figure 1: Gene and ontology terms with interaction hierarchy organization (left). Overview of hierarchical cellular processes and genotype embedding in the network (right, Kuenzi et al 2020).



Interactions are assembled as a graph, with gene names at the bottom of the hierarchy. Genes that directly interact in small cellular subprocesses are connected to a node with a unique GO term, which describes that subprocess. In turn, the subprocesses that directly interact are connected to another GO term that describes a larger cellular process. This repeats all the way to a single GO term, which is cell growth, and serves as the root of the hierarchy used in our models. Cell growth is an appropriate root because cancer is fundamentally a disease of unregulated cell growth. Every layer within this embedding encapsulates ever-increasing levels of cell functionality, which is made up of smaller children cell processes upstream. Analogous to this structure would be convolutional neural networks that perform dimensionality reduction/pooling, with each successive layer encoding larger receptive fields from the input. As the ontology network goes deeper, the "receptive field" of gene processes increases. Mutations at the gene level will have various functional impacts and affect processes downstream, and in this way it is possible to observe how mutations in one gene may alter the behavior of gene pathways seemingly disconnected from the mutated gene by tracing back these interaction networks.

Our model is based on previously published methods (Ma 2018, Kuenzi 2020) that embed genetic interaction networks into the model architecture, which results in a "visible" neural network that performs as well as the best

fully connected counterpart. Importantly, the authors use the word “visible” to describe the model’s interpretability, suggesting that they are able to look into the network and understand mechanisms governing model predictions and directly tie these to existing cellular pathways. The model learns the activation states of each node given a genetic input. The node activation states can then be interpreted as the impact a mutation set has on cell processes, since these nodes are direct representations of cellular processes.

This model will assess the mutational state of 3,008 genes in 1,225 tumor models, then pass that information forward through the network. To do this, mutations are scored by cancer driver prediction algorithms, and various methods for inputting these scores at the gene level within this hierarchy are implemented. Many mutations can cause (or “drive”) a mutation state. Some mutations may act alone in driving cancer, while others may seem benign but cause small physical changes that, when coupled with other small effect mutations, can drive the cancer state. The ability to identify and predict which of these mutations cause cancer is important, especially given the size of the human genome, the limitless ways in which the genome can be altered, and the wide range of mutations that are normal and don’t cause cancer within any population.

Genome representation plays a critical role in model function and will present one of the major challenges in this project. Previous efforts did not distinguish between mutations and instead represented mutations in a binary manner for the model, with 1 representing a mutated gene and 0 a normal (wildtype) gene. This does little to encapsulate the complex biophysical and physiological differences mutations take and may confound predictions since all mutations are weighted equally. This has very little physiologic meaning, so more sophisticated methods for genotype representation are needed. Here we update this mutation representation to reflect the likely functional impact a mutation has on a gene.

Many different algorithms have been developed for this purpose (Chen, et al., 2020), and it is only within the last few years that these algorithms have achieved sufficient predictive power to reliably encode cancer driver status for coding (protein producing) and noncoding (usually protein expression/regulation) mutations. However, major hurdles remain in algorithm bias and effectiveness in all contexts: cancer-specific driver mutation algorithms tend to overrepresent mutations in known oncogenes/tumor suppressor genes, whereas general-purpose function impact predictors tend to underrepresent cancer driver mutations that do not obliterate gene function. Additionally, these scoring algorithms may have differential performance on the same mutation - one may correctly highlight the mutation as likely to cause cancer while the other might fail. Hence, it is important to include multiple representations as the ground truth oncogenicity of a mutation might need a more democratic “consensus” approach between algorithms.

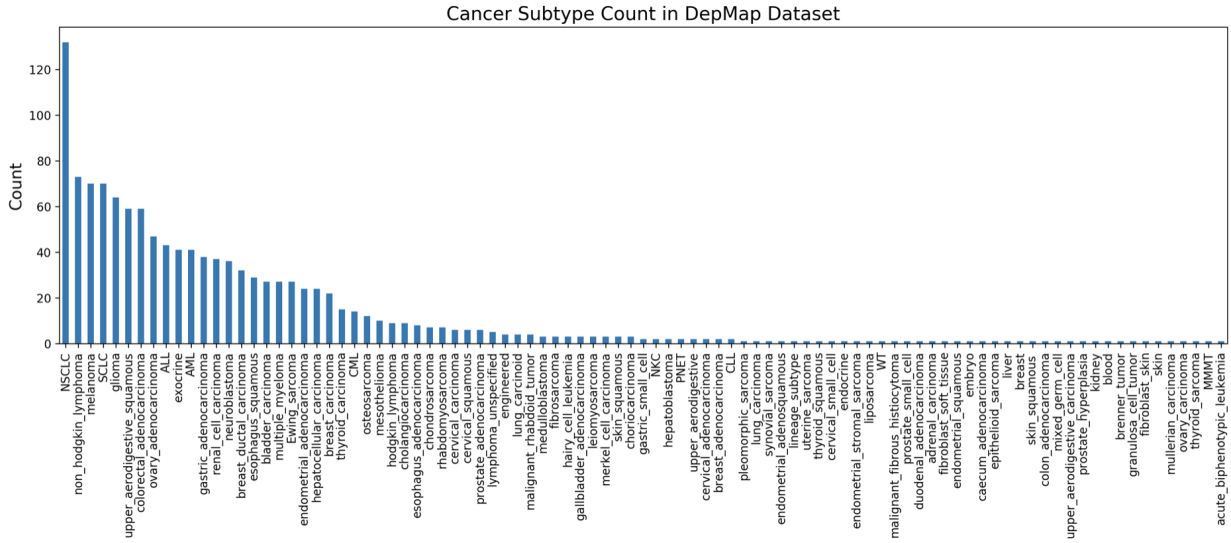
To this end we chose two high-performing algorithms to independently score our mutation set. One is cancer-specific, one is general purpose. Both perform well on benchmark cancer mutation datasets to predict cancer driver status. We will use both scores as inputs into our model in an effort to increase the biologically relevant information the model receives. Both algorithms, CHASMplus and VEST4, output a probability for a specific mutation to drive a cancer state or deleterious gene process, so the input domain will remain the same and make the numbers helpful for our purposes. We will then use both representations simultaneously to inform the model of the mutation state of a particular gene.

This method will update genome representation to a more relevant consensus approach and represents an important step forward for embedding biologic information for use in deep learning. This will open many possibilities for biologically-focused deep learning applications. The goal here is two-fold: 1) represent mutation information in a novel way for use in deep learning; 2) validate the method by predicting cancer lineage subtype as proof the model is learning more sophisticated representations of specific cancer states.

4. Dataset, Model Architecture and Mechanism

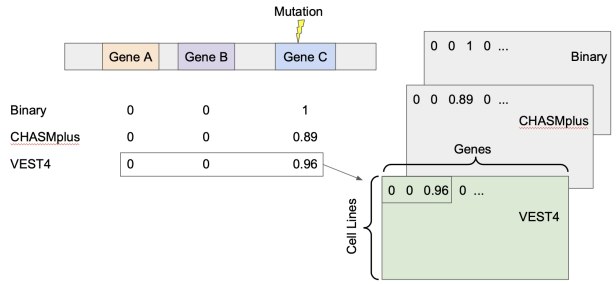
In this model we will use data from the Cancer Dependency Map (DepMap) database, which maintains a repository of almost 2,000 cancer cell lines biopsied from patients, and has undergone extensive bioinformatic characterization of each cell line, including sequencing information. Mutation annotation files (MAF) were generated for each cell line and were identically processed to highlight somatic mutations likely involved in the cancer phenotypes. The original models used the top 15% of mutated genes, for a total of 3,008 genes in 1,225 cell lines. We will keep the same structure in order to compare our results with previous representations. The database does not include negative samples, however by the nature of the annotation pipeline any cell considered to be a normal, noncancerous cell would be represented only as a series of 0's so that each gene has no probability to drive cancer. With this information the model takes in a 3,008 gene array for prediction, embeds cancer state representations within a hierarchy mirroring cell processes, and outputs a softmax classifier that predicts cancer subtype or wildtype (normal/not mutated, noncancerous).

Figure 2: Cancer subtype class distribution.



This data contains 91 unique cancer subtypes. We represent each subtype with a numerical label 0-90, with label 91 as the wildtype (noncancerous) state. This dataset immediately presents a challenge due to the small number of samples overall, so training accuracy and cancer state representation will be inherently limited. There is also a large class imbalance. The overwhelming majority of cases present was non-small cell lung carcinoma (n=132), a relatively common lung cancer. The next few subtypes present in high numbers are non-hodgkin lymphoma (n=73, blood), melanoma (n=70, skin), small cell lung cancer (n=70, lung), and glioma (n=64, brain). Importantly, there are 39 classes with only a single occurrence, which will lead to class imbalance issues during training. There are also very few training examples, as whole genome sequencing has only recently become viable as a method for standard tumor analysis due to sequencing costs approaching $1,000/sample. Even still, quality mutation variant calling still needs expert curation. Complicating things further, obtaining patient tumor samples and expanding cells for use in sequencing is often difficult and expensive. Due to these factors the number of fully characterized tumor samples is relatively limited, with only 1,225 samples available for this analysis. We opted to maximize the number of training examples and did not partition samples for a test set. Instead we settled on using 1,100 samples for training (~90%), and the remaining for validation. We also opted to not discard low frequency samples, theorizing that since many cancer subtypes share common mutations (BRAF V600E is a common variant in many subtypes) training on all samples may help the model learn important mutations better.

Figure 3**:** Overview of mutation input generation with various scoring schemes (left). Each mutation is evaluated using binary, CHASMplus, or VEST4 algorithms, and scores for each gene are stored as an array, with one array for each scoring scheme. Each row corresponds to the mutational representation of one cell line.



Mutation information is scored with Binary, CHASMplus, and VEST4 algorithms, and the scores are stored in separate, algorithm-specific arrays that will be used as inputs to the model. To make a subtype classification prediction, the model iterates through each cell line in the input and concatenates mutation scores from CHASMplus/VEST4 for each gene, then performs a linear transformation and feeds these feature arrays into a bank of neurons.

Three different model architectures were created to input genetic information into the network (Figure 4). The first model (4A) passes a weighted sum of a gene's mutation scores into an array. In this array each gene is represented by a single value, and all the gene values connected to a specific GO term are concatenated and fed into a bank of neurons representing that GO term. The resulting GO term neuron values are nonlinearly activated with the tanh activation function and batch normalized. This creates an activated GO term neuron representation. Similar to genes, GO term neurons are concatenated with other GO term neurons that are connected to a common, larger parent GO term. This process is repeated throughout the hierarchy until the root node, which is another bank of neurons representing the cell growth processes. After the root node we have attached a linear classifier layer with 92 outputs and softmax activation to predict sample subtype. This architecture constrains the information to a single bank of neurons, but it is important to not concatenate information from higher in the network as this would confound biologic interpretation. Instead, we expand the bottom layer to include more neurons. Since the first nonlinear activation occurs at the top ontology layer, we named this model the Ontology-Level model.

The next two models expanded on this architecture. The Gene-Level model adds a layer of neurons for each gene, then feeds the concatenated mutation scores into this layer and activates/batch normalizes the resulting values (4B). After this, all the activated gene neurons for the common GO term are concatenated and fed to that GO term's neurons. The Feature-Level model adds a second layer of neurons for each gene. The scores are fed into this feature layer and activated/batch normalized before being fed into the gene-level neurons (4C). The advantage of these two models is they add additional nonlinearities into the system while preserving the cell interaction network. This should allow the model to learn additional representations of the data to make more accurate predictions. The Feature-Level model has the added benefit of learning which feature scores are important for the model decision and can directly attenuate feature signal up or down. This allows for high granularity detail on all levels of biological information, from biologic inputs to large cellular processes, which makes the model more interpretable.

We concatenated the CHASMplus/VEST4 scores as a consensus method for evaluating each mutation's potential as a cancer driver. However we also used the original Binary vector to compare models for relative improvement. For these tests we concatenated the binary score with a zero to preserve model dimensionality. In this way we were able to compare single scores with consensus representations.

Figure 4: Various model architectures used. A) Ontology-Level model. Feature scores are concatenated and linearly transformed into an array as a single value representing a gene. All of these values are concatenated and fed into the ontology layer neurons, where the first nonlinear activation occurs and the mutation signal is propagated into the network. Here the representation for Gene C is highlighted in yellow. B) Gene-Level Model. Scores are concatenated and linearly transformed into an array, which is then fed into a bank of neurons representing each gene (here the neuron bank representing Gene C is highlighted in yellow) then nonlinearly activated. C) Feature-Level model. Scores are transformed into an array that are then fed into a bank of neurons representing the gene's features, which are unique for each gene and are directly fed into their respective gene neurons.

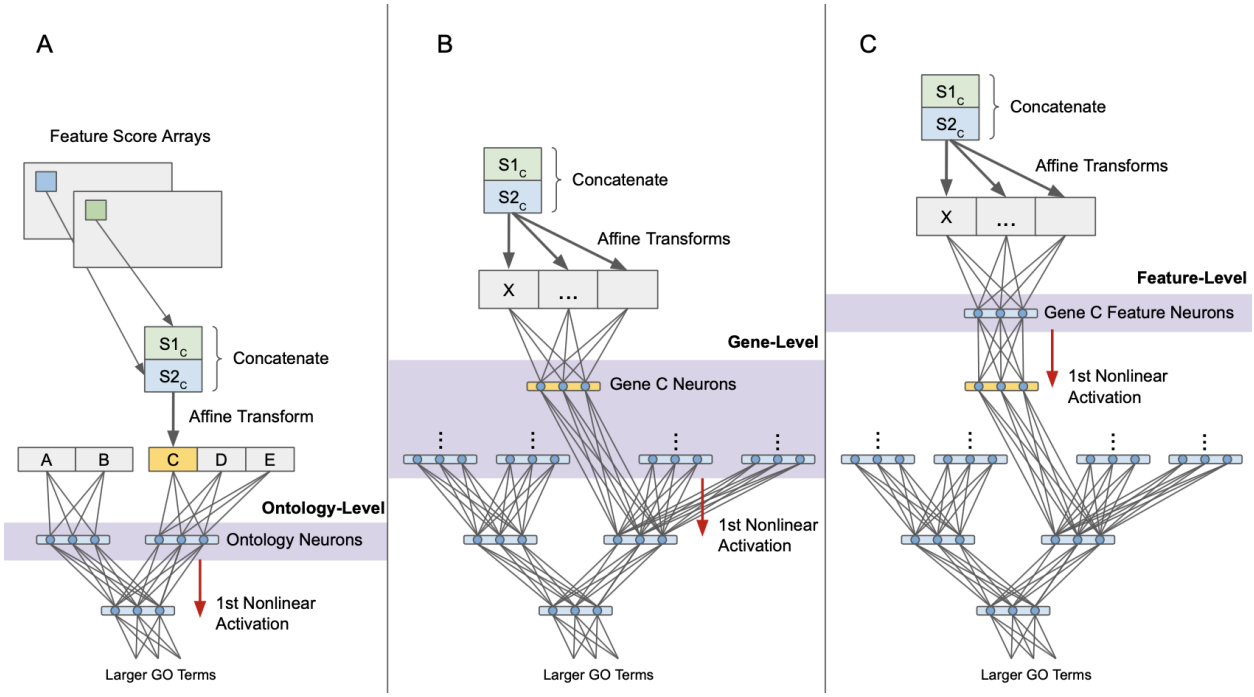


Table 1: Construction of each of the layers in the ontology-level model. The ontology layer was repeated for all terms in the ontology until the root node was reached, then the classifier layer was added at the end for label prediction.

Layer(s)	Construction	Parameters
Gene	concatenate(score 1, score 2) nn.linear	Size: [1, # features] Size: [# features, # hidden units]
Ontology	concatenate(connected child GO terms' and gene's values) nn.linear → tanh → batchnorm	Size: [# hidden units, variable length] Size: [variable length, # hidden units]
Classifier	nn.linear → softmax	Size: [# hidden units, # subtypes]

Table 2: Construction of each of the layers in the gene-level hierarchical model.

Layer(s)	Construction	Parameters
Feature	concatenate(score 1, score 2) nn.linear	Size: [1, # features] Size: [# features, # hidden units]
Gene	nn.linear → tanh → batchnorm	Size: [# hidden units, # hidden units]
Ontology	concatenate(connected child GO terms' and genes' activations) nn.linear → tanh → batchnorm	Size: [# hidden units, variable length] Size: [variable length, # hidden units]
Classifier	nn.linear → softmax	Size: [# hidden units, # subtypes]

Table 3: Construction of each of the layers in the feature-level model.

Layer(s)	Construction	Parameters
Feature	concatenate(score 1, score 2) nn.linear nn.linear → tanh → batchnorm	Size: [1, # features] Size: [# features, # hidden units] Size: [# features, # hidden units]
Gene	nn.linear → tanh → batchnorm	Size: [# hidden units, # hidden units]
Ontology	concatenate(connected child GO terms' and genes' activations) nn.linear → tanh → batchnorm	Size: [# hidden units, variable length] Size: [variable length, # hidden units]
Classifier	nn.linear → softmax	Size: [# hidden units, # subtypes]

5. Experiments

We performed a series of experiments to optimize model hyperparameters as well as test the mutation representation methods themselves in order to establish whether changing mutation feature representation to a combination of relevant cancer driver scores actually allows the model to learn more nuanced cancer state representations and better predict cancer subtypes. These experiments were repeated for all three different types of models.

Kuenzi et. al had previously varied parameters for a similar network to the ones tested here, and established performance for learning rate (0.001), batch size (5,000), epochs (100), and the number of hidden units to use in the linear layers (6). We will refer to these parameters as the "default" values.

We randomly partitioned the data to use 10% of it as the validation set and the remainder as the training set. Due to the limited number of training examples we chose not to further partition the data to make a test set. To evaluate each model and compare models relative to one another we used cross entropy loss and prediction accuracy. For each experiment we report the cross entropy loss on the training and validation sets at the final epoch, as well as the prediction accuracy on the best performing model.

We also created two different random models: one that shuffled sample labels to decorrelate mutations from subtype, and another random class generator that chose labels based solely on subtype frequency. These two methods show whether genetic information is necessary in decision making, and whether the model is just learning to predict cancer subtype frequency.

Since we have a small dataset and this data is hard to augment, we experimented using k-fold cross validation methodology. The experiment is conducted with 10 folds.

a. Random Model

We created a model that sampled from the subtype frequency distribution to predict class values. If the model were to learn to predict values based on subtype frequency alone, we would expect the resulting accuracy to be sampled from this distribution. Simulating 100,000 predictions we found the average random accuracy to be 4.49% (SD 2.5%), with a maximum cutoff of 7.12% to still be considered part of the random distribution (1-sided hypothesis test, 95% cutoff). Hence, any results will be considered significantly different from random if accuracy exceeds the 7.12% threshold.

b. Ontology-Level Model

We trained the default Ontology-Level model using the binary input, the CHASMplus input, the VEST4 input, and the concatenation of CHASMplus and VEST4. All four models significantly outperformed the random control in training loss and validation accuracy, but their validation loss values were only slightly lower across the board. Despite having a higher validation loss, the combination of CHASMplus and VEST4 inputs resulted in higher accuracy than the individual inputs, but the binary input model still outperformed it in every metric.

Table 4: Results for various inputs using identified hyperparameters with the Ontology-Level model. Random in this experiment refers to label randomization.

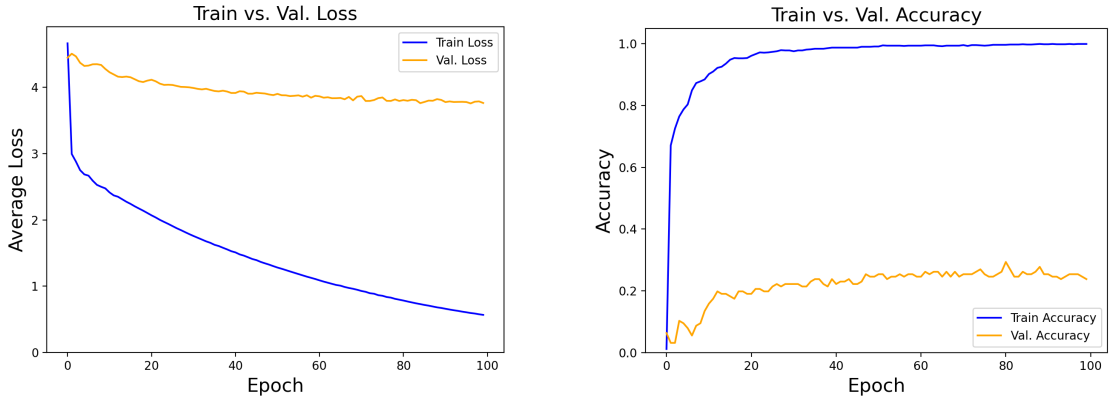
Model	Final training loss	Final validation loss	Best validation accuracy
Binary	2.5253	4.0864	16.67%
CHASMplus	2.7163	4.2048	10.32%
VEST4	2.7257	4.1104	13.49%
CHASMplus/VEST4	2.6145	4.2767	15.87%

Table 5: Results from comparing various hidden unit sizes of the ontology neurons using the Ontology-Level model with binary inputs.

Hidden Units	Final training loss	Final validation loss	Best validation accuracy
6	2.6406	4.2095	10.32%
10	2.0452	4.2604	8.73%
20	0.8416	3.909	13.49%
22	0.7105	3.8383	17.46%
24	0.5676	3.7649	25.4%
26	0.5085	3.886	21.43%
28	0.4185	3.7849	20.63%
30	0.3438	3.966	13.49%

Using binary inputs for the Ontology-Level model, we then tried varying the amount of hidden units. The results in Table 5 show that 24 hidden units yields the best overall performance, with a validation accuracy of more than 25%. The default six hidden units seems to not pass enough information through the model, but increasing it beyond 24 causes the model to overfit.

Figure 5: Training vs. validation loss (left) and accuracy (right). Model is Ontology-Level Binary with the optimal hidden features (24) obtained through experimentation



c. Gene-Level Model

For the Gene-Level model, we tested the binary input and the CHASMplus/VEST4 input against the random negative control. We had data from an extra experiment using the binary inputs and noticed that despite having very similar loss values, the validation accuracies varied drastically. One run performed even worse than the random control model, and one run significantly outperformed the weighted dual input. Since CHASMplus/VEST4 performed better than the average of the two binary runs, we decided to use it for the rest of the Gene-Level experiments.

Table 6: Results from comparing various inputs using the Gene-Level model. Loss values are from the last epoch of training, and accuracy values are from the best performing model.

Model	**Final training loss**	**Final validation loss**	**Best validation accuracy**
Binary	4.188	4.332	3.175%
Binary	4.2005	4.3396	13.49%
CHASMplus/VEST4	4.1915	4.3485	10.317%

We noticed in the experiments of Table 6 that the validation accuracy barely changed across epochs. We realized that this meant our default learning rate was too low, so we tried increasing the learning rate. Increasing it by a factor of 10 caused the training loss to decrease significantly, but we observed that the model was overfitting. We then performed a sweep of learning rates between the two and found that we got the best validation loss when the learning rate was 0.006 and the best validation accuracy when the learning rate was 0.005 (Table 7).

Figure 6: Training vs. validation loss (left) and accuracy (right). Models shown by row, using CHASMplus/VEST4 inputs with learning rate = 0.001 (top), learning rate = 0.01 (middle), and learning rate = 0.006 (bottom).

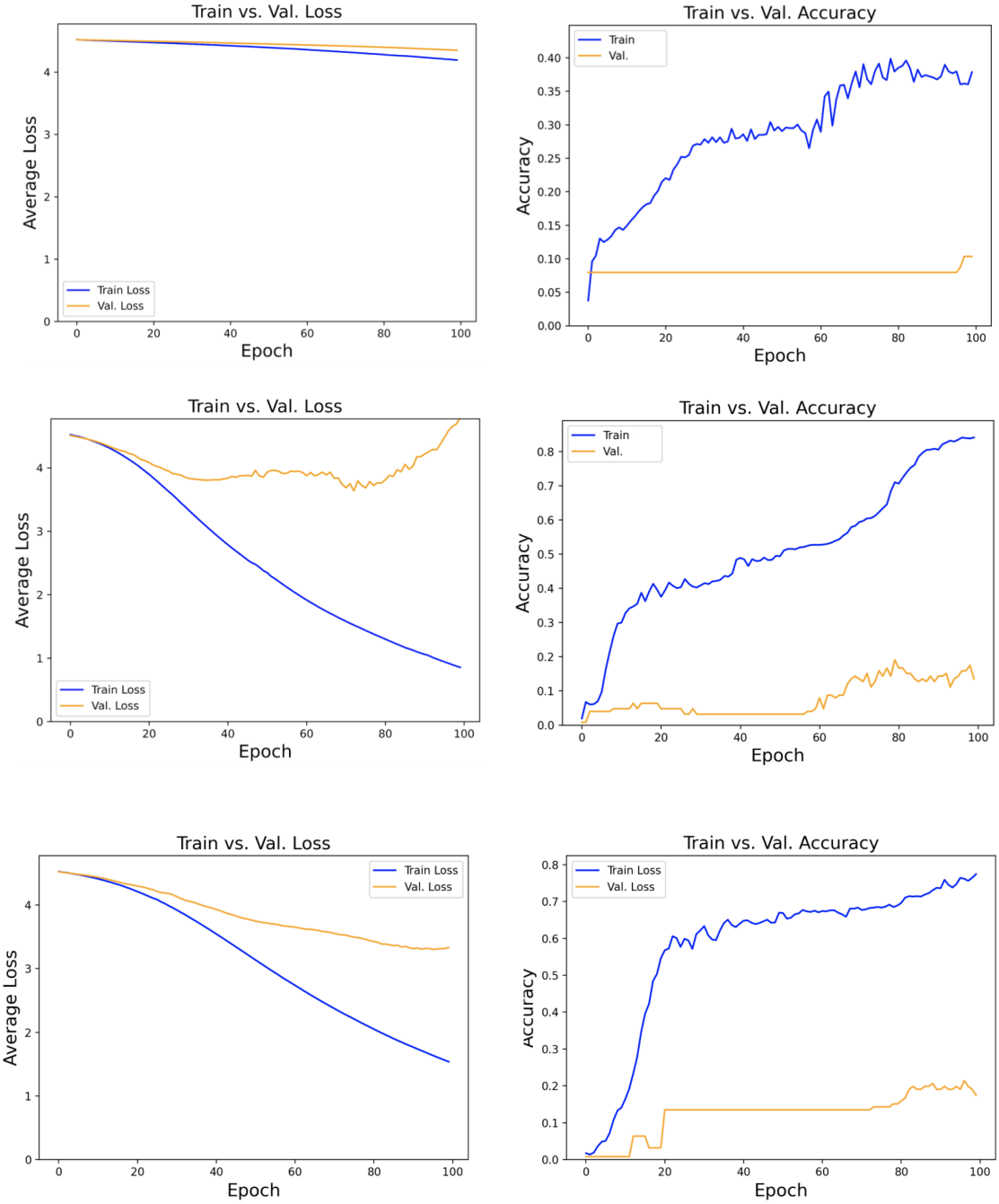


Table 7: Results from comparing various learning rates using the Gene-Level model with CHASMplus/VEST4 inputs.

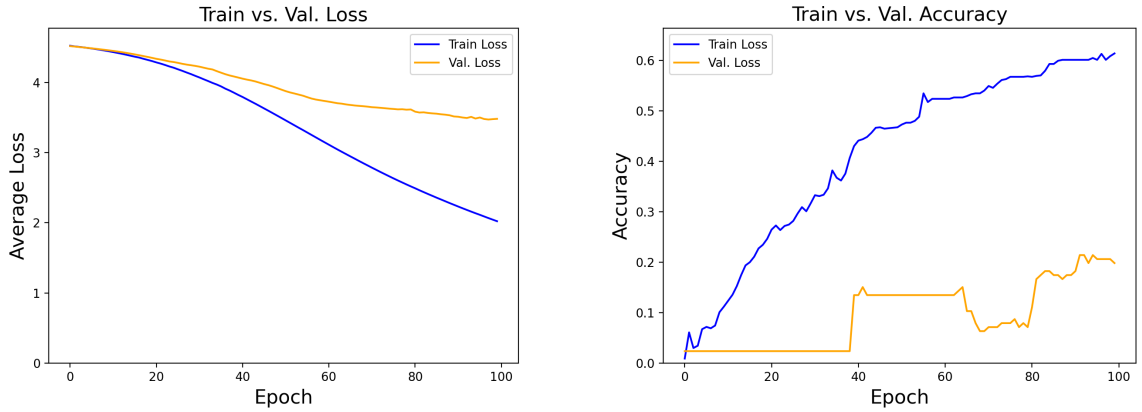
Learning Rate	**Final training loss**	**Final validation loss**	**Best validation accuracy**
0.001	4.1915	4.3485	10.32%
0.005	1.9353	3.5309	18.25%
0.006	1.538	3.3333	17.46%
0.007	1.327	3.5759	14.29%
0.008	0.9815	3.9177	11.11%
0.009	1.0417	4.2569	11.11%
0.01	0.856	4.792	13.49%

We also experimented with different numbers of gene layer hidden units when the learning rate is fixed at 0.005 (Table 8). We found that 22 hidden units performed best in terms of validation accuracy, while 20 hidden units performed better in terms of validation loss. One interesting thing we noticed was that odd numbers of hidden units tended to result in much lower accuracy than their even number neighbors. Our hypothesis is that there is some sort of symmetry in the information passed from the genes, which leads to a loss in information when there is an odd number of units.

Table 8: Results from comparing various hidden unit sizes using the Gene-Level model with CHASMplus/VEST4 inputs.

Learning Rate	**#Hidden Units**	**Final training loss**	**Final validation loss**	**Best validation accuracy**
0.005	6	2.029	3.5526	17.46%
0.005	10	1.9332	3.4365	15.87%
0.005	20	1.9315	3.2961	19.05%
0.005	21	1.9960	3.6314	17.46%
0.005	22	2.0234	3.4829	19.84%
0.005	23	1.9379	3.4317	15.08%
0.005	24	2.0046	3.5350	19.04%
0.005	25	2.0588	3.5884	14.29%
0.005	30	1.9694	3.3651	17.46%

Figure 7: Training vs. validation loss (left) and accuracy (right). Model is Gene-Level CHASMplus/VEST4 with the optimal learning rate (0.006) and hidden features (22) obtained through experimentation



d. Feature-Level Model

Because we noticed the large variation between the two runs with binary input in the Gene-Level model, we tested two runs with each input type for the Feature-Level model. Once again, we noticed much more variation with the binary inputs than with the CHASMplus/VEST4 inputs. However, the best binary input performance was worse than our worst CHASMplus/VEST4 performance. Thus, we used CHASMplus/VEST4 for the rest of the feature-level experiments.

Table 9: Results for various inputs using identified hyperparameters with the Feature-Level model. Results are from four independent runs of the same training/validation data.

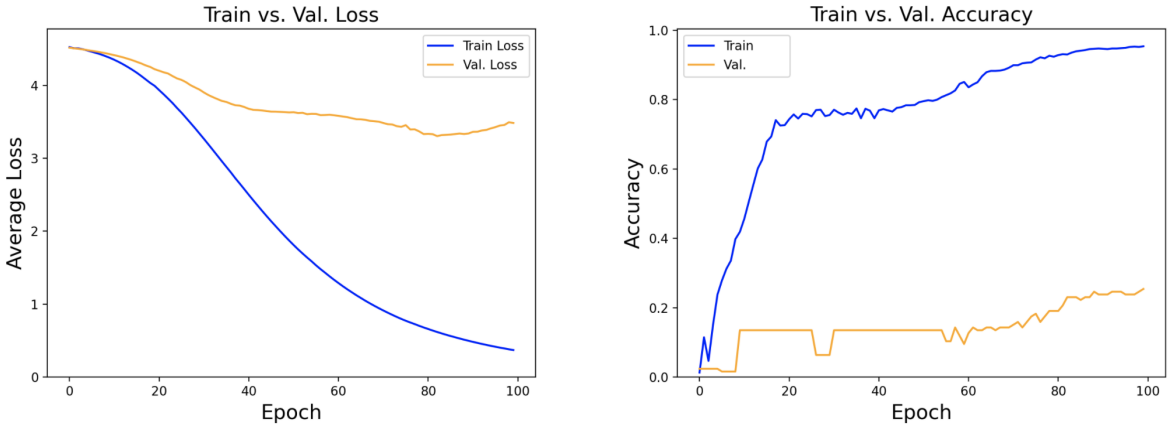
Model	**Input**	**Final training loss**	**Final validation loss**	**Best validation accuracy**
Feature	Binary	2.7843	4.1141	11.11%
Feature	Binary	2.0710	3.6504	16.67%
Feature	CHASMplus/VEST4	1.9352	3.8238	18.59%
Feature	CHASMplus/VEST4	1.9140	3.4700	19.05%

Table 10: Results from comparing various hidden unit sizes using the Feature-Level model with CHASMplus/VEST4 inputs.

Learning Rate	**#Hidden Units**	**Final training loss**	**Final validation loss**	**Best validation accuracy**
0.006	6	1.5383	3.3333	17.46%
0.006	10	0.9971	3.2596	23.02%
0.006	12	0.7612	3.5311	23.80%
0.006	14	0.4961	3.5154	18.25%
0.006	16	0.4614	3.6244	19.05%
0.006	18	0.3697	3.4824	25.40%
0.006	20	0.3170	3.5927	22.22%

We also experimented with the number of hidden units in the feature layer (Table 10). Because we observed that performance dropped with odd numbers of hidden units, we only observed even numbers for this experiment. We found that 10 hidden units gave us the best validation loss, while 18 hidden units gave us the best validation accuracy.

Figure 8: Training vs. validation loss (left) and accuracy (right). Model is Feature-Level CHASMplus/VEST4 with the optimal learning rate (0.006) and hidden features (18) obtained through experimentation



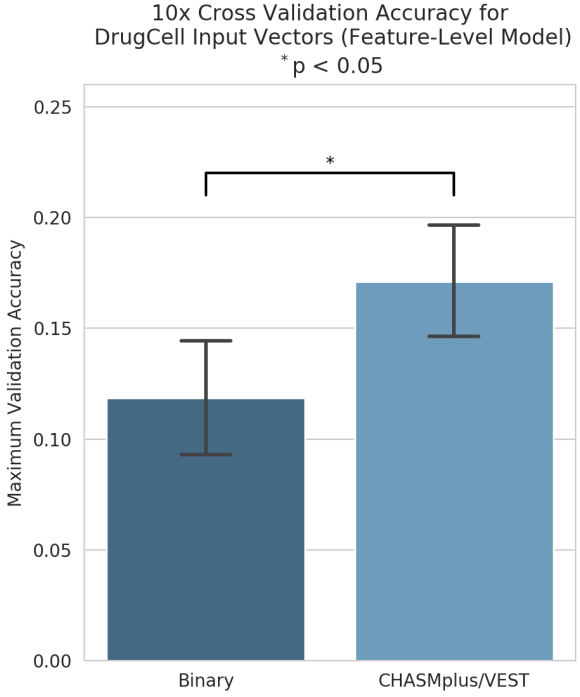
e. Cross Validation

From the identical Feature-Level experiments it appears that the Feature-Level model is able to better contextualize the biologic input information and make more accurate class predictions. To test this we randomized the data and performed 10-fold cross validation on the input data. Indeed, the concatenated CHASMplus/VEST4 input performed significantly better than the binary input vector ($p = 0.0143$, Welch's t-test). Both models outperformed the random frequency-based model, however the overall accuracy for the binary vector dropped compared to the Gene-Level and Ontology-Level models (11.85% vs. 17.11%). This model implemented the previously identified learning rate, however no other hyperparameter optimization was implemented, which suggests accuracy can be increased.

Table 11: Results summarizing 10-fold cross validation experiments. Random model is the frequency-based random class generator (statistics averaged from 100,000 runs).

Model	Input	Average Validation Accuracy	Standard Deviation
Random	N/A	4.16%	2.17%
Feature-Level	Binary	11.85%	4.04%
Feature-Level	CHASMplus/VEST4	17.11%	4.18%

Figure 9: Accuracy plots for different input types on the feature-level model with 10-fold cross validation. Colored bars represent the average maximum validation accuracy on all runs, and error bars represent the standard deviation.



6. Discussion

One of the biggest challenges we faced when training our model was working with the severely imbalanced dataset. When we looked at the outputs of the model, we saw that the model was only predicting the most common classes in our dataset, which was expected considering how few samples were available. Future studies should investigate the role this imbalance played on model predictions, loss, and whether any additional information was learned by including the mutation information for the low frequency classes. The primary advantage of this model is its inherent interpretability, but we did not implement the analysis required to trace back model decisions due to time constraints. The next step in this analysis is to trace back activation states at each of the hierarchy nodes to specific biologic features and identify mutations that were important to subtype classification. Corroborating these results with the literature would increase confidence in model predictions and help inform the next generation of model architectures.

Initial analysis of our data shows differential performance of our input vectors based on model construction. The first series of models trained were the Ontology-Level models, which took weighted sums of the biologic feature inputs and represented genes as that weighted sum value. These models required fewer parameters, took much less time to train, and still managed predictive performance significantly higher than random models. Using the hyperparameters established by Kuenzi, et al., we were able to achieve 10.32% accuracy with the binary input vector. Performance increased to 25.4% by increasing the number of hidden units in the ontology layers, indicating that increasing the number of neurons within the network has a large impact on the model's ability to encode mutation information and make classification predictions. This is further supported by the train vs. validation graphs for loss and accuracy. The ontology model performed better than Feature- and Gene-Level models, with steep, relatively smooth increases in training accuracy in early epochs that eventually level off. The validation accuracy is much lower, but increased within the first 10 epochs to a steady value at around 20%. This shows the Ontology-Level model learns classifications quickly and with minimal computational expense.

The binary input performed best in the Ontology-Level model with 16.67% accuracy, followed by CHASMplus/VEST4 at 15.87%, then VEST4 alone with 13.49% and CHASMplus alone with 10.32% (individual experiments). Interestingly, model performance appears inversely proportional to scoring specificity. CHASMplus is the only cancer-specific scoring system in this analysis and is state of the art in cancer driver analysis. VEST4 is a general purpose algorithm that still performs well predicting cancer drivers. It is surprising that Binary, which makes no biologic or physiological distinction between mutations, performs best in this model for classification. This system simply informs the model that a gene is mutated in a particular sample and allows the model to learn the association between the set of mutated genes and the subtype classification. Another interesting result was that increasing the number of hidden units available to the ontology input caused a ~2x increase in accuracy. This suggests that while the input merely informs which genes are mutated at the beginning, the Ontology-Level model is able to learn more sophisticated representations based on the dimensionality of the neuron layers over the course of training.

We performed the most extensive hyperparameter testing on the Gene-Level model, which performed about as well as the Ontology-Level model in subtype classification. We first used the default parameters described by Kuenzi, et al., and found that CHASMplus/VEST4 performed best (10.32%), and binary performed worst (3.21%). But when we reran these results the Binary input performance jumped to 13.49%. This kind of instability was infrequent, however not entirely unexpected, which is caused by aberrant weight initialization leading to variable learning potential. Subsequent tests showed performance variability, but overall nothing as large as what was initially observed. This may be because we swept learning rate, ontology hidden unit number, and gene hidden unit number, which increased the dimensionality of the model and may have overcome some of the issues posed by weight initialization in the smaller model. The lowest training loss was observed with a learning rate of 0.006, so this learning rate was implemented for the rest of the models. Setting the number of hidden units at the gene layer to 20 increased CHASMplus/VEST model performance to 19.04%, and setting the number of hidden units in the feature layer increased the model performance to 23.02%. CHASMplus/VEST4 performed marginally better than Binary, but more extensive cross validation will be needed to corroborate those results.

The largest difference in predictive performance between inputs was observed in the Feature-Level model. Initial experiments showed that CHASMplus/VEST4 performed better than binary (18.59% CHASMplus/VEST4 vs. 11.11% Binary), which was confirmed by an independent, identical experiment (19.05% CHASMplus/VEST4 vs. 16.67% Binary). We then 10-fold cross validated the model to further investigate the effect this model architecture seemed to have, which showed significant differences in performance (17.11% CHASMplus/VEST4 vs. 11.85% Binary). CHASMplus/VEST4 performed better in 8 out of the 10 cross validation experiments.

Overall, it appears that the Feature-Level model performs best with the highest base-level accuracy (19.05%, CHASMplus/VEST with base parameters). In addition, the Feature-Level model underwent no hyperparameter tuning prior to testing, suggesting model performance can be improved. According to the Ontology- and Gene-Level models, increasing the number of hidden units available to the feature inputs leads to a large increase in model performance. It is not unreasonable to assume that increasing the hidden units in the feature and/or gene layers will increase accuracy and further separate input performance.

The Feature-Level model was the only model that was able to distinguish between inputs, thereby showing that increasing model depth allows the model to learn finer granularity details regarding biologically relevant information provided to the network. This may be the most important consequence of our model architectures. The inclusion of the feature layer allows for the model to learn feature importance through backpropagation and can ostensibly weed out individual features not important for biologically-relevant learned states. This greatly increases the interpretability of the hierarchical model because cell system perturbations that cause disease can now be traced back to individual features, providing a built-in prediction for the mechanism of the disease. Implementing a bank of

neurons before the gene neurons can also allow for feature type mixing. In this analysis we used only scores for specific mutations, but there is no reason why the feature level cannot take in gene expression, epigenetic, or other kinds of data to more accurately inform the model about the specific state of a gene. Genes have complex behaviors, so embedding many kinds of information within the gene neuron representations may present an opportunity to have the model make more sophisticated predictions.

7. Conclusion

Three model architectures were implemented to analyze the difference between mutation representations in a cancer subtype prediction task. While the model performance metric was accuracy, the goal was not to have the most accurate model but rather to find a model capable of identifying biologically relevant features from less relevant representations. To that end, we identified a model capable of incorporating relevant biological information capable of learning more nuanced cancer states. While future work will have to address small sample sizes and class imbalances, the prospect of aggregating disparate biological data types for more accurate neural networks could help bridge the gap between deep learning, research, and precision medicine.

8. References

1. Kuenzi, Brent M, et al. Predicting Drug Response and Synergy Using a Deep Learning Model of Human Cancer Cells. *Cancer Cell*. 21 Oct. 2020, www.cell.com/cancer-cell/pdf/S1535-6108(20)30488-8.pdf.
2. Ma, J., Yu, M., Fong, S. *et al.* Using deep learning to model the hierarchical structure and function of a cell. *Nature Methods* 15, 290–298 (2018). https://doi.org/10.1038/nmeth.4627
3. Tokheim, C., Karchin, R. CHASMplus Reveals the Scope of Somatic Missense Mutations Driving Human Cancers. *Cell Systems* (2019), https://doi.org/10.1016/j.cels.2019.05.005
4. Carter H, Douville C, Yeo G, Stenson PD, Cooper DN, Karchin R (2013) Identifying Mendelian disease genes with the Variant Effect Scoring Tool. *BMC Genomics*. 14(3) 1-16.
5. Chen, H., Li, J., Wang, Y. et al. Comprehensive assessment of computational algorithms in predicting cancer driver mutations. *Genome Biol* 21, 43 (2020). https://doi.org/10.1186/s13059-020-01954-z

Team Contributions

Patrick Wall: Acquired model code/data, identified scoring algorithms and scored all individual mutations. Built input vectors, built 2 input Ontology-Level, Gene-Level, and Feature-Level models. Built and tested Random models. Debugged code for all models. Performed sweeps for learning rate, hidden gene units, hidden feature units, as well as repeat experiments to corroborate results. Tested Feature-Level model. Constructed train and validation sets, as well as 10-fold cross validation sets, and generated cross validation results. Wrote major report sections, created project drawings for report/presentation.

Benjamin Yang: Isolated the hierarchical gene network structure from the available code as the base for all models, modifying dimensions to match our data and labels. Performed input type experiments and hidden units sweep on the ontology-level model and gene-level model. Performed learning rate sweep on the gene-level model. Performed debugging and resolved some errors in the code. Reviewed others' changes, identified bugs and proposed suggestions. Organized and analyzed results from all experiments for the report and the presentation.

Nikhil Karnwal: Implemented 2 Input feature code for ontology level model. Reviewed others' changes, identified bugs and proposed suggestions. Did sweep on hidden units for ontology model for binary input. Implemented k-fold cross validation and made it generic to be used for integrating with all models. Wrote down method and results for 2 input features for ontology level and hidden units sweep.

Yang Li: Performed data cleaning and data analysis for the targets. Process the targets by assigning each unique mutation type a unique integer. Explored the imbalance of the datasets and its potential consequences. Performed debugging and resolved some errors in the code. Found out some command line arguments were not correctly parsed and unitized, and fixed code and bash script to take in and parse correct arguments. Experimented with the classification layer. Experimented with different learning rates.

Yucheng Tu: Removed drug parts and modified train, revised and deleted all drug parts, added classification layer, trained the drug less model. Experimented with different learning rates.