Aggregation Tool for Genomic Concepts (ATGC): A deep learning framework for sparse genomic measures and its application to tumor mutational burden

Jordan Anaya[1], John-William Sidhom[2,3,4], Craig A. Cummings[5], Alexander S. Baras[1,2,4], and the AACR Project GENIE Consortium[6]

[1]**Department of Pathology, Johns Hopkins University School of Medicine, Baltimore, MD, USA**
[2]**The Sidney Kimmel Comprehensive Cancer Center, Johns Hopkins University School of Medicine, Baltimore, MD, USA**
[3]**Department of Biomedical Engineering, Johns Hopkins University School of Medicine, Baltimore, MD, USA**
[4]**Bloomberg Kimmel Institute for Cancer Immunotherapy, Sidney Kimmel Comprehensive Cancer Center, Johns Hopkins University School of Medicine, Baltimore, MD, USA**
[5]**Department of Oncology Biomarker Development, Genentech, South San Francisco, CA**
[6]**American Association for Cancer Research, Philadelphia, PA**

Corresponding author:
Alexander S. Baras[1,2,4]

Email address: baras@jhmi.edu

ABSTRACT

Deep learning has the ability to extract meaningful features from data given enough training examples. Large scale genomic data are well suited for this class of machine learning algorithms; however, for many of these data the labels are at the level of the sample instead of at the level of the individual genomic measures. To leverage the power of deep learning for these types of data we turn to a multiple instance learning framework, and present an easily extensible tool built with TensorFlow and Keras. We show how this tool can be applied to somatic variants (featurizing genomic position and sequence context), and explore various aggregation methods on artificial data for a range of tasks (classification, regression, Cox regression). In addition, we confirm the model can also achieve high performance on real-world data, accurately classifying samples according to whether they contain a specific variant (hotspot or tumor suppressor) or whether they contain a type of variant (microsatellite instability). We then apply our model to the calibration of tumor mutational burden (TMB), an increasingly important metric in the field of immunotherapy, across a variety of commonly used gene panels. Regardless of the panel, we observed improvements in regression to the gold standard whole exome derived value for this metric, with additional performance benefits as more data were provided to the model (such as noncoding variants from panel assays). Our results suggest this framework could lead to improvements in a range of tasks where the sample-level metric is determined by the aggregation of a set of genomic measures, such as somatic mutations that we focused on in this study.

INTRODUCTION

It is often said the whole is greater than the sum of its parts. While this may be true, the whole can often still be identified by the sum of its parts. Cancers can be classified by their mRNA or miRNA levels, protein expression, genomic methylation, chromosomal copy number, or somatic mutations (Hoadley et al., 2018; Jiao et al., 2020). Despite this, it remains a challenge to use genomic scale data in traditional machine learning tasks due to the curse of dimensionality—the high dimensional space makes the data sparse and the models prone to overfitting (Altman and Krzywinski, 2018). Current approaches to this problem include manually reducing the dimensionality through feature selection, or with unsupervised and supervised dimension reduction techniques such as singular value decomposition, negative matrix factorization, and various types of autoencoders. However, it is not always clear how to best reduce the dimensionality of the data for the specific learning task, and dimensionality reduction carries the risk of losing important information.

Of the various genomic measures, perhaps the most common yet complex is somatic mutations. With 3 billion base pairs and alterations which can range from single nucleotide polymorphisms (SNPs) to large insertions or deletions (InDels), there are essentially a limitless number of unique mutations. Somatic mutation data differs from the more familiar germline GWAS mutation data in that tumors don't share an evolutionary history with each other and as a result share very few variants. While with germline mutations it is possible to build a model by selecting a set with a given allele frequency cutoff in the population (Romero et al., 2016), with somatic mutation data this would result in throwing away most of the data leaving only a few hotspots (Chang et al., 2018). To a large extent, this has restricted the application of deep learning approaches to the resolution of the gene rather than specific mutations (Kim et al., 2018; Palazzo et al., 2019; Peng et al., 2020), and most work with mutational data has involved simply summarizing it by counting the total number of mutations, total mutational burden (TMB), or by defining mutational signatures based on the type of SNP and neighboring five prime and three prime nucleotides (Chalmers et al., 2017; Alexandrov et al., 2019).

While focusing on which genes are mutated, how many mutations there are, or what types of mutations are present are valid approaches, ideally a single model would be able to encode all of this information, and do so in a manner which allows it to learn the salient features. For example, maybe it's not the total number of mutations that matter, but the number of a given type. Maybe it's not the gene that matters, but the position within a gene. Maybe it's not the neighboring 5'/3' nucleotides that matter, but sequences further away. Deep learning excels at tasks like these where it's unclear what features are important and sufficient data exists for the model to detect patterns.

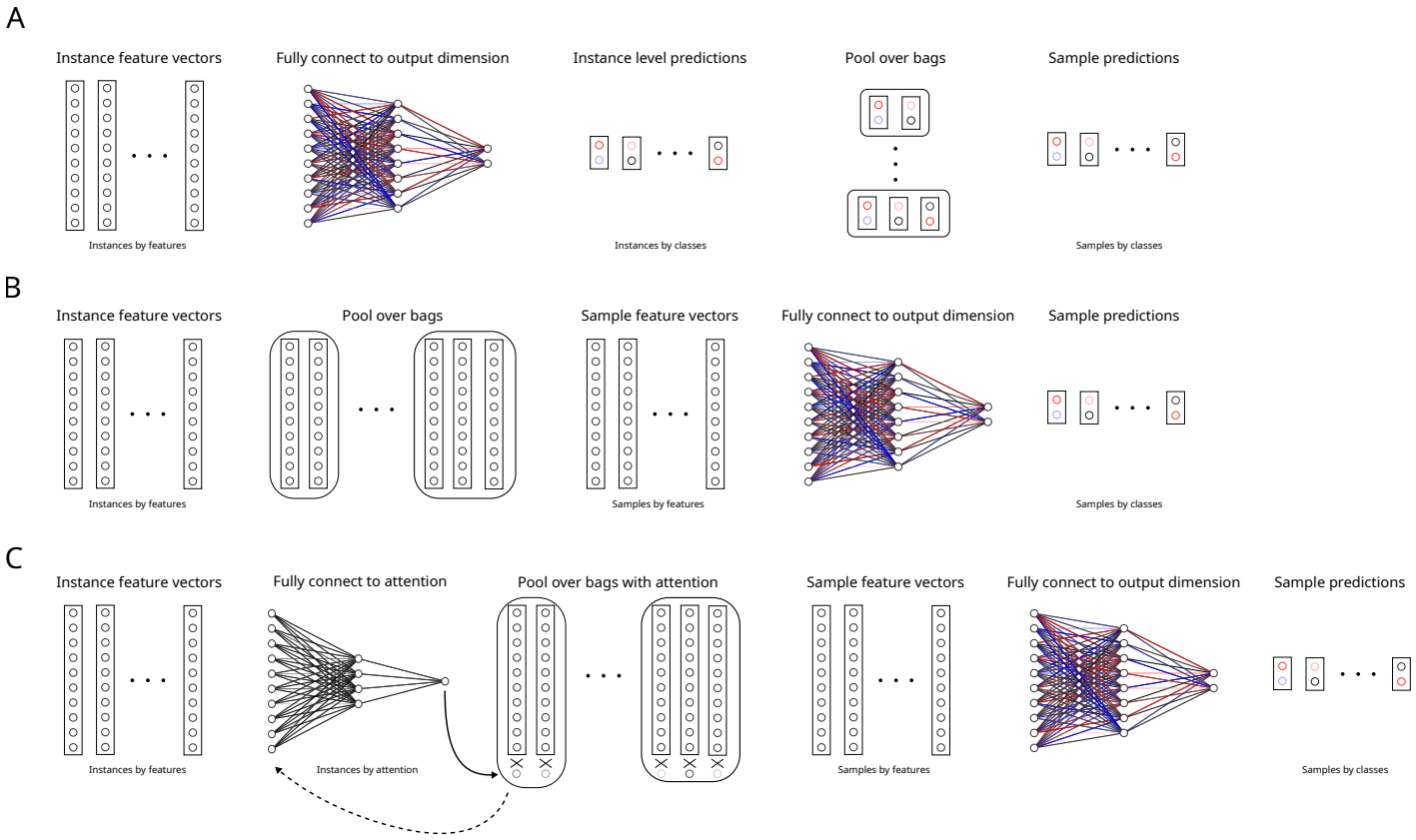

Figure 1. Multiple instance learning approaches. (A) *MIL based on an instance classifier.* The instance feature vectors are used to determine the class of the instances, and then aggregated. (B) *MIL based on a sample classifier.* The instance features are first aggregated into a sample feature vector, and then the sample feature vectors are used to classify the samples. (C) *MIL with attention.* The instances are first reduced to a single dimension, whose value is interpreted as the attention of an instance, and this attention is multiplied against each corresponding instance vector. The resulting weighted instance feature vectors are then aggregated, and the resulting sample feature vectors are then used to classify the samples. It is possible to send the sample feature vectors back to the instances and calculate the attention a second or more time(s).

Although it is clear that we would like to use deep learning to extract relevant features from somatic variants, it is less clear how to apply traditional deep learning approaches to this class of data. In traditional supervised learning each instance would have an associated label. For example, a model that will be able to recognize images of cats would be given a set of training images that are labeled as a cat or not. In contrast, with somatic mutations we have a set of mutations for each cancer sample, and only have a label at the level of the cancer. This label could be type of cancer, response or non-response to therapy, overall survival, etc. In technical terms, this class of problem is a form of weak supervision since each instance is not strongly attached to a specific label, and is known as the multiple instance problem. This has been a long recognized problem in the machine learning community, beginning with trying to identify which conformation of a drug was responsible for its activity (Dietterich et al., 1997). Since then multiple instance learning (MIL) has been applied to a range of tasks (Carbonneau et al., 2018), including problems in bioinformatics (Eksi et al., 2013; Jawa et al., 2013; Agarwal et al., 2015). Given the success of MIL in this area we are interested in the results it can achieve with somatic mutation data.

In MIL, at some point the instance features for each bag must be aggregated into a single sample-level vector that can then be used to compare against the bag label. There are two straightforward ways of accomplishing this: either bringing the instance features to the dimension of the output and then aggregating (the instance model, Figure 1A), or aggregating first and then bringing the sample vectors to the output dimension (the sample model, Figure 1B). The first approach coincides nicely with the traditional formulation of MIL—each positive bag is thought to contain one or more key instances and the negative bags do not contain these instances. The problem is essentially reformulated as classifying the instances and then pooling these classifications. The appeal of this approach is that the model makes it very clear which instances were thought to be associated with each class. However, if the problem of MIL is reenvisioned as the bag label being determined by global properties of the bag instead of single instances, for example a positive bag being defined as containing multiple types of instances or amounts of instances, then only the sample model will be able to identify these properties and correctly classify the bags. As a result, the sample model is considered to be a more general solution to the multiple instance problem, but the model no longer makes it clear which instances were deemed key instances.

In an attempt to get the best of both worlds, Ilse et al. (2018) proposed to utilize the sample model, but to give each instance an attention (Figure 1C). This attention is a single value for each instance, is derived directly from the instance feature vectors, and is multiplied back against the instance feature vectors as a weighting factor. The larger the value, the more important the model views an instance. In addition to giving the sample model a level of explainability, it improves model performance by upweighting the important instances and downweighting the unimportant instances (noise reduction). One issue with the attention is that its value is not only a function of an instance's importance, but also its rarity. If two instances are equally important, but one is present at a much lower fraction (low witness rate), then the rarer instance will be given a higher attention, and if a key instance is very frequent it may be given very little attention if any. To combat this and make the attention clearer, we added L1 regularization on the output of the attention layer to force the values to trend either towards 1 or 0. The more regularization added the clearer the separation between key instances and negative instances, but at the risk of decreased model performance. Another potential issue with attention is that it is independent of the bag. It may be the case that a key instance is only a key instance when it occurs in a certain bag environment, but the model will not make this distinction at the level of the attention layer. If desired, the attention can be made dynamic (Yan et al., 2018) by sending information about the samples back to the instances (Figure 1C), which may provide additional information about how the model is making decisions.

RESULTS

Instance Featurization of Somatic Mutations

The instance features of each bag can be manually generated, for example onehot of gene or whether a variant is a SNP or InDel, or they can be the result of a dimensionality reduction technique such as the latent layer of a variational autoencoder, or they can be the output of an encoder which is trained along with the rest of the graph. Although there is merit for the first two cases, in this manuscript we will focus on features extracted from an encoder. The concepts of two such encoders are shown in Figure 2, one for sequence, and one for position. The sequence concept has the ability to tell the model the

exact mutation that occurred, what type of mutation occurred, etc. In contrast the position concept has the ability to tell the model where in the genome the mutation occurred (gene). We believe these two concepts allow the model to extract most of the features we may be interested in, but it is easy to think of additional concepts such as an allele frequency concept, reading frame concept, etc.

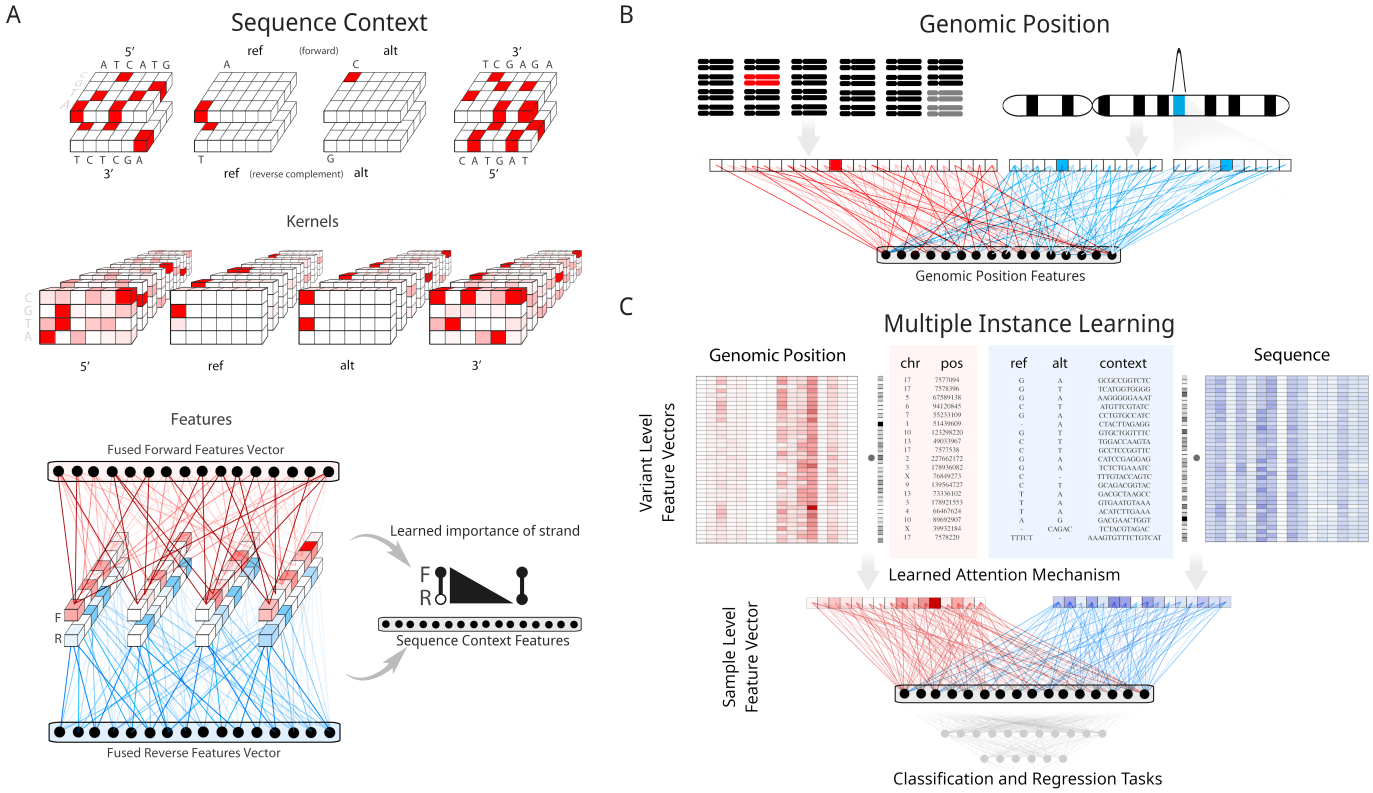

Figure 2. Featurization of mutation data and subsequent application of multiple instance learning. *Sequence and genomic position concepts.* (A) 5', 3', ref, and alt sequences are featurized independently and then fused together in a strand informed manner. Nucleotides are onehot encoded with padded nucleotides given a zero vector. The reverse complement sequences are stacked with the forward sequences such that the representation of a variant is equivalent in both directions. After a single convolution layer the forward features are fused separately from the reverse features, and then these two feature vectors are collapsed with a weighted strand aware aggregation. (B) Chromosome number is onehot encoded along with the position within the chromosome. The fractional position within an intra-chromosomal bin is then sent through two custom activation functions and then all 3 feature vectors are fused. *Sample aggregation* (C) Generally, in mutation profiling a sample is composed of a set of mutations, and a given mutation is defined in some standardized format (MAF/VCF). Each mutation is featurized according to (A) and (B) above and with MIL a weight/attention for each feature vector is developed. These weights are then used to construct a weighted sum (or average) for these feature vectors resulting in a sample-level feature vector. This sample-level feature vector can subsequently be employed for a variety of classification or regression tasks.

We are unaware of previous attempts to featurize somatic variants as they appear in mutation annotation format (MAF). For the sequence encoder we use a single convolutional layer to featurize each sequence component individually (5', 3' ref, alt), and then concatenate the resulting features. Because of the symmetry of DNA and the fact that it is impossible to know which strand a mutation occurred on, there are two equivalent representations for any given variant. This further increases the sparsity of the already sparse mutation data, so to combat this we give the model variants in both the forward and reverse directions. To collapse this extra dimension of data we perform a max reduce, but to give the model the ability to remember strand information should that information be important, we perform a weighted max reduce (Figure 2A). Our sequence concept allows for any length of sequence, with the restriction that the length is even. This is due to how we handle large insertions and deletions. When insertions or deletions are larger than the requested size we define the insertion or deletion as the sequences at the edges, with an equal amount of sequence on each side.

For position our goal is for the model to possibly learn what gene the variant is in, and maybe what part of a gene it is in. As a result, we onehot the chromosome and then use the fractional position of the start position along the length of the chromosome to assign a bin to the start position (Figure 2B). With 100 bins and 24 chromosomes, this gives the model 2,400 unique combinations. In the event that more than one gene falls in a bin combination, or a specific location within a gene is important, we also provide the model the fractional position of the start position in its assigned bin. If our only goal was to give the model gene information we could have simply onehotted the 20k genes—highlighting the fact that a concept is a user's interpretation of how the problem should be framed.

The fractional position in the bin is sent through custom adaptive activations: adaptive sigmoid unit (ASU) and adaptive rectifying unit (ARU), see methods. If you are only interested in variants in the CDS, it may not make sense to use a coordinate system that spans 3 billion base pairs when only ∼30 million of those base pairs are being used. For increased resolution it is possible to remap the genomic coordinates to another coordinate system, potentially one with a different number of chromosomes. This decreases the chance that more than one gene ends up in the same bin combination, but increases the chance a gene is split across bins.

Variant-Level Supervised Learning

To confirm that our sequence and position concepts are accurate representations of a variant and allow the model to learn features we first performed some "positive control" experiments. For the data we used the public MAF provided by The Cancer Genome Atlas (TCGA) MC3 working group (Ellrott et al., 2018). We extracted the reference and alternative sequences directly from the MAF, padding SNPs and InDels and clipping InDels where necessary. Five prime and three prime sequences were pulled from the human genome (GRCh37) using the provided start positions. For each sequence element in the concept we used six nucleotides (more or less nucleotides can be used, keeping in mind however that longer sequences will be prone to overfitting).

When defining a SNP with the pyrimidine as the reference sequence there are only six types of SNPs (T>C, T>G, T>A, C>T, C>G, C>A), and when considering the neighboring five prime and three prime nucleotides there are $6 \times 4 \times 4 = 96$ contexts. It is these 96 contexts which have been used to define mutational signatures (Alexandrov et al., 2013, 2019, 2020). We labeled the variants in the MAF with these 96 contexts if the variant was a SNP, and gave a separate label for other variant types, resulting in a total of 97 classes. Figure 3A shows the results of the 97 class classification problem. Our concept achieved near perfect accuracy, giving us confidence that our sequence concept is a valid representation of a variant.

While learning the 96 contexts showed that our sequence concept could handle variants coming in forward or reverse, it still remained to be seen if the concept could keep track of which features it extracted came in forward or reverse, which is necessary when strand context matters. To test this aspect of the model we decided to see if the concept could learn variant type as provided by the MAF, specifically the consequence/variant classification of: frameshift deletion, frameshift insertion, inframe deletion, inframe insertion, missense, nonsense, silent, splice site, noncoding (5' UTR, 3' UTR, intron). Learning missense vs. nonsense vs. silent classes will require the concept to learn the 64 codons in all 6 reading frames, and forward and reverse sequences are no longer equivalent in the eyes of the classification task.

Obviously this problem requires giving the model the reading frame, but we first decided to see how well it can do without it (Figure 3C). As expected the model was able to learn the InDels and splice sites since those classes don't require reading frame, and without any identifying features the model was unable to distinguish noncoding mutations from the other classes. When providing the reading frame concept in the form of strand and CDS position modulo 3 (and a zero vector when the variant did not fall in a CDS), the sequence concept was now able to correctly classify missense vs. nonsense vs. silent mutations (Figure 3D), indicating that our weighted max reduce of the forward and reverse sequences does allow the model to learn which strand a feature was on. The concept's relatively poor performance on what should be easy categories (inframe InDels) may be due to the limited training examples for those classes or the fact that we clipped InDels at 6 nucleotides and thus the concept is blind to InDels larger than that.

For the position concept we are primarily interested in seeing if our binning of the fractional position along each chromosome provides enough resolution to distinguish genes from one another. As a test we performed another multiclass

classification task where we labeled variants according to some genes of interest and put all other variants in an other category. Figure 3B shows that our position concept was able to classify variants based on the gene they occurred in with high precision. In this example the negative class (other) comprises 99.77% of the data, and it may not be a coincidence that the two most frequent genes in this example (TP53, PTEN) had the best recall, suggesting there may be a lower limit of detection for this concept (around .008% in this case). Given that genes can be right next to each other it's unsurprising the results are not as clear cut as the sequence concept, with the model able to separate nearby genes (for example MSH2 and MSH6 on chromosome 2) in some runs but not in others.

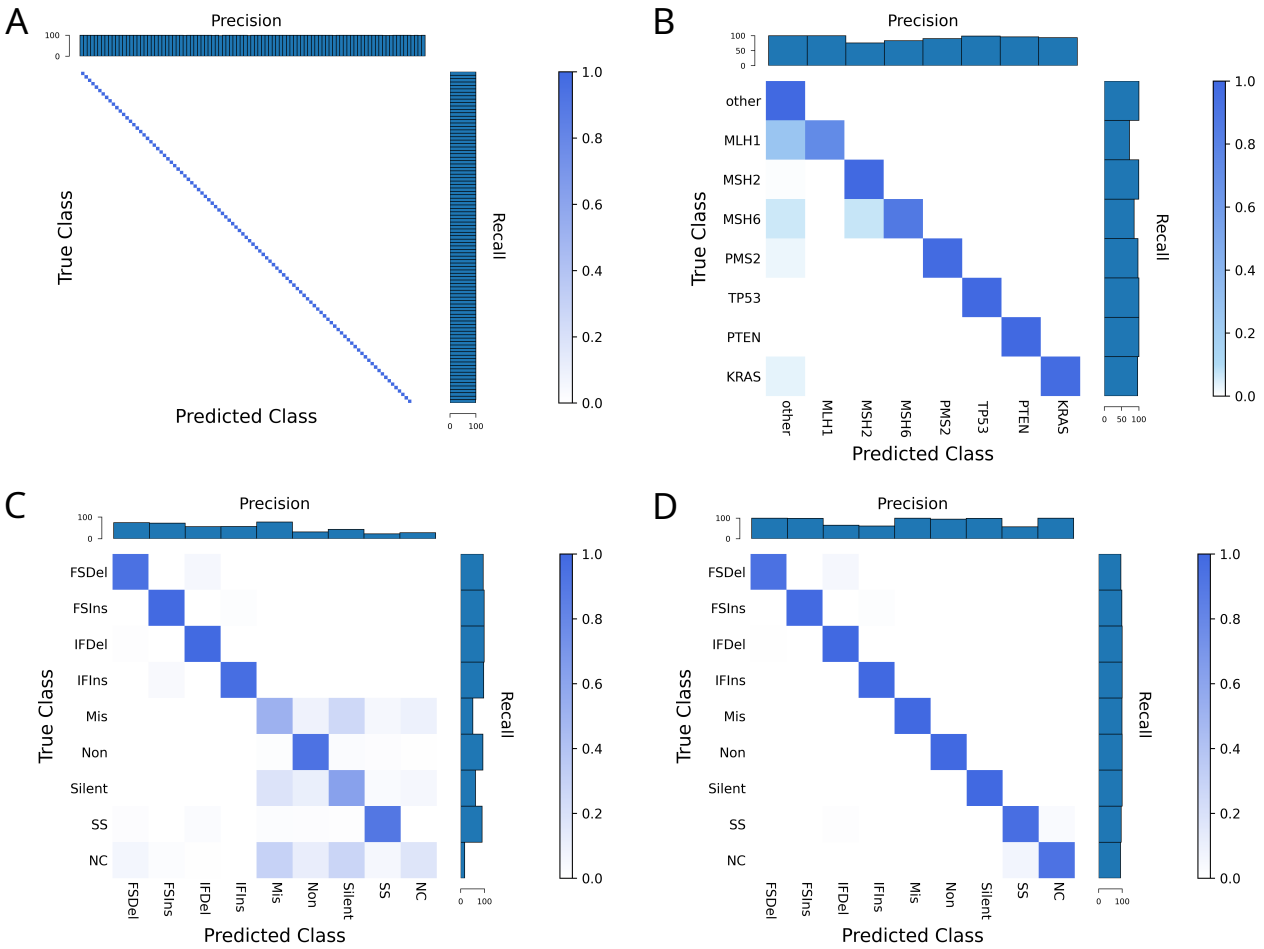

Figure 3. Characterization of performance at the mutation (instance) level. (A) 97 way classification (96 nucleotide contexts and an other/out group category) in which the model is able to accurately learn the conventional trinucleotide mutational contexts. (B) Expanding beyond simple trinucleotide context, the model also accurately can learn a genomic position/gene label. (C & D) Extending further, we show that the model can learn variant consequence encoding as defined by variant effect predictor (VEP) without and obviously better with reading frame information, in (C) and (D) respectively. *FSDel=Frameshift deletion, FSIns=Frameshift insertion, IFDel=InFrame Deletion, IFIns=InFrame Insertion, Mis=Missense, Non=Nonsense, SS=Splice Site, NC=noncoding. All four plots show row normalized confusion matrices.*

Sample-Level Supervised Learning: Aggregation of Genomic Concepts

The coding complexity of MIL stems from the fact that the number of inputs (instances) does not match the number of outputs (samples), and each sample can have a variable number of instances (potentially none). Ideally there would be a framework in a popular deep learning library that allows a user to apply MIL to their problem of interest given that they know how they want to featurize instances. We are not aware of such a framework, so we turned to TensorFlow 2 and tf.keras and built a framework around ragged tensors. The aggregation module is separate from the instance and sample encoders and handles all of the ragged operations, allowing users to write their encoders just as they would normally.

Because ragged tensors are fully supported by TensorFlow, the model has access to all of the default TensorFlow tools such as datasets, sample weighting, losses, and metrics. The model revolves around taking as input the encoders of various genomic concepts and aggregating the resulting feature vectors (Figure 2C), and as a result we refer to our tool as Aggregation Tool for Genomic Concepts (ATGC).

MIL models can vary in performance depending on the characteristics of the data (Carbonneau et al., 2018). Witness rate, bag size, bag composition, number of training examples, problem type (regression vs. classification), label noise, unrepresentative negative space, etc., can all uniquely affect a specific model's performance. It is no wonder then why MIL models are typically designed for a specific problem (most often an image classification), but we would like our model to be a one-stop shop for any MIL problem that involves genomic measures. As a result, we explored various MIL approaches across a range of tasks to try and understand the performance characteristics of each paradigm. Regardless of the approach used, due to the order of the instances of each bag being irrelevant, the aggregation must be a permutation-invariant operation to summarize the contents of each bag. Some examples of typical operations used include mean, sum, and max, with max being particularly relevant when the witness rate is low. For the purposes of our simulated problems we focused on comparing mean and sum as they seemed most suited for the tasks.

In total we tested eight different models. The first six models we tested were: instance model with sum or mean, sample model with sum or mean, attention model with sum or mean. We then began to wonder whether the best model might be one that somehow combines the information of a mean and a sum, so we developed a model we refer to as "both" which uses the weighted average for aggregation like the attention mean model, but then this sample vector is given the sum of the attentions in case the model wants to revert the mean back to a sum. We also took a stab at a dynamic model where the weighted average is sent back to the instances to calculate a second attention which is then used to calculate a weighted sum. We tried to make the architectures of the various models identical, but the models with attention will naturally have extra parameters, and the "both" model was given an extra layer to give the model the opportunity to mix the attention sums and weighted averages if it deems it necessary. Although the extra parameters may partly explain some of the minor performance differences, for the most part when a model is unable to solve a problem no amount of layers will allow it to do so.

Experiments with simulated data

For all simulated data the bag sizes are random, albeit an attempt was made to have the bag size distribution somewhat resemble what one might expect in panel data (with the important exception of each bag having at least one variant). In all cases 1000 samples were generated, with 500 used for training, 300 for validation, and 200 for test. For the classification experiments approximately half of the bags were positive, and half were negative. As a result, a random classifier would have 50% accuracy. Key instances were always variants with a specific sequence (the same 5', 3', ref, and alt), and negative variants were randomly generated. Consequently, the encoder for each model was our sequence encoder. None of the experiments were meant to be particularly hard (witness rate was generally high and there was limited label noise added if any), but that fact underscores a model's inability to solve a problem when it doesn't perform well.

We performed five different classification experiments on all eight models (Figure 4). The first experiment corresponds to the classical definition of MIL: the positive bags contain one or more key instances not observed in the negative bags. As a result, we expected all models to succeed. For the most part all the models solved the problem, but there was a wide range of performance when looking at the categorical crossentropy loss. In particular the models with attention performed better than either the instance models or the sample models without attention, perhaps due to the fact that the bag size is random along with the witness rate, i.e. it is possible to have a large bag with very few key instances. All the attention models showed clear attention, with the dynamic model's attention varying per bag as expected.

In the second experiment we diverged from the classical MIL formulation, and required the model to learn a bag-level property: the presence of two different key instances. Because the problem can no longer be reduced to classifying an instance into a specific class, the instance model is thought to fail in this scenario (Amores, 2013), and our results are consistent with this. With two different key instances the attention in the attention models now have two different bands (although difficult to see in some cases), and many different bands in the case of the dynamic model.

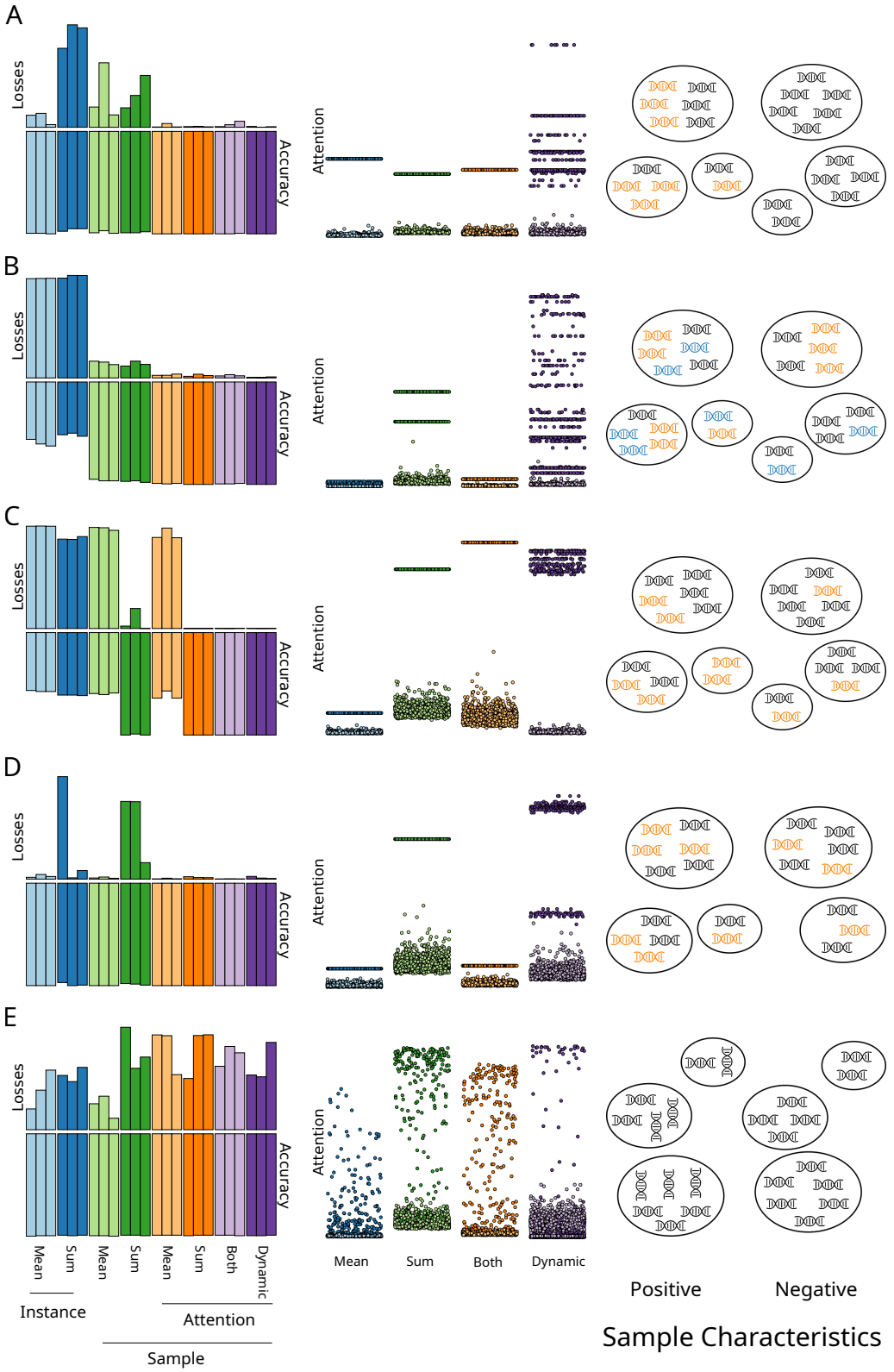

Figure 4. Sample-level classification with simulated data. (A) Positive bags were labeled with a random fraction of a key instance, with negative bags having random variants. (B) A random fraction of each positive bag was filled with two key instances in equal proportions, and negative bags were composed of a random fraction of one of the two instances. (C) Positive bags were given a set amount of key instance regardless of bag size, and negative instances were given a different amount of key instance regardless of bag size. (D) Positive bags were labeled with a specific fraction of key instance, while negative bags were labeled with a different fraction. (E) Positive bags were labeled with random amounts of randomly generated InDels while negative bags only contained SNPs. Losses were normalized relative to the highest loss for purposes of plotting, while accuracy was plotted as is. Attention was also plotted as its raw value, with the key instances plotted as a darker shade. The evaluations and attentions are based on the test samples. Three runs were performed per model with the exact same train/valid/test split.

When it comes to somatic mutations we expect some problems to require the model to learn the amount of a type of variant, and as a result labeled bags with two different specific amounts (Figure 4C). We expected the sum aggregation in this scenario to outperform the mean aggregation, but unexpectedly the instance sum aggregation did not solve the problem. Upon reflection, the output of the model immediately goes into the softmax of the crossentropy loss function, and as a result for the instance model and classification the distinction between a sum and mean is somewhat blurred. The fact that

the sample model with a mean cannot solve this problem highlights a potential issue with using mean, which is common in the MIL literature.

We can also envision problems with somatic mutations that require the model to be able to recognize the fraction of a type of variant, so we next labeled bags with two different fractions of an instance. As a first thought it might seem that only the mean aggregation will be able to solve this problem, but the sum contains all of the same information as the mean, and just requires further operations to convert the sum to the mean. With that said, the models with a mean did perform noticeably better. Interestingly the dynamic model showed two groups of attentions, corresponding to the two bag labels (Figure 4D).

As a final test we labeled positive bags with a random amount of a type of variant (random InDels). Surprisingly there wasn't much difference among the different models, and none of the models achieved a loss as low as in the other experiments. It's possible that with the bag sizes/numbers used and the wide range of possible InDels the model wasn't able to sample enough of the InDel space. This idea is consistent with not all InDels being given attention in the attention models (Figure 4E).

In addition to succeeding in classification tasks we would like our model to also perform well in regression tasks. To convert our model from one that solves a classification task to one that solves a regression problem all we have to change is the final output layer and loss function (from two output nodes and categorical crossentropy for a two class classification problem to a single output node and either mean squared error or negative partial likelihood for standard regression and Cox regression respectively). For the standard regression experiments we again tested all eight models. In Supplemental Figure 1A we had the bag label be a linear function of key instance count. Only the models which included a sum for aggregation were able to solve the problem. As before with classification, the instance model is attractive for its interpretability since its output is exactly how much each instance contributed to the bag label. However, the only regression problems the instance model sum can solve are when the bag label linearly varies with a key instance. This is seen in Supplemental Figure 1B where the bag label is now a quadratic function of key instance count. We also did a regression where the bag label depended on the fraction of bag filled with a key instance, and both the sample means and sums were able to the solve the problem, mirroring what we saw with classification (Supplemental Figure 1C).

At this point if a problem requires knowing the amount of an instance we are confident that the mean does not contain the necessary information, so for the Cox regression experiments we only tested the five models that contain a sum. For the first experiment we had the log risk increase with increasing amount of key instance in a monotonic manner. All five models solved the problem and the model predictions showed the expected pattern, although the models with attention seemed to have tighter predictions (Supplemental Figure 2A). The appeal of using a neural net for Cox regression is that it can find nonlinear relations, so we tested this expected ability by having the amount of key instance affect the log risk in a nonmonotonic fashion (the value initially increases and then decreases with increasing key instance). Interestingly, only the models with attention picked up this pattern (Supplemental Figure 2B).

When it comes to including sample information into the model there are two obvious locations to do this: either immediately before the aggregation or immediately after. If added before, the attention will now be a function of instance and sample features. To explore what this might look like we performed some experiments with sample information. As seen in Figure 5, there wasn't a clear performance difference whether the sample information was incorporated before or after the aggregation. The attention in Figure 5A is what we might hope to expect from including the sample information before the aggregation. In this experiment the key instances contribute to the bag value differently depending on the sample type, and this is correctly reflected in the attention. However, in Figure 5B the sample information appears to be impacting both the value of the key instances and background instances for the models with a sum, indicating that interpreting the attention when it is a function of instance features and sample features may not always be straightforward. Although in these examples there wasn't a striking performance difference, there were only three sample types. If the sample information is high dimensional then the instance-sample space will be very sparse, and how the sample information is included in the model may then impact performance.

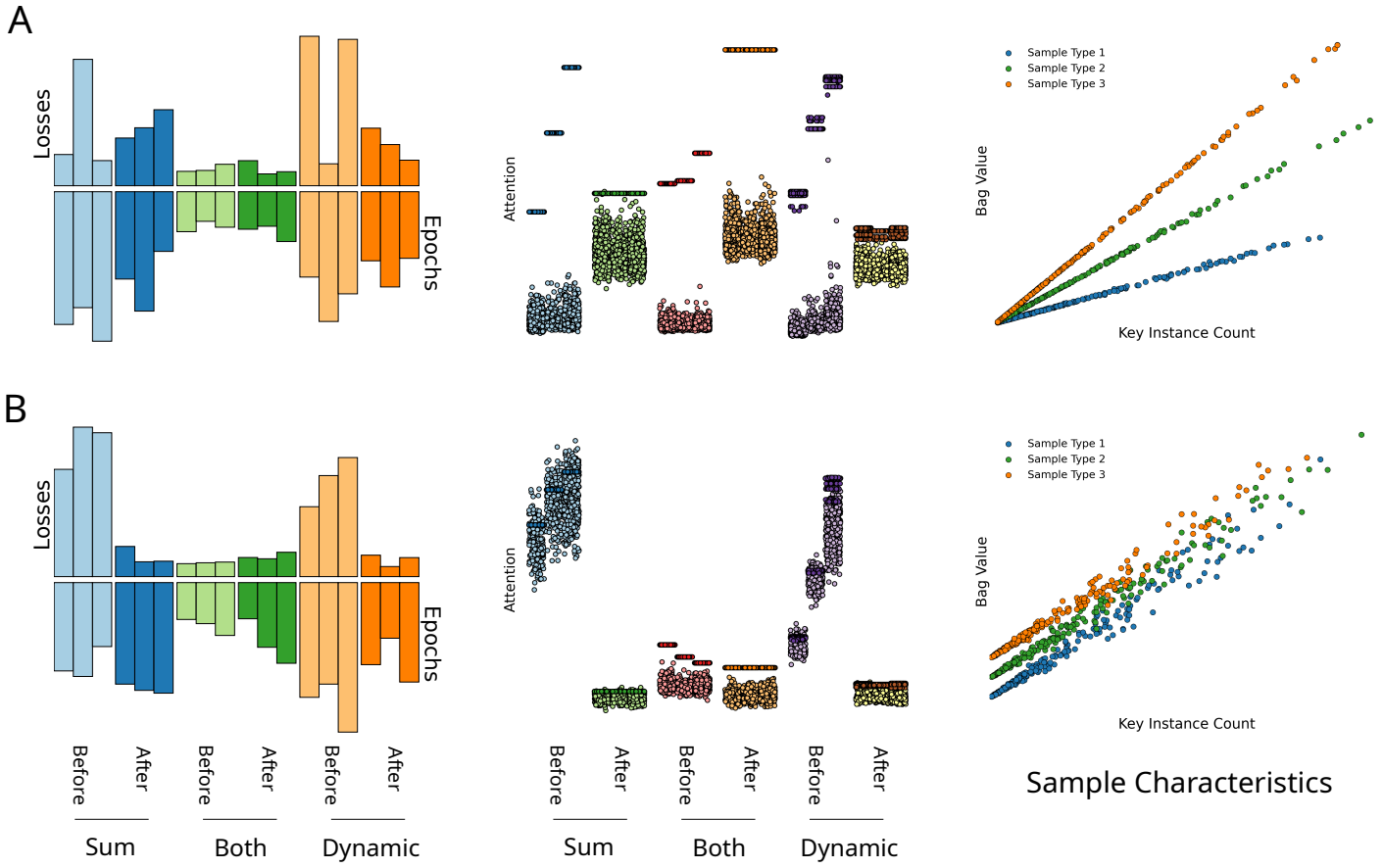

Figure 5. Incorporating sample information with simulated data. Both experiments are regression tests where the bag value is determined by the key instance count in a linear manner, but in (A) the sample type affects how much each key instance contributes to the bag value, while in (B) it does not, but the bag value is affected by the sample type. Losses and epochs are normalized to the largest value, attention is plotted as is. Evaluations and attentions are based on the test samples. In the plot of attention the three columns per model are the three sample types, and key instances are plotted as a darker color. Three runs were performed per model with the exact same train/valid/test split.

Experiments using real-world genomics data

Our experiments with artificial data were promising, but did not mimic the exact characteristics we might see in real data. As a result, we checked whether our model could identify samples containing either mutational hotspots or tumor suppressors, and at the levels that these would be present in actual data. Given that a tumor suppressor is characterized by mutations throughout the gene, there is not a specific sequence for the model to find. As a result, we will rely on our position concept to classify samples according to whether they have a mutation in a specific tumor suppressor. And given that hotspots are characterized by changes in a specific amino acid (which corresponds to a specific nucleotide sequence) we will use our sequence concept to identify these samples. It would be possible to use our position concept for both tasks, but mutations near a hotspot would be picked up by the position concept and be false positives. Our simulation experiments suggest there isn't a clear choice among the attention models (although a mean shouldn't be used for problems requiring knowing amount), but we decided to use the weighted sum for the real-world data.

For the tumor suppressor task we chose to label TCGA MC3 samples based on whether or not they had a mutation in the PTEN gene. The model's recall (99.89%) was higher than its precision (93.83%), indicating that there may be mutations near the PTEN gene which the model cannot differentiate. Looking at the output of the attention layer (Supplemental Figure 3A) we see that most variants were given a weight close to 0 while a few were given higher attentions. This is what we would expect—samples contain upwards of thousands of mutations but only a single variant is responsible for the label of the sample so the model needs to ignore all the other variants. Looking at the variants which received the most attention

most were indeed mutations in PTEN, but the others were in neighboring genes, which is consistent with the precision metric (giving attention to these variants could result in false positives).

The PTEN mutations fall in two different bins in our position concept, it can take some time for the model to pick up both bins during training, and some other genes also fall in the bins. Clearly our position concept is not ideal for labeling variants according to the gene they fall in, so why wouldn't we just give each gene its own bin? The answer is that we would like our position concept to be gene agnostic because definitions of genes differ, and we would prefer the model to define its own idea of what a gene is. Additionally, our model can adapt to all types of variant including the myriad of possible noncoding variants (for example TERT promoter mutations), which would represent a computational challenge to onehot encode, and becomes necessary when working with non-exomic data (such as whole genome sequencing). With that said, if a user believes a task is driven by mutations in specific genes, it would make sense to onehot those genes.

A hotspot is defined by an amino acid change and not a nucleotide change, so it is possible for a hotspot to be characterized by several different nucleotide sequences. To keep things simple we labeled samples by whether they had a mutation at a specific nucleotide in the BRAF gene, corresponding to the BRAF V600 hotspot. The model was able to achieve perfect precision and recall. As seen in (Supplemental Figure 3B) the model again downweighted most variants and only gave attention to a specific variant, the BRAF V600 mutation. Because all the BRAF V600 variants had the exact same sequence they were all given the exact same attention.

Real-World Example of Aggregation: Microsatellite Instability

Our sample-level tasks up to this point were fairly simple, with samples being labeled with varying amounts of one or two key instances. A more complex problem requiring both correct variant attention and aggregation that our model should be able to solve is microsatellite instability classification given full (unfiltered) exome data. Characterized by deficiencies in the mismatch repair proteins (MLH1, MSH2, MSH6, PMS2), these tumors accumulate InDels at microsatellites due to polymerase splippage. Given that microsatellites are characterized by short repetitive sequences, and the InDels are multiples of these repeats, a deep learning model which claims to learn variant-level and sample-level features should be expected to solve this problem.

Current bioinformatic approaches to predicting MSI status rely on manually featurizing variants. For example, the MANTIS tool uses a BAM file to calculate the average difference between lengths of the reference and alternative alleles at predefined loci known a priori to be important (Kautto et al., 2017). Similarly the MSIpred tool calculates a 22 feature vector using information from a MAF file and loci again known to be important, specifically simple repeat sequences (Wang and Liang, 2018). Both of these approaches are valid and produce accurate predictions, but they rely on knowing the nature of the problem, and which variants to focus on. If it was unknown a priori what properties of a variant were important for determining MSI status then these manual featurizations would likely not be very effective.

Because mutations characteristic of MSI occur at many different repeat sites, and because the repeats have a distinctive sequence, we chose to use our sequence concept for this problem. For the data we opted to use the controlled TCGA MC3 MAF rather than the public MAF since the public MAF excludes variants in introns and most simple repeats fall in introns. The TCGA has ground truth labels as defined by PCR assay for some tumor types, and we were able to obtain labels for UCEC (494), STAD (437), COAD (365), READ (126), ESCA (87), and UCS (56) tumor samples. For the sequence concept we went out to 20 nt for each component (5', 3', ref, and alt) to give the model enough space to learn about microsatellites.

We believe MANTIS and MSIpred are considered state of the art when it comes to MSI classification performance, and as can be seen in Figure 6A our model (ATGC) performs comparably. Given the performance of our model we were excited to see what was driving the performance. To classify whether a variant falls in a repeat region or not MSIpred relies on a table generated by UCSC (simpleRepeat.txt). Using this file we labeled variants according to whether they fell in these regions, and as seen in Figure 6B variants which occurred in a simple repeat region were much more likely to receive high attention than variants that did not. It is important to emphasize that all we provided the model were the 5', 3', ref, and alt sequences of the variants as defined in the MAF file, and it was able to learn through the sample labels to pay more attention to variants in simple repeats.

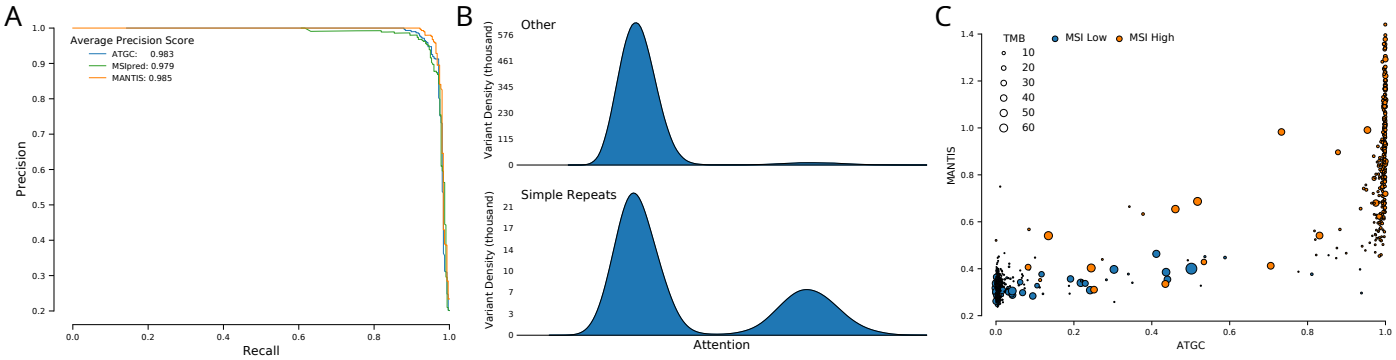

Figure 6. Application of ATGC MIL framework accurately predicts MSI status by learning which are the salient variants. (A) Average precision recall curves over 9 stratified k-fold splits for MSIpred and ATGC, precision recall curve for MANTIS for samples which had a MANTIS score. (B) Probability density plots of the output of the attention layer stratified by whether a variant was considered to be in a simple repeat from the UCSC genome annotations. (C) MANTIS scores plotted against ATGC output probability showing high concordance of ATGC model output to MANTIS scores. Samples are color coded by the PCR based MSI status label, and the size of each sample corresponds to its total mutational burden (in thousands).

While our model and MSIpred output a probability ranging from 0 to 1 with most predictions at the extremes, MANTIS provides a score ranging from 0 to 2, with 0.4 used as the cutoff to separate MSI stable vs. unstable tumors. We were curious if our model and MANTIS were wrong for the same samples, and if when our model was unsure MANTIS was also unsure. In general when our model was very confident that a sample was MSI high the MANTIS value was also high and when our model was very confident a sample was MSI low the MANTIS value was also low Figure 6C. There are a few cases where the sample is labeled MSI high by the PCR, but both MANTIS and our model are confident the sample is MSI low, perhaps indicating the PCR label may not be 100% specific (or alternatively indicates an issue with the BAM files for these samples). There are also cases where MANTIS predicts the correct label but our model doesn't, or our model predicts the correct label but MANTIS does not, suggesting that the best MSI predictor may be one that incorporates predictions from multiple models.

We did not build our model with the intent of learning how to classify tumors according to MSI status, and it's possible our sequence concept is not ideal for this purpose. We also did not do an extensive parameter search, e.g. what length of sequence to use, or add other concepts into the model such as position, or include sample information, so it is entirely possible that the performance on this dataset could be improved upon. We simply wanted to see if a deep learning model could classify samples based on biologically meaningful features extracted from variants, and with performance comparable to manual attempts. While in the case of MSI we knew that there were relevant variant features to extract and what they were, this is not always the case, and we expect deep learning methods to be required for these problems.

Potential Application: Tumor Mutational Burden

One metric involving somatic variants where it's unclear if or what variant features may be involved is tumor mutational burden (TMB). Although defined simply as the number of mutations in a tumor, TMB can be more narrowly defined as the mutations that occur in the exome, the mutations which occur in coding sequences, mutations which occur in genes which are expressed, nonsynonymous mutations, or even a very specific type of mutation such as frameshift InDels. The reasons for potentially being interested in a subset of mutations rather than every mutation is the more neopeptides a tumor has the more likely it is to be recognized by the immune system, and certain mutations contribute more to the generation of neopeptides than others. TMB has generally been shown to correlate with response to immunotherapy (Goodman et al., 2017; Samstein et al., 2019), and the FDA has recently approved a tumor agnostic indication for immunotherapy (FDA, 2020) based on TMB along with a previous approval for immunotherapy for tumors displaying microsatellite instability (Marcus et al., 2019).

The gold standard for assessing nonsynonymous mutation burden would be from whole exome sequencing (WES); however, this is not a routine part of clinical care due to its costs and tissue sample requirements. Instead, TMB is

often estimated with panel-based targeted sequencing (as was the case for the recent FDA approval for the "TMB-High" indication). These panels only cover a fraction of the exome, often around 1-5 Mb of ∼33 Mbs, and to complicate matters there are many different panels used across academic and commercial laboratories, each with potentially different sample processing protocols, sequencing technologies, and variant calling pipelines (Fancello et al., 2019). While we cannot explore the effects of all these variables, we can explore the effect of downsampling to a specific set of genes while assuming the other variables are being held constant by artificially downsampling TCGA's exome data.

Panel	Input	Model	MSE	MAE	R-squared
MSK-IMPACT341	Nonsyn	Counting	0.3668	0.4992	0.4394
		ATGC no Concepts	0.1497	0.3093	0.7712
	CDS	ATGC no Concepts	0.1253	0.2822	0.8085
		ATGC with Position	0.1044	0.2505	0.8405
		ATGC with Sequence	0.1058	0.2542	0.8383
	CDS with NCI-T	ATGC no Concepts	0.0727	0.2034	0.8884
MSK-IMPACT468	Nonsyn	Counting	0.2802	0.4325	0.5717
		ATGC no Concepts	0.1290	0.2870	0.8029
	CDS	ATGC no Concepts	0.1066	0.2600	0.8370
		ATGC with Position	0.0903	0.2326	0.8620
		ATGC with Sequence	0.0910	0.2357	0.8610
	CDS with NCI-T	ATGC no Concepts	0.0641	0.1924	0.9016
MDA-409-V1	Nonsyn	Counting	0.2453	0.4062	0.6250
		ATGC no Concepts	0.1056	0.2586	0.8386
	CDS	ATGC no Concepts	0.0857	0.2325	0.8691
		ATGC with Position	0.0760	0.2162	0.8839
		ATGC with Sequence	0.0772	0.2182	0.8820
	CDS with NCI-T	ATGC no Concepts	0.0569	0.1820	0.9127
VICC-01-R2	Nonsyn	Counting	0.2307	0.3889	0.6474
		ATGC no Concepts	0.1303	0.2848	0.8009
	CDS	ATGC no Concepts	0.1065	0.2579	0.8373
		ATGC with Position	0.0984	0.2452	0.8496
		ATGC with Sequence	0.0999	0.2483	0.8474
	CDS with NCI-T	ATGC no Concepts	0.0670	0.1962	0.8972
DFCI-ONCOPANEL-3	Nonsyn	Counting	0.2488	0.4056	0.6197
		ATGC no Concepts	0.1392	0.2975	0.7872
	CDS	ATGC no Concepts	0.1137	0.2679	0.8262
		ATGC with Position	0.0960	0.2406	0.8532
		ATGC with Sequence	0.0973	0.2427	0.8513
	CDS with NCI-T	ATGC no Concepts	0.0657	0.1956	0.8991

Table 1. TMB regression metrics for TCGA MC3 samples

To explore the ability of our model to estimate exomic TMB given panel data, we returned to the public TCGA MC3 MAF. We defined TMB as nonsynonymous mutations per megabase, used log-transformed values, and to prevent outliers from affecting the model excluded samples with a TMB greater than 64. Applying this filter along with some quality control filters (see methods) resulted in a total of 9,942 samples. For restricting the TCGA data to panel coordinates we relied on the AACR GENIE 7.0 definitions (AACR Project GENIE Consortium and others, 2017).

A conventional method for estimating nonsynonymous TMB from panel data is to simply count the number of non-synonymous mutations seen in the panel and then divide by the panel coding sequence size to get to mutations per Mb

Panel	Input	Model	MSE	MAE	R-squared
DFCI-ONCOPANEL3	Nonsyn	Counting	0.2637	0.4154	0.2548
	CDS	ATGC no Concepts	0.1023	0.2511	0.7108
		ATGC with Position	0.0990	0.2441	0.7202
		ATGC with Sequence	0.1046	0.2531	0.7045
	All	ATGC no Concepts	0.0788	0.1985	0.7772
		ATGC with Position	0.0521	0.1781	0.8528
		ATGC with Sequence	0.0589	0.1883	0.8336
VICC-01-R2	Nonsyn	Counting	0.2563	0.4088	0.2757
	CDS	ATGC no Concepts	0.1047	0.2533	0.7042
		ATGC with Position	0.0980	0.2420	0.7229
		ATGC with Sequence	0.1019	0.2503	0.7121
	All	ATGC no Concepts	0.0854	0.2215	0.7585
		ATGC with Position	0.0678	0.2040	0.8084
		ATGC with Sequence	0.0785	0.2182	0.7783

Table 2. TMB regression metrics for PCAWG ICGC samples

(Samstein et al., 2019). To compare the performance of our model against this standard we restricted the TCGA data to the footprint of several panels and ran our model with no concepts, our position concept, or our sequence concept based on inputs of all CDS variants as compared to restricting to nonsynonymous variants contained in these panels, see Table 1. ATGC with no concepts (effectively a nonlinear regression with all variants having equal weight) is a major improvement over the naive counting approach in terms of predicting the whole exome derived metric. Additionally, we observed that the inclusion of all CDS variants over just nonsynonymous variants improved performance. When featurizing variants with either our position or sequence concept we see further improvements in the fit as defined by either mean squared error, mean absolute error, or R-squared. We applied random permutation testing in this context by shuffling variants with respect to which sample they came from while keeping the total number of variants of a given sample the same as in the observed data. The performance benefit of variant featurization was abrogated with the application of this random permutation testing (Supplemental Table 1).

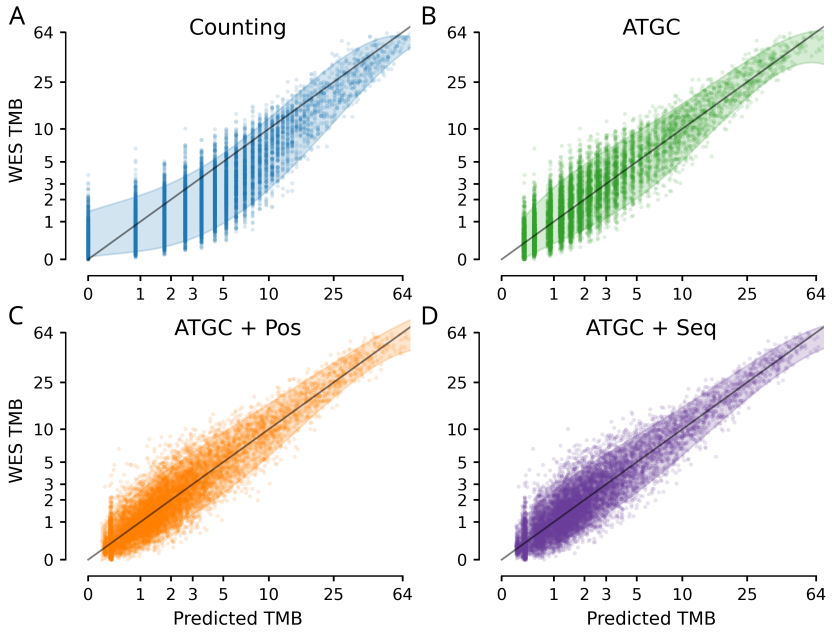

Figure 7. Comparison of panel to exomic nonsynonymous TMB regression shows improved regression characteristic with the application of ATGC and variant featurization as compared to naive counting based approaches. Predicted vs whole exome sequencing derived TMB based on the TCGA MC3 data using the footprint of the MSK-IMPACT468 panel. Shaded area represents the upper and lower quantile estimates for 90% of the data (i.e. quantile regression). The ATGC model predictions are from 8 different K folds. (A) Counting—count of nonsynonymous mutations divided by size of coding sequence covered by panel. (B) ATGC model applied with no featurization concepts, i.e. all variants have equal weight of 1. (C & D) ATGC with position or sequence concepts allowing variants to be differentially weighted.

Based on Table 1, it was evident that the inclusion of additional variants (outside of those used to derive the whole exome-level metric) resulted in improved performance. The TCGA MC3 data is WES derived and therefore the vast majority of variants reported in these data are contained within exons. In contrast, panels often include non-exomic regions in their design (Supplemental Figure 4). For example, the DFCI-ONCOPANEL-3 is reported to have more intronic sequence than exonic sequence coverage, and the VICC-01-R2 is approximately half non-CDS sequences (which is expected to allow for fusion detection in these panels). As noted above, models trained on TCGA data were limited to input variants that fell in the CDS with a regression target of whole exome derived nonsynonymous mutation rate. To further explore whether the inclusion of additional variants (for example intronic variants) would improve model performance we turned to whole genome data available from The Pan-Cancer Analysis of Whole Genomes (PCAWG). As with the TCGA MC3, we applied the footprint of the panels to the PCAWG whole genome data and set the regression target to the nonsynonymous mutation rate in PCAWG data restricted to the exome. As can be seen in Table 2 the inclusion of non-exomic variants helped all models except the naive counting based model (which doesn't change); however, the additional variants appeared to help the models with variant featurization more than the nonlinear model.

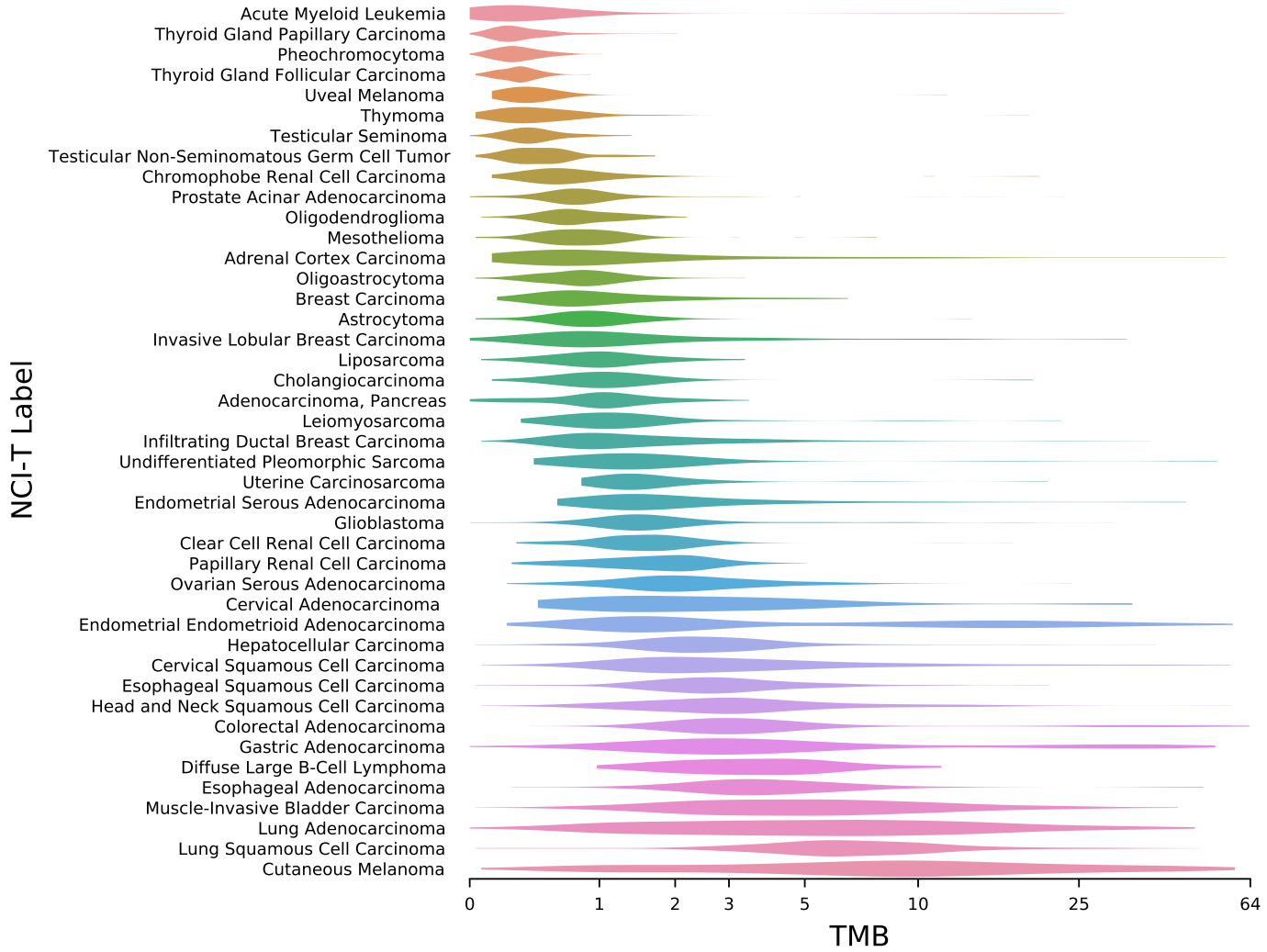

Figure 8. TMB distributions by NCI-T Label.

To further explore the difference in fit between the different models we looked at the predicted TMBs vs actual TMBs. As seen in Figure 7, the counting method initially severely underestimates TMB since it has to predict 0 any time there are no variants the panel, and then classically overestimates TMB. The nonlinear regression is able to make the residuals more

symmetrical, but the model must predict the same value for a given number of panel variants (illustrated in the spaces between columns of predictions). With variant featurization the model is now able to give each variant a different weight, so two samples with the same number of panel variants can have different exome predictions. This is illustrated in both the position model and sequence model plots by the absence of the stepwise predictions.

Sample information can provide a lot of information about the expected TMB value. For example, when looking at the TMB distributions of the MC3 data by NCI-T label many groups show a tight spread around a small average TMB value (Figure 8). We wondered how much incorporating this information into the model would improve performance, and how it compared to the performance improvement we see with variant featurization. As can be seen in Table 1, simply adding this information into our model with no concepts resulted in a larger performance gain than we see with variant featurization. This performance improvement is not specific to grouping samples by NCI-T label, as each TCGA study label (BRCA, PRAD, etc.) similarly has a characteristic TMB distribution as has been previously reported (Lawrence et al. (2013), Supplemental Figure 5), and using those labels provides similar performance gain (Supplemental Table 2).

DISCUSSION

Many genomic technologies generate data which can be considered "large p, small n", wherein the number of possible features per sample greatly exceeds the number of samples. For example, somatic mutations can occur anywhere in the genome, thus creating an enumerable number of possible unique features per sample. Similar considerations apply to CHIP-SEQ, methylation data, and protein modifications. MIL is a natural solution to these problems given that it can extract relevant features from the instances while at the same time learning sample-level properties informed by the learned instance features. Furthermore, as these measurements begin to be used in the clinic and researchers turn to deep learning for their analysis, it will likely be necessary for the models to explain their decisions given a patient's right to understanding treatment decisions (Goodman and Flaxman, 2017). Again we see that MIL with attention is a natural solution for this requirement, and we assume the lack of its use in analyzing genomic data when it has taken off in many other fields (particularly image analysis) is likely due to the coding complexities MIL presents. To counteract this we developed an easily extensible model that simply requires a user to know how they want to featurize their instances. Once a user chooses or develops their own instance featurization, then our model will aggregate those features and with an appropriate loss function can be pointed to any sample-level metric.

When it comes to applying MIL to genomics problems an important consideration will be the signal to noise ratio. For example, if trying to identify samples according to a mutation in a specific gene this task will be easier with panel data which often only has a handful of variants per sample. You are essentially looking for a needle in a haystack so you want the smallest haystack possible. In contrast, if your problem involves learning a type of mutation or some sample-level property, then performance will increase with more variants per sample and exome-level or genome-level data will be preferred. It is for this reason we used (and most models use) exome-level data to predict microsatellite instability.

We could have chosen to apply our model to other genomics data types, but chose somatic mutation data given how common this data is becoming and the lack of deep learning in this area. Although technically variants have been featurized before by interpreting the aligned reads as an image (Poplin et al., 2018), our work represents the first attempt to featurize variants after they have been called and processed into a MAF file. Although featurizing the variants with the raw reads is a consideration, it was not practical for this study given that the BAM files are not always available, and even if they are this featurization would be considerably more computationally intensive.

Here we demonstrated the potential of featurizing somatic variants by achieving similar performance to tools which use manually curated features to predict microsatellite instability. Importantly, our model was only given the sequence context and through the attention mechanism pointed to the important features of these manually developed methods. We then attempted to use our model for the increasingly relevant problem of TMB calibration, with variant featurization improving the regression metrics, and observing further improvements when more variants were used as input.

In the TMB experiments we utilized TCGA MC3 data and AACR GENIE panel footprints to generate estimates of what these panels would be able to measure from the TCGA MC3 samples, an approach that has been previously described

(Endris et al., 2019; Buchhalter et al., 2019; Budczies et al., 2019; Merino et al., 2020). While not entirely novel, this study represents the first attempt to our knowledge in which variant featurization is used to reweight variants. Panel data typically overestimates TMB due to the panels covering commonly mutated genes in cancer, so this could be one feature informing TMB prediction. In fact, some protocols call for removing common mutations when calculating exomic TMB from panel data (Chalmers et al., 2017). While a strong rationale, the question then becomes which variants to count and which ones to ignore, and whether this is a binary vs continuous concept. And does the list change depending on the panel or in the context of specific cancers? Ideally TMB calibration would not require manually filtering the data for each use case with potentially competing filtering protocols. Another potential reason for panel data overestimating TMB is the higher depth of sequencing, which may allow the panels to call variants with lower variant allele frequencies, but this is a bias that's difficult to explore without matched panel/exome data.

When thinking about how variant featurization may be improving the regression metrics it seems likely that the model is trying to identify types of variants which are overrepresented or underrepresented in the panel, potentially in a tumor specific manner. The model is trying to learn to predict the number of nonsynonymous mutations in the exome given a subsampling of the total mutations as represented in a given panel. Arguably, for most tumors this is primarily determined by the background mutation rate of the tumor. Indeed, calculating the background mutation rate has been proposed as a method for estimating TMB from panels (Yao et al., 2020). However, calculating the background mutation rate requires incorporating gene expression and replication timing data, which will likely not be available during model application, and thus would have to be inferred from another data source. Variant featurzation may also be partially learning cancer type, which we saw provides the model a lot of information about the expected TMB value. It is currently unclear whether a model with variant featurization or a model that incorporates cancer type will be more likely to generalize to another data set (if the cancer type is unknown then the only option would be featurization).

It is important to note that when training a deep learning model one must ensure the training data is reasonably representative of the potential test data. One issue with using the TCGA MC3 data is that it only includes exomic sequences, while panels usually include noncoding sequences (generally for the purposes of detecting fusions). Interestingly, we observed that the inclusion of variants detected in panels outside of the strict definition for the whole exome derived metric improved model performance, as has been suggested previously (Buchhalter et al., 2019). Another consideration is that TCGA data is matched tumor-normal data, which thereby allows for direct filtering of germline mutations. In contrast, many panel-based methods do not include a matched normal sample of the patient and instead filter the identified variants against large population-scale exome data sets (gnomad.broadinstitute.org, for example) for common/known germline variants; however, it is known that this approach will allow for a small number of so-called "private variants" to pass through these types of filtering approaches and incorrectly be considered somatic alterations. As a result, it would be inappropriate to use a model trained on somatic mutation data from TCGA for these types of panels. One potential consideration is to reincorporate germline mutations into the TCGA data and then filter based on public germline databases (as mentioned above) prior to subsampling based on the panel genomic footprint.

Furthermore, when applying a model trained on TCGA MC3 MAF data you are working under the assumption that the data processing which led to the MAF file was similar to the processing of the data you are applying the model to. It is for this reason that it is not necessarily the case that our model (or MSIpred for that matter) should be selected over MANTIS for MSI predictions despite the fact that our model may outperform MANTIS on TCGA data given enough concepts and hyperparameter adjustments. MANTIS operates directly on the BAM files, and as a result is less biased by the data processing pipelines in TCGA data.

Tumor type is not the only sample-level information that could be added to the model. Age, sex, grade, smoking status, etc., could all easily be added to the model and likely improve performance. However, it is up to a researcher to determine if they believe the samples in the training set (in this case TCGA data) will generalize to their data. For example, a researcher who has panel data for lung cancer samples would likely either want to only train on the TCGA lung cancer data, or train on all the data with cancer type included. However, this application of the model would be assuming that the lung cancer samples in the TCGA are representative of the researcher's lung cancer samples. Ideally, laboratories would generate a set of whole exome and panel-level somatic mutation results on a set of common samples and then train our models in that

context (potentially also leveraging TCGA data as well) to arrive at the most generalizable and applicable model for their panel and specific aggregate measure they are trying to estimate.

Exomic TMB is associated with response to immunotherapy, which is in part why there is interest in accurately estimating exomic TMB from panel data. However, if you have the ability to extract features from variants which can be used to predict any sample-level metric, why would you use those features to predict exomic TMB? A future consideration is to directly label the samples with respect to response to therapy and then attempt to train a model to make that distinction from panel-level data. While we believe this approach is more powerful, there does not exist an equivalently sized dataset to the TCGA with patients that received immunotherapy and containing whole exome data.

METHODS

Data and Code

All code for processing data, running models, and generating figures is available at GitHub: https://github.com/OmnesRes/ATGC2, and has been archived at Zenodo: http://doi.org/10.5281/zenodo.4542577. Code is written in Python 3 and TensorFlow 2. All intersections were performed with PyRanges (Stovner and Sætrom, 2020). We leveraged NVIDIA V100s with 32GB of RAM for much of the computation, however most of the computations here could be reasonably performed on CPU as well within the same coding framework. All data used is publicly available, albeit the controlled MC3 MAF is restricted access.

Model

We used a combination of TensorFlow 2 and tf.keras for implementing ATGC. In Keras the first dimension of the inputs must match the first dimension of the outputs. To work around this constraint we developed a model around ragged tensors. The model is modular in the sense that instance and sample featurization are separate "encoder" models from the aggregation model. This allows users to write their own classes for featurizing a concept of interest, and allows for mixing and matching various concepts. Ragged tensors are fully supported by TensorFlow, so it is possible to use our model with default loss functions and dataset batching. However, because we like additional control over the sample weighting and want the ability to perform stratified batching, we prefer to use our own loss and metric classes even when the loss and/or metric already exist in TensorFlow, and we created dataset utilities built around generators. Custom losses used that are not available in base TensorFlow include QuantileLoss and CoxPH, where the output of the graph is interpreted as the log risk and the loss is set as the negative partial likelihood, as first proposed by Faraggi and Simon (1995).

Custom activation functions

Functions can be tested at: https://www.desmos.com/calculator/vruk2tomxk

Adaptive square root (ASR)

$$ASR(x,\alpha) = \sqrt{e^{\alpha} + x^2}$$

The above will be used as a core element in the activation functions below.

Adaptive Rectifying Unit (ARU)

$$ARU(x,\alpha) = 0.5 \times (x + ASR(x,\alpha))$$

Note, a bias term could be added and applied to x prior to this activation function (as is the convention) and α can be a trainable parameter that modulates the curvature of this "rectifier". The ARU can approach the shape of the commonly used rectified linear unit (ReLu) as α approaches negative infinity, but ARU is fully differentiable across all x, which does not hold for ReLu. We have tuned default initial conditions for α to match the commonly used softplus function, particularly near $x = 0$, as initial conditions for this adaptive rectifier.

Adaptive Sigmoid Unit (ASU)

$$ASU(x, \kappa_{lower}, \kappa_{upper}, \alpha_{lower}, \alpha_{upper}) = \kappa_{lower} + \frac{(\kappa_{upper} - \kappa_{lower}) \times (x + ASR(x, \alpha_{lower}))}{ASR(x, \alpha_{lower}) + ASR(x, \alpha_{upper})}$$

Again, a bias term could be added and applied to x prior to this activation function (as is the convention). We generally keep the lower and upper asymptotes fixed at 0, 1 or –1, 1 depending on the application. The lower and upper α parameters together control the curvature of the sigmoid. We have tuned default values to match the logistic function, particularly near $x = 0$, as initial conditions for this adaptive sigmoid function.

The above were motivated by the previously described inverse square root unit (ISRU). It is important to note that all input x is at most squared in this formulation and there is no application of input x as an exponent (e^x), which greatly helps to avoid numerical overflow that can occur in the context of aggregation over samples in a multiple instance learning (MIL) framework.

TCGA MC3 MAF Processing

Any variant with either FILTER = "PASS" or "wga" were included. We considered nonsynonymous mutations to be any mutation with the labels Missense Mutation, Nonsense Mutation, Frame Shift Del, Frame Shift Ins, In Frame Del, In Frame Ins, Nonstop Mutation. To ensure that a TCGA sample's capture kit covered the exome we downloaded the metadata of the read groups with the GDC API. Any sample which only had a combination of Gapfiller_7m & NimbleGen_Sequence_Capture_2.1M_Human_Exome_Array kits or tcga_6k_genes.targetIntervals & cancer_2000gene_shift170.targetIntervals BED files, were considered incomplete captures. We would advise anyone performing analyses assuming the TCGA whole exome data covers the whole exome to exclude these samples. Of note, we could not find information about the capture kit for every sample, and could not find the coordinates for most kits, so while we can exclude samples we don't believe used an appropriate kit, we cannot guarantee that the rest of the samples did use an appropriate kit. The most common kit was the Broad Institute's Agilent kit, so we used its coordinates for the assumed size of the exome.

For labeling regions CDS or exon we used Homo_sapiens.GRCh37.87.gff3, and for panel coordinates we used the GENIE 7.0 definitions.

TCGA MC3 Controlled MAF Processing

Processing was similar to the public MAF, except we allowed FILTER to equal "PASS", "NonExonic", "bitgt", "wga", or "broad_PoN_v2" to allow as many repeat variants as possible.

ICGC MAF Processing

Processing was similar to the TCGA MC3 MAF, however the MAF contained multiple samples per donor. To restrict the data to one sample per donor we dropped the duplicate rows with PANDAS.

MSI Analyses

PCR labels for TCGA samples were obtained from a combination of cBioPortal, TCGAbiolinks, and individual publications. If the label for a 5 marker call and 7 marker call were in disagreement, the 7 marker call was given preference. Both MSS and MSI-L were considered to be MSI low.

The source code for MSIpred was altered to allow it to run on Python 3, and the model was changed to output probabilities. All repeats from simpleRepeat.txt were used, but otherwise the model was run as recommended.

ACKNOWLEDGMENTS

The results here are in whole or part based upon data generated by the TCGA Research Network.

COMPETING INTERESTS

CC is an employee of Genentech, a member of the Roche Group, and holds Roche stock.

FUNDING

This research was supported by the American Association for Cancer Research (AACR) Project GENIE Consortium, Genentech, Inc., Mark Foundation for Cancer Research (19-035-ASP), and the philanthropy of Susan Wojcicki and Dennis Troper in support of Computational Pathology at Johns Hopkins.

CHANGES FROM PREVIOUS VERSION

In the previous version of this manuscript our model did not use ragged tensors, and instead placed a dummy dimension in front of all inputs and outputs. All of the results in the paper have been updated with the new model. In addition, this version of the manuscript contains new simulation results and includes TMB regression with sample information. Previously the R-squared in the tables was calculated as pred versus true, when it should have been true versus pred. That has been fixed in this version.

REFERENCES

AACR Project GENIE Consortium and others (2017). AACR Project GENIE: powering precision medicine through an international consortium. *Cancer Discovery*, 7(8):818–831.

Agarwal, V., Bell, G. W., Nam, J.-W., and Bartel, D. P. (2015). Predicting effective microrna target sites in mammalian mrnas. *elife*, 4:e05005.

Alexandrov, L. B., Kim, J., Haradhvala, N. J., Huang, M. N., Ng, A. W., Wu, Y., Boot, A., Covington, K. R., Gordenin, D. A., Bergstrom, E. N., et al. (2019). The repertoire of mutational signatures in human cancer. *BioRxiv*, page 322859.

Alexandrov, L. B., Kim, J., Haradhvala, N. J., Huang, M. N., Ng, A. W. T., Wu, Y., Boot, A., Covington, K. R., Gordenin, D. A., Bergstrom, E. N., et al. (2020). The repertoire of mutational signatures in human cancer. *Nature*, 578(7793):94–101.

Alexandrov, L. B., Nik-Zainal, S., Wedge, D. C., Aparicio, S. A., Behjati, S., Biankin, A. V., Bignell, G. R., Bolli, N., Borg, A., Børresen-Dale, A.-L., et al. (2013). Signatures of mutational processes in human cancer. *Nature*, 500(7463):415–421.

Altman, N. S. and Krzywinski, M. (2018). The curse (s) of dimensionality this-month. *Nature Methods*, 15(6):399–400.

Amores, J. (2013). Multiple instance classification: Review, taxonomy and comparative study. *Artificial intelligence*, 201:81–105.

Buchhalter, I., Rempel, E., Endris, V., Allgäuer, M., Neumann, O., Volckmar, A.-L., Kirchner, M., Leichsenring, J., Lier, A., von Winterfeld, M., et al. (2019). Size matters: Dissecting key parameters for panel-based tumor mutational burden analysis. *International Journal of Cancer*, 144(4):848–858.

Budczies, J., Allgäuer, M., Litchfield, K., Rempel, E., Christopoulos, P., Kazdal, D., Endris, V., Thomas, M., Fröhling, S., Peters, S., et al. (2019). Optimizing panel-based tumor mutational burden (TMB) measurement. *Annals of Oncology*, 30(9):1496–1506.

Carbonneau, M.-A., Cheplygina, V., Granger, E., and Gagnon, G. (2018). Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 77:329–353.

Chalmers, Z. R., Connelly, C. F., Fabrizio, D., Gay, L., Ali, S. M., Ennis, R., Schrock, A., Campbell, B., Shlien, A., Chmielecki, J., et al. (2017). Analysis of 100,000 human cancer genomes reveals the landscape of tumor mutational burden. *Genome Medicine*, 9(1):34.

Chang, M. T., Bhattarai, T. S., Schram, A. M., Bielski, C. M., Donoghue, M. T., Jonsson, P., Chakravarty, D., Phillips, S., Kandoth, C., Penson, A., et al. (2018). Accelerating discovery of functional mutant alleles in cancer. *Cancer Discovery*, 8(2):174–183.

Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71.

Eksi, R., Li, H.-D., Menon, R., Wen, Y., Omenn, G. S., Kretzler, M., and Guan, Y. (2013). Systematically differentiating functions for alternatively spliced isoforms through integrating rna-seq data. *PLoS Comput Biol*, 9(11):e1003314.

Ellrott, K., Bailey, M. H., Saksena, G., Covington, K. R., Kandoth, C., Stewart, C., Hess, J., Ma, S., Chiotti, K. E., McLellan, M., et al. (2018). Scalable open science approach for mutation calling of tumor exomes using multiple genomic pipelines. *Cell Systems*, 6(3):271–281.

Endris, V., Buchhalter, I., Allgäuer, M., Rempel, E., Lier, A., Volckmar, A.-L., Kirchner, M., von Winterfeld, M., Leichsenring, J., Neumann, O., et al. (2019). Measurement of tumor mutational burden (TMB) in routine molecular diagnostics: in silico and real-life analysis of three larger gene panels. *International Journal of Cancer*, 144(9):2303–2312.

Fancello, L., Gandini, S., Pelicci, P. G., and Mazzarella, L. (2019). Tumor mutational burden quantification from targeted gene panels: major advancements and challenges. *Journal for Immunotherapy of Cancer*, 7(1):183.

Faraggi, D. and Simon, R. (1995). A neural network model for survival data. *Statistics in medicine*, 14(1):73–82.

FDA (2020). FDA approves pembrolizumab for adults and children with tmb-h solid tumors. https://www.fda.gov/drugs/drug-approvals-and-databases/fda-approves-pembrolizumab-adults-and-children-tmb-h-solid-tumors.

Goodman, A. M., Kato, S., Bazhenova, L., Patel, S. P., Frampton, G. M., Miller, V., Stephens, P. J., Daniels, G. A., and Kurzrock, R. (2017). Tumor mutational burden as an independent predictor of response to immunotherapy in diverse cancers. *Molecular Cancer Therapeutics*, 16(11):2598–2608.

Goodman, B. and Flaxman, S. (2017). European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3):50–57.

Hoadley, K. A., Yau, C., Hinoue, T., Wolf, D. M., Lazar, A. J., Drill, E., Shen, R., Taylor, A. M., Cherniack, A. D., Thorsson, V., et al. (2018). Cell-of-origin patterns dominate the molecular classification of 10,000 tumors from 33 types of cancer. *Cell*, 173(2):291–304.

Ilse, M., Tomczak, J. M., and Welling, M. (2018). Attention-based deep multiple instance learning. *arXiv preprint arXiv:1802.04712*.

Jawa, V., Cousens, L. P., Awwad, M., Wakshull, E., Kropshofer, H., and De Groot, A. S. (2013). T-cell dependent immunogenicity of protein therapeutics: preclinical assessment and mitigation. *Clinical immunology*, 149(3):534–555.

Jiao, W., Atwal, G., Polak, P., Karlic, R., Cuppen, E., Al-Shahrour, F., Bailey, P. J., Biankin, A. V., Boutros, P. C., Campbell, P. J., Chang, D. K., Cooke, S. L., Deshpande, V., Faltas, B. M., Faquin, W. C., Garraway, L., Getz, G., Grimmond, S. M., Haider, S., Hoadley, K. A., Kaiser, V. B., Kato, M., Kübler, K., Lazar, A. J., Li, C. H., Louis, D. N., Margolin, A., Martin, S., Nahal-Bose, H. K., Nielsen, G. P., Nik-Zainal, S., Omberg, L., P'ng, C., Perry, M. D., Rheinbay, E., Rubin, M. A., Semple, C. A., Sgroi, D. C., Shibata, T., Siebert, R., Smith, J., Stein, L. D., Stobbe, M. D., Sun, R. X., Thai, K., Wright, D. W., Wu, C.-L., Yuan, K., Zhang, J., Danyi, A., de Ridder, J., van Herpen, C., Lolkema, M. P., Steeghs, N., Morris, Q., Group, P. T. S., Working, C. T., and PCAWG Consortium (2020). A deep learning system accurately classifies primary and metastatic cancers using passenger mutation patterns. *Nature Communications*, 11(1):728.

Kautto, E. A., Bonneville, R., Miya, J., Yu, L., Krook, M. A., Reeser, J. W., and Roychowdhury, S. (2017). Performance evaluation for rapid detection of pan-cancer microsatellite instability with MANTIS. *Oncotarget*, 8(5):7452.

Kim, S., Lee, H., Kim, K., and Kang, J. (2018). Mut2Vec: distributed representation of cancerous mutations. *BMC Medical Genomics*, 11(2):33.

Lawrence, M. S., Stojanov, P., Polak, P., Kryukov, G. V., Cibulskis, K., Sivachenko, A., Carter, S. L., Stewart, C., Mermel, C. H., Roberts, S. A., et al. (2013). Mutational heterogeneity in cancer and the search for new cancer-associated genes. *Nature*, 499(7457):214–218.

Marcus, L., Lemery, S. J., Keegan, P., and Pazdur, R. (2019). FDA approval summary: pembrolizumab for the treatment of microsatellite instability-high solid tumors. *Clinical Cancer Research*, 25(13):3753–3758.

Merino, D. M., McShane, L. M., Fabrizio, D., Funari, V., Chen, S.-J., White, J. R., Wenz, P., Baden, J., Barrett, J. C., Chaudhary, R., Chen, L., Chen, W. S., Cheng, J.-H., Cyanam, D., Dickey, J. S., Gupta, V., Hellmann, M., Helman, E., Li, Y., Maas, J., Papin, A., Patidar, R., Quinn, K. J., Rizvi, N., Tae, H., Ward, C., Xie, M., Zehir, A., Zhao, C., Dietel, M., Stenzinger, A., Stewart, M., Allen, J., and TMB Harmonization Consortium (2020). Establishing guidelines to harmonize tumor mutational burden (TMB): in silico assessment of variation in TMB quantification across diagnostic platforms: phase I of the Friends of Cancer Research TMB Harmonization Project. *Journal for Immunotherapy of Cancer*, 8(1).

Palazzo, M., Beauseroy, P., and Yankilevich, P. (2019). A pan-cancer somatic mutation embedding using autoencoders. *BMC Bioinformatics*, 20(1):655.

Peng, J., Zou, D., Gong, W., Kang, S., and Han, L. (2020). Deep neural network classification based on somatic mutations potentially predicts clinical benefit of immune checkpoint blockade in lung adenocarcinoma. *OncoImmunology*,

9(1):1734156.

Poplin, R., Chang, P.-C., Alexander, D., Schwartz, S., Colthurst, T., Ku, A., Newburger, D., Dijamco, J., Nguyen, N., Afshar, P. T., et al. (2018). A universal snp and small-indel variant caller using deep neural networks. *Nature biotechnology*, 36(10):983–987.

Romero, A., Carrier, P. L., Erraqabi, A., Sylvain, T., Auvolat, A., Dejoie, E., Legault, M.-A., Dubé, M.-P., Hussin, J. G., and Bengio, Y. (2016). Diet networks: thin parameters for fat genomics. *arXiv preprint arXiv:1611.09340*.

Samstein, R. M., Lee, C.-H., Shoushtari, A. N., Hellmann, M. D., Shen, R., Janjigian, Y. Y., Barron, D. A., Zehir, A., Jordan, E. J., Omuro, A., et al. (2019). Tumor mutational load predicts survival after immunotherapy across multiple cancer types. *Nature Genetics*, 51(2):202–206.

Stovner, E. B. and Sætrom, P. (2020). PyRanges: efficient comparison of genomic intervals in Python. *Bioinformatics*, 36(3):918–919.

Wang, C. and Liang, C. (2018). MSIpred: a python package for tumor microsatellite instability classification from tumor mutation annotation data using a support vector machine. *Scientific Reports*, 8(1):1–10.

Yan, Y., Wang, X., Guo, X., Fang, J., Liu, W., and Huang, J. (2018). Deep multi-instance learning with dynamic pooling. In *Asian Conference on Machine Learning*, pages 662–677.

Yao, L., Fu, Y., Mohiyuddin, M., and Lam, H. Y. (2020). ectmb: a robust method to estimate and classify tumor mutational burden. *Scientific Reports*, 10(1):1–10.