Aggregation Tool for Genomic Concepts (ATGC): A deep learning framework for sparse genomic measures

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 and features. 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, sequence context, and read counts) on a range of artificial tasks (classification, regression, Cox regression). In addition, we confirm the model can achieve high performance on real-world problems, accurately classifying samples according to whether they contain a specific variant (hotspot or tumor suppressor), groups of variants (tumor clonality), or a type of variant (microsatellite instability). Our results suggest this framework will lead to improvements on sample-level tasks that require aggregation of a set of genomic measures.

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 challenging to use genomic scale data in traditional machine learning tasks due to the curse of dimensionality. in that 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 base substitutions (SBSs) to large insertions or deletions (InDels), there are a seamingly 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, or by defining mutational signatures based on the type of SBS and neighboring five prime and three prime nucleotides (Chalmers et al., 2017; Alexandrov et al., 2020a).

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 and fuse together various types of information 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.

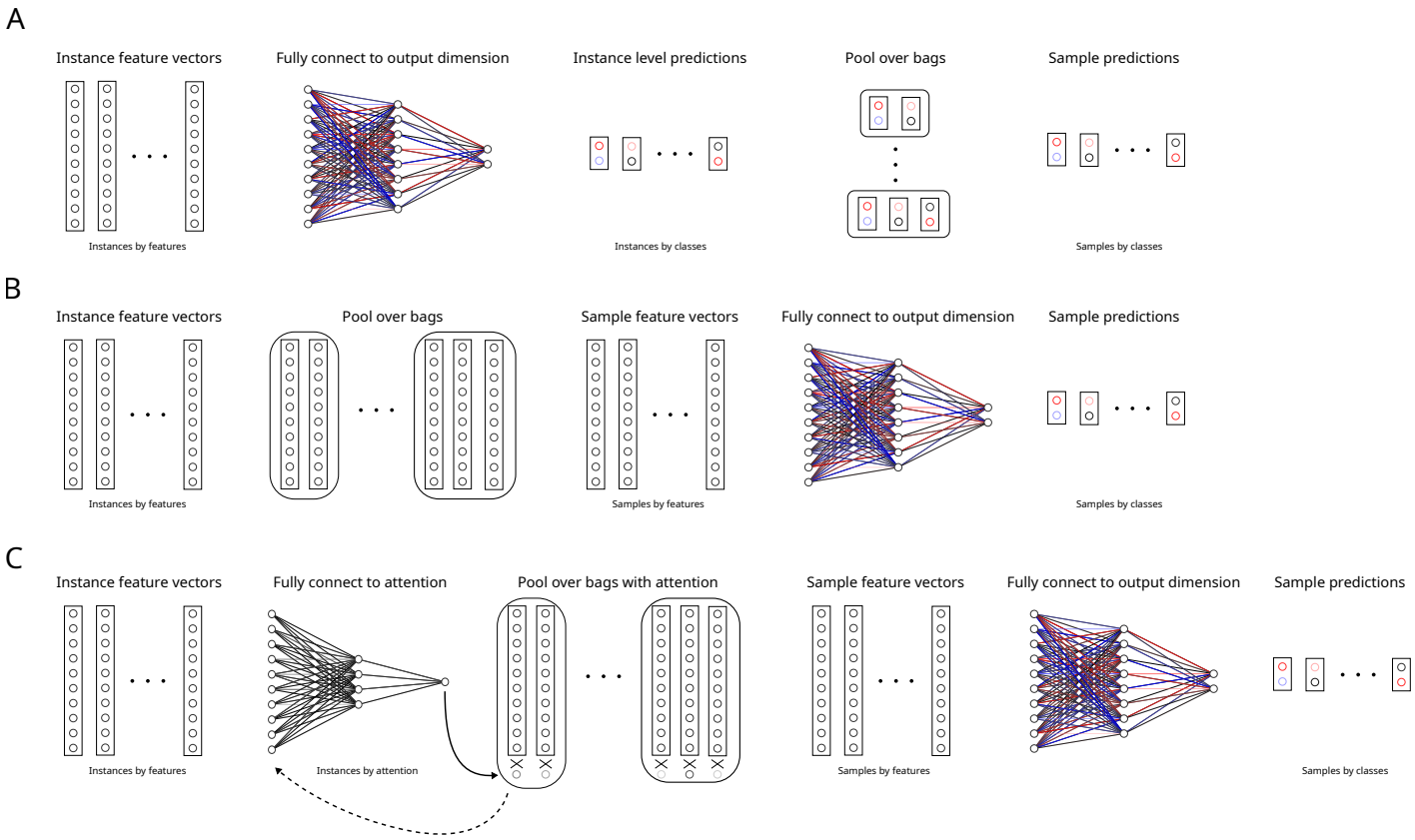

Figure 1. Multiple instance learning approaches. (A) *MIL based on an instance classifier.* The instance feature vectors are used to determine the class of the instances, and then aggregated. (B) *MIL based on a sample classifier.* The instance features are first aggregated into a sample feature vector, 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 sample that is represented by a set of mutations. 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.

FEATURIZATION OF SOMATIC MUTATION INSTANCES

The instance features of each bag can be manually generated via a variety of ; for example onehot of gene, variant class (substitution versus insertion / deletion), 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 (so-called end to end training). 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.

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

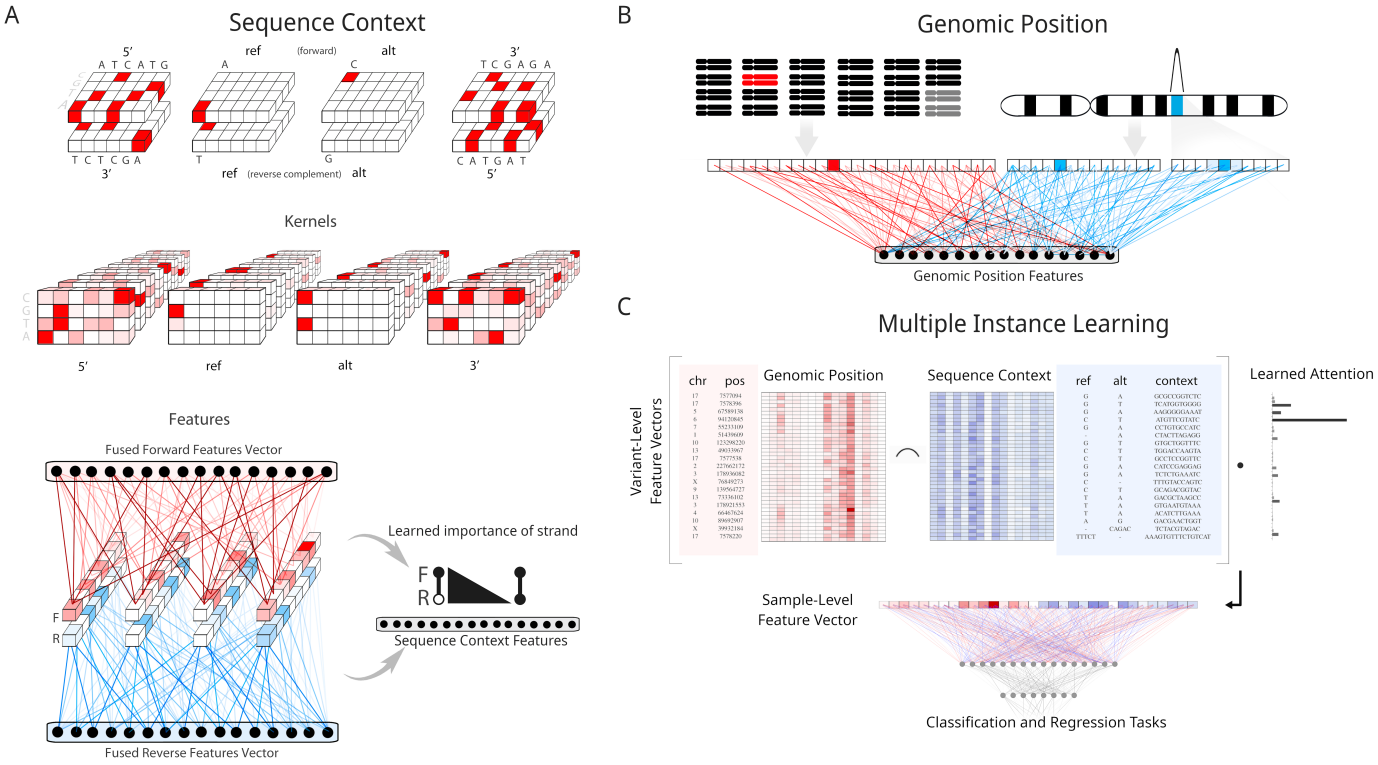

Figure 2. Featurization of mutation data and subsequent application of multiple instance learning. *Sequence and genomic position concepts.* (A) 5', 3', ref, and alt sequences are featurized independently and then fused together in a strand informed manner. Nucleotides are onehot encoded with padded 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.

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 SBSs 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 SBS with the pyrimidine as the reference sequence there are only six types of SBSs (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, 2020a,b). We labeled the variants in the MAF with these 96 contexts if the variant was a SBS, 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.

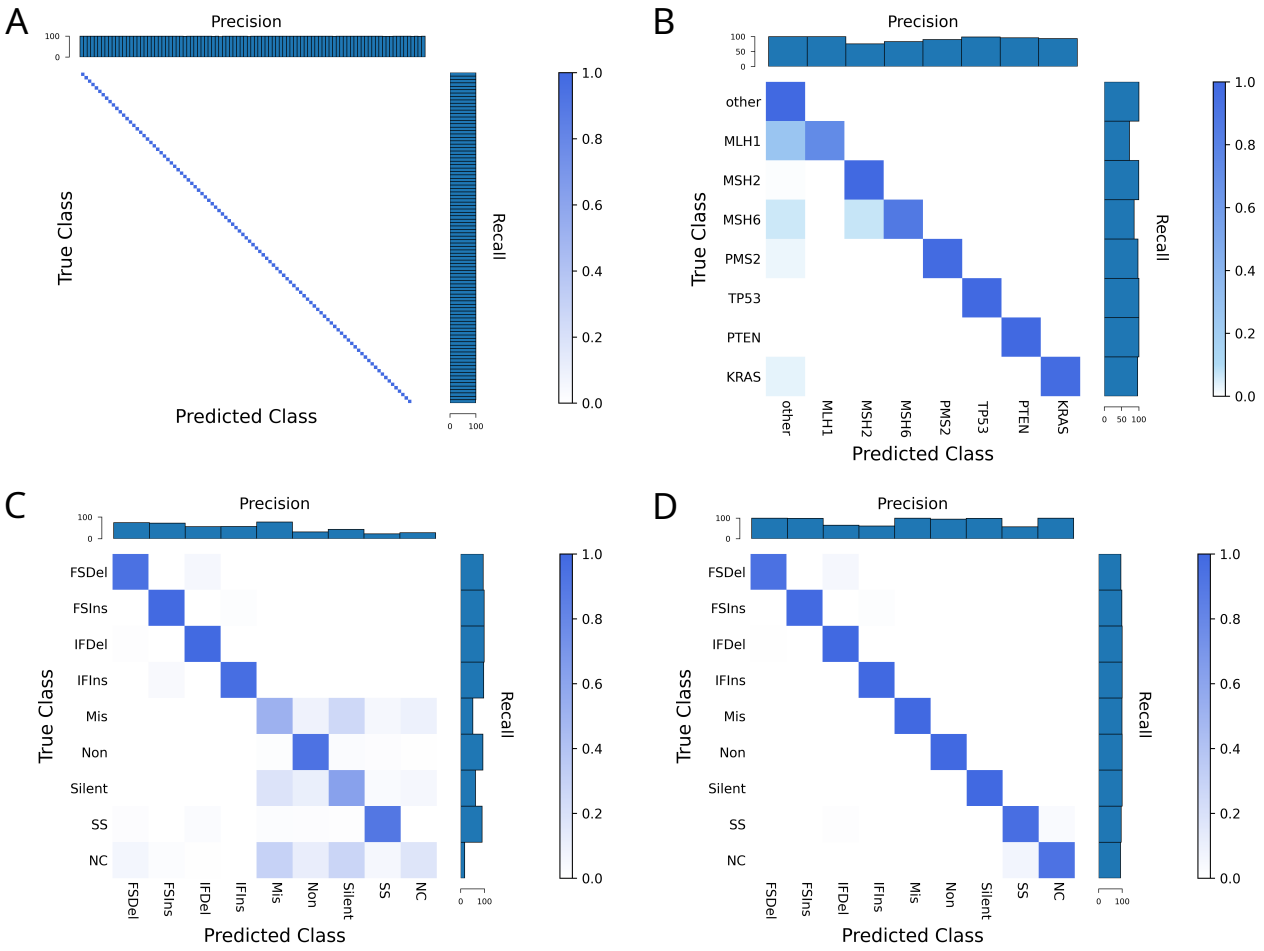

Figure 3. Characterization of performance at the mutation (instance) level. (A) 97 way classification (96 nucleotide contexts and an other/out group category) in which the model is able to accurately learn the conventional trinucleotide mutational contexts. (B) Expanding beyond simple trinucleotide context, 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).

Performance characteristics of different aggregation methods

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.

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.

Classification experiments

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.

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.

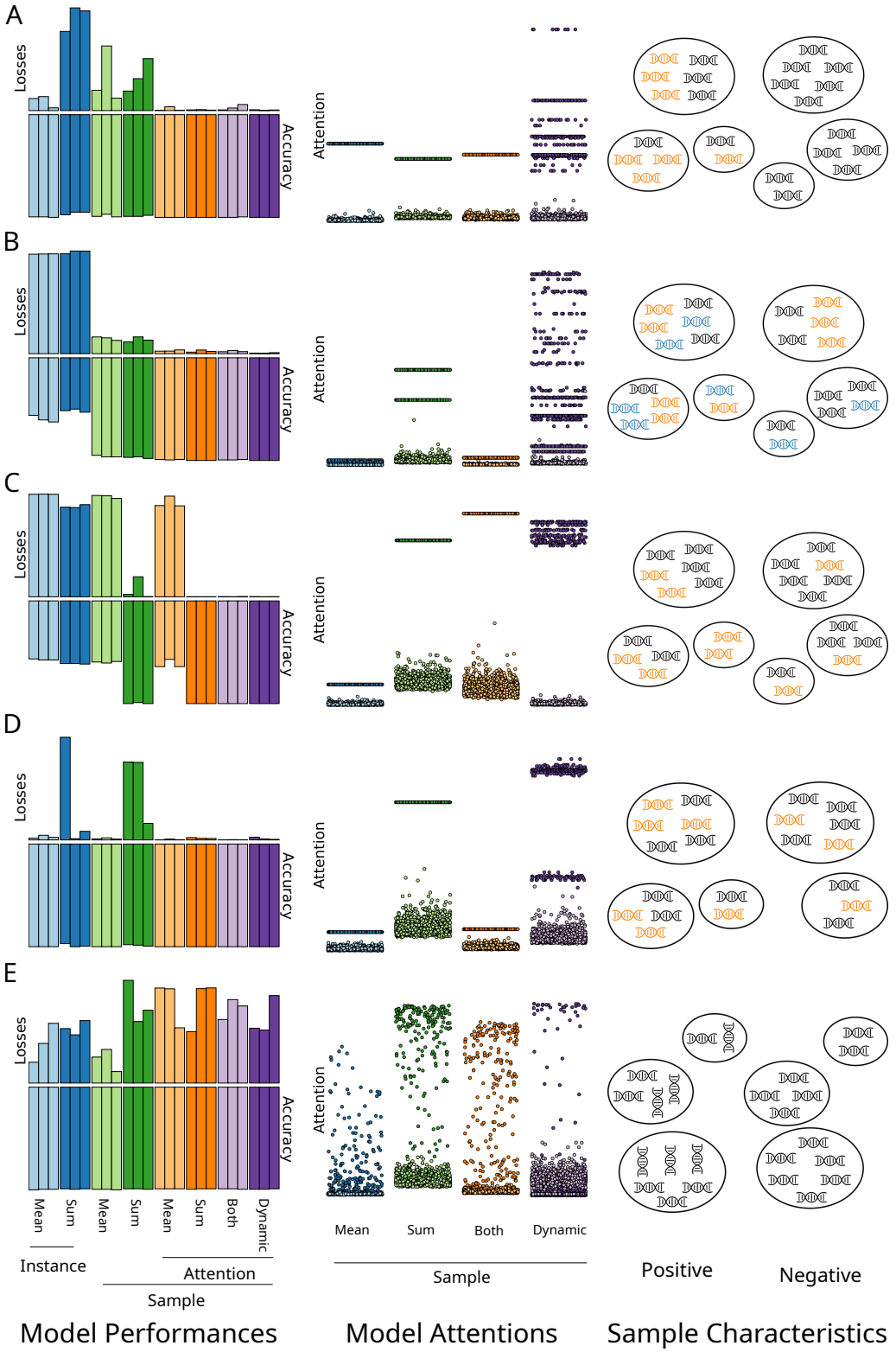

Figure 4. Sample-level classification with simulated data. (A) Positive bags were labeled with a random fraction of a key instance, with negative bags having random variants. (B) A random fraction of each positive bag was filled with two key instances in equal proportions, and negative bags were composed 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 SBSs. 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.

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).

Regression experiments

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).

Incorporating sample information

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.

Real-world applications

Our controlled experiments allowed us to understand the performance profile of different aggregation methods, and now we'd like to understand the ability of our model to solve commonly encountered problems involving somatic mutations. For example, a researcher may be interested in an effect where a single mutation is responsible for the outcome, such as drug resistance. This would be equivalent to trying to find a needle in a haystack, and corresponds to the classical formulation of

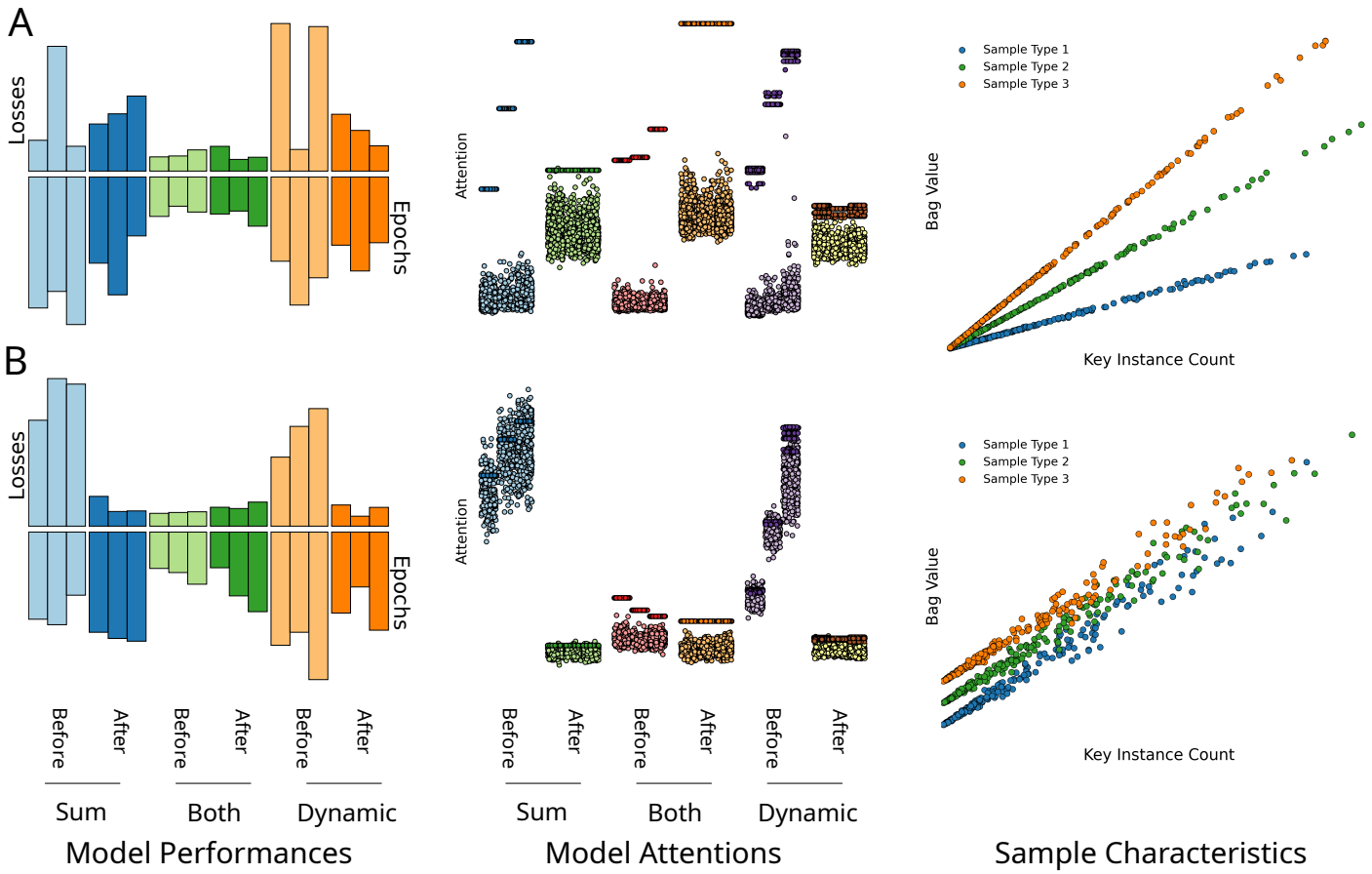

Figure 5. Incorporating sample information with simulated data. Both experiments are regression tests where the bag value is determined by the key instance count in a linear manner, but in (A) the sample type affects how much each key instance contributes to the bag value, while in (B) it does not, but 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.

MIL where a single key instance determines the bag label. Or a researcher may have a task driven by the number of clones a tumor is composed of, which is a sample-level feature and follows the current rethinking of MIL. Or a task may depend on the mutational processes occurring, which requires identifying the salient types of instances as well as quantifying their relative or absolute amounts at the sample-level. Regardless of what the model needs to learn, we would hope that with an appropriate encoder and enough training data our framework will solve the task.

Tumor suppressors and hotspots

A tumor suppressor is characterized by mutations throughout the gene, as a result there is not a specific sequence for the model to find and we will need to rely on our position concept to classify these samples. Hotspots are characterized by changes in a specific amino acid (which corresponds to a specific nucleotide sequence), so 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. For the aggregation we used a weighted sum.

We chose PTEN as our tumor suppressor, and labeled TCGA MC3 samples based on whether or not they had a mutation in the gene. The model's recall (99.76%) was higher than its precision (92.46%), 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).

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.

Tumor purity and clonality

The number of clones in a tumor can be determined by clustering the variant allele frequencies (VAF) of a tumor's mutations (Dentro et al., 2017). While numerous tools have been developed over the years to determine the number of clones in a tumor sample, a recent tool that also takes into account mutational signatures claims to achieve state of the art performance (Abécassis et al., 2021). The authors also developed a simulator based on the same model as their predictor. Using this tool we generated 10,000 samples with 1-4 clones and variable numbers of mutations and purity (see methods).

In addition to read counts, CloneSig requires tumor purity and sequence context as inputs. We felt that tumor purity was something our model could easily learn, and as a result made that an output of the model and as inputs simply provided ref and alt reads and allowed the model to learn its own concept of VAF. Figure 6A confirms that the model can learn tumor purity from read counts. As a second output we had the model predict the number of clones in the sample (Figure 6B). In comparing the predictions to CloneSig on the same samples (Figure 6C), the models performed similarly when there were 1-3 clones, but our model was significantly better at 4 clones.

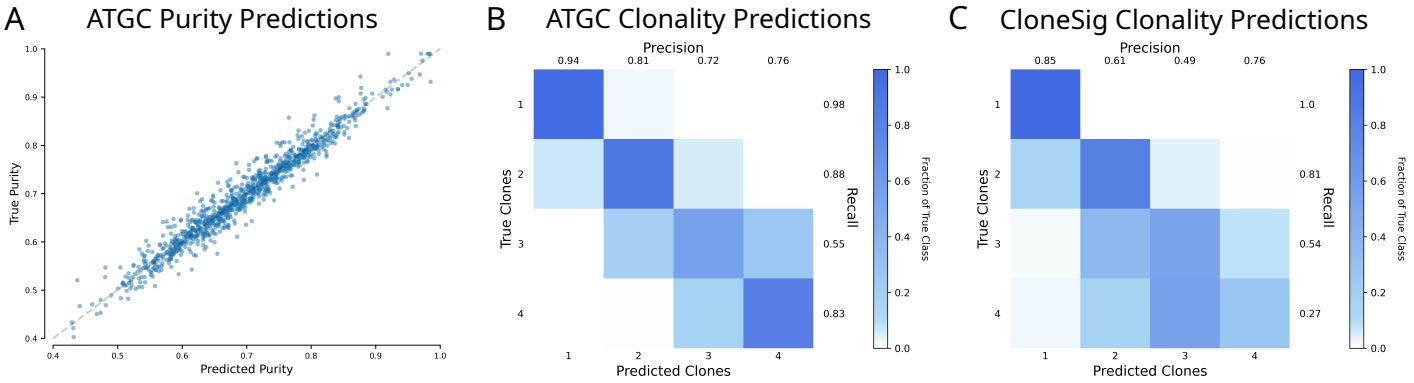

Figure 6. ATGC concurrently predicts tumor purity and clonality (A) Tumor purity predictions of ATGC. (B) ATGC clonality predictions (C) CloneSig clonality predictions. CloneSig predicted 5 clones a few times and for the purposes of this figure were ignored. Confusion matrices are row normalized.

Microsatellite Instability

Characterized by deficiencies in the mismatch repair proteins (MLH1, MSH2, MSH6, PMS2), microsatellite unstable tumors accumulate InDels at microsatellites due to polymerase splippage. 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 7A 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 7B 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.

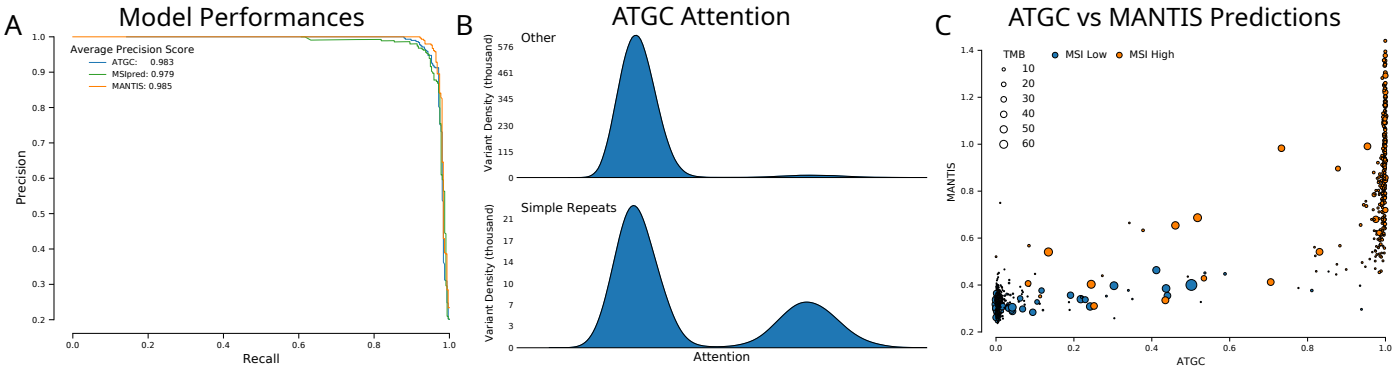

Figure 7. Application of ATGC MIL framework accurately predicts MSI status by learning which are the salient variants. (A) Average precision recall curves over 9 stratified k-fold splits for MSIpred and ATGC, precision recall curve for MANTIS for samples which had a MANTIS score. (B) Probability density 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 7C. 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.

DISCUSSION

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. The problem is similar to finding to finding a needle in a haystack so

will be easier with a smaller haystack. In contrast, if a 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, and performance would be expected to be even greater with genome-level data.

As with any deep learning model, one must ensure the training data is reasonably representative of the potential test data. When applying a model trained on TCGA MC3 MAF data it is implicitly assumed that the data processing which led to the MAF file was similar to the processing of the data the model will be applied to. For clonality assessment where we only know the true clonal structure in simulated data, any model trained on the simulated data will only be as good as the simulation process. As a result, it is not necessarily the case that our model should be selected over non-machine learning approaches. Given a large enough dataset our model (or MSIpred for that matter) may outperform MANTIS on a holdout of the same data for MSI predictions, but because MANTIS operates directly on the BAM files and therefore is not impacted by any of the biases introduced by the data processing pipelines that lead to a MAF, MANTIS would still likely perform better on unrelated datasets.

An encoder that trains along with the graph is not guaranteed to outperform or even achieve equivalent performance to manual featurization. For example, if you know going into a problem that it is mutations in a specific gene which drive the process the best featurization will be a onehot of the genes. However, if it is only mutations in a specific exon (or intron) that drive a process, then an encoder which can featurize position would be expected to outperform the onehot model given enough training examples to learn the necessary positional features. Encoders can also potentially be pretrained or concurrently trained with an auxiliary task if instance-level labels are available.

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 suitability of applying MIL-based deep learning to these data. Although 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 in a multimodal manner spanning mutation, sequence context, and genomic location. Featurizing the variants with the raw reads (as opposed to the called variants) was a consideration but not practical for this study given that the BAM files are not always available, and even then this featurization would be considerably more computationally intensive. And, it is uncertain as to what additional relevant features would be extracted since outside of the called variants much of the sequences should be shared across samples.

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 because it essentially transposes the problem—the large amount of instance data is a benefit instead of a hindrance when extracting relevant 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(s).

METHODS

Availability of Data and Materials

All data used in this publication is publicly available. The MC3 MAFs are from Ellrott et al. (2018), the MSI PCR labels are available from cBioPortal, TCGAbiolinks, or individual publications (Cancer Genome Atlas Network et al.,

2012; Levine, 2013; Berger et al., 2018; Liu et al., 2018). The UCSC simpleRepeat.txt was downloaded from http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/simpleRepeat.txt.gz. The GFF used was downloaded from ftp://ftp.ensembl.org/pub/grch37/current/gff3/homo_sapiens/Homo_sapiens.GRCh37.87.gff3.gz.

All code for processing data, running models, and generating figures is available at GitHub: https://github.com/OmnesRes/ATGC/tree/method_paper, and has been archived at Zenodo: https://doi.org/10.5281/zenodo.5649254. 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.

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. The framework allows for any number of instance or sample inputs, and any number of outputs. 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.

MSI Analyses

The PCR labels were merged and when 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.

Clonality Simulations

10,000 samples were made with CloneSigSim with approximately equal numbers of 1 through 4 clones, and the number of mutations ranging from 100 to 5000. Purity was allowed to vary according to the default parameters, the overdispersion parameter was fixed at 100, copy number was fixed, the minimal clonal fraction was set at .1, the minimal CCF difference was set to .1, and the active signatures were allowed to be chosen according to the default parameters.

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.

REFERENCES

Abécassis, J., Reyal, F., and Vert, J. (2021). Clonesig can jointly infer intra-tumor heterogeneity and mutational signature activity in bulk tumor sequencing data. *Nature Communications*, 12(1):1–16.

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. T., Wu, Y., Boot, A., Covington, K. R., Gordenin, D. A., Bergstrom, E. N., et al. (2020a). The repertoire of mutational signatures in human cancer. *Nature*, 578(7793):94–101.

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. (2020b). 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.

Berger, A. C., Korkut, A., Kanchi, R. S., Hegde, A. M., Lenoir, W., Liu, W., Liu, Y., Fan, H., Shen, H., Ravikumar, V., et al. (2018). A comprehensive pan-cancer molecular study of gynecologic and breast cancers. *Cancer cell*, 33(4):690–705.

Cancer Genome Atlas Network et al. (2012). Comprehensive molecular characterization of human colon and rectal cancer. *Nature*, 487(7407):330.

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.

Dentro, S. C., Wedge, D. C., and Van Loo, P. (2017). Principles of reconstructing the subclonal architecture of cancers. *Cold Spring Harbor Perspectives in Medicine*, 7(8).

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.

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

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.

Levine, D. A. (2013). Integrated genomic characterization of endometrial carcinoma. *Nature*, 497(7447):67–73.

Liu, Y., Sethi, N. S., Hinoue, T., Schneider, B. G., Cherniack, A. D., Sanchez-Vega, F., Seoane, J. A., Farshidfar, F., Bowlby, R., Islam, M., et al. (2018). Comparative molecular analysis of gastrointestinal adenocarcinomas. *Cancer cell*, 33(4):721–735.

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*.

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.