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

Jordan Anaya[1], John-William Sidhom[2,3,4], Craig A. Cummings[5], Faisal Mahmood[6, 7, 8, 9, 10], Alexander S. Baras[1,2,4], and the AACR Project GENIE Consortium[11]

[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]**Department of Pathology, Brigham and Women's Hospital, Harvard Medical School, Boston, MA, USA**
[7]**Department of Pathology, Massachusetts General Hospital, Harvard Medical School, Boston, MA, USA**
[8]**Cancer Program, Broad Institute of Harvard and MIT, Cambridge, MA, USA**
[9]**Cancer Data Science Program, Dana-Farber Cancer Institute, Boston, MA, USA**
[10]**Harvard Data Science Initiative, Harvard University, Cambridge, MA, USA**
[11]**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. We apply our tool to somatic variants, successfully solving artificial tasks traditional machine learning approaches fail, and achieve best-in-class performance when classifying tumors by cancer type or microsatellite status, while simultaneously providing unprecedented explainability. Our results suggest this framework will lead to novel biological insights and state-of-the-art performance on tasks that include aggregation of a set of genomic measures.

INTRODUCTION

While deep learning has made significant progress in a range of biological tasks[1], for genomics data this progress has been limited to predicting a feature of a genomic measure, such as whether a DNA sequence is a promoter[2]. Making predictions at a higher level, such as at the level of a collection of genomic measures, is complicated by the curse of dimensionality—the high dimensional space makes the data sparse and the models prone to overfitting[3]. Current approaches to this problem include manually reducing the dimensionality through feature selection, dimension reduction techniques such as singular value decomposition, negative matrix factorization, and various types of autoencoders, or the use of sparse networks which attempt to reduce the weights of the model[4]. However, reducing the dimensions of the data or capacity of the model may produce suboptimal results.

With 3 billion base pairs and alterations which can range from single base substitutions (SBSs) to large insertions or deletions (InDels), somatic mutations may be the most complex genomic measure. Somatic mutation data differs from

the more familiar germline GWAS mutation data in that tumors do not 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[5], with somatic mutation data this would result in throwing away most of the data leaving only a few hotspots[6]. To a large extent this has restricted the application of deep learning approaches to the resolution of the gene rather than specific mutations[7–9], 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 5' and 3' nucleotides[10,11].

While focusing on which genes are mutated, how many mutations there are, or what types of mutations are present are valid approaches, ideally a model would learn directly from the data what the salient features are. 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.

Our approach to applying deep learning to individual genomic measures and making predictions at the level of the sample is inspired by the recent application of deep learning to pathology images. While the whole slide images are too large to be featurized, the images can be broken up into subsections (tiles), and these individual tiles can then be featurized. Each individual tile may contain enough information to classify a slide, or only a subset of tiles may contain useful information. Similarly, with genomics data it isn't possible to encode the entire genome, and a potential solution is to break the genome into segments (tiles). Whereas with imaging it's unclear how to tile the data, the locations of the genomic alterations provide clear landmarks to center genomic tiles.

This problem, where each sample is composed of instances (tiles/mutations) and only a sample label is available, is the multiple instance problem[12–14]. Labels in the field of cancer biology may include presence or absence of cancer, type of cancer, response to therapy, or overall survival, and the instances may be somatic mutations, circulating DNA fragments, neopeptides, RNA/protein modifications, copy number alterations, or methylation sites. MIL has revolutionized the field of computational pathology, allowing researchers to identify cancer subtypes, tissues of origin, or predict survival[15–17]. To help usher in a similar revolution for sparse genomics-scale data we developed an easily extensible tool and demonstrated its application to somatic mutations.

RESULTS

Aggregation Tool for Genomic Concepts

In MIL each sample (tumor) is composed of a different number of instances (mutations). Deep learning libraries expect the first dimension of the inputs to match the first dimension of the outputs, but the first dimension of the inputs is variable and the first dimension of the outputs is the number of samples. To solve this problem we built a tool in TensorFlow 2 which handles the variable first dimension under the hood with ragged tensors. When providing data to the model users must follow our process of generating a ragged NumPy array of the data, but otherwise users will not be aware that the model uses ragged tensors and training can proceed as with any other model. The model is comprised of an aggregation model that makes the final predictions and takes encoder models as inputs. These input models are expected to encode the instance data, and are written with standard TensorFlow/Keras layers despite the ragged data type. They are expected to generate a single feature vector per instance, and these feature vectors are used by the main aggregation model to calculate a weight vector (attention), aggregate the instance features per sample, and generate predictions (Figure 1). There is no limit to the number of encoder models, and a user simply lists them in the same order as they list their input data.

Current machine learning with mutation data is limited to manual featurization (for example onehot of genes or variant classes). Our model differs in that the instance features are learned during training (known as end-to-end training). Although there can be merit for manual featurization, end-to-end training harnesses the full power of deep learning by extracting features relevant to the learning task, and would be expected to achieve the best results with enough training examples. To perform end-to-end training decisions must be made on how the raw data will be presented to the model, and how the model will encode the data. We consider the outcome of this process to be a genomic concept, and this is

the most important step in extracting relevant features. Somatic mutations are often reported in a Variant Call Format (VCF) or Mutation Annotation Format (MAF), and a genomic concept can be constructed for any measurement in these files. The concept can be as simple as a trainable embedding (for example our gene encoder), or it can be as complex as convolutional layers for the flanking nucleotide sequences along with the reference and alteration in both the forward and reverse directions (our sequence encoder).

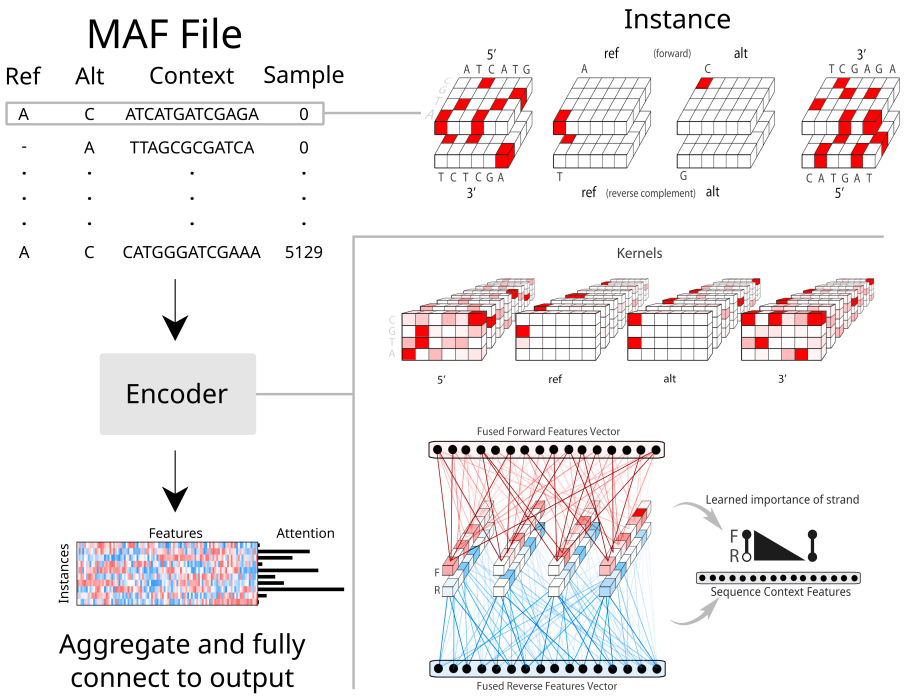

Figure 1. Model overview. From the information in a MAF various genomic concepts can be defined, and in this example sequence information is being featurized. Our concept of how to represent a variant with regards to its sequence involves splitting a variant into its 5', 3', ref, and alt sequences, onehot encoding the nucleotides, and convolving each sequence independently before fusing them together in a strand-informed manner. After the input encoder featurizes the instances the aggregation portion of the model calculates attention and performs a weighted sum per sample. At this point each sample is now represented by a single sample-level tensor of shape (attention heads, instance features) and this tensor can then be used to make predictions for classification or regression tasks.

As far as we are aware our work represents the first attempt to allow a neural network to featurize the sequence of variants as they appear in a MAF. To confirm that the encoders we developed were valid we performed positive controls, utilizing the unique mutation calls from the MC3 public MAF. Our sequence encoder was found to be a faithful representation of a variant, learning the 96 contexts and an outgroup with near perfect accuracy, and our gene encoder was able to perform a data compression without any information loss (Supplemental Figure 1). To confirm that our sequence encoder could effectively utilize strand information we asked a more difficult question, whether it could classify variants according to their consequence as provided by the MC3 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). This problem requires learning all 64 codons in 6 different reading frames, and importantly the strand the variant falls on affects the label. We first asked how well the model could do without providing reading frame, and while the model was able to learn InDels and splice sites, it was unable to distinguish noncoding mutations from the other classes, and did the best it could at associating codons with a consequence (Figure 2A). When provided the reading frame 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 2B), indicating that our weighted max reduce of the forward and reverse sequences does allow the model to learn which strand a feature was on, and correctly associate the relevant codons with consequence. 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.

To the best of our knowledge this is also the first attempt at applying MIL to this type of data. Our implementation of MIL follows that proposed by Ilse et al.[18] (Supplemental Figure 2), with some important modifications for the nature of somatic mutation data. In image analysis the aggregation function is often a weighted average, but whereas the number of tiles is unrelated to the label for an image the number of mutations may provide information about a tumor sample. Using

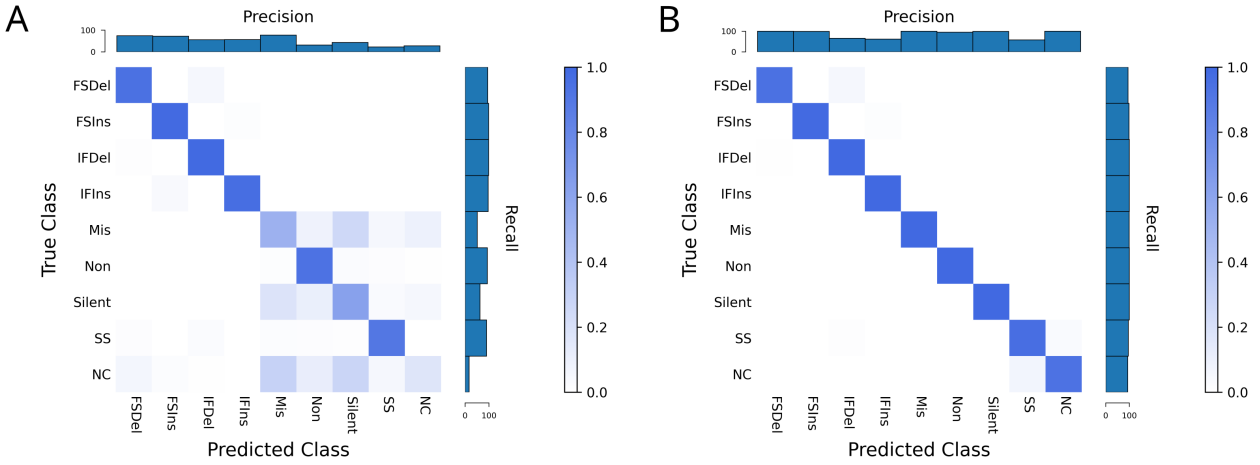

Figure 2. Predicting variant consequence. Our sequence encoder can learn variant consequence as defined by variant effect predictor (VEP) without (A) and clearly better with reading frame information (B). *FSDel=Frameshift deletion, FSIns=Frameshift insertion, IFDel=InFrame Deletion, IFIns=InFrame Insertion, Mis=Missense, Non=Nonsense, SS=Splice Site, NC=noncoding.* All four plots show row normalized confusion matrices.

simulated data we explored various MIL implementations along with traditional machine learning approaches, and found our attention-based MIL with a weighted sum performed well (Supplemental Figures 3-6). In order for a weighted sum to be meaningful the instance features must be activated, and this results in potentially large values on the graph. To account for this we perform a log of the aggregation on graph. We also developed dropout for MIL, wherein a random subset of instances is given to the model each gradient update, but then all instances are used during evaluation. This allows training of our model with large samples, for example those that might be seen with whole genome sequencing, and also helps with overfitting since the samples are altered every batch.

Cancer Type Classification

A readily available label in cancer datasets is the cancer type, and this task can have practical importance for when the tumor of origin for a metastatic cancer cannot be determined[19]. The types of mutations of a cancer are influenced by the mutational processes of its etiology while the genomic distribution of its mutations is influenced by histology of origin, and these features of somatic mutations have been shown to be capable of classifying cancers[20–23]. However, these previous studies relied on manually generated features—the onehotting of mutational context and binning of genomic location. We were interested in seeing whether we would see a benefit using a model which can learn its own features from the data. For this task we used the TCGA MC3 public mutation calls, which are exome-based. To understand the baseline performance on this data using current approaches we created two different manual features: the 96 SBS contexts with a 97th outgroup so that no data was discarded, and a onehot of genes. We opted to use genes as opposed to 1 Mb binning of the genome (as done in previous studies with whole-genome sequencing) given that the exome only covers a small fraction of the genome. For each manual feature we ran a logistic regression, random forest, and a neural net with the input vector being the sum of each sample's onehotted instances. For our MIL model we used either our sequence encoder or gene encoder, with the input being either a 6 nucleotide window of each sequence component or the gene as a categorical variable respectively. While our gene featurization essentially works with a onehot of the genes like the manual featurization approach, the efficient data compression could potentially lead to performance gains, and the embedding matrix that trains along with the graph may be informative to analyze.

Table 1 shows the results of test folds from five-fold cross validation for the different models and TCGA project codes (a 24 class problem). We also ran the models classifying samples according to their NCIt code and obtained similar results (a 27 class problem, see Supplemental Table 1). Interestingly, for sequence context a random forest performed comparably to a neural net while for genes logistic regression performed comparably to a neural net. In all cases our model achieved the highest performance, with a significant boost in performance with sequence context (∼10% increase over the best

Model	Input	Accuracy	Weighted Accuracy	AUC
Logistic Regression	Context	43.7%	44.8%	.911
Random Forest		50.1%	48.7%	.925
Neural Net		49.9%	51.5%	.934
ATGC		**55.3%**	**57.8%**	**.948**
Logistic Regression	Gene	55.8%	50.8%	.927
Random Forest		48.5%	41.7%	.898
Neural Net		56.0%	53.3%	.929
ATGC		**59.1%**	**54.8%**	**.931**

Table 1. Tumor classification performance metrics. Every model was trained with the same sample weighting and 5-fold cross validation.

alternative model).

To investigate the performance further we looked at the precisions and recalls for each model (Figure 3A). In general there wasn't a single cancer type that explained the benefit of our model, although the precision for LGG was noticeably higher for our model with sequence context (likely due to identifying IDH1 mutations). When looking at the confusion matrices for our model some cancers which might be expected to be confused for each other were, for example kidney cancers (KIRC and KIRP), and lung cancers (LUAD and LUSC). Notably, for some cancers there were significant performance differences between using sequence context or gene information, suggesting combining these two features would give the best performance. If instance features are suspected to interact with each other they should clearly be fused at the instance level, but if no interaction is expected there is a range of possibilities for combining the features[24,25]. While careful consideration should be made on how to best fuse input features in the context of MIL, that was not the focus of this study and the best approach may be data dependent.

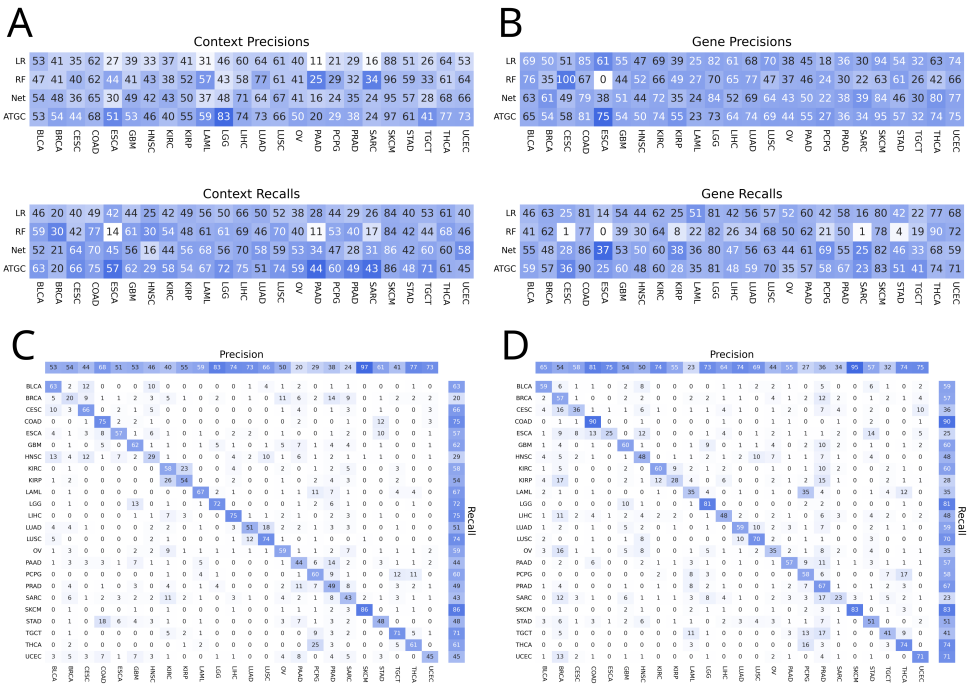

Figure 3. Tumor classification metrics. (A) Precisions and recalls for the four models and sequence context as input. (B) Precisions and recalls for the four models and genes as input. (C) Confusion matrix for ATGC and sequence context. (D) Confusion matrix for ATGC and genes. Confusion matrices are row normalized.

When investigating how the model is making decisions we can either look at how the model is featurizing the instances, or we can look at what attention the model is giving to the instances. To illustrate these two possible approaches we had the model create instance feature vectors for data it had never seen before, and looked at the feature vectors for several cancer types with known etiologies, clustering them with K-Means. Visualizing the instance feature vectors revealed a

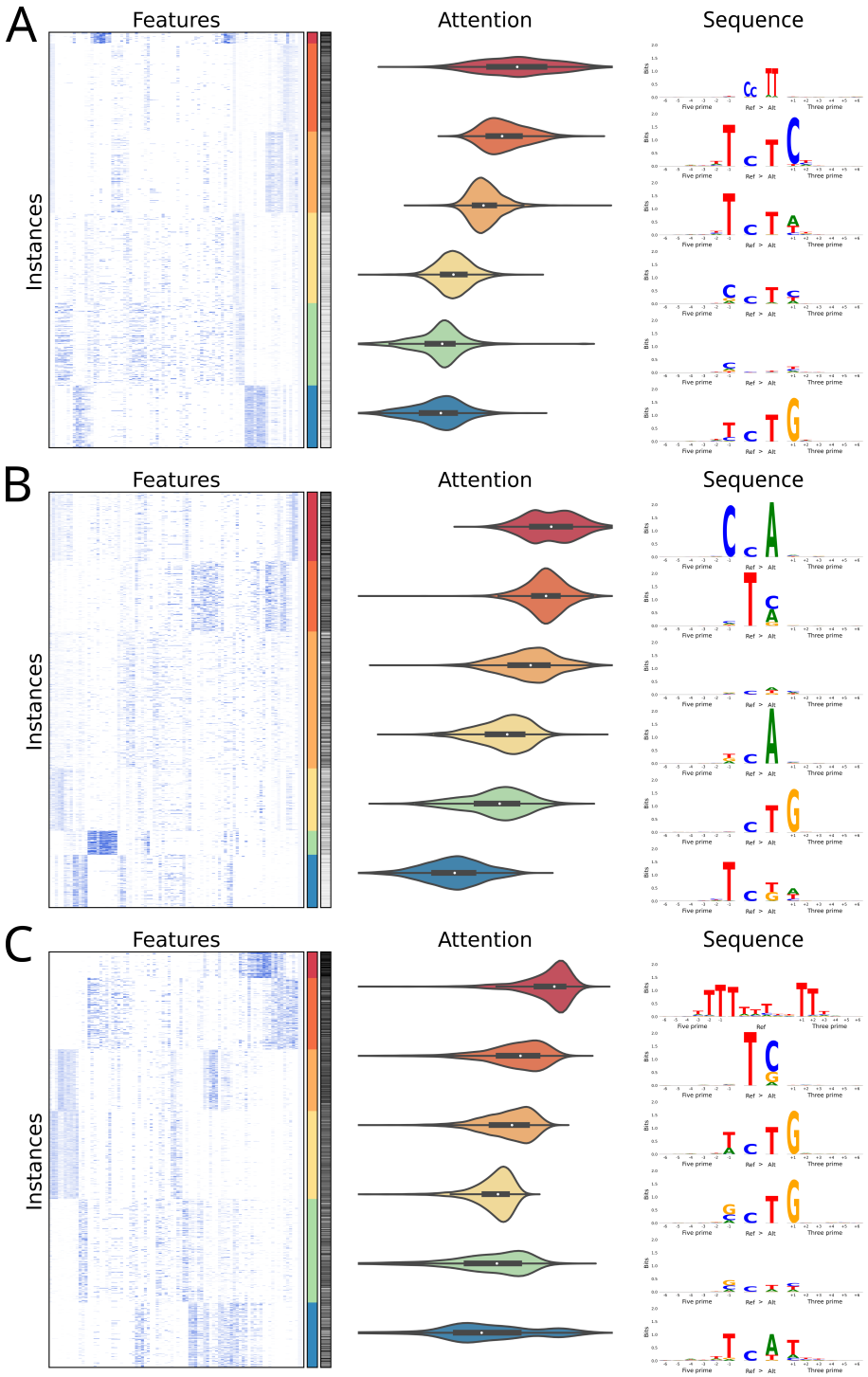

Figure 4. Instance feature vectors reveal known cancer biology. The feature vectors for the instances of a single test fold for (A) SKCM, (B) LUAD, and (C) COAD samples were clustered with K-Means into 6 clusters and ordered by their median attention, with 2000 instances per cancer shown in the heatmaps. The assigned cluster and attention for the corresponding attention head is also shown. Violin plots of the attention values for each cluster are also shown along with sequence logos of the instances that make up each cluster.

rich feature space, with clear clusters present in each cancer type. We would expect similar instances to receive similar attention, however K-Means clustering is not guaranteed to cluster the instances according to the features which generate high attention, and the clustering may not be fine enough to identify instances which are homogeneous with regards to their attention. In Figure 4A we see that for SKCM 6 clusters did for the most part produce clusters which were either high or low in attention, while in LUAD (Figure 4B), and COAD (Figure 4C), we see clusters that contain a bimodal distribution of attention.

To investigate what sequences were present in each of these clusters we generated sequence logos. For SKCM the

highest attention cluster was composed of a specific doublet base substitution characteristic of ultraviolet radiation, with the next highest attention cluster comprised of a very specific 5' nucleotide, ref, alt, and 3' nucleotide also characteristic of ultraviolet radiation[11]. For LUAD the highest attention cluster contained sequences characteristic of tobacco smoking, while in COAD the highest attention cluster was a specific deletion occurring at a homopolymer, which is characteristic of cancers deficient in mismatch repair and is a known signature of this cancer type[11].

While in Figure 4 we used clustering to identify groups of mutations within a cancer type which may be of interest, we instead could have simply sorted all instances by attention. To explore this possibility we took the highest attention instances (top 5%) of each attention head for SBSs, insertions, and deletions, and looked at the bits of information contained in the logos (Supplemental Figure 7). For SBS mutations most of the information gain (approaching a perfect 2 bits in some cases) occurs at the alteration and flanking nucleotides, however for most heads there is still information 2, 3, or more nucleotides away from the alteration. Given that a head of attention may identify several distinct motifs as important, the high attention instances for each head may not be homogeneous and as a result these bits represent a lower bound of the information gain. For InDels there is significant information 3 or 4 nucleotides away from the alteration, and these sequences are often mononucleotide repeats. When performing a similar analysis for our gene encoder we noticed that a small number of genes appear to be given high attention in each head, and clear groupings were present in the embedding matrix, with a small cluster enriched in cancer associated genes (Supplemental Figure 8).

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. As with tumor classification, 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[26]. 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[27]. Both of these approaches are valid and produce accurate predictions, but they rely on knowing the nature of the problem.

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 most variants in non-exonic regions and most simple repeats fall in these regions. 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 nucleotides for each component (5', 3', ref, and alt) to allow the model to potentially capture long repetitive sequences.

Although we did not provide the model information about cancer type or perform any sample weighting, our model displayed similar performance across cancer types (Supplemental Figure 9A). 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 5A our model slightly performs them despite not being given information about simple repeats. When comparing our predictions to MANTIS both models were often similarly confident in their predictions (Supplemental Figure 9B), however there are 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. 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).

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 5B variants which occurred in a simple repeat region were much more likely to receive high attention than variants that did not. When clustering the instances of these samples we observed two clear clusters which are receiving high attention, a smaller cluster characterized by SBS mutations in almost exclusively intergenic regions with almost 33% labeled as a

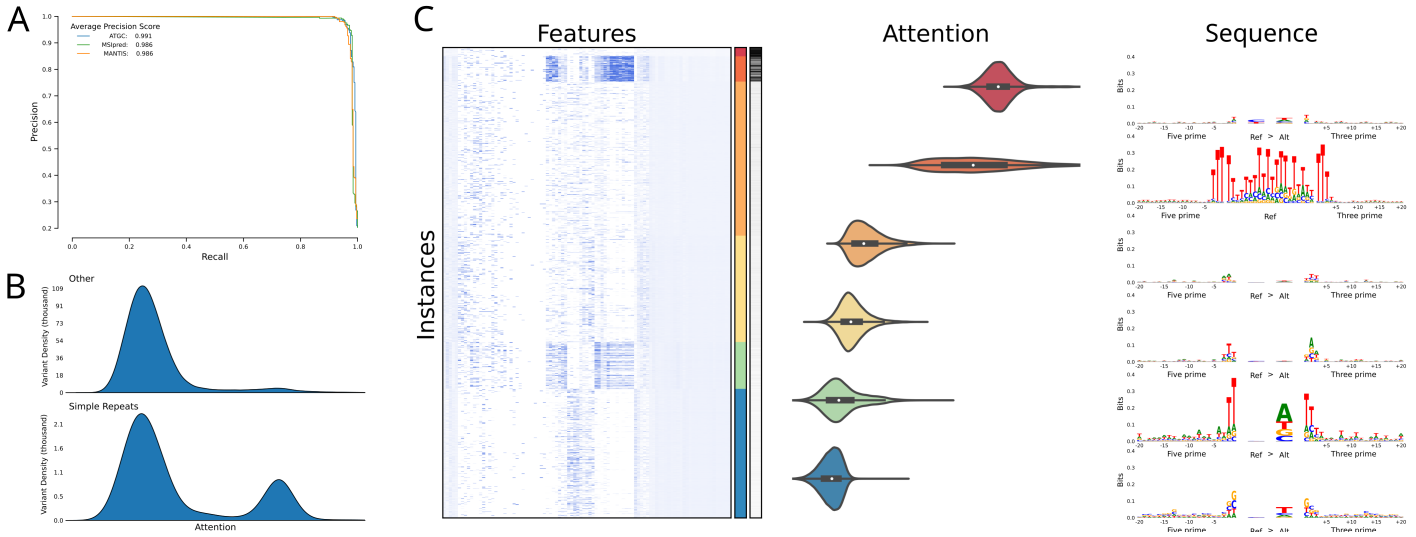

Figure 5. Application of ATGC MIL framework accurately predicts MSI status by learning which are the salient variants. (A) Average precision recall curves over 9 stratified k-fold splits for MSIpred and ATGC, precision recall curve for MANTIS for samples which had a MANTIS score. (B) Probability density 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) Features of 2000 random instances from a single test fold clustered with K-Means and the corresponding sequence logos of the clusters.

simple repeat, and a larger cluster characterized by deletions in genic but noncoding regions. The sequence logo of the intergenic cluster did not reveal a specific sequence while the logo for the deletion cluster revealed that the model is giving attention to deletions at a mononucleotide T repeat.

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 circulating DNA fragments, CHIP-SEQ peaks, methylation sites, or RNA/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. Our tool is the first to bring end-to-end training to somatic mutation data, and in doing so we introduce a new way to represent somatic mutations (our sequence and gene encoders).

When performing cancer classification with sequence context, application of our model led to a $>10\%$ improvement in accuracy over a neural net using manually generated features. When classifying samples according to their MSI status our model slightly outperformed the current state-of-the-art methods despite the other methods containing information specific to the task. The success of our sequence encoder in our simulated experiments and tumor and microsatellite classifications suggests that any model currently employing manual featurization of mutational sequence context could benefit from our encoding strategy. For high-dimensional categorical variables such as gene or bin of genomic location, our results suggest users may also see a benefit by performing a trainable embedding as performed by our gene encoder. Importantly, whereas other models require post hoc analyses to understand how they are making decisions, our tool directly reveals which instances it views as important. This is essential because 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[28].

Given the large variability in bag size for genomics problems (ranging from a few instances to potentially tens of thousands), 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 it will be easier with a smaller haystack. And in fact, MIL may not even be the best solution to this problem[29]. 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.

Research into how to best implement MIL is an active field, with a recent flurry of activity in its application to computational pathology[30]. We consider our approach to MIL fairly standard[18], with our goal being to simply demonstrate its value in the context of genomics data. As such, it is unlikely our results represent the full potential of applying MIL to this data. For example, in computational pathology there have been recent suggestions to include a clustering step at the instance level[15,31,32]. And by design we limited our instance featurization to information present in a MAF, but we can easily imagine incorporating outside knowledge into the featurization process. We hope that the success we achieved with our proof-of-concept application of MIL inspires additional work in this area.

METHODS

Availability of Data and Materials

All data used in this publication is publicly available. The MC3 MAFs are from Ellrott et al.[33], the MSI PCR labels are available from cBioPortal, TCGAbiolinks, or individual publications[34–37]. The UCSC simpleRepeat.txt was downloaded from http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/simpleRepeat.txt.gz. The Broad coverage WIGs are available at https://www.synapse.org/#!Synapse:syn21785741

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.7499923. Code is written in Python 3 and TensorFlow 2. All intersections were performed with PyRanges[38]. 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. Keras, like many deep learning libraries, requires the first dimension of the inputs to match the first dimension of the outputs. Many implementations of MIL work around this constraint by performing stochastic gradient descent, where one sample is shown to the model at a time. This precludes the ability to perform sample weighting or use loss functions which require a ranking of samples (such as the loss in Cox regression). To perform minibatch gradient descent as we would with any other model we developed our model around ragged tensors.

Our model is modular in the sense that the top aggregation model which calculates attention and generates predictions takes as input encoder models. These encoders perform operations on ragged data for instance data and normal vectors for sample data (the dataset functions automatically infer whether a ragged tensor needs to be made). This framework allows for any number of instance or sample inputs, any number of outputs, and any number of attention heads (as GPU memory permits). 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 and data augmentation (data dropout), we prefer to use our own loss and metric classes even when the loss and/or metric already exists in TensorFlow, and we created dataset utilities built around generators. We implemented a CoxPH loss with the possibility of stratification, 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 et al.[39].

Our attention is inspired by the attention first proposed by Ilse et al.[18]. One issue with the proposed attention in MIL 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. In addition it's possible for the model to find a

solution that is characterized by the absence rather than presence of key instances, which results in key instances being given a lower attention. To correct for this issue and make the attention clearer, we added L1 regularization on the output of the attention layer. 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[40] by sending information about the samples back to the instances (Supplemental Figure 2C), which may provide additional information about how the model is making decisions. For the purposes of experimentation we developed our own version of dynamic attention.

We provide users several options for the aggregation function (mean, sum, dynamic). When using an aggregation function that includes a sum the instance features should be activated, and this can result in potentially large sums. To counteract this we log the aggregations on the graph when a sum is performed. Depending on the data there may be too many mutations to fit onto the graph, so we developed dropout for MIL where a random fraction of instances per batch are sent into the model during training, but then all the instances are used for evaluation. This has the additional benefit of helping with overfitting as the samples are constantly changing during training, and is essentially a form of data augmentation.

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)[41]. 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.

MAF Processing

For the MC3 public MAF variants which had a "FILTER" value of either "PASS" , "wga", or "native_wga_mix" and fell within the coordinates of the corresponding coverage WIGs were retained. For the MC3 controlled MAF variants which had a "FILTER" value of "PASS", "NonExonic", "wga", "bitgt", "broad_PoN_v2", "native_wga_mix" or combination thereof and were called by more than one mutation caller were retained. The MC3 working group was inconsistent in its merging of consecutive SBSs, so these were merged into a single mutation if the maximum difference between the average alternative or reference counts and any single alternative or reference count was less than 5 or the maximum percent difference was less than 5%, or the maximum VAF deviation was less than 5%.

Tumor Classification Analyses

We performed tumor classification with both the original project codes and an NCIt ontology. For the NCIt classification, tumors were mapped to their NCIt code by Alex Baras, a board certified pathologist, using the available histology description. All models were weighted by tumor type, and the K-Folds were stratified by tumor type.

Logistic Regressions

"LogisticRegression" from "sklearn.linear_model" was used with default parameters.

Random Forests

We used "RandomForestClassifier" from "sklearn.ensemble" and we explored how the different parameters affected performance with "gp_minimize" from "scikit-optimize". We found most parameters to have limited effect, however we did find the default number of estimators was far from optimal, and set "n_estimators" to 900 and "min_samples_split" to 10.

Neural Nets

Neural nets were carefully optimized separately for context and gene classification, with different layer numbers, layer size, dropout, and regularization explored. Inspired by the performance of sklearn's logistic regression which performs L2 regularization by default, we found that adding a L2 penalty on the first layer of the models was highly beneficial. For the context model we found the best model to be one with a layer of 512 and L2 kernel_regularizer of .01, followed by a dropout layer of .3, a layer of 256 followed by another dropout layer of .3, a layer of 128 followed by another dropout layer of .3, to a final layer connecting to output dimension. For the gene model the best model was found to contain a layer of 1052 and L2 kernel_regularizer of .02, followed by a dropout layer of .5, a layer of 512 followed by another dropout of .5, to a final layer connecting to output dimension. All layers have relu activation except the final output layer which is logits and has no activation.

MIL Models

In order to increase the interpretability of the models we set the number of attention heads to that of the number of classes. This results in a sample tensor with multiple feature vectors, one for each class. We found that giving each vector its own subnetwork to generate a prediction for its corresponding class led to the best performance. For our context model we performed a data dropout of .4, which essentially results in augmented samples containing a random 60% of their instances each batch. We had our sequence encoder convolve each sequence component of 6 nucleotides with 16 independent kernels and our ARU activation, and fused the features at a dimension of 128 with a relu activation and .01 L2 kernel regularization, and a dropout of .5 was performed on this feature vector. There was a single layer to attention activated with our ASU activation and no regularization, and a weighted sum was performed with this attention vector. Following aggregation each subnetwork was comprised of a single layer of 256 with relu activation followed by a dropout of .5 and to a final layer of 1 with no activation, which is the prediction for the corresponding class. For our gene model we performed a data dropout of .5, and encoded the genes at a dimension of 128 with L2 regularization of .01 and relu activation, calculated attention with a single layer and ASU activation, performed a weighted sum, and then had each subnetwork directly make their predictions with a layer of 1 and no activation.

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, but otherwise was run as recommended. Models were not weighted by tumor type, but tumor type was used for K-Fold stratification. For our MIL model because there were only 2 classes we considered it a binary classification task and used a single head of attention. A data dropout of .4 was used, and the sequence components were given 8 independent kernels with ARU activation and fused to a dimension of 128 with a relu activation and .01 L2 kernel regularization, a dropout of .5 was performed, then attention was calculated with a single layer and ASU activation and .05 L1 activity regularization to force the positive class to receive higher attention. Following aggregation a layer of 256 and relu activation was used followed by .5 dropout, a layer of 128 and relu activation followed by .5 dropout, to the final prediction and no activation.

Sequence Logos

Logomaker[42] was used for the sequence logos. The probabilities were calculated separately for single base substitutions, doublet base substitutions, insertions, and deletions.

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

1. Ching, T. *et al.* Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface* **15,** 20170387 (2018).
2. Routhier, E. & Mozziconacci, J. Genomics enters the deep learning era. *PeerJ* **10,** e13613 (2022).
3. Altman, N. S. & Krzywinski, M. The curse (s) of dimensionality. *Nature Methods* **15,** 399–400 (2018).
4. Elmarakeby, H. A. *et al.* Biologically informed deep neural network for prostate cancer discovery. *Nature* **598,** 348–352 (2021).
5. Romero, A. *et al.* Diet networks: thin parameters for fat genomics. *arXiv preprint arXiv:1611.09340* (2016).
6. Chang, M. T. *et al.* Accelerating discovery of functional mutant alleles in cancer. *Cancer Discovery* **8,** 174–183 (2018).
7. Kim, S., Lee, H., Kim, K. & Kang, J. Mut2Vec: distributed representation of cancerous mutations. *BMC Medical Genomics* **11,** 33 (2018).
8. Palazzo, M., Beauseroy, P. & Yankilevich, P. A pan-cancer somatic mutation embedding using autoencoders. *BMC Bioinformatics* **20,** 655 (2019).
9. Peng, J., Zou, D., Gong, W., Kang, S. & Han, L. Deep neural network classification based on somatic mutations potentially predicts clinical benefit of immune checkpoint blockade in lung adenocarcinoma. *OncoImmunology* **9,** 1734156 (2020).
10. Alexandrov, L. B. *et al.* Signatures of mutational processes in human cancer. *Nature* **500,** 415–421 (2013).
11. Alexandrov, L. B. *et al.* The repertoire of mutational signatures in human cancer. *Nature* **578,** 94–101 (2020).

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

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

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

15. Lu, M. Y. *et al.* Data-efficient and weakly supervised computational pathology on whole-slide images. *Nature Biomedical Engineering* **5,** 555–570. https://doi.org/10.1038/s41551-020-00682-w (2021).

16. Lu, M. Y. *et al.* AI-based pathology predicts origins for cancers of unknown primary. *Nature* **594,** 106–110. https://doi.org/10.1038/s41586-021-03512-4 (2021).

17. Chen, R. J. *et al. Whole Slide Images are 2D Point Clouds: Context-Aware Survival Prediction Using Patch-Based Graph Convolutional Networks* in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021* 339–349 (Springer International Publishing, 2021). https://doi.org/10.1007/978-3-030-87237-3_33.

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

19. Pavlidis, N. & Pentheroudakis, G. Cancer of unknown primary site. *The Lancet* **379,** 1428–1435 (2012).

20. Salvadores, M., Mas-Ponte, D. & Supek, F. Passenger mutations accurately classify human tumors. *PLoS computational biology* **15,** e1006953 (2019).

21. Jiao, W. *et al.* A deep learning system accurately classifies primary and metastatic cancers using passenger mutation patterns. *Nature Communications* **11,** 728. ISSN: 2041-1723 (2020).

22. Danyi, A., Jager, M. & de Ridder, J. Cancer type classification in liquid biopsies based on sparse mutational profiles enabled through data augmentation and integration. *Life* **12,** 1 (2021).

23. Sanjaya, P., Waszak, S. M., Stegle, O., Korbel, J. O. & Pitkanen, E. Mutation-Attention (MuAt): deep representation learning of somatic mutations for tumour typing and subtyping. *bioRxiv* (2022).

24. Li, Y. C., Wang, L., Law, J. N., Murali, T. & Pandey, G. Integrating multimodal data through interpretable heterogeneous ensembles. *Bioinformatics advances* **2,** vbac065 (2022).

25. Lipkova, J. *et al.* Artificial intelligence for multimodal data integration in oncology. *Cancer Cell* **40,** 1095–1110 (2022).

26. Kautto, E. A. *et al.* Performance evaluation for rapid detection of pan-cancer microsatellite instability with MANTIS. *Oncotarget* **8,** 7452 (2017).

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

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

29. Akerman, O., Louzoun, Y., Isakov, H. & Levi, R. Counting Is Almost All You Need. *bioRxiv* (2022).

30. Gadermayr, M. & Tschuchnig, M. Multiple Instance Learning for Digital Pathology: A Review on the State-of-the-Art, Limitations & Future Potential. *arXiv preprint arXiv:2206.04425* (2022).

31. Li, J. *et al.* A multi-resolution model for histopathology image classification and localization with multiple instance learning. *Computers in biology and medicine* **131,** 104253 (2021).

32. Sharma, Y. *et al. Cluster-to-conquer: A framework for end-to-end multi-instance learning for whole slide image classification* in *Medical Imaging with Deep Learning* (2021), 682–698.

33. Ellrott, K. *et al.* Scalable open science approach for mutation calling of tumor exomes using multiple genomic pipelines. *Cell Systems* **6,** 271–281 (2018).

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

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

36. Berger, A. C. *et al.* A comprehensive pan-cancer molecular study of gynecologic and breast cancers. *Cancer cell* **33,** 690–705 (2018).

37. Liu, Y. *et al.* Comparative molecular analysis of gastrointestinal adenocarcinomas. *Cancer cell* **33,** 721–735 (2018).

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

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

40. Yan, Y. *et al. Deep multi-instance learning with dynamic pooling* in *Asian Conference on Machine Learning* (2018), 662–677.

41. Carlile, B., Delamarter, G., Kinney, P., Marti, A. & Whitney, B. Improving deep learning by inverse square root linear units (ISRLUs). *arXiv preprint arXiv:1710.09967* (2017).

42. Tareen, A. & Kinney, J. B. Logomaker: beautiful sequence logos in Python. *Bioinformatics* **36,** 2272–2274 (2020).