Max Kelsen

Representation Learning in Cancer Genomics

Athon Millane

Honours Thesis

School of Information Technology and Electrical Engineering
University of Queensland
Australia

04/11/2019
Dear Professor Abosh,

In accordance with the requirements of the degree of Bachelor of Engineering (Honours) in the division of Electrical and Computer Engineering, I present the following thesis entitled “Representation Learning in Cancer Genomics”. This work was performed under the supervision of Dr Sally Shrapnel.

I declare that the work submitted in this thesis is my own, except as acknowledged in the text and footnotes, and has not been previously submitted for a degree at The University of Queensland or any other institution.

Yours sincerely,

Athon Millane
ATHON MILLANE.
I would first like to thank my supervisor Dr Sally Shrapnel for her significant guidance and feedback throughout the course of my project. Her diligence and enthusiasm was ever present and a source of inspiration throughout the project. I extend thanks to Dr Maciej Trzaskowski for providing industry supervision, and to Nick, Ryan and the rest of the Max Kelsen team for supporting my ideas. I would like thank Dr Nic Waddell and Mr John Pearson at QIMR/GenomiQa for their valuable contributions on biological directions to explore. For their support and encouragement I’d like to acknowledge my family.
Abstract

Genomic information is becoming an abundant resource for understanding our biology, and identifying mechanisms of disease. Cancer is one such disease, which is highly heterogeneous across causes and effects. This project is concerned with how biologically informed representations can help decode the complex hierarchical structure of the genome. We learn distributed representations of genes according to shared expression in biological pathways, demonstrating the ability to cluster pathological genes, and to differentiate oncogenic pathways within a pathological gene set. We combine gene vectors with somatic mutation summaries, to add a biological context to cancer cases, and demonstrate a 3.2% improvement over existing studies in the task of tumour classification across 29 tumour types. In a second study, we evaluate sequential representations of DNA, and investigate the impact of fixed and variable length tokenisation schemes on learning. We show the value of variable length tokenisation for visualising entropy across regions of the genome, and identify short sequence motifs consistent with biological literature. Learned sequential representations are evaluated on functional lncRNA/mRNA classification, and the best representations outperform a comparable study by 2.9%.
# CONTENTS

Acknowledgements iii  
Abstract v  
List of Figures ix  
List of Tables x  

1 Introduction 1  
1.1 Chapter outline ........................................ 5  

2 Background 7  
2.1 Machine learning and genomics .................................... 7  
2.1.1 Representation Learning .................................... 7  
2.1.2 Representations in Genomics .................................. 8  
2.1.3 Somatic Mutations ........................................... 8  
2.2 Distributed representations .................................... 10  
2.2.1 Word2Vec .................................................. 11  
2.2.2 Biological vectors .......................................... 12  
2.3 Dimensionality reduction ..................................... 12  
2.3.1 Principle Component Analysis ................................ 13  
2.3.2 t-Distributed Stochastic Neighbour Embedding ............. 13  

3 Study 1: Distributed representation of genes 15  
3.1 Introduction .................................................. 16  
3.2 Aims ......................................................... 16
3.3 Source domain .................................................. 18
  3.3.1 Biological basis ........................................... 18
  3.3.2 Dataset and preprocessing .............................. 18
  3.3.3 Gene2Vec .................................................. 21
3.4 Target domain .................................................. 28
  3.4.1 Representations of mutations ......................... 28
  3.4.2 Gene aggregated mutation indicator .................. 31
  3.4.3 Dataset and Preprocessing .............................. 31
3.5 Extended representation ...................................... 34
  3.5.1 Clustering and image manipulation ................. 34
  3.5.2 Core technique ............................................ 37
3.6 Results .......................................................... 39
  3.6.1 Experiment 1A: Cancer classification with MutInd ... 41
  3.6.2 Experiment 1B: Cancer classification with GeneVec .. 42

4 Study 2: Sequential representation of DNA ................. 45
  4.1 Introduction ................................................. 45
  4.2 Aims .......................................................... 46
  4.3 Data and preprocessing ..................................... 47
    4.3.1 Whole genome data .................................... 47
    4.3.2 Task specific data .................................... 48
  4.4 Nucleotide Tokenisation .................................... 49
    4.4.1 Fixed Length Tokenisation ............................ 49
    4.4.2 Variable length tokenisation ......................... 51
  4.5 Results ........................................................ 55
    4.5.1 Experiment 2A: Unsupervised sequence representation learning with recurrent neural networks .......... 55
    4.5.2 Experiment 2B: Task specific sequence classification .................. 56

5 Discussion ....................................................... 59
  5.1 Experimental design ........................................ 59
  5.2 Limitations - Study 1 ....................................... 60
5.2.1 Gene vector learning ........................................... 60
5.2.2 Domain extension of mutations .............................. 61
5.2.3 Classification task ............................................. 63
5.3 Limitations - Study 2 ............................................. 63
5.3.1 Nucleotide tokenisation ..................................... 63
5.3.2 Sequence classification ................................. 65
5.4 Summary .......................................................... 66

List of Figures

1.1 Cost of whole genome sequencing since the first human genome in 2001, compared to the exponential progress of Moore’s law (dotted line). Adapted from [3]. ......................... 2
1.2 Extrinsic vs. intrinsic representations. ............................ 4
2.1 Continuous bag of words (CBOW) and Skip-gram architectures. Adapted from [48]. .......................................... 11
3.1 Distributional properties of all MSigDB pathways, by length. .......... 19
3.2 Gene pairs selected from biological pathways serve as positive samples for the gene2vec technique. ........................................... 22
3.3 PCA2 plots of GeneVec, examining cross section of embedding dimension and training iterations. ..................................... 23
3.4 Scree plots of cumulative variance of PCs across embedding dimension and training iteration. ........................................ 24
3.5 PCA2 Plot of 1024 dim gene vectors. Genes coloured by cancer pathway. ................................................ 25
3.6 PCA2 plot of cancer gene2vec genes only. Genes coloured by associated pathway. ................................................ 26
3.7 tSNE plot of cancer gene2vec genes only. Genes coloured by associated pathway.

3.8 Distributions of interest for unprocessed data from the TCGA data portal.

3.9 Distribution of mutation indices by co-occurrence within the TCGA dataset.

3.10 Subsampling, clustering and sorting operations applied to GeneVec matrix.

3.11 Evaluating different clustering strategies within matrices.

3.12 Plot thickening strategy applied on a single sample for 32 and 256-dimensional gene vectors.

3.13 Constructing extended representation, validated on tumour classification task.

3.14 Classification accuracy with CV confidence interval for best models in each category.

3.15 Confusion matrix for logistic regression with MutInd vectors.

3.16 CNN with GeneVec matrices.

4.1 Distribution of sequence lengths for the 2 task specific datasets.

4.2 Computation time for sequential and vectorised fixed-length tokenisation strategies.

4.3 Distribution properties of tokens for variable length tokenisation strategies.

4.4 Continuous colour map of tokenised DNA according to token length.

4.5 Continuous colour map of tokenised DNA according to token prevalence in dataset.

4.6 Cross entropy performance for next token prediction using fixed and variable length tokenisation.

4.7 Testset classification accuracy on mRNA/lncRNA classification task using fixed and variable length tokenisation.
List of Tables

3.1 Chosen parameters for \texttt{gensim} model. \hspace{1cm} 21
3.2 Fields extracted from the TCGA data portal. \hspace{1cm} 32
3.3 Parameters evaluated in grid search. \hspace{1cm} 42
3.4 Classification performance using 2 representations. \hspace{1cm} 44
4.1 Classification accuracy across best sequence models. \hspace{1cm} 57
Chapter 1

Introduction

The future of humanity relies on understanding our biology. In the past 100 years, the average human lifespan has doubled, a feat attributed to magnifying biology and examining the interaction of subsystems we were previously unaware of [53]. In the 1800’s, the germ theory of disease [50], lead surgeons to wash their hands before delivering babies or performing surgery. In the 1900’s, the advent of microbiology and ability to see cells as regenerating, interacting agents led to a sophisticated enough understanding of the immune system to engineer our bodies against microbial invasion. We identified causal factors, and then probed enough in the space of outcomes to realise when and where interventions can have a positive affect. In this century, we finally have sufficient technology to peer all the way to the source of this complexity, the 3.2 billion letters of a four letter alphabet \{A,C,T,G\}, that define our cells, bodies and brains, behaviour, societies and ecosystem.

In 2003, after 13 years and 5 billion of investment, The Human Genome Project reported coverage of 99% of the euchromatic human genome with 99.99% accuracy [42]. Since that time, millions of entire genomes have been sequenced [45]. Hidden in this data is an encyclopedia of our evolutionary history, an instruction manual for every cell in our bodies, in a million different revisions, with correlated results on display. Now comes the task of decoding this information, learning the language, the syntax and semantics, to reveal the hierarchy of information leading from single molecules to intelligent life.
The motivation for learning the genomic language extends beyond curiosity or philosophical interest. The second leading cause of death globally, responsible for an estimated 9.6 million deaths in 2018 [21], is the disease of the genome known as cancer. Cancer arises when the genetic information inside a normal cell is changed or ‘mutated’ in such a way that cells proliferate in an uncontrolled way[24]. Cancerous tissue can spread throughout the body and displace tissue, bone and organs, interfering with the regular function of systems in the body. Mutations may be small - a single letter in the sequence of billions as an ‘A’ where the should be a ‘T’, or structural, with large fragments of code being shifted, deleted or duplicated [47]. Such mutations may be caused by a UV ray penetrating through the cell membrane on the surface of the skin [4], or exposure of the cells in the lungs to the benzene, ethylene oxide, or acetaldehyde in tobacco smoke [30]. Recognising this correlation has led to ‘SunSmart’ exposure safety campaigns [51] and smoking reduction strategies put in place [32], likely saving millions of lives. In remaining cases, such clear causes and preclusions often aren’t available.
The reason cancer still poses a massive burden to human health is in large part due to it’s heterogeneity [11]. Mutations can be caused by so many different factors that developing any general causal hypothesis is impossible. Moreover, even identifying the mutations in order to diagnose or treat cancer is challenging, since many mutations may be completely benign, while many others are healthy expression of genetic variance (germ-line mutations) [20]. To complicate things further, cancers evolve over time, spreading into numerous sub-populations where even if the majority of the cancer is destroyed through targeted treatment, the strongest variant of mutation may survive and then spread even more rapidly than before. While this behaviour seems chaotic, or impossibly complex, decoding the language of the genome would reveal this as phenomena emergent from rules that we could control or intervene upon.

A promising technique in this decoding effort is machine learning [17]. Machine learning (ML) uses computationally efficient statistical techniques to learn from large data domains, without the need for human-coded rules. In recent years, ML techniques have become very popular and have been adopted for tasks across many scientific domains, including in genomics [68]. ML often follows a supervised paradigm, involving taking data and labels, and partitioning the data to minimise an objective function, typically the difference between predictions and labels. Such supervised learning approaches have been successful for a number of genomics tasks, such as variant calling [52], and DNA-binding domains for proteins [2], significantly outperforming previous approaches.

Other tasks in cancer genomics, such as tumour classification from somatic mutation data, still achieve relatively low accuracy, largely due to cancer heterogeneity [67]. The hypothesis underpinning this project is that incorporating biological priors into descriptions of mutations can make them easier to identify and interpret. Representation learning, a principle outlined at the genesis of ML [34], gives us techniques for summarising biological substructure, such as genes or functional sequence regions. These representations can then be recombined in a new context, allowing for emergent information more complex than the original representation domain.

In this project, we have taken two distinct approaches to learning representations, which we separate by the terminology of extrinsic and intrinsic representations. Extrinsic repres-
entations learn a description of an entity according to its relationship with neighbouring, external entities. *Intrinsic* representations describe an entity according to its internal sub-structure, including the order and type of the sub-entities it contains.

![Extrinsic vs. intrinsic representations.](image)

The biological application of this abstract distinction, is accomplished over two studies. Study 1 explores an extrinsic representation, by learning distributed representations of genes according to shared co-expression in biological pathways. These learned gene vectors are combined with somatic mutation signatures to construct an extended representation of a tumour sample. The quality of this representation is evaluated qualitatively through visualisation, and quantitatively on the task of tumour type classification across 29 tumour samples.

Study 2 learns an intrinsic representation of DNA regions, by using sequence models to learn vector representations of sequences and classify functional structure across the genome. We investigate the effect of how DNA sub-entities are defined by comparing fixed and variable length tokenisation schemes. Representations are evaluated according to quality of learning, and classification accuracy on the task of mRNA/IncRNA functional sequence classification.
1.1 Chapter outline

The remainder of this work is detailed in the following chapters:

**Chapter 2** introduces the field of representation learning, including the history of the field and present day value. A number of different representational paradigms are covered, including distributed, sequential and intrinsic vs. extrinsic representations. Theory is grounded in the application to genomics, and literature at the intersection of domains is introduced.

**Chapter 3** presents the first study, an investigation into how distributed gene representations can be used to support somatic mutation data on a disparate sample set, for the task of tumour type classification. The problem domain is covered, followed by exploratory data analysis. A baseline experiment sets benchmarks, followed by a novel deep learning representation for comparison.

**Chapter 4** presents the second study, exploring how sequential representation of bases can be used to identify functional structure across large regions of the genome. Strategies for sequence tokenisation strategies are described and explored in detail, followed by experimental results. Experiments assess general representational ability and task specific performance for supervised sequence classification.

**Chapter 5** is a discussion of major findings from both studies. Models are assessed against current literature and biology. Future extensions of the work are explored.
Chapter 2

Background

2.1 Machine learning and genomics

2.1.1 Representation Learning

Information processing tasks can be made easier or more difficult depending on the representation of the data provided. To push the frontiers of science, often breakthroughs are in representations. As an example, in quantum computing, Dirac’s Bra-Ket notation [14] offered a representation to simplify operations on quantum states. In machine learning, it is very common to engineer time features by applying a sinusoidal operator to map from a linear space to a circle - ensuring 11:59pm is close to 12:01am.

In deep learning, a good representation is generally one which makes a subsequent learning task easier [27]. Deep neural networks aim to improve representation at every layer, with each subsequent layer more abstract than the next. Convolutional neural networks learn a representation which is spatially invariant, while recurrent neural networks learn to represent relationships within a temporal sequence. A supervised neural network adds a final fully connected layer either for regression or classification in $n$ dimensions.
2.1.2 Representations in Genomics

DNA mutations are often represented in Variant Call File (VCF) format, alternatively Mutation Annotation Format (MAF). These store the index (related to reference genome) of mutations, the type of mutation, the value of the reference nucleotide and result of the mutation. This is in essence a git diff of the changes in one sample versus the agreed reference.

For both DNA and RNAseq data, it is common to represent sample specific information aggregated at the gene level to reduce dimension. Somatic mutations can be mapped to genes using a one-hot encoding (1 or more mutations present in gene) or a count encoding (number of mutations present in gene). RNAseq gene expression data similarly aggregates read counts of mRNA found in a selection of cells - when normalised this represents a single continuous dimension per gene.

2.1.3 Somatic Mutations

Mutations in cancer are somatic mutations, or mutations acquired by a person’s cells throughout their life. Certainly not all somatic mutations are cancerous. Everyone who interacts with the environment will have mutations in their cells throughout their life, and in fact biopsies of healthy tissues have shown just as many mutations as those found in cancer tissue [65]. The fact remains that through our current understanding of biology, certain mutations are linked to cancer. Environmental factors causing cancer-correlated somatic mutations factors may include, smoking, UV exposure, chemicals in food, or cell degeneration with ageing.

Somatic mutations are often categorised by the way in which a sequence of bases is varied. If a single base-pair within the genome sequence is changed (say by interaction with a UV ray), this is known as an SNP (single nucleotide polymorphism). An example of an A to C SNP in the 4th base pair of our sequence is demonstrated below.
2.1. MACHINE LEARNING AND GENOMICS

SNP(\(x, 4, A, C\)) : \(\{x = \text{ACT } \underbrace{A}_{A \rightarrow C} \text{GCGAAT} \rightarrow \text{ACTCGCGAAT}\}\)

Such a mutation can be described by two pieces of information. Firstly, the index as compared to the reference genome is required, for example: \{base_pair: 123,497,128\} or \{chromosome: Y, gene: RBMY2EP, start_pos: 2139576\}. Additionally, the new base pair that the original base pair mutates to will define which new amino acid is coded in its place. Since there is a many to-one mapping of 3-mers to amino acids, there is a chance that the mutation has no effect on the resulting amino acid. Such a mutation is known as a silent mutation. A change that maps to a new amino acid is a missense mutation, and a change with no mapping is a nonsense mutation, and will signal to the cell to end translation prematurely.

Another common type of variants are insertions and deletions, together known as indels. These entail the insertion or deletion of short sequences of DNA (typically less than 50 base pairs), at a specific location within the genome. An indel can be characterised by a start index, an end index (if deletion) and an insertion sequence (if insertion).

\[
\text{INS}(x, 2, (AGA)) : \left\{ x = \text{AC } \underbrace{\ast}_{\text{AGA}} \text{TAGCGAAT} \rightarrow \text{ACAGAATGCGAAT} \right\}
\]

\[
\text{DEL}(x, 2, 4) : \left\{ x = \text{AC } \underbrace{TA}_{\text{delete}} \text{GCGAAT} \rightarrow \text{ACGCGAAT} \right\}
\]

During translation, base pair sequences are converted in lengths of three (3-mers) to amino acids. An insertion or deletion which is not a multiple of 3 long is known as a frame-shift mutation, since the remainder of the bases in the gene will be mis-translated. This can drastically alter the function of the DNA in that region.

Larger rearrangements of the genome (typically greater than 50 base pairs) are known as structural variants. This may include large INDELs, complete rearrangements of large DNA fragments or an abnormal number of copies of a chromosome being observed.
Whilst the existence of such variations in the genome have been linked to many human disorders, they are very difficult to characterise or localise due to the large combinatorics of possible changes. An indicator specifying existence of structural variant in the genome, and broad localisation only as specific as which chromosome the variant occupies, could be provided. Compared to small SNPs and INDELs, which can be localised to a specific base pair index and substitution, structural variants are poorly determined and differentiated, making them incompatible as raw genomic features, and difficult to draw conclusions from.

2.2 Distributed representations

Distributed representations, as introduced by Hinton et al. in 1986 [34], entail the many to many mapping of parameters to entities. This sits in contrast to a local representation, where a piece of information, such as a word, is localised to specific place in memory. Instead, distributed representations capture concepts with patterns of activity. The authors of this seminal work cite 3 major advantages of distributed representations:

1. Their constructive characteristics, allowing for approximate description with imperfect information,
2. Their ability to generalise to novel situations
3. Their tunability to changing environments.

The same authors, later that year, introduce error back-propagation as the method of learning distributed representations [54], and modern neural network theory was born. Distributed representations, while not always referred as such, are ubiquitous in deep learning. In fact, any layer in a network not directly reflecting information or labels, is by definition, a distributed representation. Recent literature has shown the fascinating properties of representations from intermediate layers of neural networks, such as Gabor filters [22] and edge detectors in early layers of image models, and more developed sub-structure (such as ears
or whiskers for a cat) in later layers [66]. In a majority of ML contexts, representations are just a by-product of some supervised optimisation problem, such as classification or regression.

### 2.2.1 Word2Vec

In the field of language understanding, distributed representations have become the center of attention. Mikolov et al. [48] propose two neural network architectures for learning continuous vector representations of words on very large datasets, training on 1.6 billion words in less than a day.

![Figure 2.1: Continuous bag of words (CBOW) and Skip-gram architectures. Adapted from [48].](image)

The main efficiency gains of this technique are that where previous works used a feed-forward neural network with non-linear hidden embedding layer H, the authors opt to remove this layer entirely. Instead, they only maintain a projection layer of size \( N \times D \), where \( N \) is a sparse number of connections, typically \( N = 10 \), and \( D \) is the projection dimension [49]. The result is a simpler but much more efficient architecture. In typical models, the
CHAPTER 2. BACKGROUND

dominating term for training complexity is the connections between the vocabulary $V$ and
embedding layer $H$, resulting in $Q = N \times D + H \times V = O(H \times V)$, where $Q$ is computational
complexity per training example [48]. The CBOW architecture predicts the current word
based on the context, which is a small window around the word, with a resultant com-
plexity of $Q = N \times D + D \times \log_2(V)$, while Skip-gram architecture predicts surrounding
words given the current word with complexity of $Q = C \times (D + D \times \log_2(V))$, where $C$ is
maximum word distance, typically $C = 5$ [48]. Even with simpler learning architecture,
this technique showed significant performance improvement over any previous models for
tasks such as sentence completion (next word prediction) [49].

2.2.2 Biological vectors

With the significant impact of this technique on many facets of natural language learn-
ing, distributed representations have found application in biology. Gene2Vec [15] uses
the Skip-gram architecture to learn distributed vector representations of genes. The au-
thors use default configurations of the original implementation, with a 200 dimensional
projection layer. By curating a dataset of gene expression terms from the Gene Expression
Omnibus (GEO) [16], the authors develop a dataset of gene pairs known to share context
with one another. They use the open source Python implementation gensim to complete
experiments.

2.3 Dimensionality reduction

Part of the value of working with distributed vector representations is the ability to visu-
alise the vector space of interest, and identify clusters or relationships between entities.
Since these embedding spaces are usually in a high dimension of size (200 - 1000) [49],
for visualisation we aim to compress the most salient factors into 2 or 3 spatial dimensions,
for viewing.
2.3.1 Principle Component Analysis

Principle component analysis uses orthogonal transformations to convert a set of observations of potentially correlated variables, into a smaller set of orthogonal, linearly uncorrelated variables known as principle components. This is practically achieved by capturing the most possible variance of the data with a single component, then defining consecutive components that are orthogonal to the PC with the same objective of maximising variance in the data. In order to maximise variance, the first weight vector $w^{(1)}$ mapping the data matrix $X$ to a PC thus has to satisfy [38]:

$$w^{(1)} = \arg \max_{\|w\| = 1} \left\{ \sum_i (x_i \cdot w)^2 \right\} = \arg \max_{\|w\| = 1} \left\{ \|Xw\|^2 \right\}$$

For further components, the $k$th PC is found by subtracting the first $k-1$ PCs from $X$:

$$\tilde{X}_k = X - \sum_{s=1}^{k-1} Xw^{(s)}w^{(s)T}$$

Then maximising the variance of this residual data matrix:

$$w^{(k)} = \arg \max_{\|w\| = 1} \left\{ \|\tilde{X}_k w\|^2 \right\}$$

Numerically, an easy way to compute PCA is via Singular Value Decomposition [26], which is the default solver in the sklearn Python library.

2.3.2 t-Distributed Stochastic Neighbour Embedding

t-Distributed Stochastic Neighbour Embedding (t-SNE) is another dimensionality reduction technique, especially well suited for high dimensional datasets [63]. Rather than performing linear operations on the data, t-SNE builds a map in two dimensions, such that
similar samples are modelled by nearby points and dissimilar objects by distant points. The distance between points \( y_i \) in the map are determined by minimizing the Kullback-Liebler divergence \([41]\) between the distribution of points \( p_{ij} \) and \( q_{ij} \) in sample space \( P \) and embedded space \( Q \).

\[
KL(P\|Q) = \sum_{i \neq j} p_{ij} \ln \frac{p_{ij}}{q_{ij}}
\]

This minimisation is performed using gradient descent, resulting in a map that reflects the similarities between the high-dimensional inputs well.
Chapter 3

Study 1: Distributed representation of genes
3.1 Introduction

The first study uses distributed representations to learn biologically informed gene vectors. These vectors are recombined in the new domain of somatic mutations, for tumour classification. These techniques are similar to those described as ‘domain adaption’ or ‘transfer learning’ in machine learning [66]. A case involves the training of supervised or unsupervised models on one data domain, with the aim that statistical structure within this domain can be transferred to a new, disparate domain. A drawback of this application is that by focusing on transfer from source domain to target domain, an implicit requirement is that the target domain has the same statistical properties of the source domain. Instead, we take a representations perspective, with the aim of learning shared information at different hierarchies. This is very amenable to biology, since, as established in Chapter 2, genomics contains nested structure at every scale.

To model information at one level of this biological hierarchy, we learn a distributed representation of genes, henceforth "gene vectors", according to how they co-occur in gene expression pathways. This representation is learned by only observing the extrinsic properties of genes. We hypothesise that by combining these learned representations of genes and compositing them together with known information about somatic mutations in samples1, we can build a richer representation of the sample of interest. To evaluate whether this is possible, we choose the established task [59] of tumour type classification using somatic mutations of DNA as the only available feature for each sample.

3.2 Aims

The objectives of this study can be summarised as follows:

1. **Source domain** Establish a compatible and large dataset of unsupervised informa-

---

1Here, a ‘sample’ is all genomic features of a single sequenced tissue biopsy, following conventional terminology in bioinformatics [6]. Be careful to note that in section 3.3.3, ‘positive’ and ‘negative samples’ correspond to genes, also following literature [49], but in a different context.
3.2. **AIMS**

In this case for all experiments, genes within the human genome are the chosen domain resolution. We’ll designate the term GeneVec to describe the learned representation of fixed length ‘gene vectors’ during this phase.

2. **Target domain** Choose a disparate domain which shares some dimension with the source domain. In all experiments, this is human somatic mutation data from TCGA, mapped to samples and aggregated to a single binary indicator for each gene \(1_{gene} = [0, 1]\). We coin this unaltered target domain as MutInd (portmanteau of Mutation Indicator).

3. **Extended representation** Source and target domains are now combined together in a way that improves the quality of information in the target domain; empirically, improving the performance on the designated task. This is achieved by using genes as a shared dimension, acting as the intersection between vector representations for source and target domain.

4. **Qualitative exploration** Build visualisation tools to explore the qualitative properties of our learned representation. This entails dimensionality reduction via Principle Component Analysis (PCA) and t-Stochastic Neighbour Embedding (t-SNE). Ablation studies are completed exploring cross sections of clustering techniques, PCA dimensions, image manipulation techniques and visual spectra.

5. **Quantitative assessment** To establish whether embedding biological context into our target domain actually creates a ‘better’ representation, we evaluate this representation on a classification task with corresponding performance metrics. This task is cancer type classification for 29 cancer types on the TCGA dataset. Performance for the MutInd representation is evaluated across an array of models, and the best performance is compared with the GeneVec extended representation using a Convolutional Neural Network (CNN). Results are also compared with existing studies on the same task.
3.3 Source domain

3.3.1 Biological basis

In deciding which biological context to explore to facilitate cancer classification, two factors are considered: shared dimension and shared context. Shared dimension is a necessity between source and target domain when hoping to aggregate information. Here, we chose genes, since they provide a reasonable resolution for partitioning a sample of somatic mutations across the genome ($\sim 10^5$ features per sample). With the choice of genes, the necessity of shared context becomes relevant. Whatever properties we wish to learn about must be contextually useful when aggregated together to consider a particular human sample. A landmark study on the ‘Hallmarks of Cancer’ [28] gives a strong direction, indicating that altered biological pathways, the series of observed molecular interactions that lead to cell change, are implicated in all human cancers [37]. Furthermore, the pathway altered will affect acquisition of specific biological capabilities, varying among tumours of different types. [28]

3.3.2 Dataset and preprocessing

With sufficient biological context, we resolve to map out the context of genes that share biological pathways, on the assumption that shared pathway context indicates shared biological function\(^2\). Using the MSigDB v6.2 collections [44], all collections of 17,809 pathways were downloaded for exploration. This collection is far reaching, including the Hallmark Gene Sets [43] as well as positional gene sets, microRNAs, immunologic signatures [25] and gene ontology terms [5]. This data was downloaded in a single package, without any identification of different pathway sets. Plotting the distribution of these sets by length (number of genes contained within), reveals an approximately bi-modal distribution, with the typical drop off moving away from zero, but with a very large number of pathways con-

\(^2\text{All code, data pipelines and experimental notebooks for this project with supporting environment configuration scripts can be found at github.com/athon-millane/thesis.}\)
3.3. **SOURCE DOMAIN**

taining exactly 200 genes. With further inspection of MSigDB literature, this was revealed to be the C7 collection of Immunological signatures [25], found to correspond to the top or bottom genes (FDR < 0.02 or maximum of 200 genes) ranked by mutual information. These make-up 4872 of the gene sets, more than a quarter of available data.

![Figure 3.1: Distributional properties of all MSigDB pathways, by length.](image)

This data was then processed into pairs of genes, with every pair accounting for a known co-occurrence of genes together in a pathway. With 32,479 gene identifiers spread across 17,809 pathways, the combinatorics of this task become very large, and code to generate pairs must be strategic. By using `itertools` and Python's standard library `set` operation, alongside `pandas.pipe` operators, a vectorised pair algorithm was constructed and run across the entire dataset (see Listing 1.1). Even with optimisation, the code still took a number of hours to run on a 256 thread cloud machine, and resulted in 144,603,456 unique gene pairs. Concerned a the size of this number, the number of possible pairs from the set of all gene ids was calculated for comparison, using the equation for combinations.
CHAPTER 3. STUDY 1: DISTRIBUTED REPRESENTATION OF GENES

\[ p = \frac{n!}{k!(n-k)!}, \quad \text{here } n=|\text{genes}|=32479, \text{ and } k=2, \text{ so} \]
\[ = \frac{n!}{2!(n-2)!} \]
by induction, this is the same as:
\[ = \frac{n(n-1)}{2} \]
\[ = 527,426,481 \text{ unique gene pairs} \]

The result here is that a subset approximately a quarter the size of all possible gene pairs is curated, thus capturing shared pathway information. Listing 3.1 demonstrates the Python code used to construct pairs, using vector set operations and pandas pipelines.

Listing 3.1: Python code for generating gene pairs.

```python
def generate_gene_pairs(gene_set):
    """Generate gene pairs with set operation."""
    from itertools import combinations
    pairs = combinations(list(gene_set), 2)
    return set(pairs)

def get_pairs_set(df):
    """Return the set union of all pairs from every gene set within the df."""
    return set.union(*df['pairs'].tolist())

def load_pairs():
    """If gene pairs file exists, load it. Otherwise, create it and save to gene pairs file."""
    print("Creating new gene pairs file.")
    pathways_df['pairs'] = (pathways_df.set
                           .apply(generate_gene_pairs))
    gene_pairs = pathways_df.pipe(get_pairs_set)
    with open (GENE_PAIRS, 'wb') as f: pkl.dump(gene_pairs, f)
    return gene_pairs
```
3.3. SOURCE DOMAIN

3.3.3 Gene2Vec

We now provide implementation details of the Gene2Vec strategy. We adapt the most popular python implementation of Word2Vec, gensim, for use on our set of gene pairs. The gensim API is designed to receive sentences of written text, parse them with a window and apply a parametrised version of either the Continuous Bag of Words (CBOW) or Skip-Gram algorithm, both described in [48]. This implementation is highly optimised for memory efficiency and multi-threaded implementation, using cython and BLAS. To train our model, we initialise the gensim model according to parameters identified in Table 3.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Choice</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2V_DIMENTION</td>
<td>-</td>
<td>dimension of the embedding</td>
</tr>
<tr>
<td>W2V_NUM_WORKERS</td>
<td>96</td>
<td>number of worker threads</td>
</tr>
<tr>
<td>W2V_MIN_COUNT</td>
<td>1</td>
<td>ignores all genes with frequency lower than this</td>
</tr>
<tr>
<td>W2V_SG</td>
<td>1</td>
<td>sg =1, skip-gram, sg =0, CBOW</td>
</tr>
<tr>
<td>W2V_WINDOW_SIZE</td>
<td>1</td>
<td>window size for sampling genes</td>
</tr>
<tr>
<td>W2V_NEGATIVE</td>
<td>3</td>
<td>number of negative samples</td>
</tr>
</tbody>
</table>

Table 3.1: Chosen parameters for gensim model.

W2V_NUM_WORKERS used as many threads as were available on the cloud instance used at the time. W2V_MIN_COUNT was kept low to capture as much information as possible. As Skip-Gram performed better than CBOW in the original study [48], this was the chosen architecture. W2V_NEGATIVE was set to the default value of 3. Since all provided ‘sentences’ are actually just pairs of genes, or sets of size 2, W2V_WINDOW_SIZE is set at 1, the default for pairs. For the embedding dimension, a grid search across the set of the following dimensions was completed: W2V_DIMENTION = {16, 32, 64, 128, 256, 512, 1024}.
Figure 3.2: Gene pairs selected from biological pathways serve as positive samples for the gene2vec technique.

Figure 3.2 gives an overview of the entire process. Positive pairs of genes with shared pathways are sampled from MSigDB, while negative pairs are sampled uniformly from the set of all possible pairs. It is important to note that this relies on the assumption that the average randomly sampled pair will not share context (negative sample). This assumption works well in NLP domains, where the likelihood of 2 random words existing in context of each other is very low, however in the section above, we already demonstrated that the set of gene pairs with shared pathways is approximately one quarter of the cardinal set. This was an oversight during training and will be further elaborated in the discussion of Chapter 5.

Training of Gene2Vec took 16 hours training with 96 workers on a cloud VM. In total, 10 training epochs were completed. Loss curves only provide a limited indication of the quality of vectors learned, so we instead opted for a qualitative assessment. To do this, we observe the latent space to see if structure is forming. Ideally, a t-distributed stochastic
neighbor embedding (tSNE) [62] plot would be run across \( n \) dimensions. Unfortunately, due to the high dimensionality of this problem, this kind of cluster analysis is not tractable. A feasible alternative is to visualise plots of the first two principle components (PCs). Principle Component Analysis (PCA) plots were fit to the vectors of all 32,479 genes. While Figure 3.3 shows some clustering structure, there is little change with the number of training iterations or embedding dimension.

Figure 3.3: PCA2 plots of GeneVec, examining cross section of embedding dimension and training iterations.
To further evaluate these PCA plots, we consider what fraction of the entire variance is captured by the 2 PCs observed. This is established through the use of a Scree plot [7], which plots the cumulative proportion of variance captured across PCs. From a linear algebra perspective, PCs are the eigenvectors of a Singular Value Decomposition [26], and their variance is equal to their corresponding eigenvalue.

Figure 3.4: Scree plots of cumulative variance of PCs across embedding dimension and training iteration.
Figure 3.4 illustrates the issue of examining only two PCs. As a fraction of the entire variance, the first two PCs only capture a maximum of 35% of variance in the best case. The shape of the scree curve is revealing. Considering the cumulative variance only plateaus after 800 or embedded dimensions (bottom row), this would support the value of working with higher dimensional embeddings.

There was an opportunity at this stage to try to decode biological meaning from the clusters, by labelling genes and establishing whether structure in the plots aligns with some existing hypothesis. The challenge here is that any explanation would be very incomplete. With over 30,000 genes plotted, manually labelling or validating gene co-expression from biological literature could only be achieved for select genes. For a small investigation, we take the top 10 cancer pathways, as outlined in [55]. According to the authors, 57% of tumors have at least one potentially actionable alteration in these pathways. Corresponding pathways are searched for by name in MSigDB, and from these, sets of genes are recorded for each pathway.

Figure 3.5: PCA2 Plot of 1024 dim gene vectors. Genes coloured by cancer pathway.

Figure 3.5 illustrates how these genes sit in comparison to entire gene set, according the
first 2 PCs. It is interesting to note that they all fall within the central region, and not in any of the well defined clusters around the outside of the plot. This is likely indicative of the significant heterogeneity of cancer behaviours, making corresponding genes and pathways difficult to isolate. While not clustered away from the rest of the gene set, it is positive to see that almost all implicated genes are within small region.

Encouraged by this central cluster of cancer pathways, we further examine this subset. The entire process of gene2vec is reinitialised, this time only running on the cancer pathway gene pairs, rather than all available from MSigDB. New gene vectors are trained, this time only for the subset of 7281 genes involved in cancer pathways. Applying the same PCA2 analysis shown in Figure 3.5, we can visualise the largest principle components of the cancer pathway genes. See Figure 3.6.

![Figure 3.6: PCA2 plot of cancer gene2vec genes only. Genes coloured by associated pathway.](image)

This visualisation is much more promising. There are a number of well isolated clusters and separate gene pathways can be distinguished. In a few cases, single pathways are broken into many smaller clusters, potentially indicating that pathways could be decon-
3.3. SOURCE DOMAIN

structured into further sub-pathways. With this smaller subset, tSNE is tractable. The python package tsne-cuda is used, to enable highly parallel computation. tSNE is run with default parameters, and applied to the first 50 PCs generated by a PCA. See Figure 3.7.

![Figure 3.7: tSNE plot of cancer gene2vec genes only. Genes coloured by associated pathway.](image)

The non-linear dimensionality reduction of tSNE shows clean clusters when applied to gene vectors of dimension 1024. Generally, clusters are tighter than for PCA, to be expected from tSNE’s minimisation objective and isotropic property. Many clusters correspond to a single pathway only, but quite a few are spread across two or more. A common pairing appears to be the P13K and MYC pathways. These pathways are demonstrated to have significant cross-pathway interaction in [55].
CHAPTER 3. STUDY 1: DISTRIBUTED REPRESENTATION OF GENES

3.4 Target domain

3.4.1 Representations of mutations

In Chapter 2, we developed an approximate understanding of causes of mutation in the genome, types of variation, and established a causal link between specific somatic mutations and proliferation of cancer cells. We also understand that a large number of tumours which have metastasised cannot be readily identified by their primary origin, leading to confusion in how best to treat the tumour and predict its behaviour. Given that the somatic mutations give the ‘initial conditions’ of a tumour, progress in identifying tumour type from somatic mutations alone may be promising for identifying primary origin.

Tumour type classification is a supervised learning problem, since for a given tissue sample, we have a profile of genomic information or features, and accompanying labels:

\[ X_i \in \mathcal{X}^d, y_i \in \mathcal{Y}^c, \quad i \in (1, \ldots, n) \]

Here \( n \) represents the number of samples in the dataset and \( d \) is the dimension of each sample, or number of features in \( \mathcal{X} \), while \( c \) is the number of classes that exist in \( \mathcal{Y} \). We wish then to define a classification model with parameters \( g \) which can be trained on a labelled set of tumour biopsies to learn best possible mapping between spaces \( \mathcal{X} \) and \( \mathcal{Y} \), by maximising some objective function \( f(x, y) \):

\[
g : \mathcal{X} \rightarrow \mathcal{Y}, \quad g(x) = \operatorname{argmax}_y f(x, y)
\]

The goal is that with a new, unlabelled tumour sample \( X_j \in \mathcal{X}^d \), our model will identify the correct tumour label, \( y_j \in \mathcal{Y}^c \). To consider the the dimension of the \( \mathcal{X} \) and \( \mathcal{Y} \) spaces we’ll be dealing with, let’s refer back to the TCGA dataset which is the source of somatic mutation data \( \mathcal{X} \), with corresponding tumour type labels \( \mathcal{Y} \). Figure 3.8 shows distributions of interest.
3.4. TARGET DOMAIN

The top left pane shows distribution of samples per cancer type, while the top right shows a log_{10} count of the number of somatic variants per case. The bottom left pane shows log_{10} count of variants per variant type, variant unfiltered. Bottom right shows log_{10} count of variants per gene, across all genes in the dataset.

At first inspection, one begins to appreciate the true challenge of learning from somatic mutations. Clearly, the class distribution for this problem is significantly unbalanced, with a number of classes having fewer than 200 samples to learn from. Matters are made worse by the fact that cases are characterised by as many as 10,000 variants and as few as a single variant, spread throughout 3,003,366 unique mutation indices. With only 3,594,006 mutations in total, this means that 83.6% of mutations exist entirely in isolation, with only one incidence of that mutation index in the entire dataset. Figure 3.9 shows the distribution of counts of mutation index co-occurrences. We observe that a very small percentage of
mutations occur frequently enough to meet statistical power requirements.

Figure 3.9: Distribution of mutation indices by co-occurrence within the TCGA dataset.

Assigning numerical values to the dimensions $d$ and $c$ of our feature and label spaces respectively, the naive one-hot representation of this data would be:

$$X_i \in \mathcal{X}^{3003366}, \ y_i \in \mathcal{Y}^{33}, \ i \in (1, \ldots, 10189)$$

where every feature in $\mathcal{X}$ is a possibly mutation index. The resultant feature vector for the data would be of size $3 \times 10^9$ parameters, to contain only $3 \times 10^6$ mutation values. Besides the issue of size and sparsity, if all features are orthogonal, the majority of mutations will be redundant for learning, since they share nothing in common as features. The only option is to perform a spatial aggregation of mutations along the genome. [6]
3.4.2 Gene aggregated mutation indicator

During analysis of the TCGA dataset, we observed a mapping of 22,109 genes, with each gene boundary defined according to correspondence with an observed phenotype. Taking a top down perspective, these genes could be further deconstructed into intronic and exonic regions then further still to amino acids and base pairs. While not intrinsic to the information contained by the DNA sequence, all of this hierarchical structure has meaning in correlation with observations in biology. Thus genes offer a heuristic for modular structure in the genome.

In order to reduce the cardinality of representations for mutation data, some aggregation must be performed across mutation indices. To group mutations spatially is a reasonable inductive bias because of the biological understanding of hierarchical structure - base pairs close to each other are (in general) more likely to share influence since they share a module, be that amino acid, protein, or regulatory function. Taking gene boundaries as the window for mutation grouping, we could take a single indicator $\mathbb{1}_{\text{gene}} = [0, 1]$ if a gene contains 1 or more mutations. Our representation for $X$ is then:

$$X_i \in \{0, 1\}^{|G|}, \quad G = \{g_i | \forall g_i \in \text{genome}\}$$

This greatly reduces the dimensionality of our feature set, and significantly increases the number of feature co-occurrences in the dataset. A tumour is now characterised by which genes contain somatic mutations.

3.4.3 Dataset and Preprocessing

To obtain tumour data, a subsection of data was retrieved from the TCGA BigQuery public portal, from the Somatic_Mutation table. A subsection of fields was determined to contain important information for the learning task. Each row within the table records a single mutation. project_short_name indicates the study data was collected from. Since studies were isolated to particular tumours, this acts as the label for tumour type.
case_barcode identifies the sample the mutation pertains to, and is required for grouping of data. start_position and end_position indicate the relevant location of the mutation within the genome, while hugo_symbol identifies the corresponding gene region for the mutation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>project_short_name</td>
<td>TCGA associated study. Used as tumour type label.</td>
</tr>
<tr>
<td>case_barcode</td>
<td>ID for a specific sample. Used to identify unique cases.</td>
</tr>
<tr>
<td>hugo_symbol</td>
<td>Gene identifier. Used to aggregate mutations to gene level.</td>
</tr>
<tr>
<td>start_position</td>
<td>Start index of mutation. Used to localise mutation.</td>
</tr>
<tr>
<td>end_position</td>
<td>End index of mutation. Used to localise mutation.</td>
</tr>
<tr>
<td>variant_classification</td>
<td>Class of variant (missense, silent, etc). Used for filtering.</td>
</tr>
<tr>
<td>variant_type</td>
<td>SNP, Deletion or Insertion. Used for filtering.</td>
</tr>
</tbody>
</table>

Table 3.2: Fields extracted from the TCGA data portal.

A number of filtering techniques were applied to the data, removing samples that contain insufficient signal for the purposes of experimentation. One consideration was genes which appear fewer than 100 times across the entire dataset. When aggregated as features, these would be more likely to create spurious correlations for a classifier. Additionally, the 18 variant types were reduced to 9, on the advice of Dr Nic Waddell at QIMR. This is because a number of variant types are clinically identified to have minimal phenotypic effect, such as silent mutations. The remaining variant_classification categories may then be aggregated across genes to establish gene mutation status for a given sample.

In addition, 3 aggregations were performed on pairs of tumour types in the space of labels, reducing the total number of labels to 29. Colon cancers and rectal cancers are commonly aggregated to the broader class of Colorectal cancer, while Glioblastoma and low grade Glioma are both forms of brain Glioma. Stomach and esophageal cancers are also aggregated into a single class, consistent with research indicating a continuum between histology of sub-types, and no clear boundary between the tumour types [60].

Another large filter on genes was realised to be necessary due to computational and al-
3.4. TARGET DOMAIN

gorithmic restrictions. It was recognised in early experiments that running preprocessing, model training and evaluation on the full set of genes would be infeasible for deep architectures such as CNN, for running many experiments. In favour of cross-validation for robust results, and a short enough iteration cycle to make changes if necessary, the number of genes, representing the number of features in every experiment, needed to be reduced by an order of magnitude.

Gene subset selection was completed according to two separate strategies. Firstly, to build on prior art, a dataset of cancer driver genes was sourced from a meta-study evaluating the evaluation of cancer driver genes [61]. The study evaluated 7 established methods for identifying driver genes, then introduced an extension technique known as 2020+ which uses 24 engineered features and the feature importance ranking of a random forest predicting whether the described gene is an oncogene (gene contributing to cancer) or tumour suppressor gene. This method was a top performer, and was only bested on some metrics by an existing method known as MutSigCV from the Broad Institute [8]. MutSigCV is an algorithmic approach which takes mutation tallies per gene and applies scores, then selects a subset of genes above a significance threshold as cancer driver genes.

For consistency and ease of future experiments, the top 1000 genes were selected according to both cancer driver metrics. For MutSigCV, this cutoff was at a $p$-value of 0.28, well above the threshold of statistical significance of $p < 0.05$ according to hypothesis testing, which is established with only the top 369 genes. According to the 2020+ driver metric, the $p$-value reaches statistical significance after the top 1452 samples, indicating that valuable information may be lost from contributing genes that weren’t in the top 1000. For the sake of experiment, the top 1000 genes will be sufficient to establish performance of both MutInd and extended representations.
3.5 Extended representation

3.5.1 Clustering and image manipulation

Our new representation allows the mutations found in a cancer tissue biopsy to be visualised as an image, opening up the ability to apply ML models such as the CNN. A convolutional kernel’s translation invariance property may be useful for detecting visible structure within the image. Here, the intersection of gene vectors and the specifics of which genes contain mutations can be displayed visually, and the variance of these factors observed in the images.

Figure 3.10: Subsampling, clustering and sorting operations applied to GeneVec matrix.
3.5. **EXTENDED REPRESENTATION**

To make the most of this visual structure, we perform further operations to each image in the sample set to improve learning by the CNN. The first strategy employs clustering in gene vector space, applied to the column vectors of the GeneVec matrix as a sort operation.

In Figure 3.3 we observed many small clusters in the PCA2 components, but then found that most of the mutational genes sit in a larger central cluster. Acknowledging this as a limitation, we apply clustering strategies so the objective is to recognise these groups of genes and cluster them together in the GeneVec matrix. A number of clustering algorithms were audited, including Mini-Batch K Means [56], DBSCAN [18], and Spectral Clustering [46]. All algorithms allow for a variable number of clusters to be learnt, which was desired since the number of clusters is unknown.

In addition, a 2-dimensional version of Spectral Clustering [13] was tested, in which sorting operations are allowed on both dimensions. Figure 3.11 shows results of applying different clustering techniques to rearrange matrix columns, and rows in the case of Spectral BiClustering, compared to the unsorted, randomly assigned gene locations. By qualitative assessment, all techniques seem to find bands with high variance, and group together other
regions with lower variance. With BiClustering, not only are bands visible but a spectrum can be observed within bands, and some 2-dimensional structure can be observed as patches in the lower variance regions. Worth noting is that the BiClustering technique must be applied after the GeneVec matrix is combined with the MutInd vector samples, since rearrangements are not possible when the two representations are separated. This adds trivial computational overhead however, since only a list of index rearrangements is required, which only need be computed once.

Figure 3.12: Plot thickening strategy applied on a single sample for 32 and 256-dimensional gene vectors.

To boost representational power further, an additional image processing step is tested and applied to generated GeneVec/MutInd sample images. We coin this technique "plot thickening", since the effect is to increase the width of the column vectors representing the genes that are mutated within a particular sample. Figure 3.12 displays how this technique...
amplifies the presence of mutated genes and helps highlight shared motifs for a specific sample.

This technique takes further advantage of the spatial invariance property of the CNN: while gene index is preserved, it is also duplicated into the blank space of non-mutated genes, meaning that band thickness is determined by the location of the next mutation. The effect is that samples with many mutations in a cluster will occupy similar regions of the image, but for samples with very few, sparse mutations, these bands will be magnified to occupy as much space as their neighbours allow them. This forces any resulting model not to over-fit to the presence or non-presence of genes, and make better use of the learned gene-vector representations, which have captured biological context.

### 3.5.2 Core technique

With a learned extrinsic representation of our source domain, and a curated dataset of MutInd represented samples in the target domain, we can now look at extending mutation data to incorporate these learned biological priors. With the statistically significant cancer driver genes reducing the effective dimension, we can now selectively sample from the pool of learned gene vectors to construct our new representation. Figure 3.13 is a visual overview of how learned gene vectors are aggregated into a single GeneVec matrix which forms a template from which to construct extended MutInd samples.  

---

\(^3\)The approach of intersecting gene vectors and mutations was first identified in a blog post, available via Towards Data Science [29]. To date no other literature can be found on the topic. The blog does not provide code or methods for replication of results. Attempts were made to contact the author for more experimental details but we were unable to get a response.
Listing 3.2 shows the core functions implemented to construct the extended representation. `gene_intersection` returns a list of genes which intersect the two data domains according to gene Entrez ID. While there was a risk that slight mis-spelling or typographic error could result in genes missing out, generally a maximum of 20 to 30 genes would be lost from the intersection of 1000, and manually searching for causes would be prohibitively time consuming. Once a gene subset is established, the `embed_gene_vectors` will receive two ID-mapped `pandas` DataFrames, and multiply gene mutation indices by the gene matrix dimension, according to element-wise multiplication or Hadamard product.
### 3.6. Results

With many iterations and only visualisation tools to observe qualitative phenomena, we have taken a number of steps to augment the way we represent somatic mutation data sampled from a tumour. What remains is to assign a performance metric and establish whether the steps have helped or hindered the dataset for a particular task. It is important
to note that while steps up to this point have incorporated biological hypotheses to support our task, the learned gene vectors have captured general biological priors, and are largely task agnostic. The same general representation extension could be useful for improving other tasks on somatic data. These could be in domain, such as classification across a different tumour dataset, or cross domain, on other oncological subtyping tasks such as classifying cancer by progression/stage. With that disclaimer, we detail experiment specifics for the chosen task, tumour type classification.

The objective is to classify cancer type from the labelled TCGA dataset of 29 tumour after preprocessing as described in Section 3.3.2. The classification task is used to evaluate the mutation indicator (MutInd) sample representation, across a range of ML algorithms to establish a baseline for performance. The extended GeneVec/MutInd images are then used to train and evaluate a CNN on the same tumour type prediction task. As is necessary for any effective comparison across models, cross-validation was employed with 10-folds of repetition, and variation of results recorded. Furthermore, given the significant class imbalance of the label set, stratified random sampling was employed for every round of training and cross-validation.

Figure 3.14: Classification accuracy with CV confidence interval for best models in each category.
3.6. RESULTS

Figure 3.14 gives a summary of results observed for representation and model combinations in Study 1. Bars show the mean accuracy scores observed across folds, with confidence intervals showing 2 standard deviations of variance in results. The chosen metric was classification accuracy, as is common across existing studies [59]. Sections 3.6.1 and 3.6.2 give further details.

3.6.1 Experiment 1A: Cancer classification with MutInd

Using the MutInd representation on the same subset of genes, four different model architectures were tested for classification performance on the representation.
These architectures were, logistic regression, linear stochastic gradient descent classifier, random forest, and ensembles of gradient boosted decision trees. Whilst not comprehensive to the range of learning architectures, these techniques allow us to cover the most popular parametric and non-parametric classification techniques.

After training all models with 10-fold, stratified random sampling using `sklearn`'s model selection utility, validation scores were recorded with confidence intervals around performance. For further evaluation of classification performance at a class by class level, confusion matrices were plotted. See Figure 3.15 for the confusion matrix of the top performing classifier, logistic regression.

### 3.6.2 Experiment 1B: Cancer classification with GeneVec

Since the GeneVec extended representation is a new contribution, the need for breadth of experiments was realised. A grid search was instantiated, running the same cross-validation strategy across combinations of configurations. Table 3.3 details the configurations explored.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeneVec embedding dimensions</td>
<td>[32, 128, 256]</td>
</tr>
<tr>
<td>Clustering algorithm</td>
<td>['MiniBatchKMeans', 'GaussianMixture']</td>
</tr>
<tr>
<td>Image colour palettes</td>
<td>['viridis', 'seismic']</td>
</tr>
<tr>
<td>Image normalisation</td>
<td>['non-zero', 'abs-value']</td>
</tr>
<tr>
<td>Model architecture</td>
<td>['resnet18', 'resnet34', 'resnet101']</td>
</tr>
</tbody>
</table>

Table 3.3: Parameters evaluated in grid search.

Embedding dimension and clustering strategy were discussed in Section 3.5. Image colour palette and normalisation are both post-processing steps applied once GeneVec matrices have been generated. Colour palettes determine the spectrum displayed by the images generated, and were chosen to maximise range. Normalisation was required since matrices must be mapped to the domain \([0, 1]\) to be converted to pixels. Abs-value takes
3.6. RESULTS

The absolute value of matrices before normalisation, effectively folding the space of gene vectors around the origin. Non-zero shifts all non-zero values linearly to the domain [0, 1], and leaves zero values at zero, such that the most negative value in the data set is mapped next to zero.

The representation is trained on variants of the ResNet architecture, a CNN with residual connections. The three architectures tested are well established in literature, available to be imported directly from the PyTorch model zoo. The three architectures are progressively more complex, containing 18, 34 and 101 layers respectively. Figure 3.16 shows the confusion matrix of a top performing architecture selected from the grid search.

![Confusion Matrix](Confusion_Matrix.png)
CHAPTER 3. STUDY 1: DISTRIBUTED REPRESENTATION OF GENES

When comparing this confusion matrix with Figure 3.15, we observe that mislabelled samples are much more spread across tumour types, whereas for the simpler representation and model, falsely classified samples are mostly grouped across BRCA and PRAD classes.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Test Loss</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MutInd only</td>
<td>2.23</td>
<td>49.7%</td>
</tr>
<tr>
<td>GeneVec/MutInd</td>
<td>1.94</td>
<td><strong>52.6%</strong></td>
</tr>
</tbody>
</table>

Table 3.4: Classification performance using 2 representations.

Table 3.4 gives a comparison of loss and testset accuracy for the best performing models from each experiment. This represents a 3.4% improvement over [59].
Chapter 4

Study 2: Sequential representation of DNA

4.1 Introduction

This study further investigates how biological context can be extracted from a source domain and applied to specific tasks, now through the lens of intrinsic representations. Where Study 1 characterised genes by shared co-expression in biological pathways, we now characterise regions of DNA according to the sequence contained within them. As laid out by the Central Dogma [10], all biological expression originates from the DNA sequence, via transcription to RNA, then translation to protein, resulting in the huge variety of phenotype expression seen in biology. In theory, a sufficiently complex model should be able to infer phenotype from DNA sequence alone. In the same regard, any region comprised of a sequence should also be able to summarised and compared to other sequences according to a learned representation. This representation could be considered as a computational phenotype: the ‘expression’ of a DNA sequence in the vector space shaped by a model during optimisation.

To learn a representation of sequences, we take inspiration from the large body of existing
work in Natural Language Processing. In recent years, significant progress has been made
in modelling language as sequences of tokens, defined by characters, words or sub-word
components. By training a sequence model on the task of next-token prediction only, the
model is forced to learn a representation that is task agnostic, such that for any token
or sequence of tokens, a vector is defined which characterises that sequence. Interested
in how the resolution of tokens in DNA sequences affects the representation learned, we
investigate fixed and variable length tokenisation strategies, and demonstrate how variable
length tokenisation can be used to learn motifs across the genome.

We then evaluate these representations on a classification task, to identify functional prop-
erties of DNA from sequence alone. The functional property evaluated is whether the DNA
region of interest codes for RNA that is converted to protein, messenger RNA (mRNA) or
RNA that regulates transcription and other processes in the genome, known as long non-
coding RNA (lncRNA). This binary classification task is analogous to differentiating the
ingredients in a recipe from the cooking instructions, and is a strong signal that the intrinsic
representations learned are a useful summary of sequence function.

4.2 Aims

The objectives of study 2 can be defined as follows:

1. **Data domain** Define the domain of data for learning representations. In this case
we process and parse the entire human reference genome of 3.2 billion bases into
batches for training. For task adaption, the data domain is sections of the genome
between 200 and 20,000 bases in length, each corresponding to either an mRNA or
lncRNA strand.

2. **Nucleotide tokenisation** Investigate different tokenisation schemes applied to whole
genome and task specific sequence data. Tokenisation strategies considered are 3-
mer, 5-mer and 7-mer fixed length tokenisation and variable length token vocabular-
ies of corresponding cardinality. Variable length tokens are learned from data using
4.3. DATA AND PREPROCESSING

the SentencePiece [40] algorithm. Explore distribution properties of variable length tokens, across the dataset and within specific sequences.

3. **Unsupervised sequence modelling** Train representations on the tokenised whole genome using the AWD-LSTM recurrent neural network architecture. Evaluate learning progress according to metrics of cross entropy for next token classification. Compare quality of results for fixed and variable tokenisation schemes.

4. **Supervised function classification** Adapt representations to mRNA/IncRNA classification task. Use fine tuning techniques to further improve classification accuracy. Evaluate accuracy for fixed and variable tokenisation schemes, and compare to state of the art results in literature.

4.3 Data and preprocessing

4.3.1 Whole genome data

The chosen dataset for unsupervised learning is the Homo Sapiens reference genome, assembly GRCh38.p13. The data was downloaded in *fasta* format to disk, consuming 3.3 GB uncompressed. Using the SeqIO library functionality of *BioPython*, data was read in batches of 2000 base-pairs into rows of a pandas DataFrame, resulting in 1,465,634 rows containing a total of 2.931 billion base-pairs. Due to the size in memory and challenges processing at this scale, a smaller DataFrame was parsed in chunks of 20,000 rows each for training and validation. Note that using this data only, the unsupervised model has had access to less than 3% of the genome. To train on more of the genome, more training and validation batches could be fed in during the training procedure.
4.3.2 Task specific data

For domain data, the choice is to evaluate representations on the binary sequence classification task of Long-Non-Coding RNA (lncRNA), vs Messenger RNA (mRNA). The original data is available from GENCODE Release 25 [9], which contains comprehensive annotation of long non-coding and protein coding RNA transcript sequences. The data was segmented into unsupervised fine-tuning and classification fine-tuning datasets, as outlined in a number of existing works [33] [64]. The unsupervised training set is of uncapped length, with some sequences extending to 20,000 base pairs. These were used unaltered for language model domain adaption. The classification dataset is a subset of 16,000 samples of lengths between 200 and 1000 base pairs. The distributions of these 2 datasets is visualised in Figure 4.2.

![Unsupervised Fine-Tuning](image1)

![Classification Fine-Tuning](image2)

Figure 4.1: Distribution of sequence lengths for the 2 task specific datasets.
4.4 Nucleotide Tokenisation

For training a sequence model, information must be presented according to a chosen vocabulary, or set of tokens. In the English language, the character vocabulary would be the set of numbers and letters, and their combination would make up the word vocabulary. For the genome, the set of base pairs \{A, C, T, G\} is analogous to characters. Sequences of tokens may be grouped together, into larger tokens, known as n-grams, where n is the number of alphabet components per-token. This is analogous with the terminology in genomics of k-mers, where is k is the number of base-pairs contained in the biological sub-sequence. In Natural Language processing, it is very common to use n-gram techniques for the following reasons:

- Sequence models such as RNN’s are constrained to a finite memory. LSTM’s attempt to reduce this issue but still struggle with very long sequences. Tokenisation groups elements in sequence together and reduces sequence length.

- A reduced sequence length also reduces the number of sequential steps required, enabling greater parallelism in training.

- Increasing the cardinality of the vocabulary can allow more expressivity of the data. This removes some of the burden from the sequence model to determine meaning of sub-token components.

4.4.1 Fixed Length Tokenisation

A simple tokenisation strategy is n-gram tokenisation. This strategy entails taking a window of n characters, with a chosen stride, in order to break a sequence into fixed-length tokens of length n. These tokens then represent the new sequence. For windowing the size of the token, no token overlap exists, and the sequence length will be reduced by a factor of n. The cardinality of the newly tokenised dataset, in the case of k-mers, increases from
4 (the cardinality of the set of base pairs) to

$$|V_k| \leq |V_1|^k = 4^k.$$  

This bound reaches equality if every possible sequence of $k$ 1-mers is represented in the dataset.

For the first experiment, we’re interested in comparing fixed tokenisation, as described, with variable length tokenisation, described hence, for a number of different learning tasks. A custom tokeniser was implemented to be incorporated into the FastAI library using existing API classes Tokeniser, Numericaliser and Processor [35]. Initial attempts at tokenisation involved a for loop which increments token index and segments sequences accordingly, however this requires $O(n^2)$ operations. A custom vectorised approach was established using the numpy library function ndarray.strides. Relative time performance of the two techniques for sequence lengths of exponentially increasing size is shown in Figure 4.2.

Figure 4.2: Computation time for sequential and vectorised fixed-length tokenisation strategies.
4.4. NUCLEOTIDE TOKENISATION

4.4.2 Variable length tokenisation

In 2018, Google introduced a variable-length tokenisation strategy named SentencePiece [40]. SentencePiece is language-agnostic and does not treat whitespace differently from other tokens. This makes it attractive as a strategy to transfer to the genomics domain since our data contains no whitespace and would typically cause trouble in language or alphabet specific tokenisation. SentencePiece implements 2 sub-word segmentation algorithms, byte-pair encoding [57] (BPE), and unigram language model [39]. BPE segmentation typically requires $O(n^2)$ and is improved to $O(n \log(n))$ by SentencePiece, but unigram only requires $O(n)$, linear to the size of the input data. BPE and unigram showed comparable performance on benchmarks in the paper, and so within time and computation constraints, unigram was the favoured strategy.

The unigram subword regularisation strategy assigns an independent probability to the occurrence of variable length subwords (k-mers). For some sequence of base pairs $x = (x_1, x_2, ..., x_n)$, the probability of this sequence is formulated as:

$$P(x) = \prod_{i=1}^{n} p(x_i), \quad \forall x_i \in V, \sum_{x \in V} p(x) = 1$$

where $V$ is some determined vocabulary of tokens. The most probable sequence to segment tokens is then given by:

$$x^* = \arg \max_{x \in S(X)} P(x)$$

Where $S(X)$ is a set of possible variable-length token candidates, built from the input sequence. If the vocabulary $V$ is available, and $p(x_i)$ are hidden or too hard to compute, these can be estimated using Expectation Maximisation (EM) [12] which maximises the likelihood $L$:

$$L = \sum_{s=1}^{|D|} \ln(P(X^{(s)})) = \sum_{s=1}^{|D|} \ln \left( \sum_{x \in S(X^{(s)})} P(x) \right)$$
By computing this loss for every token (k-mer) in the vocabulary, and selecting the top \( n\% \) of k-mers, a probabilistic vocabulary can be recursively constructed, which minimises Shannon Entropy, or code length, for a given sequence. This compressed representation is desirable because in the process of learning it will find common motifs within the genome, potentially much longer than the mean k-mer length. Thus the process of interpreting potential semantics within the DNA structure may be initiated. The hope is that by optimising information compression in sequences, challenges for the RNN architecture in learning can be reduced, hopefully reducing required training time or improving accuracy when compared to fixed-length tokenisation strategies.

To implement this tokenisation strategy, SentencePiece models were trained on the same data corpus as for fixed tokenisation. SentencePiece has been implemented within the FastAI API, which was convenient for experimentation. Some adaption of FastAI source code was required since training parameters for SentencePiece were hidden within helper functions and needed to be made accessible for customisation to genomic data.

In an attempt to fairly evaluate fixed vs. variable-length tokenisation schemes, 3 different vocabulary sizes were chosen for each scheme. For fixed-length tokenisation, 3-mers, 5-mers and 7-mers were generated, resulting in vocabularies with cardinality \( |V| = 64, 1024, 16328 \). To correspond with this variation, variable-length vocabularies were generated of the same respective cardinality, using the ‘vocab-size’ hyperparameter during model training. Training on a single batch of 20,000 rows containing sequences of 2,000 base-pairs each, took each SentencePiece model about 2 hours on a Google Cloud computing instance with 96 CPU cores. Visualisations of the tokenisations obtained with the resulting models are provided in Figure 4.3.
Examining the frequency distribution of tokens for each vocabulary size, this distribution appears to be approximately exponential. For all models, the single base $T$ was the most common token by an order of magnitude and was removed from the distribution for visualisation purposes. The fact that this nucleotide, the molecule Thymine, is identified in isolation, while all other single-base tokens are assigned much less prevalently, may indicate an unknown biological designation of this molecule. While no literature can be found to support this, statistically, this token holds smaller entropy than $A$, $C$ and $G$.

Other observations are that token length is distributed approximately normally with a mode length equal to the length of the corresponding fixed-length token, for instance, 5-mers are most prevalent for vocabulary size $4^5 = 1024$. In all cases, the distribution is biased towards shorter tokens, with the interesting exception of token length 16, which occurs comparatively more frequently for the two larger vocabularies. This is an artifact of maximum token length being limited at 16 for the sake of computation and suggests that a number of motifs of length $\geq 16$ were prevalent within the training set.
CHAPTER 4. STUDY 2: SEQUENTIAL REPRESENTATION OF DNA

Figure 4.4: Continuous colour map of tokenised DNA according to token length.

Figure 4.5: Continuous colour map of tokenised DNA according to token prevalence in dataset.

In order to visualise this tokenisation scheme within the DNA code, a tool was created using the Jupyter HTML widget, to provide a continuous colour mapping to a text-based visualisation of DNA sequence data. These visualisations are provided in Figure 4.4 and Figure 4.5. Two dimensions of interest were the relative length of tokens, and the relative likelihood of occurrence in the batch of sequences that was analysed.
4.5 Results

For reliable experimentation and replicable results, we choose to use the training methodology outlined in Unsupervised Language Model Fine-Tuning for Text Classification (ULMFiT) [36]. This technique uses a practically accessible model in terms of size and convergence time. Transformer models like BERT and XLNet are known to be SOTA but require models with billions of parameters for incremental improvement. The key details of the ULMFiT training scheme are:

- AWD-LSTM sequence model, with embedding size of 400, 3 layers and 1150 activations per layer.

- Training is completed in 3 phases: general domain language model (LM) pretraining, target domain LM fine-tuning and target class classifier fine-tuning. A significant contribution of the paper is to show that for natural language, fine-tuning the language model can greatly improve classification results.

- Variable dropouts across layers are applied. These are also altered across the 3 training phases.

- Discriminative fine-tuning and slanted-triangular learning rates.

4.5.1 Experiment 2A: Unsupervised sequence representation learning with recurrent neural networks

Experiment 2A details results from the domain LM fine-tuning phase of model training. For DNA sequence learning, this task is next token prediction applied to the unsupervised fine-tuning dataset, as described in Section 4.3.2. Metrics for cross-entropy loss and classification accuracy are assessed across the 3 fixed and 3 variable length tokenisation schemes. Figure 4.6 shows loss trajectories with number of training epochs for training and validation sets of fixed and variable length tokens of increasing cardinality.
For the sake of comparison of representations, early stopping wasn’t provisioned, and we see that models overfit slightly, especially for the two larger cardinality representations, where validation accuracy seems to reach a minima after 8 epochs of training, then begin to increase. The cardinality 64 tokenisation has strange convergence properties especially for the variable length tokenisation, where cross entropy has a negative spike then jumps back up and plateaus similarly to the training set. The theory here is that during stochastic sampling of batches, some batches contain particularly lower entropy due to the token length of sequences sampled. This adds complexity to the comparison of fixed and variable tokenisation, further analysis in the Discussion, section 5.3.1.

4.5.2 Experiment 2B: Task specific sequence classification

Experiment 2B is focused on the domain classification task. The classification model is trained using One Cycle scheduling [58], discriminative learning rates and gradual unfreezing of layers. The embedding learned during unsupervised learning is transferred to the classification model with the addition of a linear classification head. At the beginning of training, all embedding layers are frozen, to prevent losses from the untrained linear head from propagating through the learned language model. After the linear layer is
4.5. RESULTS

trained, embedding layers are unfrozen, and discriminative learning rates update according to $\eta^t = \eta / 2.6$ as per findings in [36].

Figure 4.7: Testset classification accuracy on mRNA/IncRNA classification task using fixed and variable length tokenisation.

Figure 4.7 shows task classification accuracy over this fine-tuning phase. In all cardinalities, representations trained on fixed length tokens perform better than their variable counterparts. This gap in classification accuracy is smallest for vocabularies of size 1024.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRU Ensemble (Hill et al.)</td>
<td>87.5%</td>
</tr>
<tr>
<td>Fixed tok (5-mer)</td>
<td><strong>90.4%</strong></td>
</tr>
<tr>
<td>Variable tok (1024)</td>
<td>87.2%</td>
</tr>
</tbody>
</table>

Table 4.1: Classification accuracy across best sequence models.

Table 4.1 summarises results this supervised binary classification task. Across best performing representations, fixed 5-mer tokenisation outperforms variable schemes, and outperforms [33] by 2.9%.
Chapter 5

Discussion

5.1 Experimental design

The nature of these studies evolved as they were carried out; as a result the scope of experimentation is broad. During experimentation, an ongoing trade-off had to be made on whether to iterate many times on a single strategy, or pursue strategies in parallel to better understand the landscape of the problem. The resultant strategy was shaped by the resources available. A number of exploratory grid searches across parameters of interest would be set up to run for hours or days on a GPU, then CPU resources were used to exploit what appeared to be the optimal strategy as new information arrived. Unfortunately, as new hypotheses developed, the breadth of possible parameters grew more and more, and this lead to a lot of time spent on engineering a system which could be used to examine every cross section of interest.

In retrospect, it was found that many of the parameters explored made little difference to the final result. As an example, the top 36 hyper-parameter combinations out of 72 only corresponded to a 1.4% change in results. An alternative approach would involve being more parsimonious with parameter choices. Working with a smaller subset of data would have resulted in faster iteration times for steps in the pipeline, and allowed for more
iterations. The risk of working this way is that the subsets used may not be representative of the entire data. The fact that there were multiple intersecting data domains for both studies further complicated things, so the choice was made to use complete data to avoid missed learning opportunities.

5.2 Limitations - Study 1

5.2.1 Gene vector learning

Study 1 covered a number of separate techniques, including Gene2vec, domain extension of somatic mutations, and cancer classification with the resultant representations. Each of these techniques had a number of pitfalls. In learning a distributed representation of genes, an existing algorithm known to perform well in natural language domains was applied, replacing words and sentences for genes and pathways. This analogy is not perfect however. The pathways obtained from MSigDB are sets, and are not ordered, as the name would imply. Where for sentences a window is passed along the sentence, and only words within a short region of each-other are used for context, the same could not be applied for genes. Instead, in the absence of distance information within pathways, the approach was to choose every combination of gene pairs within the pathway set.

The drawbacks of this strategy are twofold. Firstly, this meant that the subset of all pairs marked as positive samples (genes with shared context) is very large, at approximately one quarter of the cardinal set. Comparatively, in the NLP domain this ratio would be tiny, given the rarity of two random words co-occurring in a sentence. As mentioned, the implementation chosen requires uniform random sampling of negative samples from the cardinal set, on the assumption that drawing a positive sample is highly unlikely. This is a requirement of the algorithm, as computing the entire set of negative samples to sample from would be impossible for most NLP tasks. As a result, one quarter of the samples provided to Gene2vec that are labelled negative, will be falsely labelled as positive. Clearly this would hamper learning.
Secondly, and likely of greater concern, is that the definition of ‘context’ for defining gene pairs is likely too broad. As seen in Figure 3.1, a number of pathways were very large, and while the figure cut off the ‘tail’ of the distribution at 400, the actual distribution contained a few pathways of length over 1000. The derivation for the size of the cardinal pair set reveals that the number of pairs \( p = \frac{n(n-1)}{2} = O(n^2) \) as \( n \to \infty \). Due to this property, larger pathways have quadratically larger contribution to the training set, and very large pathways may have dominated the set entirely. This problem was realised after many experiments relying on the gene vectors learnt had been completed. Figures 3.6 and 3.7 are the results of using a greatly reduced group of pathways with shown association to oncogenic factors. This was an addendum and showed that indeed pathway differentiation is clearer than for Gene2vec models trained on every pair combination.

Given that the Gene2vec technique extends the Word2vec findings [48], there are a number of different studies that coin the term, using the same general technique but different data domains [15], [69]. In [15], the authors instead use shared Gene Ontology (GO) terms [5] to generate a set of 270,704 positive samples and 40,879,714 negative samples via gene-pairs that did not share any GO term or children GO terms. At the time of experimentation, our comparatively broader context set and much larger sample size seemed like an asset, however contains the limitations discussed above. As gene vectors were evaluated in a different context, the results of [15] are difficult to compare with our own, however qualitatively, our dimension reduced tSNE plots appear to be better defined with clearer clusters. In future work, we would further investigate the best way to generate positive pairs including evaluation of the GO term approach. Additionally, a correct negative sample set would be curated and used in place of random samples.

### 5.2.2 Domain extension of mutations

During the domain extension phase, learned gene vectors are used to extend somatic mutation (MutInd) samples from vectors to matrices, as shown in Figure 3.13. The rationale behind this technique is that gene vectors will embed pertinent gene properties, so that a region containing mutations and biological context for that region can be leveraged for
learning. Challenges in practical implementation of this technique are mostly due to the need to reduce the number of genes considered. This is a practical limitation of the feature dimension that would fit in memory for training a CNN. Furthermore, many genes may contain somatic mutations that are not causal to tumour growth. For these reasons, cancer driver genes from a metastudy of cancer driver identification techniques [61] were used, resulting matrices just under 1000 dimensions wide.

An oversight in this process was the assumption that because driver genes were used, they would be commonly implicated in MutInd. After sets of images were generated, many samples were realised to be very sparse, with only a small number of non-zero columns indicating a mutated gene. When fed as a sparse vector into simple linear model, a single parameter may be sufficient to draw a boundary between different classes. However, with the added representational power of a matrix feature set and CNN, comes the cost of a much larger optimisation space. The convolutional kernel of the CNN is an inductive bias designed to pick up local structure in 2d space, however single genes standing in isolation without the context of other mutations do not provide this. The ‘plot thickening’ strategy demonstrated in Figure 3.12 was an attempt to combat this, by replicating sparse mutation signatures across the matrix. While this serves well for visual inspection, the technique introduces its own inductive biases and makes the original index of mutations difficult to identify.

It is believed that the domain extension strategy will perform best on samples with a large number of mutated genes, where many mutations have shared contribution to tumour characteristics. This count of mutations is known as Tumour Mutational Burden, (TMB), and has recently been considered a bio-marker for Immunotherapy response [19], and survival rate in general [1]. Moreover, cases of Cancer of Unknown Primary (CUP) are shown to have high TMB, more often than average [23]. Given that the task of cancer classification may largely be beneficial for diagnosing the CUP subset of tumours, a technique which favours high TMB would be biased in the right direction. To quantify whether this is the case, future work would stratify classification performance by TMB and compare representations. To complete this, cross validation would be required so that the entire sample set could be used for metrics, as further stratifying data would otherwise lead to
data partitions too small to draw conclusions from.

5.2.3 Classification task

The classification task carried out in this study required the prediction of 29 tumour types from somatic mutation data. For this task, the MutInd representations are used to train 4 common ML architectures, with logistic regression yielding top performance. GeneVec/MutInd representations are used to train a CNN with ResNet architecture. While other factors such as sample stratification for CV is maintained, a comparison of representations performance may be confounded by classification architecture.

A number of corrections could be made to attribute performance improvements more precisely. Future work could take two identical SGD architectures and apply them to the two representations of different dimension. If wanting to use a CNN, the MutInd samples could be augmented to a control set of images that contains only binary, filled in columns for mutated genes, rather than the learned gene vectors. In the range of a few percent accuracy, even controlling architecture, results will vary with stochastic optimisation, so a distribution of results should always be considered.

5.3 Limitations - Study 2

5.3.1 Nucleotide tokenisation

For Study 2, the focus was shifted to representations learned from sequences of DNA base pairs. An important factor when considering sequence models is how tokens are defined. Increasing token length increases the cardinality of the feature space, and decreases the length of sequences, posing an interesting trade-off for sequence learning. Fixed token lengths of 3, 5 and 7 were evaluated, cardinalities of $4^3, 4^5, 4^7$ respectively, then these cardinalities were used as the vocabulary size hyperparameters for training variable length
A strategy that was investigated early in experimentation but not pursued was striding of tokens. In our implementation, consecutive tokens for fixed and variable vocabularies have no overlap with one another, meaning that the stride, or distance from start character of one token to the next, would be the size of that token. In other implementations, such as [31], the authors perform fixed length tokenisation and use strides smaller than the token length, to allow for overlap of tokens. This introduces a prior to the task of next token prediction, since some content of the token will be in the next token. In the extreme case of strides of size 1, regardless of k-mer length, there are only 4 possible k-mers that can come next. The authors of [31] discuss how using strides can speed up learning, but results in models quickly converging to an ‘informed random’ loss, which is just the entropy of random guesses in the space of next tokens given the prior of the previous token.

The main novelty of this study was in introducing variable tokenisation to genomics data and comparing to fixed length approaches. We realised including stride in experiments would preclude fair comparison based on cross entropy loss, according to the discussed above. Token cardinality was standardised between variable and fixed tokenisation for the exact purpose of ensuring trustworthy comparison for next token prediction. Re-examining the distributions in Figure 4.3, specifically for token length (bottom row), we realise that while distribution modes are equivalent to their fixed length counterparts, they don’t correspond to mean token length. This is most clear for the \( |V| = 64 \) distribution, which includes many single character tokens, and mean token length of 2.4, and not 3.

Reflecting on the the loss curves for variable tokenisation seen in Figure 4.6, this uneven token distribution explains a lot. First consider fixed length tokenisation, of 3-mers, vocab size 64. The cross-entropy will be the sum of negative likelihood for each token, and assuming the model starts with a uniform prediction distribution across classes:

\[
\text{Loss} = -\sum_{i=1}^{|V|} y_{o,c} \ln(p_{o,c}) = -\ln\left(\frac{1}{4^3}\right) = \ln\left(4^3\right) = 4.16
\]

This corresponds exactly with the initial training loss observed for fixed 3-mer tokenisation
in Figure 4.6. Comparing this with variable tokenisation, found to have mean token length 2.4, a uniform distribution results in cross-entropy \( \ln(4^{2.4}) = 3.33 \), also approximately consistent with the starting points (and plateau points) of loss for variable tokenisation shown in the figure. For higher cardinality token sets, the starting loss difference between fixed and variable is much smaller, corresponding to smaller mean token length and hence mean entropy.

5.3.2 Sequence classification

A surprising result when evaluating classification performance was to see fixed tokenisation outperform variable for all vocabulary sizes. Low performance shows some correlation with issues during language model learning in the previous phase. For example, where validation loss plateaued for variable 3-mers, we see the poorest classification accuracy of all experiments, dropping as low as 60%. This is a testament to the importance of a good learned representation - with the same linear models and training regime, the weak embedding results in a 15% gap in accuracy.

A plausible explanation for this performance difference may relate to data choices during representation learning. Due to computational constraints, both SentencePiece tokenisation and unsupervised language model training were completed on only 20,000 rows or 40 million base pairs in the genome, accounting for only about 1.3% of the genome, constrained to a specific region. While sequence model training had an additional domain transfer phase with an unsupervised mRNA/IncRNA context set, SentencePiece did not.

Future work would ideally gather more computational resources and train models on larger regions of the genome, or the entire genome. Given that current NLP models continue to improve with billions of samples, a larger intersection of the genome could allow SentencePiece to exhibit similar performance improvements to those found for language [40]. To continue experiments on a small region of the genome, SentencePiece would likely perform better on task specific metrics when run on a domain specific data subset.
5.4 Summary

This project presented two studies, with the objective of demonstrating the value of representations to decode the complex hierarchical structure of the genome. A distributed representation of genes according to co-expression in biological pathways was found to cluster cancer pathways in a learned embedding space, and support empirically established interactions within pathways. Combining this gene representation with somatic mutation signature of tumour biopsies across 29 cancer types shows an improvement in classification accuracy of 2.9% over the best somatic only representation, and 3.2% over the best performing comparable study [59].

A sequential representation of DNA sequences was learned through next token prediction across the human reference genome. The token size in k-mers was found to have an impact on learning trajectories, as was the choice of fixed or variable length tokens. Distributions of learned variable length tokens allows for visualisation of entropy distribution across regions of the genome. Motifs found by this strategy are consistent with findings of transcription factors in literature. Sequential representations are fine-tuned to a sequence function dataset, and evaluated on the functional classification task of lncRNA/mRNA classification. Results are positive, with 5-mer fixed tokenisation attaining classification accuracy of 90.4%, a 2.9% improvement over an existing study [33].
Bibliography


