**DEPARTMENT OF COMPUTER APPLICATIONS**

**COCHIN UNIVERSITY OF SCIENCE AND**

**TECHNOLOGY COCHIN-22**

**M.Sc COMPUTER SCIENCE WITH**

**SPECIALIZATION IN DATA SCIENCE**

MINI PROJECT ON

# VARIANT CALLING IN GENOMIC SEQUENCING DATA USING DEEP LEARNING

III SEMESTER

NOVEMBER 2023

**DEPARTMENT OF COMPUTER APPLICATIONS**
**COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY COCHIN-22**

# CERTIFICATE

This is to certify that the mini project entitled "Variant calling in genomic Sequencing data using Deep learning" is a bonafide record of the project work done by, ABIJITH T K (Reg No:- 35922002), DEVIKA MOHAN (Reg No:- 35922013 ), AJITH PRASAD (Reg No:- 35922027), and SHIBU C (Reg No:- 35922031) under our supervision and guidance in partial fulfillment of the award of Degree of Masters of Science in Computer Science during the year 2022-24 at Department of Computer Applications, Cochin University of Science and Technology, Cochin.

Internal Examiner

Head of the Department

Department of Computer Applications
CUSAT

Department of Computer Applications
CUSAT

## ACKNOWLEDGEMENT

With great pleasure, we hereby acknowledge that the help given by various individuals throughout the project itself is an acknowledgment to the inspiration-driven and technical assistance contributed by many individuals. We convey our reverential salutation to **Almighty God**, for enabling us to take up and complete the mini-project successfully.

We are pleased with our indebtedness to **Dr. M.V. JUDY**, Head of the Department, Department of Computer Application, CUSAT for her gracious encouragement. We also take this opportunity to express my heartfelt gratitude and thanks to our class coordinator **Dr. Sabu M.K**. for the support and guidance. We are also thankful to **Dr.ARUN K S**, our project guide.

Also, we are obliged to the teaching staff for being helpful and cooperative during the period of the project. We extend our heartfelt thanks to parents, friends, and well wishes for their support and timely help.

## ABSTRACT

In the era of genomic sequencing accurate detection of genetic variants is essential for understanding the basis of genetic diseases. This mini project presents a novel and simple approach that harnesses the power of deep learning to improve variant calling from sequencing data. The goal of our project is to identify and predict diseases using the strurtural variations in the genome sequence reads. Structural variations refer to genomic variations, such as deletion, insertion, single nucleotide variation, and inversion. As the first step we convert the genome reads into a matrix using tools like Matplotlib, Numpy, Tensorflow, and Scikit-learn. And we use that matrix to identify the variants. We use a simple Convolutional Neural Network to train our model. The images we created from the matrix are used as the inputs for the model, and it is used for training. We design the network output to call a candidate a “heterozygous variant”, “homozygous variant”, “non-variant”, or “complex-variant”. Besides classifying the variant type of the candidate sites, the network is also trained to predict the possible variant bases with minimal tuning.

## TABLE OF CONTENTS

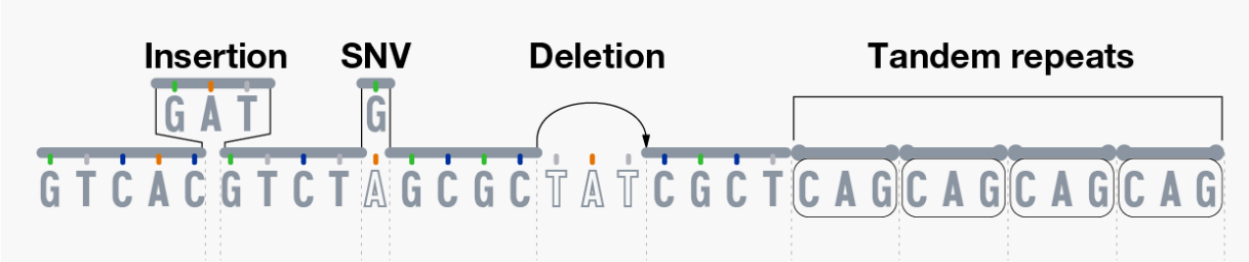
# 1. INTRODUCTION

Genome sequencing is a process that determines the order of nucleotides in a DNA molecule, which can reveal information about the genetic makeup of an organism. Variants are changes in the DNA sequence that can affect the function of genes and proteins and may be associated with diseases or traits. Variant calling serves as an important component within genomic analysis, enabling the identification of genetic alterations, mutations, and structural changes within an individual's DNA sequence.

This project uses a convolutional neural network (CNN) to extract features from the genome sequencing data, and a fully connected layer to perform the classification task. Network is trained and tested on simulated and real datasets, and achieves high accuracy and efficiency compared to existing methods.

Variant calling allows researchers to identify and catalog genetic variations within populations. Genetic variants are often associated with diseases. Identifying variants related to diseases helps researchers identify the genetic basis of disorders, paving the way for targeted therapies and precision medicine.

Different types of variation are




- Insertion variation: A type of genomic variation that involves the addition of one or more nucleotides to a DNA sequence.
- Deletion variation: Refers to the removal or loss of one or more nucleotides from a DNA sequence.
- Simple Nucleotide Variant: Also known as Single Nucleotide Polymorphism (SNP), refers to a type of genetic variation that involves the substitution of a single nucleotide (A, T, C, or G) at a specific position within the DNA

sequence. SNVs are among the most common types of genetic variations in the human genome.

- Tandem variation: Refers to a type of genomic variation where a specific sequence of nucleotides is repeated consecutively within a DNA strand. These repetitive sequences, known as tandem repeats or tandem elements, can vary in length and are typically found adjacent to each other.

We can classify the variations into other types, Homozygous, Heterozygous, Non-Variant, and Complex variants.

- Homozygous variant: Refers to genomic locus where an individual has identical alleles on both homologous chromosomes. For example, if the variant site has two copies of the same allele (AA or GG), it is considered homozygous.

- Heterozygous Variant: Occurs when an individual has different alleles at a specific genomic locus on the homologous chromosomes. For example, if the variant site has one copy of each allele (AG or CT), it is considered heterozygous.

- Non-Variant: A non-variant (or wild-type) refers to a genomic locus where there is no observed variation from the reference genome. This could be represented as the reference allele (e.g., if the reference is A, then AA or TT).

- Complex Variant: More intricate variations, such as insertions, deletions, or rearrangements, that go beyond a simple single nucleotide change. This might involve changes in the length or structure of the sequence at the variant site.

Variants in the genome may manifest as homozygous or heterozygous states, as well as exhibit complexities such as complex insertions, deletions, and single nucleotide variations (SNVs). In our project we are identifying these types of variations.

# 2. LITERATURE SURVEY

Poplin et al. [1] introduce a deep learning approach for genomic variant calling named “DeepVariant: Highly Accurate Genomic Variant Calling Using Deep Neural Networks”. It utilizes a convolutional neural network (CNN) to analyze aligned reads and predict genomic variants. The model is trained on a dataset with known variants, enabling it to learn complex patterns associated with variant calling. The model takes aligned reads as input, with a focus on understanding the error patterns associated with variant calling. DeepVariant's architecture enables it to make highly accurate variant calls, and it has been shown to outperform traditional variant callers in terms of precision and recall.

Bože, et al [2]. introduces “DeepSV: accurate calling of genomic deletions from high-throughput sequencing data using deep convolutional neural networks” - a novel approach employing deep convolutional neural networks (CNNs) for precise calling of genomic deletions from high-throughput sequencing (HTS) data. DeepSV leverages a deep learning architecture to automatically learn complex patterns within aligned reads, capturing informative features associated with deletions. The model is trained on a diverse dataset containing samples with known deletions, enabling it to generalize and make accurate predictions on new sequencing data.

Alzahrani, et al.[3] introduced a deep learning-based variant caller named “DeepVCF: A Deep Learning-Based Variant Caller for RNA-Seq Data” that is specifically designed for RNA-seq data. It uses a CNN to classify each RNA-seq read as either a reference allele or an alternate allele. DeepVCF is able to achieve high accuracy in calling SNVs and indels in RNA-seq data, and it is also able to call variants in low-coverage regions of the genome.

Dilthey A, et al,[4] is a deep learning-based variant caller named “DeNovoCNN: A Deep Learning Approach to De Novo Variant Calling in Next Generation Sequencing Data” that is specifically designed for de novo variant calling. It uses a CNN to classify each sequencing read as either a de novo mutation (DNM) or an inherited variant (IV). DeNovoCNN is able to achieve high accuracy in calling

DNMs in next-generation sequencing (NGS) data, and it is also able to call DNMs in low-coverage regions of the genome.

Luo, et al.[5] proposed a method for using population data to improve variant calling using deep learning named “Improving Variant Calling Using Population Data and Deep Learning”. The method uses a CNN to classify each sequencing read as either a reference allele, an alternate allele, or a population allele. The population allele is the allele that is most common in the population from which the sample was drawn. The method is able to achieve high accuracy in calling variants in both single samples and cohorts.

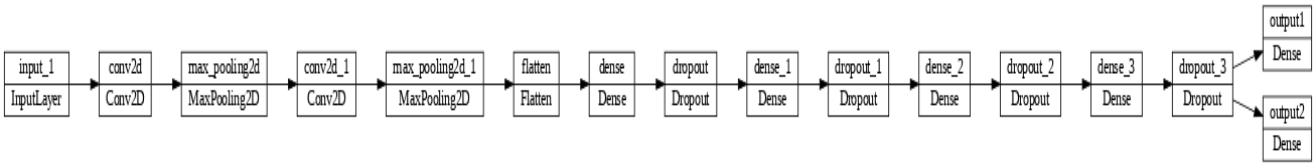
Each approach employs convolutional neural networks (CNNs) to scrutinize aligned reads and forecast diverse genomic variants. DeepVariant prioritizes comprehensive accuracy in variant calling, with DeepSV focusing on genomic deletions. Tailored for RNA-seq data, DeepVCF attains precision in calling SNVs and indels, while DeNovoCNN is crafted for de novo variant calling. Luo et al. introduce a method leveraging population data to heighten variant calling accuracy through deep learning. These methodologies underscore the adaptability of deep learning in refining the accuracy of genomic variant calling across diverse applications and data modalities.

# 3. DESIGN AND IMPLEMENTATION

## 3.1 Design

### 3.1.1 System Architecture Design

We used a Convolutional Neural Network for the classification of variants. A CNN consists of Convolutional layers, Pooling layers, and Fully connected or Dense layers.
Our network consists of 2 Convolutional layers, and 4 Fully Connected layers. The activation functions we used are elu and softmax.




Our model is a simple neural network for calling heterozygous or homozygous variants from alignments of single molecule reads to a reference genome. The system design and architecture of the model are as follows:

- **Input**: The model takes as input a BAM file containing the alignments of reads to a reference genome. For each candidate variant site, the model extracts three 15 by 4 matrices from the alignments, encoding the expected reference sequence, the observed bases before the site, and the observed bases after the site. The matrices use one-hot-like encoding for the four possible bases (A, C, G, T) and also include the number of reads that aligned to each position.
- **Output**: The model outputs two groups of values for each candidate variant site. The first group contains four values that indicate the possible base of the site, using a sigmoid activation function. For example, if the site has a base “C”, the model outputs [0, 1, 0, 0]. If the site has a heterozygous variant, such as “A/G”, the model outputs [0.5, 0, 0.5, 0]. The second group contains four values that indicate the variant type, using a softmax activation

function. The four possible types are homozygous reference, heterozygous, homozygous alternative, and no-call.

- **Network structure**: The model consists of two convolutional layers, each followed by a max-pooling layer, and three fully connected layers. The convolutional layers use filters of size 2 by 4 and 3 by 4, respectively, and apply an exponential linear unit (ELU) activation function. The max-pooling layers use pool sizes of 7 by 1 and 3 by 1, respectively, and do not mix different bases. The fully connected layers use 48 units each and also apply an ELU activation function. The model uses dropout layers with a rate of 0.5 between the fully connected layers to prevent overfitting. The model uses mean square loss for the first group of output values and cross-entropy loss for the second group of output values.

### 3.2 Implementation

The implementation process of our model involves data preprocessing, input encoding, CNN-based feature extraction, classification, and variant calling.

Firstly, the input sequencing data is preprocessed to remove errors and prepare it for the CNN. This includes trimming reads, converting them to base pairs, and aligning them to the reference genome. Subsequently, the aligned reads are encoded into a numerical matrix tensor of shape 15*4*3 that the CNN can understand.

The encoded reads are then passed through the CNN, which extracts features from the data. The CNN's convolutional layers apply filters to the input data to extract features, while the pooling layers reduce the size of the representation by summarizing the features from the convolutional layers. We design the network output to call a candidate as a “heterozygous variant”, “homozygous variant”, “non-variant”, or “complex-variant”. Besides classifying the variant type of the candidate sites, it is also trained to predict the possible variant bases.

## 4. DATA COLLECTION AND PREPROCESSING

The data collection and preprocessing steps involved in Our model are:

- Data collection:

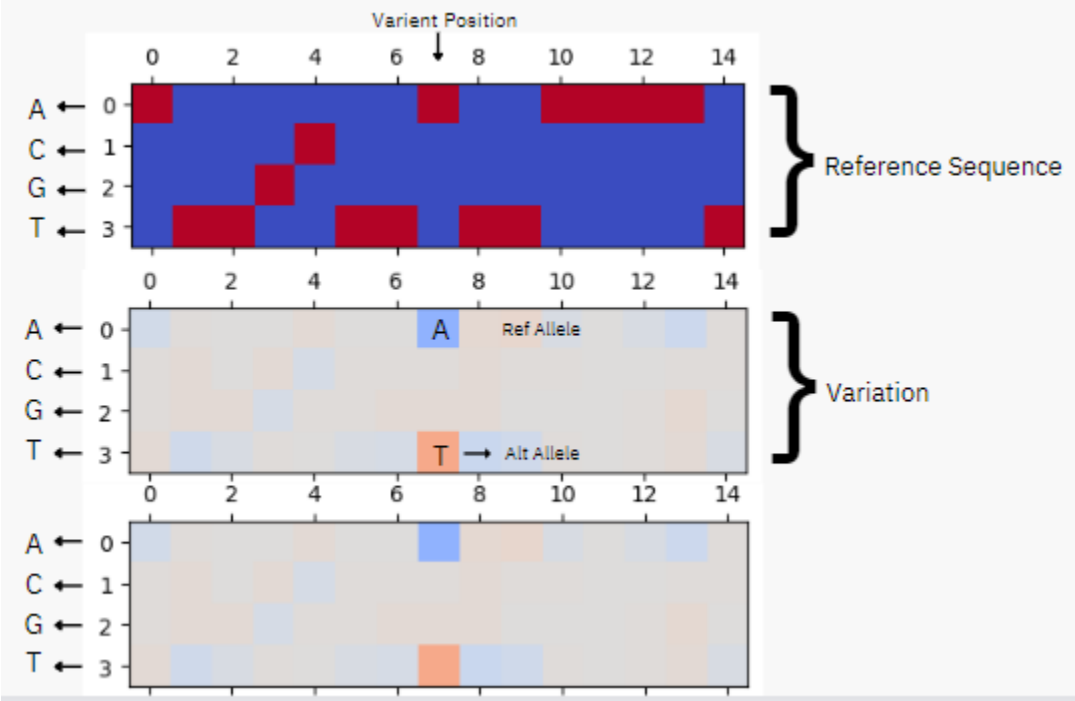
Our model requires two types of data as input: a reference genome and a set of aligned reads. The reference genome is a sequence of nucleotides (A, C, G, T) that represents the consensus of a species or population. The aligned reads are short fragments of DNA that are mapped to the reference genome. The aligned reads contain information about the genomic variants, such as single nucleotide polymorphisms (SNPs), insertions, deletions, and structural variations.

- Preprocessing: Our model performs several preprocessing steps to transform the raw data into a suitable format for the deep learning model.

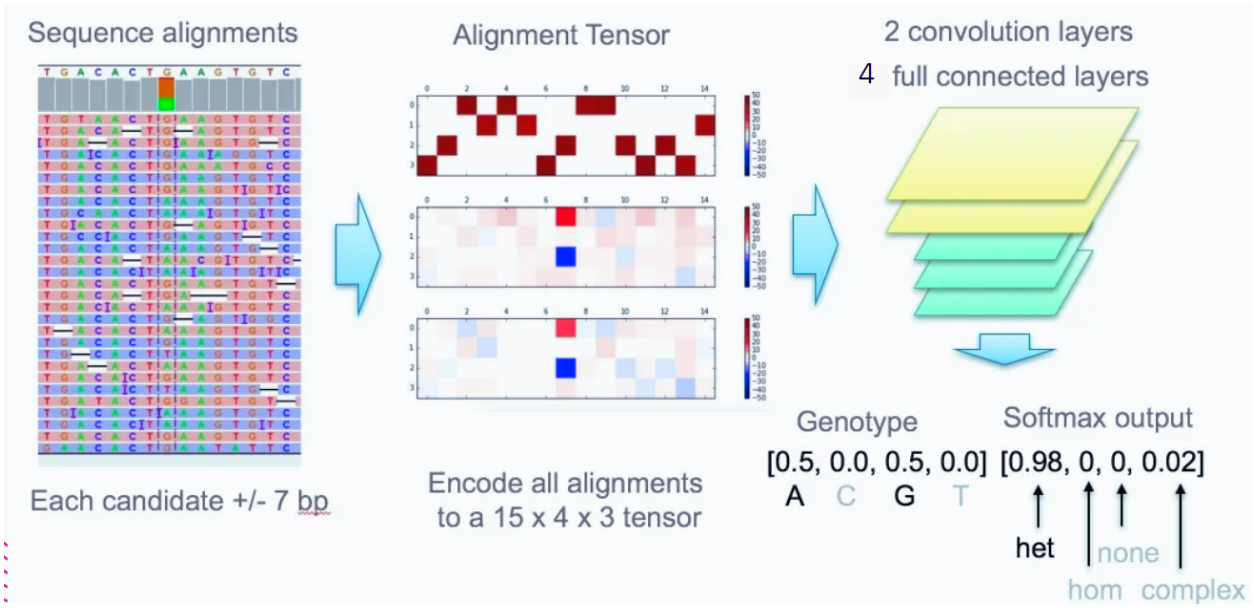
For this, we summarized the alignment information into three 15 x 4 matrices. We first do some simple statistics to build a list of variant candidates. For each candidate, we add 7 base flanking bases on both sides. Including the variant candidate site, we consider 15 bases in the reference at once.

For each position, we use one-hot-like code to generate the matrix. Namely, for each position, we track 4 counting numbers for each base of “A”/”C”/”G” and “T”. One matrix tracks the baseline encoding the reference sequence and the number of supported reads through a position. Two other 15 by 4 matrices track the differences between the sequencing data and the references. One matrix is a design to track the difference from all bases from the sequence data and the other one focus on missing bases on the reference.

The preprocessed image will look like this



# 5. METHODOLOGY




The sequence reads are aligned to identify the variant position. And the variant is converted to three matrices having shape 15 x 4 where the 15 represents the position.

# 6. EXPERIMENTAL RESULTS AND ANALYSIS

## Loading the dataset

Loading the dataset

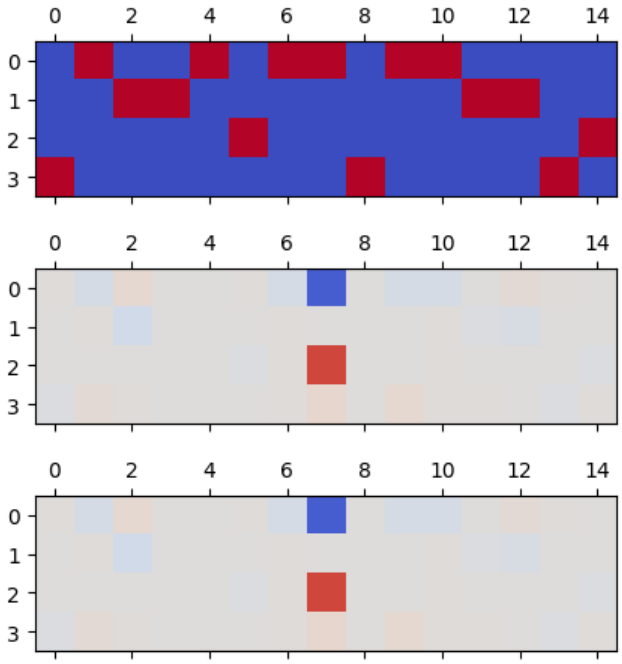
```
[ ]  Xarray, Yarray, pos_array = get_training_array("VariantNet_wd/wd/aln_tensor_chr21",
                                "VariantNet_wd/wd/variants_chr21",
                                "VariantNet_testing_data/testing_data/chr21/CHROM21_v.3.3.2_highconf_noinconsistent.bed" )
```

```
[ ]  print("Shape of Xarray : ",Xarray.shape)
     print("Shape of Yarray : " ,Yarray.shape)
     print("Length of position array : ",len(pos_array))
```

```
Shape of Xarray :  (53537, 15, 4, 3)
Shape of Yarray :  (53537, 8)
Length of position array :  53537
```

```
i = 200
plt.figure(figsize=(5, 2))
plt.matshow(Xarray[i,:,:,0].transpose(), vmin=0, vmax=50, cmap="coolwarm", fignum=0)
plt.figure(figsize=(5, 2))
plt.matshow(Xarray[i,:,:,1].transpose(), vmin=-50, vmax=50, cmap="coolwarm", fignum=0)
plt.figure(figsize=(5, 2))
plt.matshow(Xarray[i,:,:,2].transpose(), vmin=-50, vmax=50, cmap="coolwarm", fignum=0)
```

```
<matplotlib.image.AxesImage at 0x7f1ac4b3e860>
```



## Creating the model

```
[ ] from tensorflow.keras.layers import Input, Dense, Conv2D, MaxPooling2D, Flatten, Dropout
    from tensorflow.keras.models import Model
```

```
inputs = Input(shape=(15,4,3))

conv1 = Conv2D(filters=64, kernel_size=(2,4),padding="same",activation="elu")(inputs)
pool1 = MaxPooling2D(pool_size=(7,1),strides=1)(conv1)

conv2 = Conv2D(filters=56, kernel_size=(3,4),padding="same",activation="elu")(pool1)
pool2 = MaxPooling2D(pool_size=(3,1),strides=1)(conv2)

flatten = Flatten()(pool2)

dense1 = Dense(128,activation="elu")(flatten)
drop1 = Dropout(0.50)(dense1)

dense2 = Dense(64,activation="elu")(drop1)
drop2 = Dropout(0.50)(dense2)

dense3 = Dense(32,activation="elu")(drop2)
drop3 = Dropout(0.50)(dense3)

dense4 = Dense(16,activation="elu")(drop3)
drop4 = Dropout(0.50)(dense4)

output1 = Dense(4,activation = "softmax",name="output1")(drop4)
output2 = Dense(4,activation = "elu",name="output2")(drop4)
```

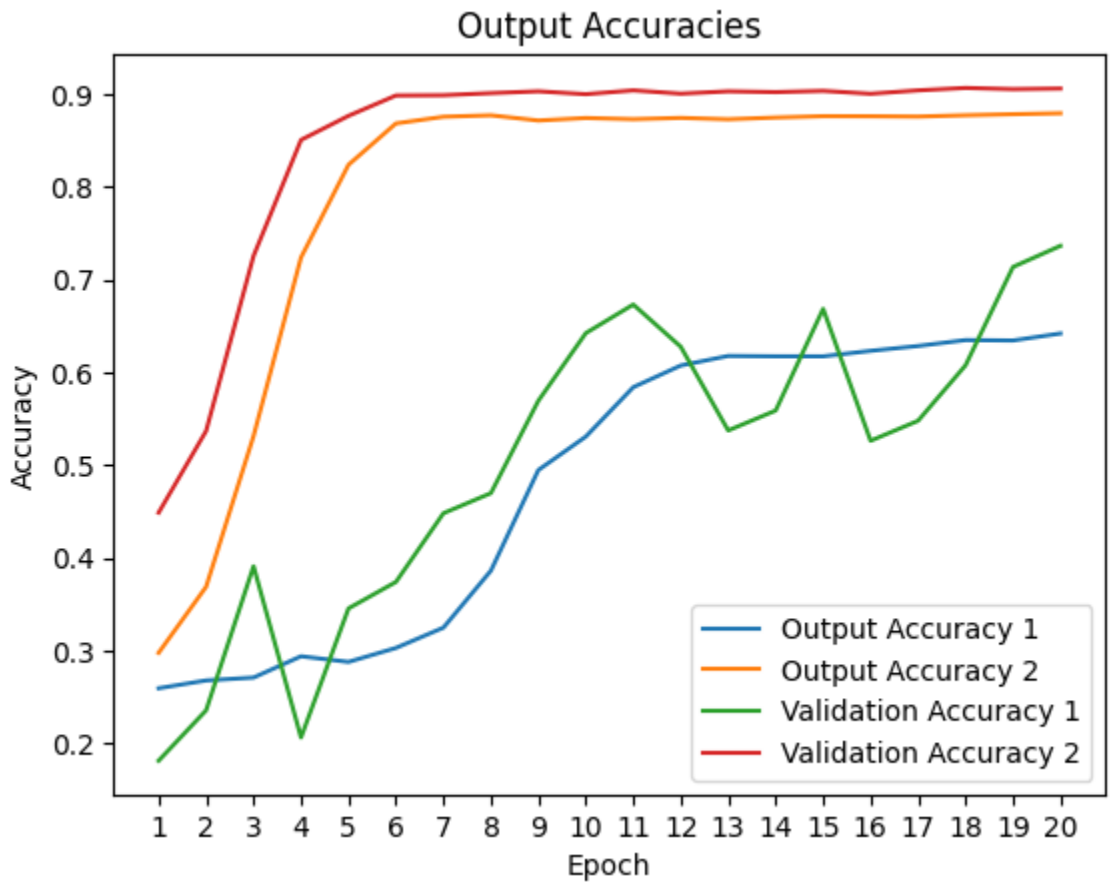
```
[ ] model = Model(inputs=inputs, outputs=[output1, output2])

    model.summary()
```

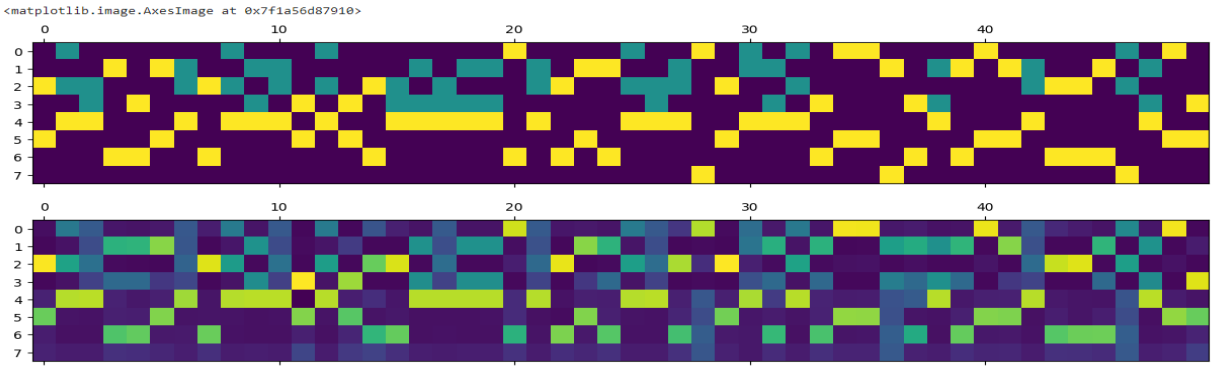
## Training the model

```
history = model.fit(x_train,{"output1":Ytrain_C,"output2":Ytrain_V},epochs=20,validation_data=(x_val,{"output1":Yval_C,"output2":Yval_V}))
```

## Plotting the graph




## Predictions



## Classification Report

```
from sklearn.metrics import classification_report

class_report = classification_report(Yarray2_V.argmax(axis=1),t.argmax(axis=1))
print(class_report)
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96     21809
           1       0.94      0.99      0.97      9691
           2       0.83      0.98      0.90     12609
           3       0.00      0.00      0.00      3125

    accuracy                           0.91     47234
   macro avg       0.68      0.74      0.71     47234
weighted avg       0.86      0.91      0.88     47234
```

# 7. CONCLUSION AND FUTURE WORKS

## 7.1 Conclusion

Our project addresses the challenge of classifying homozygous, heterozygous, variant, and complex variant genotypes in human genetics. Leveraging deep learning and alignment tensors, Our model outperforms traditional methods, particularly in discerning complex variants and low-mappability regions. Its enhanced accuracy results from learning intricate patterns in alignment data, distinguishing true variants from sequencing errors.

Our model demonstrates versatility in handling various variant types, including SNPs, indels, and multi-allelic variants. Its scalability enables efficient processing of large sequencing datasets, suitable for extensive genomic studies. The model's interpretability offers insights into genetic mechanisms related to complex traits and diseases.

In conclusion, Our model is a powerful, accurate, versatile, scalable, and interpretable tool for genotype classification. Its applications span genomic research and clinical settings, with potential in precision medicine for personalized treatment strategies based on individual genetic profiles.

## 7.2 Future Scope

1. Expanding Variant Types: Our model's future lies in expanding its scope to encompass a broader spectrum of genetic variations. This includes handling structural variants, copy number variations, and epigenetic modifications. This would provide a more comprehensive understanding of the genetic landscape.

2. Integrating Multimodal Data: Our model could integrate with other data sources, such as transcriptome and proteome data. This would create a more holistic view of genetic variants and their impact on biological processes. It would facilitate a deeper understanding of gene-environment interactions and disease mechanisms.

3. Enhancing Interpretability: Enhancing the interpretability of Our model would enable researchers to gain a deeper understanding of the specific patterns and features that drive variant predictions. This would facilitate the development of more accurate and reliable variant calling tools.

4. Personalized Medicine: Our model could inform the development of more effective personalized treatment plans with fewer side effects. It could also identify individuals at risk of developing certain diseases, enabling preventive measures and early interventions.

## 8. REFERENCES


1. “DeepVariant: Highly Accurate Genomic Variant Calling Using Deep Neural Networks” - Poplin, R., Ruano-Rubio, V., DePristo, M.A., et al. Genome Research, 2018.

2. “DeepSV: accurate calling of genomic deletions from high-throughput sequencing data using deep convolutional neural networks” - Boža, V., Brejová, B., Vinař, T., et al Journal: PLoS ONE, 2019.

3. “DeepVCF: A Deep Learning-Based Variant Caller for RNA-Seq Data” Alzahrani, A., et al. (2018), Nucleic acids research, 46(11), e61.

4. “DeNovoCNN: A Deep Learning Approach to De Novo Variant Calling in Next Generation Sequencing Data” Dilthey, A., et al. (2019), Genome Biology, 20(1), 101.

5. “Improving Variant Calling Using Population Data and Deep Learning” Luo, R., et al. (2020), Nature Genetics, 52(4), 423-433

6. “Structural variant calling: the long and the short of it” Medhat Mahmoud, Nastassia Gobet, Diana Ivette Cruz-Dávalos, Ninon Mounier, Christophe Dessimoz & Fritz J. Sedlazeck , 20 November 2019

7. National Center for Biotechnology Information company: https://www.ncbi.nlm.nih.gov/

8. 1000 genomes project: https://www.genome.gov/27528684/1000-genomes-project