How Effective is Machine Learning in Classifying COVID-19 Genomics Sequence Data Transformed for Image Recognition?

Yael Lyshkow

Contents

1. Introduction

The role of Artificial Intelligence(AI) in the computer science field is rapidly growing, with image classification being a significant part of this expansion. The field of computer vision, a subset of Artificial Intelligence typically does not apply image recognition to data not typically presented in an image format. This investigation is researching this concept applied to textual nucleic acid sequences converted into digital images for image recognition.

This paper seeks to investigate how effective machine learning and computer vision are in classifying images created from COVID-19 genomic sequences, with each image created from a single sequence. If this process can be shown to be effective in classifying COVID sequences, it could help in fast diagnosis of genetic diseases or rapid detection of increasingly lethal mutations in COVID-19 variants

This concept has previously been applied to topics such as malware classification[1], wherein malware binaries, essentially malware text files, were converted into grayscale images and passed through a convolutional neural network (a deep learning architecture) for classification, a strategy proposed by Nataraj et al[2]. Through this, they managed to classify the parent family of malwares that had been mutated, allowing for easier development and application of anti-malware software. Along similar lines, the use of machine learning to analyse non-image data is seen in work done by Justin Salamon and Juan Pablo Bello[11] . Salamon and Bello investigated classification of environmental sounds in the form of images of? spectrograms (a visual way of representing the "loudness" of a sound over time). They used convolutional neural networks, employing 8732 labelled sound clips converted into images to train a model that could accurately classify typical environmental sounds. Their trained model was capable of classifying recorded sounds with labels such as drilling, gun shots, sirens, or street music, which brought to their attention the possibility of using this algorithm for crime detection.

Similar to the classification of malware families in Nataraj et al, the intent of this paper is to investigate whether image representations of COVID-19 sequences can be created and classified successfully, or if Machine Vision cannot be used applied to this non-image type data.

As applications of computer vision are more frequently being applied to diagnosing health issues, a method of converting genomic data into a visual format easily recognisable by machine learning algorithms could have many benefits even beyond COVID sequences. For example, it is conceivable that specific cardiac or neurological disorders like stroke, depression or Alzheimer's Disease could be diagnosed through classification of ECG data converted into images.

To investigate the ability of machine learning algorithms to classify COVID genomics data lineages identified as variants of concern (VOC) (see appendix 6.5), multiple python programs were written to prepare COVID sequences, transform them into images used for training, validate and test novel images using the created AI model. The COVID-19 genomic data used to do this was released to the public by the COVID-19 Genomics UK Consortium (COG-UK), a group dedicated to generating SARS-CoV-2 sequences and analysis in order to allow further research into the topic. With over almost 900,000 sequences available for conversion into images, this is an excellent source of data for training AI models, which require large amounts of data to gain accuracy.

2. Background

Image recognition and computer vision is a field becoming steadily more popular and growing in potential in recent years. Computer vision is defined as the ability of computers to interpret and understand the visual world[3] and find meaning in visual data such as images and videos. Image recognition is the task of identifying images and categorizing them into predefined distinct classes[4]. As it's growing in popularity, image recognition algorithms have been applied to countless ~~things~~ faucets of modern life including security, product identity, medical diagnosis, etc. Additionally, image recognition is often used to perform tasks such as identifying people in data banks of pictures or in the context of self-driving cars, identifying traffic lights and their status. Large companies such as Facebook and Google are devoting much time and money into developing their computer vision capabilities[5] and extensive global research is being put into developing computer vision algorithms.

2.1 Machine Learning

Classically, computer programs require step-by-step instructions to be written in order for the program to be executed correctly. Although this approach is useful for solving many problems it is likely to be inadequate when given a task whose solution does not conform to specified programmatic steps. When classifying images, the breadth of variation in colours, stylistic details and composition exceeds the capability of most sequential programs, which leads to the employment of machine learning.

Machine learning, a branch of artificial intelligence, and the umbrella term in which computer vision falls under, is the ability of a computer to learn from data, identify patterns and make decision from it[6]. To develop computer vision algorithms further, large amounts of data must be fed into algorithms to allow the computer to locate patterns, creating models, and essentially ‘learning’ from annotated data, hence machine *learning*. The main goal driving machine learning is the recently uncovered ability for computers to perform tasks they haven’t been previously programmed to do.

Computer Vision

The main concept of computer vision is teaching computers to process images at the pixel level and gain understanding from patterns within the images coupled with a label defining each image[7]. The most common tasks computer vision is used for are object classification and identification. Object classification, also called segmentation, parses data and classifies an object into a pre-defined category, e.g. finding a person in an image. Object identification does the same, but instead searches for a *particular* object in an image, e.g. finding a specific person in an image of many.

Image recognition systems function by detecting salient image features, which are sections of the image that contain the most information about the object. This is done by isolating the most informative portions of the data and ignoring the features that are of less interest through training with different images with the same ground truth label. By passing large amounts of data paired with labels through a machine learning algorithm, the model is trained to learn the difference between classes[8] while learning how to identify important features of each class. COVID-19 data was an optimal choice for this investigation due to the large amount of accurately labelled data available. This Image recognition process works by extracting pixel features from an image, wherein each pixel is represented by a single (8 bit) number for greyscale images, or multiple bit set of numbers (up to 24 bits) for coloured images. The range of those numbers is known as the colour depth[9]. A large number of features, or characteristics are derived from the image, and the label from the image coupled with the features extracted are used to train a model.

2.2 Deep Learning

Most recent developments in computer vision stem from the use of deep learning, a subset of machine learning which utilises a specific type of algorithm called a neural network. Similar to other machine learning algorithms, neural networks extract features from large amounts of data, but neural networks are unique in that there they were inspired by the structure of the human brain, specifically, the connections between the neurons in the cerebral cortex[7]. One of the more prominent vendors of neural network hardware and software is NVIDIA. Their main hardware products are Graphic Processing Units (GPU), and one NVIDIA Graphic Processing Unit (GPU) chip contains thousands of computer processors, each able to emulate a neuron. A neural network is typically split into three layers (see Figure 1) : the input layer, the hidden layer(s), and the output layer. The layer that receives external data is the input layer, and the layer that returns the final result is the output layer, leaving the hidden layer or layers in between. Through training, the middle layer(s) are trained to identify features of images or other inputs by teaching itself with a large, labelled data set, leaving it with the ability to identify new data given to it. This type of network is called a convolutional neural network.

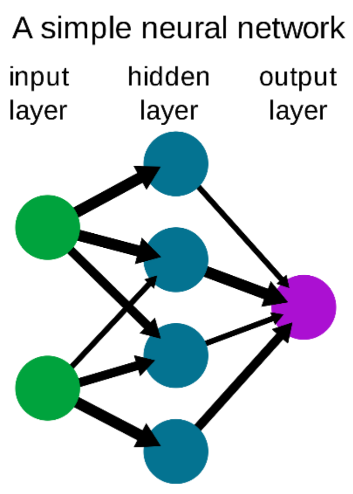


Image 1
A visual representation of the layers in a CNN and their many neuron-like interconnections[17]

Convolutional Neural Networks

Twenty-four bit colour images are represented through the use of three distinct components of numerical data, one 8-bit pixel value each for red, green and blue, stacked on top of each other and acting together to represent a pixel. These pixels represent the intensity of each colour using a number between 0 and 255 for standard 8-bit depth, with 1-byte per component. In convolutional neural networks, a small array of pixels, or “filters”, slide across the given image and preforming operations, which is termed convolution. Separate filters recognise different features withing the image, for example horizontal vs. vertical edges, as is done in the visual cortex of animals and humans as found by Hubel and Wiesel[18]. Multiple filters are used to extract a large amount of data from the image.

Each layer of a CNN uses three types of calculations: convolution, which uses only addition and multiplication to scan the image, nonlinearity, which involves applying an equation to the output from the convolution filter to find more complicated relationships, and pooling, which reduces the amount of computation requires and increases efficiency[10].

Convolutional Neural Network Platforms

There are a number of Convolutional Neural Networks platforms which simulate and apply artificial neural networks (ANNs) and are intended for practical applications such as data classification and recognition[17]. These platforms often use~~d~~ simple static ANNs that allow easy configuration by non-expert users.

As this investigation focused more on the ability of Machine Learning to classify data rather than the inner workings of a convolutional neural network, Google AutoML, an example of one of these platforms, was employed to simplify the process and focus on whether image recognition could be used to analyse and classify COVID genomics data so as to identify VOCs.

2.5 Google Cloud AutoML

AutoML, or automated machine learning, is the process in which machine learning applied to large amounts of data is automated, allowing real-world problems to be solved easier, and making machine learning more accessible to non-experts. It is seen as an artificial intelligence- based resolution to utilizing machine learning[16].

AutoML is an ANN platform which employs code free deep learning (CFDL) for image analysis, meaning no code was required to build or train the model. However, coding was required once the model was created for testing unlabelled images and creating a response from the model. This platform is hosted by Google, a company which, as previously mentioned, has devoted a lot of time and resources to develop their machine learning and artificial intelligence capabilities. Google Cloud AutoML provides a graphical user interface (GUI) for uploading, labelling, training and testing image data. After data is uploaded, automated machine learning is utilized.

AutoML Vision, a subcategory of machine learning provided through the AutoML platform, enables users to execute supervised learning, i.e., training the computer to recognise patterns using labels fed in by the user.

3. Methodology

3.1 Constructing the dataset

The dataset used to train the model was constructed from COVID-19 sequence data converted through a number of python programs into pixelated images.

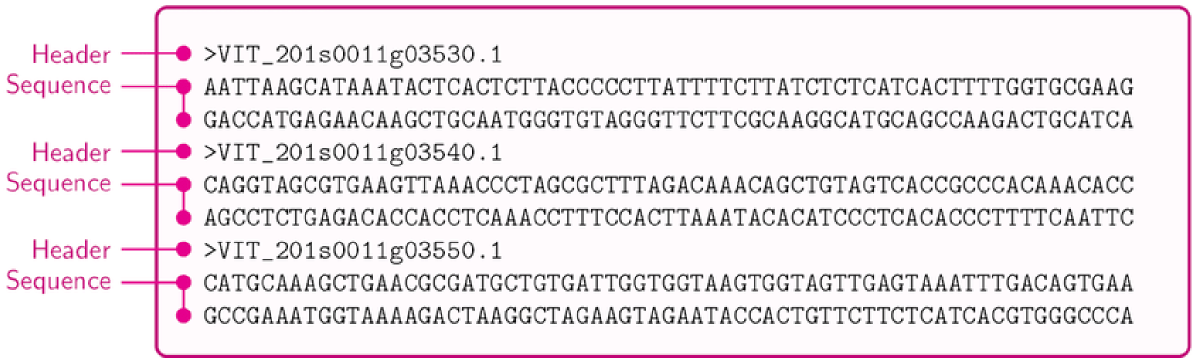
Initially, genomic data downloaded from Cog-UK[19] was downloaded and saved as a monolithic 12 Gigabyte (GB) FASTA file. This file contained the COVID-19 DNA sequences along with their corresponding names. The first step was splitting up this file into individual files, one per sequences, as accessing data in it repeatedly would be inefficient. The data was split at the end of each sequence into around 440,000* separate FASTA files, each containing one COVID sequences and saved under the name of the corresponding label (see appendix 6.1 for python code).

In bioinformatics (a field in which methods and software is developed to understand biological data), FASTA format is a text-based format representing nucleotide sequences, wherein the nucleotides are represented using single-letter codes[14]. The FASTA format defines a single metadata item and precedes the nucleotide sequence data.

In addition to the sequence data, another matching file containing the metadata, such as the sequences' lineage and country of origin, was downloaded and likewise split up in a similar fashion (see appendix 6.2 for python code). One text file was created for each lone of metadata and then a program was written to match the sequence data file to the metadata file (see appendix 6.4 for python program).It was found that 50 thousand sequences lacked sufficient, or any, metadata, and those files were therefore discarded.

The files had to be split into folders containing 10,000 files per folder, as Windows could not handle the number of files being placed in a single folder.

*The number of COVID-19 sequences available at the time of training.



```
>VIT_201s0011g03530.1
AATTAAGCATAAATACTCACTCTTACCCCCTTATTTTCTTATCTCTCATCACTTTTGGTGCGAAG
GACCATGAGAACAAGCTGCAATGGGTGTAGGGTTCTTCGCAAGGCATGCAGCCAAGACTGCATCA
>VIT_201s0011g03540.1
CAGGTAGCGTGAAGTTAAACCCTAGCGCTTTAGACAAACAGCTGTAGTCACCGCCCACAAACACC
AGCCTCTGAGACACCACCTCAAACCTTTCCACTTAAATACACATCCCTCACACCCTTTTCAATTC
>VIT_201s0011g03550.1
CATGCAAAGCTGAACGCGATGCTGTGATTGGTGGTAAGTGGTAGTTGAGTAAATTTGACAGTGAA
GCCGAAATGGTAAAAGACTAAGGCTAGAAGTAGAATACCACTGTTCTTCTCATCACGTGGGCCCA
```

Image 2 An example of a multi-FASTA file[15]

The next data preparation step was choosing a method for transforming the genetic sequences into image data to allow for image recognition software to be used. In the case of COVID, which is an RNA sequence, the nucleotides are presented as ‘A’, ‘T’, ‘C’ and ‘G’. For the sake of converting the sequences into images, each letter was assigned a colour: red, blue, green, and yellow, respectively. Each COVID sequence is around 29900 nucleotides in length, so an image matrix of a square root of that was defined for representing the sequences. Each letter corresponded to one pixel in the final image, with the proportions being 173x173 pixels (√29900) letters in length. Sections of data with unknown values, represented by ‘N’s in the original file, were saved as black pixels as seen in image 3.

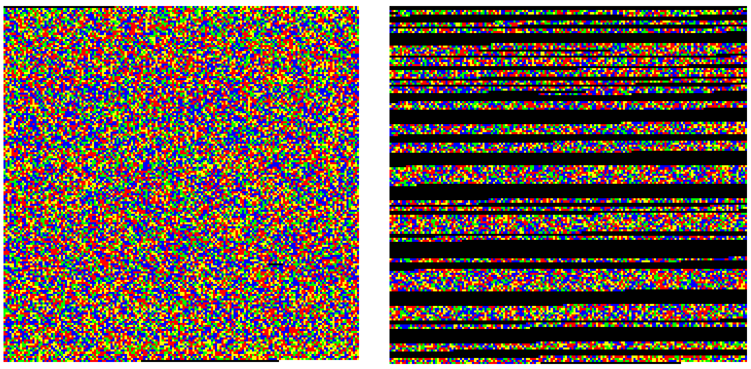


Image 3

First image with no missing data, in the second, missing data is represented with black pixels.

A majority of the images contained gaps of missing data, something which Machine Learning is very capable of extracting features from. Differing cleanliness of the data files, such as mixed use of both lower and upper cases to represent nucleotides, led to some issues, but overall, data provided by Cog-UK was organised enough to be processed efficiently. After being split up to ensure there are enough for both training the model and measuring the accuracy of it, each image along with its corresponding label will be fed into the classifier.

AutoML, the platform through which classification takes place, requires the data represented by an image and corresponding label, to be split into training, validation, and testing sets. The training set is used to train the model, the validation set is used to validate the results and the test set contains the data the model is tested. The validation set is used by AutoML to know when to stop training the model, and the test set is used to measure the model's accuracy after it has been trained.

A program was written to split a specified lineage into three separate subfolders: "TRAIN", "VALIDATION" and "TEST", along with a corresponding .csv file for each stating the location of each image in the google cloud database and the label linked to it so for example, "TEST,gs://covid_1_test_uscentral1/England_ALDP-1489703_2021.png,B_1_1_7".

3.2 Training of the Model

The main source of results in this investigation come from primary experimental data produced by a trained computer vision model. During initial testing, a model was trained through the Google Cloud AutoML platform using three variants, with 300 images per variant, to gauge the capability of the platform. This test was completed on the following variants: B.1, B.1.1.7 and AD.2, and the results were as follows:

	AD.2	B.1	B.1.1.7
AD.2	0.97	0.03	0.00
B.1	0.01	0.88	0.11
B.1.1.7	0.00	0.00	1.00

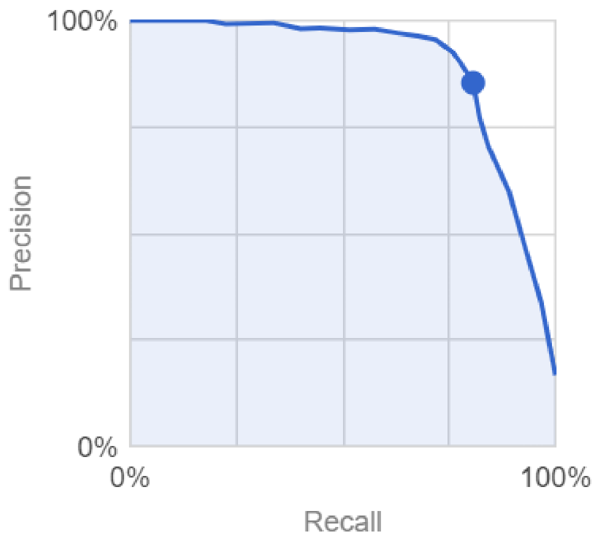
Confusion matrix of initial testing

This data represents the confusion matrix, and shows how often the model classified each label correctly (blue), and the labels that were frequently confused (grey). The model's performance can be used to evaluate and compare models by showing whether the images are being classified correctly and accurately. An ideal model would have high values along the diagonal, and low values in all other sections. As can be seen from the results outputted by the initial test, the model produced highly accurate results, with only 5% of the data being classified incorrectly. These results indicate that Computer Vision is capable of accurately classifying genetic sequences converted to images despite large amounts of gaps.

A new model was then created and trained on six VOCs with 300 images per variant.

	B.1.351	B.1.1.7	B.1.617.2	B.1.525	P.1	B.1.617.1
B.1.351	0.87	0.03	0.00	0.04	0.01	0.04
B.1.1.7	0.13	0.71	0.01	0.05	0.02	0.08
B.1.617.2	0.00	0.03	0.92	0.04	0.01	0.00
B.1.525	0.05	0.03	0.06	0.73	0.00	0.13
P.1	0.10	0.02	0.00	0.04	0.76	0.08
B.1.617.1	0.01	0.00	0.02	0.05	0.01	0.90

Confusion matrix of model trained on variants of concern.



Precision-recall curve.

The ML model produced a positive prediction model (PPV) (precision) of 96.34% and a sensitivity (recall) of 67.59%. The PPV indicates the proportion of positively classified data items that were true positives[23] i.e. were given the correct label, and the sensitivity states how valid the results were, or the accuracy. As can be seen by the precision rate of the model, missing stretches of data don't appear to be a limiting factor when using this method, confirming that machine learning is an apt approach for a task of this sort.

Similar to the initial testing, high values along the diagonal of the confusion matrix suggest an accurate, well-trained model. Though these values are high, compared to the initial test, there values which are lower. This may be due to the larger number of labels fed into the model, providing more labels and causing "confusion", or could relate to the genetic similarities between the variants fed into the second model. Compared to the randomly picked variants picked for the initial test, all variants chosen for training the second model are classified as "variants of concern", so more frequent confusion when training the second model may indicate a higher frequency in genetic resemblances.

In order to test the accuracy of the model, the model had to the proven to be able to accurately classify the COVID-19 variants. After the multi-label classification model was created and trained with the VOC and the accuracy of the model was evaluated, the model was deployed so that new, unlabelled images could be fed into the model in order to test its ability. For this to happen, a batch of images was fed into the model asynchronously using the AutoML "BatchPredict" Application Program Interface (API), which applies labels to the test images based on the primary object of the image that the model predicts[20]. A python program written according to the API to pass the test images through the deployed model using a csv file with the corresponding paths and returned a json file with the corresponding label given by the trained model.

Due to the quantity restriction on Google AutoML, ten random sequences were selected per day between the dates 05/03/21 and 04/07/21, amassing in around 1200 sequences to be classified.

The json file returned data in the following format:

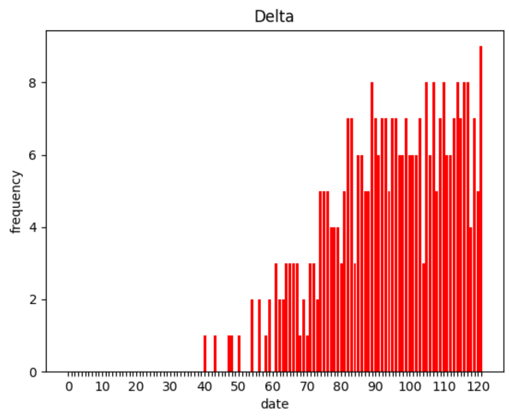
{"ID":"gs://covid_1_batchprediction_uscentral1/England_ALDP-16B8A97_2021.png","annotations":[{"annotation_spec_id":"8889664760266620928","classification":{"score":0.8513818399999999},"display_name":"B_1_617_2"}]},

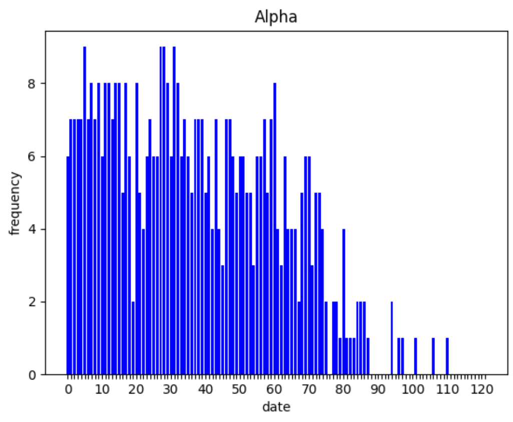
Line taken from outputted jsonl file

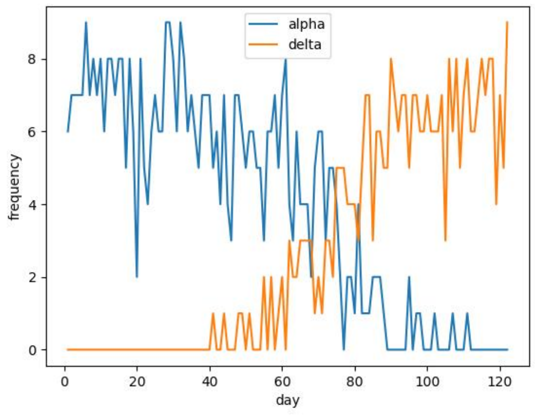
This provides a classification 'score', or a representation on the model's confidence in its categorization, along with the matching label returned from the model. With the use of a python program all results returned from the model were parsed, returning a simple list of dates along with an alpha and delta sequence count for each day, as between the two dates listed above, 99% of the recorded sequences were either Alpha or Delta according to the data collected by Cog-UK.

4. Results

From this data several graphs were produced using the matplotlib module on python(see Appendix something), providing image representations of the data produced.





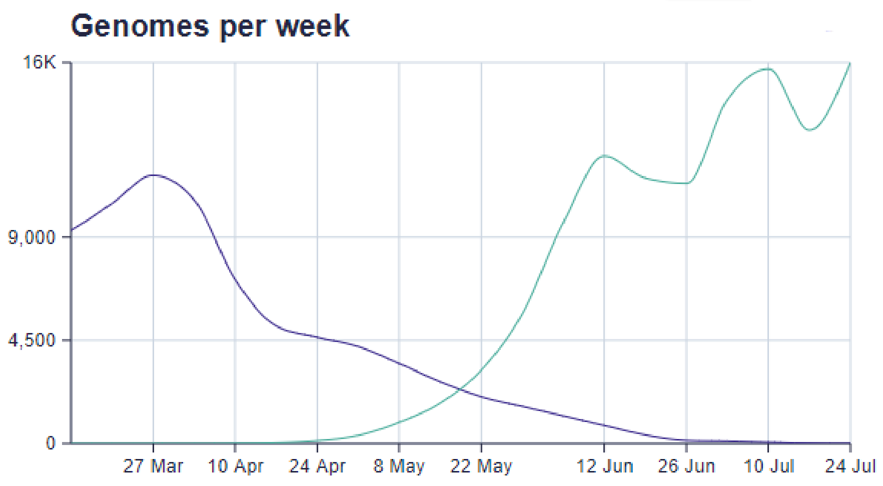


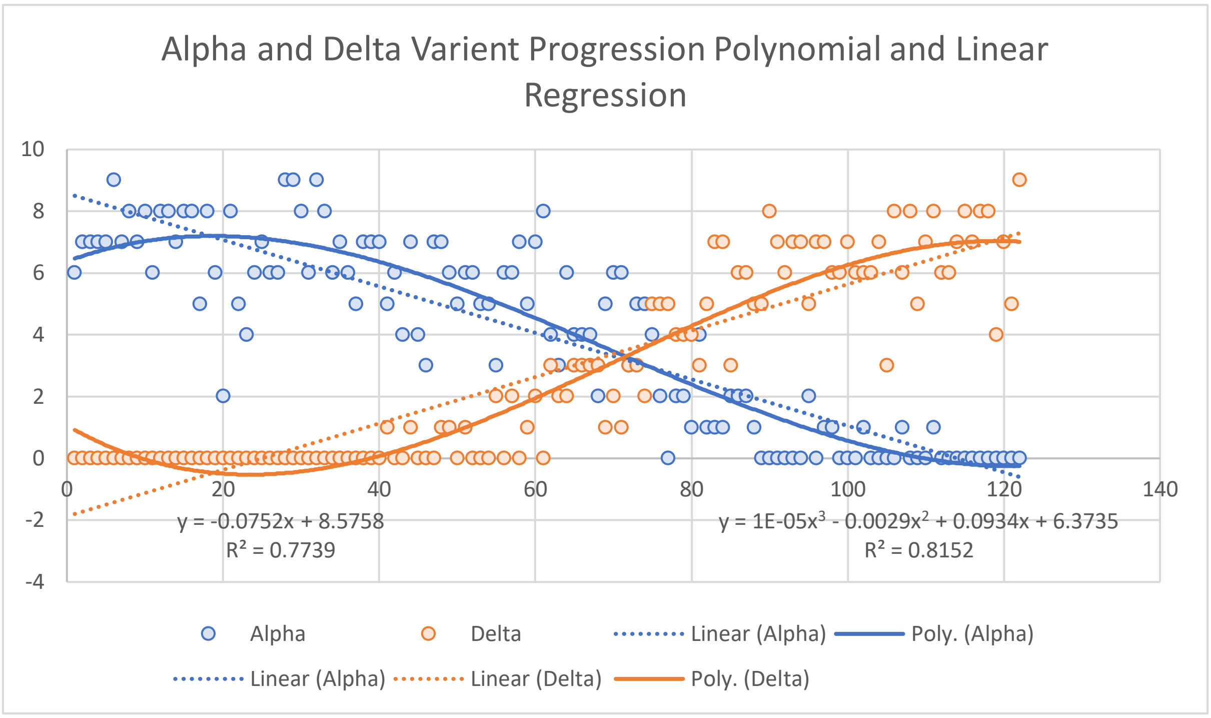
Above:
Histogram of Alpha and Delta variants over set dates

Left:
Line plot of variants

These results were produced from a randomised subset of the total data (meaning variation in quantity between days in the original dataset was not considered), but despite that, these results clearly mirror the known trends of these two variants of concern between the dates chosen. A polynomial trendline and regression equations were calculated in Excel for the data (as an exponential fit would not be able to take into account the zero values):

To compare against data collected through non-AI methods, a graph taken from the Sanger Institue COVID-19 Genomic Survelliance page[24] showing the recorded Alpha and Delta variants between March and July is shown below:





Through the use of computer vision an artificailly intelligent image classification model, trends in COVID-19 data were discovered and documented accuratelly. The goal of this paper was to investigate the effectiveness of machine learning and computer vision in classifying data not typically presented in an image format, and explore a novel way to encode and classify sequence data. The results seen in the figure taken from the Sanger Institute are clearly mirrored in the results produced by the ML model shown in figure x, demonstrating that the model was able to efficiently and effectivly categorize the data.

5. Conclusion

In this paper, the feasibility of using machine learning to classify genomic sequences was investigated on the Google AutoML platform. The results show that even a model trained and tested on a limited number of COVID-19 genomic sequences converted into images could be correctly classified even with significant gaps in the sequence data.

The high accuracy rate of 96.34% when evaluating the training of the model suggests that machine learning is a capable method for classifying genomic sequences in image format even with large amounts of missing data present, which with the use of other classification processes can cause problems. This indicates that the model takes into account features that cannot easily be seen by human inspection.

A model trained on a wider array variants with the use a much greater amount of data per variant should be more capable of performing classification tasks, and could be applied to…

Hopefully, this investigation of machine learning will allow for a consideration of new applications regarding the use of image classification software on data not usually presented as images. This idea of creating data capable of being passed through a computer vision machine learning model could have many applications and may be able to provide an efficient method of classifying data.

6. Applications

Machine learning, or more specifically deep learning, has the ability to recognise aspects and features in data undistinguishable by the human eye, as seen in a study by Korot E. et al, wherein automated deep learning succesfully classified sibject's sex from retinal fundus photographs, despite ophthalmologists not previously having the capability to do this[25]. The ability of machine learning to classify features more accurately than humans possibility of many new discoveries through more research into the field, but in terms of biosurveillance, and more specifically classifying COVID-19 sequences, it allows for the possibility of early detections of new dangerous variants through the monitoring of automatically produced data.

Applications within the field of genomics with sequences converted into images could include biosurveillance, meaning the rapid detection of new variants and mutations. A model like this could be deployed onto a point of care device for rapid classification instead of requiring large amounts of sequence data to compare against.

7. Bibliography

[1] "M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang and F. Iqbal, "Malware Classification with Deep Convolutional Neural Networks," *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2018" accessed 04/06/21

[2] Malware Images: Visualization and Automatic Classification https://www.researchgate.net/profile/Shanmugavadivel-Karthikeyan/publication/228811247_Malware_Images_Visualization_and_Automatic_Classification/links/0deec53bee6c992ef1000000/Malware-Images-Visualization-and-Automatic-Classification.pdf, accessed 10/06/21

[3] "Computer Vision" https://www.sas.com/en_us/insights/analytics/computer-vision.html, accessed 10/06/21

[4] "Image Recognition with Deep Neural Networks" https://www.altexsoft.com/blog/image-recognition-neural-networks-use-cases/ , accessed 04/0s6/21

[5] "Facebook Outlines Advances in Computer Vision and Object Identification Tech" https://www.socialmediatoday.com/news/facebook-outlines-advances-in-computer-vision-and-object-identification-tec/599399/ , accessed 10/06/21

[13] “What is an API” https://www.mulesoft.com/resources/api/what-is-an-api accessed 11/06/21 accessed 11/06/21

[14] “FASTA format” https://en.wikipedia.org/wiki/FASTA_format#:~:text=In%20bioinformatics%20and%20biochemistry%2C%20the,comments%20to%20precede%20the%20sequences. accessed 11/06/21

[15] “A sample of the multi-FASTA file” https://www.researchgate.net/figure/A-sample-of-the-Multi-FASTA-file_fig1_309134977 accessed 11/06/21

[16] “Automated Machine Learning” https://en.wikipedia.org/wiki/Automated_machine_learning accessed 25/06/21

[17] “Artificial Neural Network Software” https://www.predictiveanalyticstoday.com/top-artificial-neural-network-software/ accessed 27/06/21

[18] “Eye, Brain and Vision” David H. Hubel accessed 27/06/21

[19] “All Sequences” https://cog-uk.s3.climb.ac.uk/phylogenetics/latest/cog_all.fasta accessed 18/04/21

[20] “Making batch predictions” https://cloud.google.com/vision/automl/docs/predict-batch?authuser=3 accessed 04/07/21

[21] “SARS-CoV-2 variants of concern as of 1 July 2021” https://www.ecdc.europa.eu/en/covid-19/variants-concern accessed 04/07/21

[22] “Clade and lineage nomenclature” https://www.gisaid.org/references/statements-clarifications/clade-and-lineage-nomenclature-aids-in-genomic-epidemiology-of-active-hcov-19-viruses/ accessed 04/07/21

[23] ”Machine Learning and Applied Statistics Lesson of the Day” https://chemicalstatistician.wordpress.com/2014/08/07/machine-learning-and-applied-statistics-lesson-of-the-day-positive-predictive-value-and-negative-predictive-value/ accessed 06/07/21

[24] “COVID-19 Genomic Surveillance” https://covid19.sanger.ac.uk/lineages/raw?show=B.1.1.7%2CB.1.617.2&xMin=2021-03-13&xMax=2021-07-24&area=overview&latitude=53.292845&longitude=-2.149341&zoom=5.63 accessed 20/08/21

[25] “Predicting sex from retinal fundus photographs using automated deep learning” https://www.nature.com/articles/s41598-021-89743-x.pdf accessed 10/06/21

8. Appendix

6.1 Code for splitting data

```
import os.path
#437262

save_path = r"C:\Users\yaell\OneDrive\Desktop\EE\Sequences"
f = open(r"C:\Users\yaell\OneDrive\Desktop\EE\cog_all.fasta", "r")

count = 0
for line in f:
        if count%10000 == 0:
                foldername = str(int(count/10000)).zfill(4)
                directory = os.path.join(save_path, foldername)
                print(count," ",directory)
                if not os.path.exists(directory):
                        os.makedirs(directory)
        count+=1
        name = line.strip(">").strip("\n").replace("/","_")
        content = next(f)
        newpath = os.path.join(save_path, foldername)
        #newpath = os.path.join(save_path, str(filename))
        completeName = os.path.join(newpath, name+".fasta")
        file = open(completeName, "w")
        file.write(content)
        file.close()
```

6.2 Code for splitting metadata

```
import os, time
starttime = time.time()

f = open(r"C:\Users\yaell\OneDrive\Desktop\EE\cog_metadata.csv")
save_path = r"C:\Users\yaell\OneDrive\Desktop\EE\Metadata"


count = 0
for line in f:
   # f.readline()
    if count%10000 == 0:
        foldername = str(int(count/10000)).zfill(4)
        directory = os.path.join(save_path, foldername)
        print(count," ",directory)
        if not os.path.exists(directory):
            os.makedirs(directory)

    line = line.split(",")
    name = line[0].replace("/","_")
    newpath = os.path.join(save_path, foldername)
    #newpath = os.path.join(save_path, str(filename))
    completeName = os.path.join(newpath, name+".csv")
    file = open(completeName, "w")
    line = ','.join([str(elem) for elem in line])
    file.write(line)
    file.close()
    count+=1

print(count)
print("Time taken was" , time.time() - starttime)
```

6.3 Code for creating images

```
from PIL import Image
import math, numpy as np, os, time


def imagemaker(startcode, name, counter, directory):
    startcode = str(startcode).upper()
    code = ""
    for i in startcode:
        if i == 'A' or i == 'C' or i == 'T' or i == 'G' or i == 'N':
            code = code + i
    count = 0

    num = math.ceil(len(code)/173  )
    brack = 0
    pixels = []
    for row in range(num):
        pixels.append([])

    for k in range(num*173):
        if count<len(code):
            if code[count] == 'A':
                pixels[brack].append((255,0,0))
            elif code[count] == 'C':
                pixels[brack].append((255,255,0))
            elif code[count] == 'T':
                pixels[brack].append((0,0,255))
            elif code[count] == 'G':
                pixels[brack].append((0,255,0))
            elif code[count] == 'N':
                pixels[brack].append((0,0,0))
        else:
            pixels[brack].append((255,255,255))
        count += 1
        if count%173 == 0:
            brack += 1

    array = np.array(pixels, dtype=np.uint8)
    new_image = Image.fromarray(array)
    name = name.strip(".fasta")
    filepath = os.path.join(directory, name + ".png")
    new_image.save(filepath)




starttime = time.time()

paths = []
startpath = r'C:\Users\yaell\OneDrive\Desktop\EE\Sequences'
savetopath = r'C:\Users\yaell\OneDrive\Desktop\EE\Images'
for folder in os.listdir(startpath):
    paths.append(os.path.join(startpath, folder))

counter = 0
for subfolder in paths:
    for file in os.listdir(subfolder):
        fullpath = os.path.join(subfolder, file)

        with open(fullpath, "r") as f:
            content = f.readlines()
        if counter%10000 == 0:
            foldername = str(int(counter/10000)).zfill(4)
            directory = os.path.join(savetopath, foldername)
            print(counter," ",directory)
            if not os.path.exists(directory):
                        os.makedirs(directory)
        imagemaker(content, file, counter, directory)
        counter += 1


print("Run time was" , time.time() - starttime)
```

6.4 Code for linking Metadata with corresponding COVID sequence

```
import os, time, os.path
from pathlib import Path
starttime = time.time()


paths = []
findpaths = []
startpath = r'C:\Users\yaell\OneDrive\Desktop\EE\Metadata'
findpath = r'C:\Users\yaell\OneDrive\Desktop\EE\Images'


def findfolder(name):
    for folder in os.listdir(findpath):
        findpaths.append(os.path.join(findpath, folder))
    for sub in findpaths:
        fullpath = os.path.join(sub, name + ".png")
        if Path(fullpath).is_file():
            return fullpath


for folder in os.listdir(startpath):
    paths.append(os.path.join(startpath, folder))


counter = 0

bigfile = open("masterfile.csv", "w")
bigfile.write("file_name,label,is_valid,date")


for subfolder in paths:
    for file in os.listdir(subfolder):

        if counter%10000==0:
            print(counter)


        fullpath = os.path.join(subfolder, file)
        with open(fullpath, "r") as f:
            headings = f.readline()
            content = f.readline()

            content = content.split(",")
        name = content[0].replace("/","_")
        fullname = findfolder(name)
        printto = fullname,content[6],True,content[4]
        bigfile.write(str(printto).strip("(").strip(")")+"\n")
        counter += 1
        f.close()

bigfile.close()
```

6.5 Variants of Concern

NextStrain Clade	**PANGO lineage**	**Origin**
20H Beta V2	B.1.351	South Africa
20I Alpha V1	B.1.1.7	UK
20J Gamma V3	P.1	Brazil
21A Delta	B.1.617.2	India
21B Kappa	B.1.617.1	India
21C Epsilon	B.1.427, B.1.429	USA
21D Eta	B.1.525	Nigeria
21F Iota	B.1.526	USA
21G Lambda	C.37	Peru