Classifying Viral Genomes with Convoluted Neural Nets

Intro

My overarching goal for this project is to classify viruses by their genomes using a neural network. Viruses are the ideal group of organisms to test this process out on due to the relatively small size of their genomes and their great diversity. A convoluted neural net should work well to find patterns since these operate on smaller subsets of an input 'image', and will therefore be able to identify genes, repeating sequences, non-functional gene fragments, or any other pattern that can be used to classify close relatives. My method is to pass the sequences as 'images' to a convoluted deep neural network, which would ideally be able to break down the sequences into their underlying patterns and discover how best to classify them. I have built a set of models that can classify viruses of a certain family out of a set of viruses from the same classification, as well as a model that can identify those four classifications from each other. Ideally, I would be able to enter a random genome into a pipeline and get back the family /subfamily it belongs to or doesn't. That being said, the preprocessing and modeling for the larger datasets are very demanding, so time and computing power are limiting factors on making this happen.

Data Acquisition

The data acquisition phase for this project was by no means a straight path, although I am very happy with the resulting data. I began with the NCBI viral genomes main page, wherein each row entry in the main table has a virus, some information about it such as source and host category, and most importantly a link to a unique information page for each virus. The first step I took was to scrape this page in order to create a table of viruses, with the extracted links for each virus' page instead of the accession number. As I scraped this data, I also extracted features from the table that were not in the virus rows. For example, the classifications, e.g. Deltavirus, were in the table as header rows above the rows of viruses that belonged to that category, so I had to adjust my code to add this information to each virus row in my DataFrame. Certain viruses were present in the table as a series of sequences, found within another table inside some cells of the main one. I build another version of the overarching code to extract the rows from these 'minitables' and treat them like other rows, with the addition of the header to the 'minitable' as a value in an 'Extra Info' column so the header data would not be lost.

When I had all of the viruses and their links my next step was to extract the info from each page. I created a function to scrape a BeautifulSoup object, or 'soup', for each page and save it to a .csv file, in order to ensure that I didn't have to scrape each page more than once. This function was run in batches in order to reduce the amount of time lost when the connection was broken, since a break that stopped the process would prevent all of the soups obtained from the function from saving. I then added the soup for each virus into my main table. Once I had all of the soups, I attempted to extract the genomes, only to find that they were loaded into the webpages from another source, and were not present in the html soups. I did extract some information from the soups for each virus, most notably the key I needed to access each virus' full report from the NCBI Entrez database, where they are stored. I installed BioPython and

using a similar function setup to the soup acquisition, and then I extracted the report for each virus using the Entrez system and saved them to a .csv, also running this process in batches with a function.

With the reports in hand, I extracted the genomes and phylogenies into the virus table. The genomes were re-formatted using regular expressions. I then saved the virus table as a final .csv and began working in a new notebook.

Data Cleaning

Once I had the data in hand my first step was to reformat or edit columns as necessary. The length and protein columns had units removed and were converted to integers, and the phylogeny column had classifications removed wherever applicable, as these were often the first item in that list. I also modified the 'SequenceType' column such that every virus with a complete genome was labelled the same way. With the data properly formatted, I then explored it visually in order to gain a better understanding of the dataset.

Exploratory Data Analysis

I created parallel lists for the classifications of viruses and the counts for each value, and plotted these lists together, showing that double stranded DNA viruses with no RNA stage were the most common at almost 3000 viruses, followed closely by single stranded RNA viruses (Fig. 1.). The next three most common groups, had similar frequencies just over 1000, including single stranded DNA and double stranded RNA viruses, as well as unclassified RNA viruses (Fig. 2.). Next, I plotted the mean sequence lengths against classifications, finding the double stranded DNA viruses and unclassified bacterial viruses to be greatly longer than the rest. A majority of the viruses had sequences near or under 10,000 nucleotides. I next examined the relationship between mean sequence length and host type (Fig. 3.). The 'Host Type' column was quite flawed, containing specific list values that matched or partially matched other values. The best way to organize this feature would have been with a set of binary columns, but by this point in the process I had already decided I would be looking at classifications and specific phylogenies, and was more eager to get to model building.

I then looked at the different phylogenies within each classification. My goal was to find specific families that made up a large enough proportion of each classification to train a model accurately. The most common group in the double stranded DNA viruses was the Siphoviridae family (Fig. 5.). In the double stranded RNA classification, I found that roughly half of the viruses belonged to the Sedoerovirinae subfamily of the Reoviridae family (Fig. 6.). In the single stranded DNA classification, the Geminiviridae was the most common, and at a glance no other family came close to half of the classification. Lastly, in the single stranded RNA classification, the majority of the viruses were positive, (corresponding to a positive RNA strand) and a minority were negative or unclassified (Fig. 7.). After creating models to identify each of these families from their classifications, I also wanted to create a model to classify viruses from these four groups, so I plotted their relative proportions in both a bar and a pie chart to see their actual and proportional values (Fig. 8.).

Preprocessing

For the preprocessing phase I created two functions to turn my raw sequences, variable length strings of characters (a,t,g,c), into equally long lists of characters forming an 'image' to be passed to the convoluted neural nets. The first step in this process was to build a padder: this function takes a series of strings and returns that series with every string having the same length as the longest one, by filling it with zeroes. It does so first by looping through the series to identify the maximum length of a sequence, and then using an inner function to add the appropriate number of zeros to the end of the string to match that length. The next function needed was a encoder: this turns a series of strings into an list, which serves as the 'image' once it is reshaped. This function also relies an an internal function, a simple if-elif-else framework that returns a 'double' type value corresponding to each letter, or returns the same character if it is not one of the four dna nucleotides (Kwan et al 2009). The function then returns a list of these numbers, replacing the original string. In addition to applying these two functions to the input variables, I also applied the keras utility function 'to_categorical' to encode the target output variables. For some of the larger datasets, I had to run the preprocessing as .py scripts in the terminal after they caused the jupyter notebook kernel to crash. The results of the preprocessing were saved in the models subfolder in the '.npy' format, which allowed them to be loaded back in very rapidly relative to their size.

Modelling

With the models as with the preprocessing, I built the frameworks in the jupyter notebook, but after running into kernel issues, I simply copied the code into .py files in order to run and modify the models from the terminal both on my local machine and in Amazon Web Service EC2 instances. To start, I imported the preprocessed .npy files for the smallest set, the single stranded DNA viruses, and randomly split a training and testing set, with 33% of the data serving as the test. I also had the train test split stratify on the target variable to ensure that both the train and test sets had similar proportions of true 1 and 0 values (in this case, Geminiviridae and not-Geminiviridae, respectively. In order to use the lists in a 2 dimensional convoluted network, I reshaped the input variables to make them three dimensional, with a value of 1 in the first and third dimension, and the length of each sequence, or 11702, as the second. Next, I built the sequential convoluted neural network. My first model had 3 pairs of convolutional and pooling layers, a flatten layer, and two dense layers, the last of which was the output layer with a sigmoid activation (Fig. 9). All of the convolutional layers used a rectified linear unit activation, and a kernel size, the 'window' examining smaller portions of the sequence, was (1,100). The model was complied with the adam optimizer and the 'categorical_crossentropy' loss function. Before further tweaking this model I fit it with 10 epochs and found that several runs consistently scored between 94 and 98% binary accuracy on the test set, compared to a baseline of 50.37%.

Working off of this first model, I simply copy-pasted the same code to work off of for the other three classifications, and for a fifth model, (conveniently numbered 0), which would classify the viruses into these four categories. The second model, working with single-stranded RNA viruses, ran more poorly than the first one, reaching a binary accuracy of 79.34% over a baseline of 69.83% after some tweaking, mostly including changing the kernel sizes in order to capture larger patterns. With my third model, I ran into a new issue. The double stranded DNA

viruses included the longest sequences by far and just attempting to preprocess this data simply returned a MemoryError in the terminal or crashed the jupyter notebook (Fig. 2.). The provisional solution to this problem was to use only sequences shorter than 100,000 nucleotides, including 2141 of the 2787 original viruses. With this change, the data could be placed in the third model, starting at a kernel size of 1,200 and getting smaller with every convoluted layer, eventually reaching a binary accuracy of 77.58% over a baseline of 52.73%. The fourth model identified the Sedoreovirinae subfamily from the double stranded RNA viruses, and after a little bit of tweaking in kernel sizes, scored a 77.23% binary accuracy over a baseline of 70.67%. My last model worked off of a larger subset of data, all of the data from the first four models, running into the same memory errors during preprocessing. After reducing the maximum sequence length like before, I was able to process the data into padded and encoded lists, but the model wouldn't run due to the size of the data. The solution I found was to take a random sample from each of the classifications, so I took 1100 random viruses of each class and preprocessed those, and the resulting model scored a 77% over a baseline of 25%.

Discussion

This project thus far shows conclusively that a convolutional neural net can function well in classifying specific virus families or classifications, beating the baseline with varying margins in every case. Going forward, I intend to continue refining these models until they all surpass a binary accuracy of 90%, or in the case of the multi-classification model, a categorical accuracy over 90%. I fully expect to reach this, as there remain several avenues of customizability that I haven't yet begun to explore in terms of model optimization, such as different loss functions or a greater variety of layer configurations. Once all of the models work well, my next goal will be to attempt to classify the viruses in the 'unclassified' category, to see if any of them fit within the families I can identify.

The use case of these models is limited to identifying these specific four families of viruses, but with a large enough dataset, the same process could be used to classify or identify any genome or genome segment small enough to work with. Another even more alluring option would be to work with different targets; training a model to predict other features based on genomes, such as certain mechanisms or functions, could lead to the discovery of the genetic underpinnings of these other features. With ample training, a more complex neural network or set of networks may be able to predict not only the family of an unknown virus based on its genome, but how it functions, what host it needs to target, and how it can theoretically be treated, all before having to work with a live specimen.

References

Keung Kwan, Hon & Arniker, Swarna. (2009). Numerical representation of DNA sequences. 307 - 310. 10.1109/EIT.2009.5189632.

NCBI viral genomes resource.

Brister JR, Ako-Adjei D, Bao Y, Blinkova O. Nucleic Acids Res. 2015 Jan;43(Database issue):D571-7. doi: 10.1093/nar/gku1207. Epub 2014 Nov 26.

Appendices

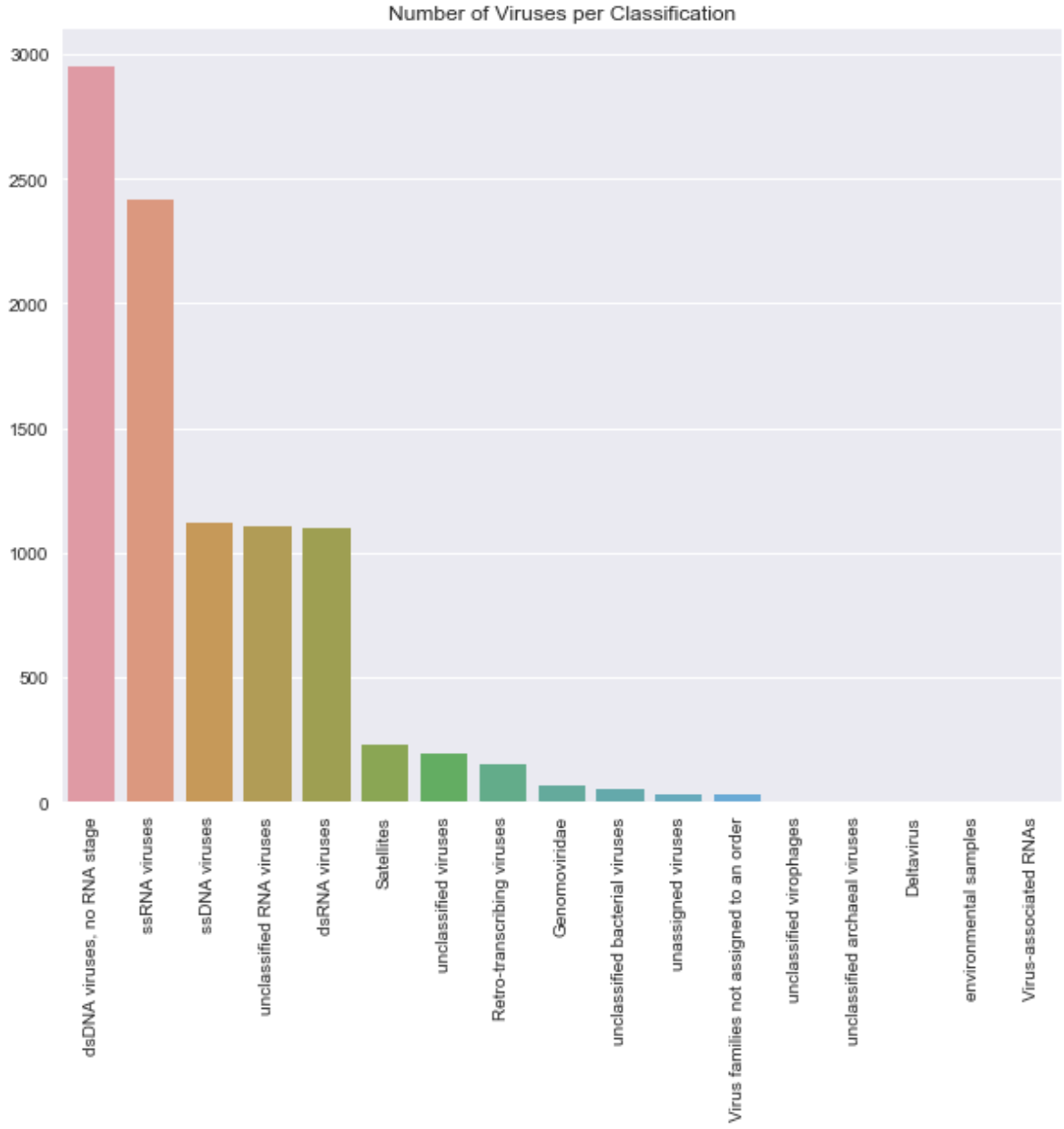

Figure 1. Total number of viruses, including both complete genomes and segments, in each classification of the NCBI viral genomes database.

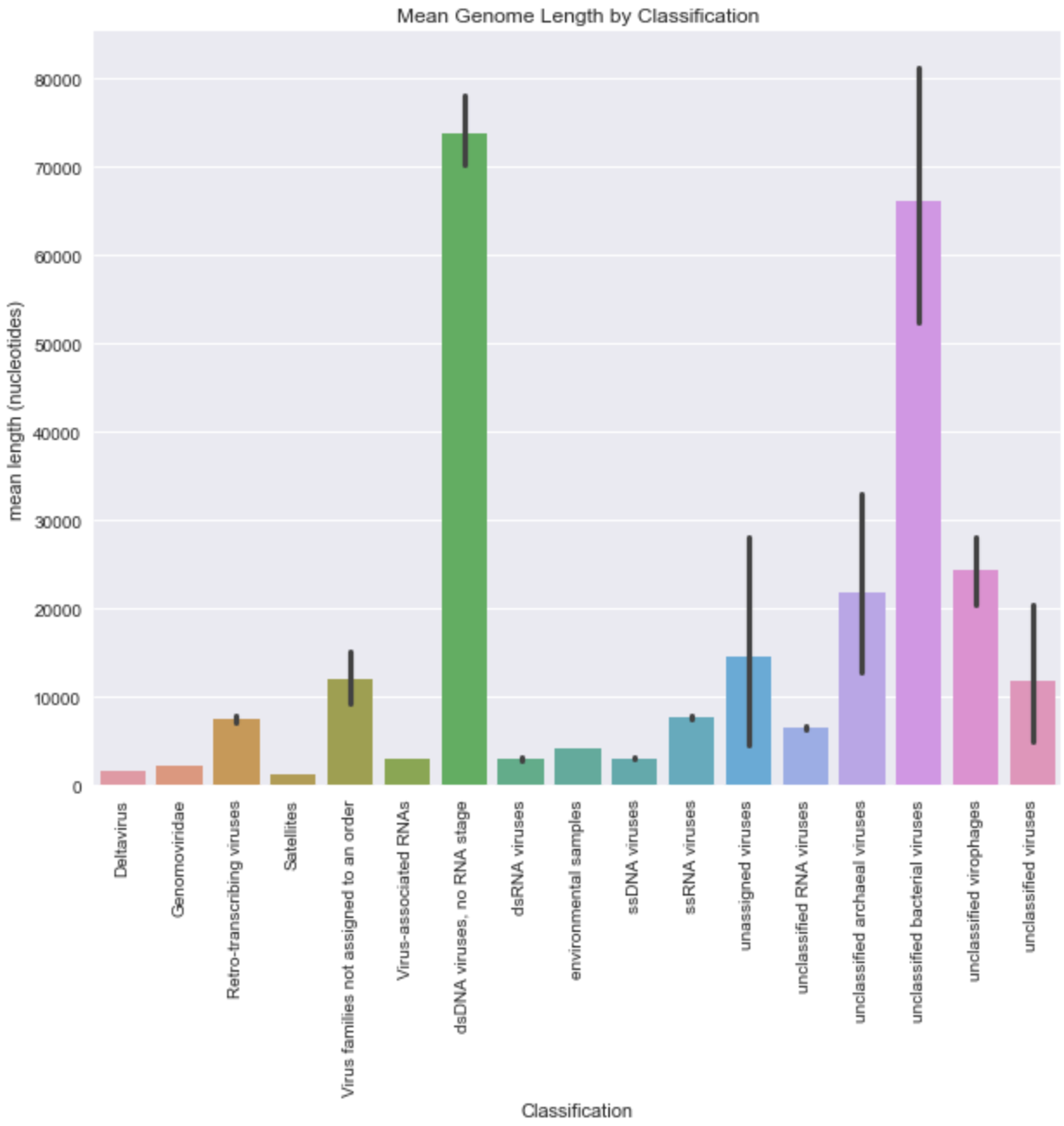

Fig. 2. Mean sequence length with error bars per classification for all sequences in the NCBI viral genome database.

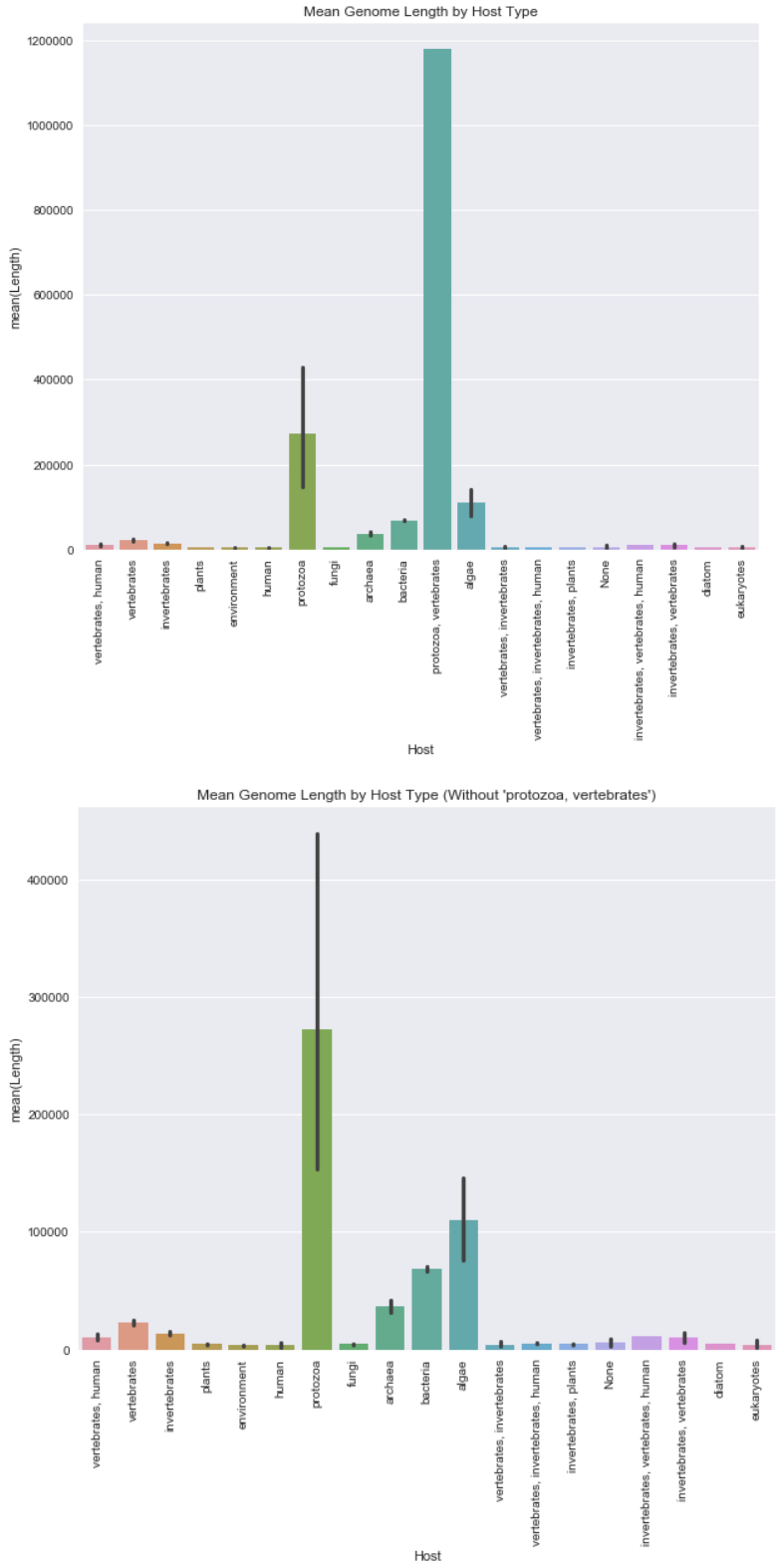

Fig. 3. Mean sequence length for viruses in the NCBI viral genome database. Including the ‘protozoa, vertebrates’ category (top) and without that category (‘bottom’)

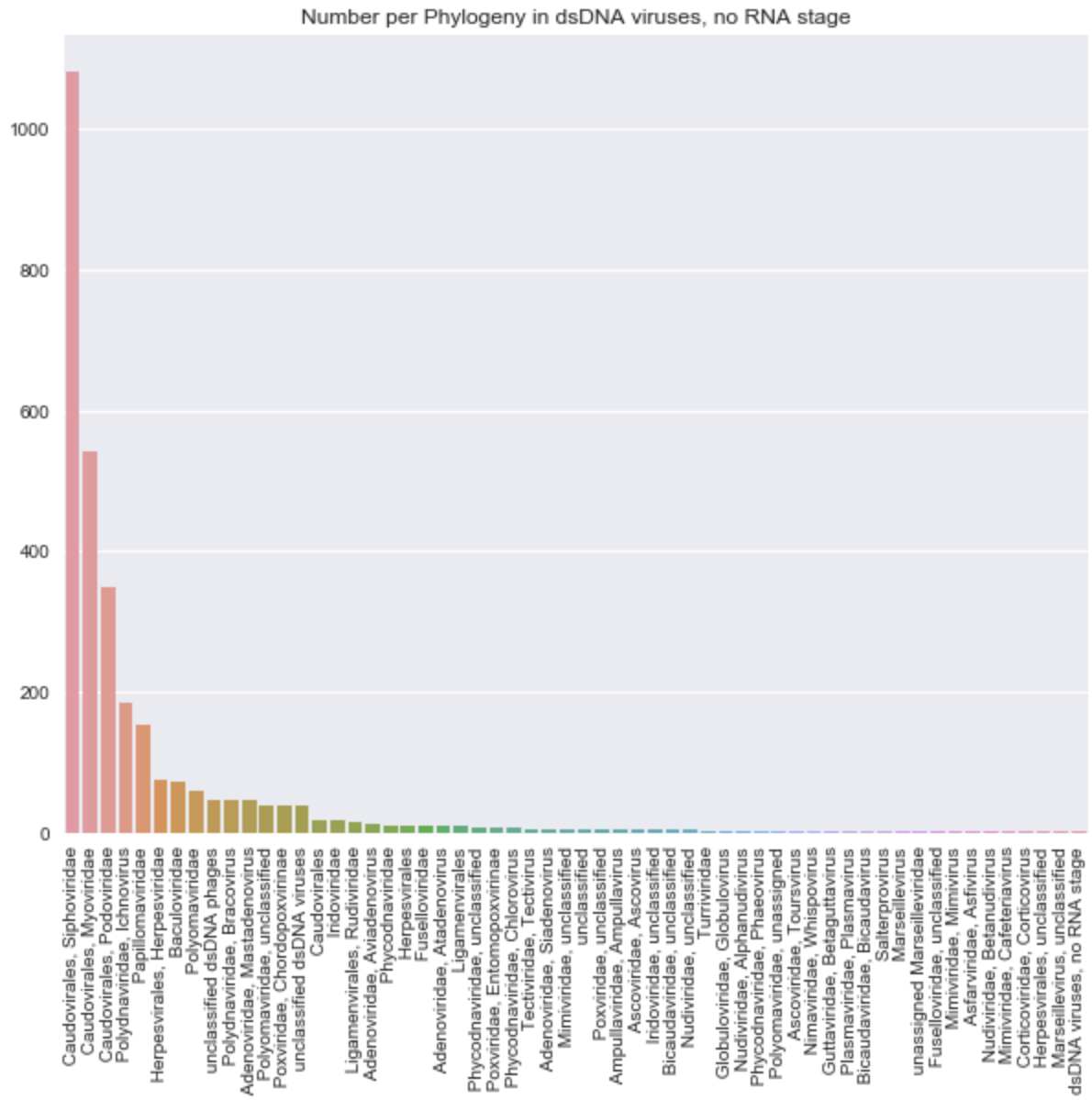

Fig. 4. Number of viruses with each phylogeny in the double stranded DNA virus classification of the viral genomes in the NCBI database.

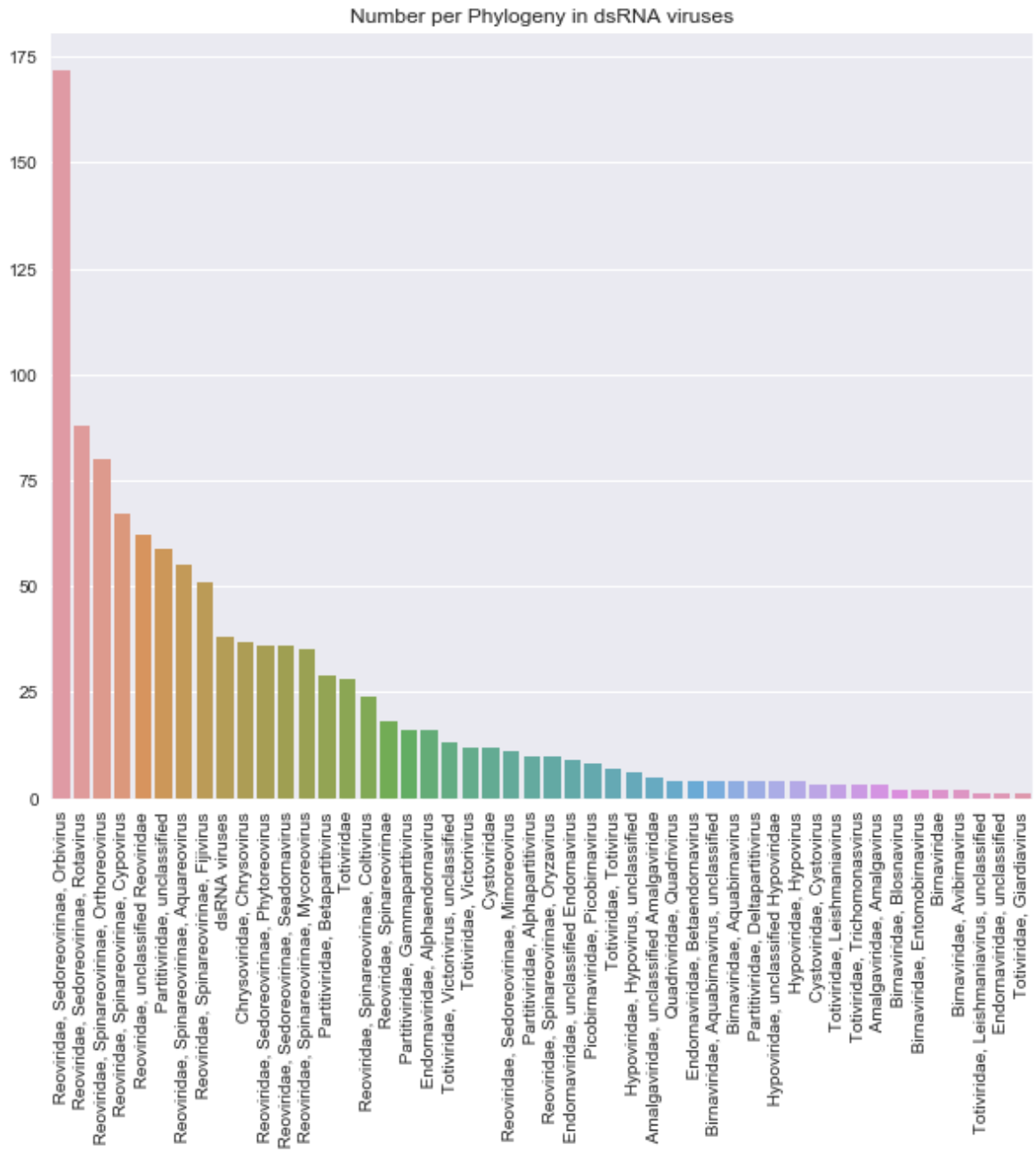

Fig. 5. Number of viruses with each phylogeny in the double stranded RNA virus classification of the viral genomes in the NCBI database.

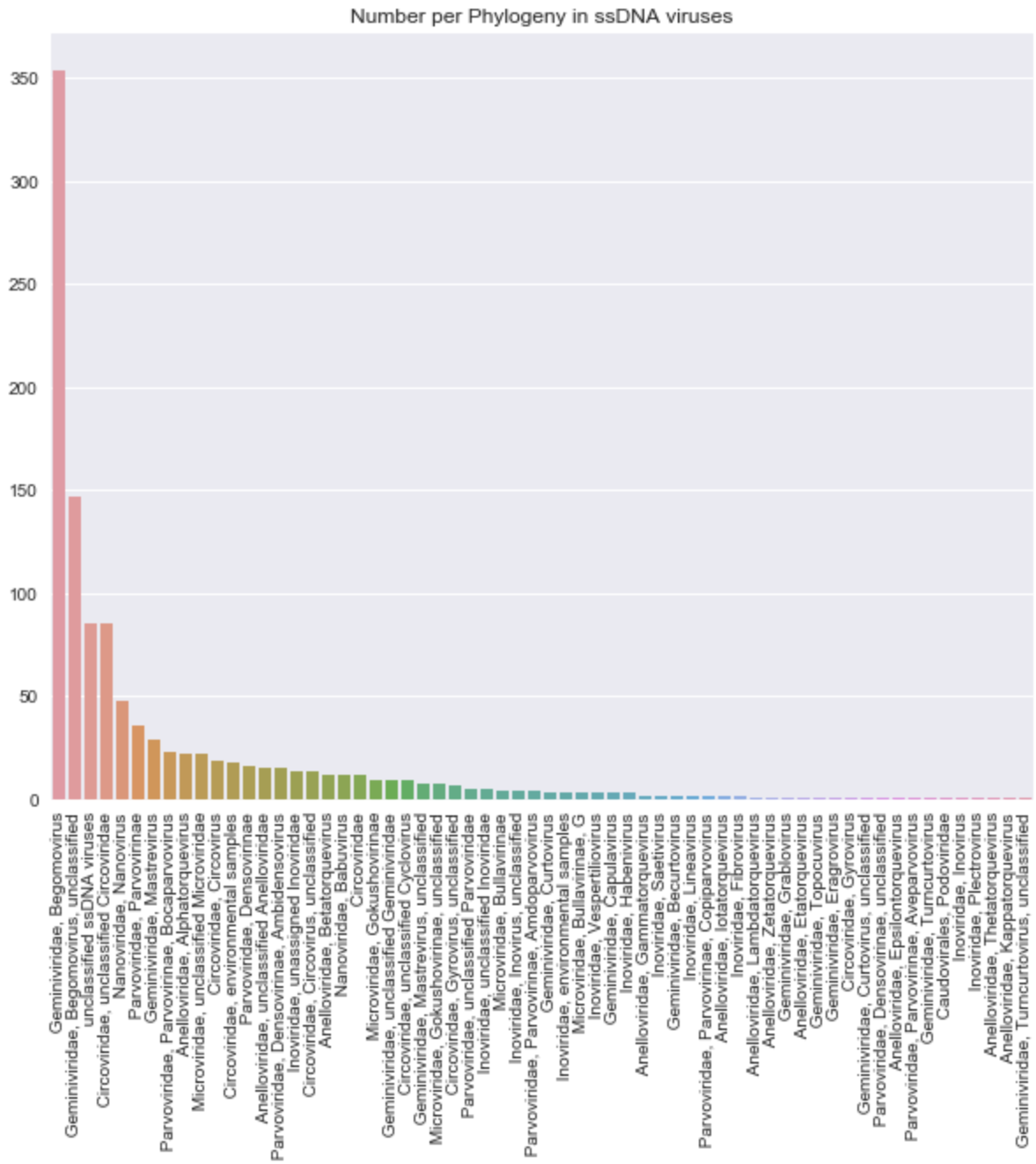

Fig. 6. Number of viruses with each phylogeny in the single stranded DNA virus classification of the viral genomes in the NCBI database.

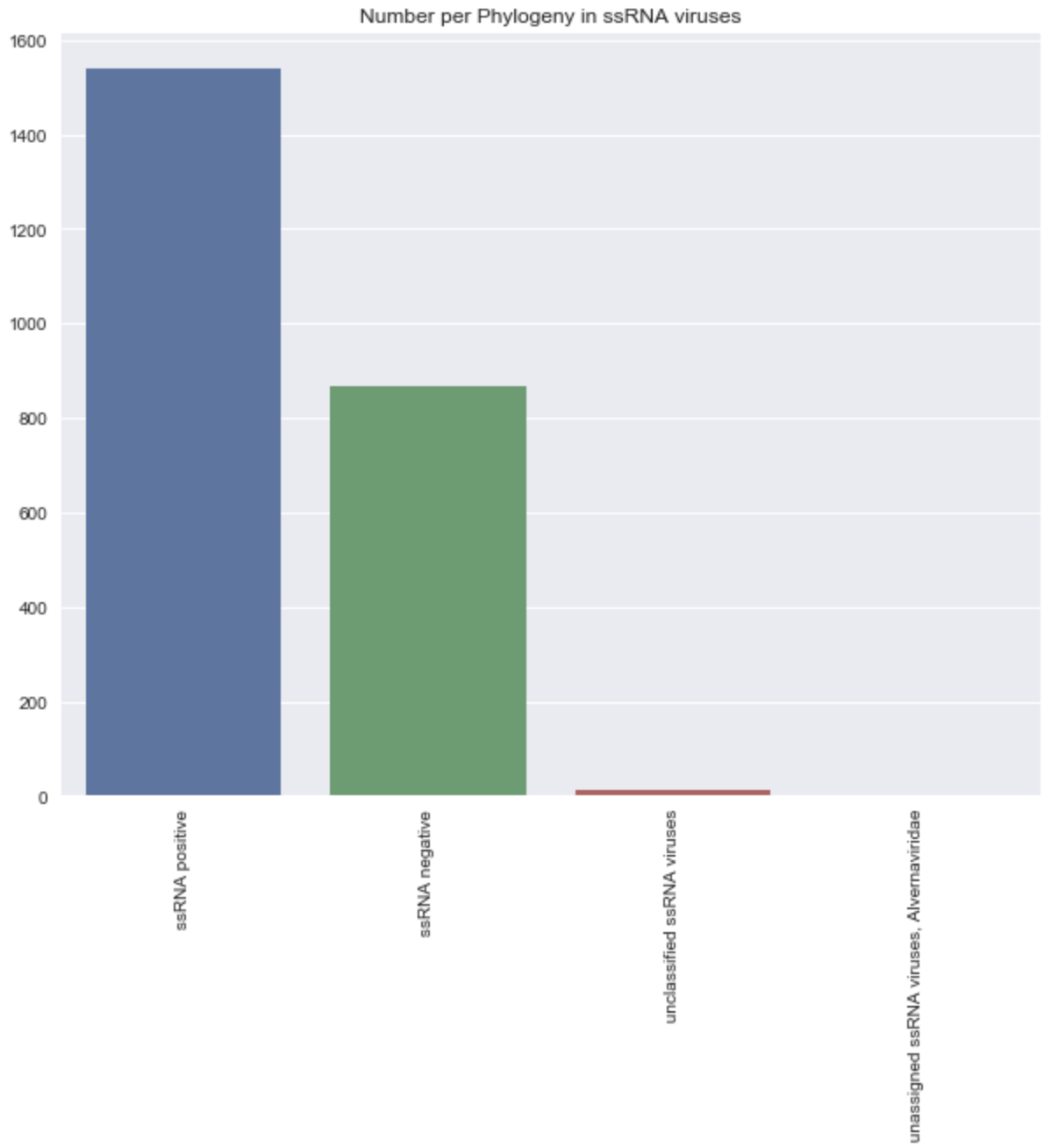

Fig. 7. Number of viruses with each phylogeny in the single stranded RNA virus classification of the viral genomes in the NCBI database.

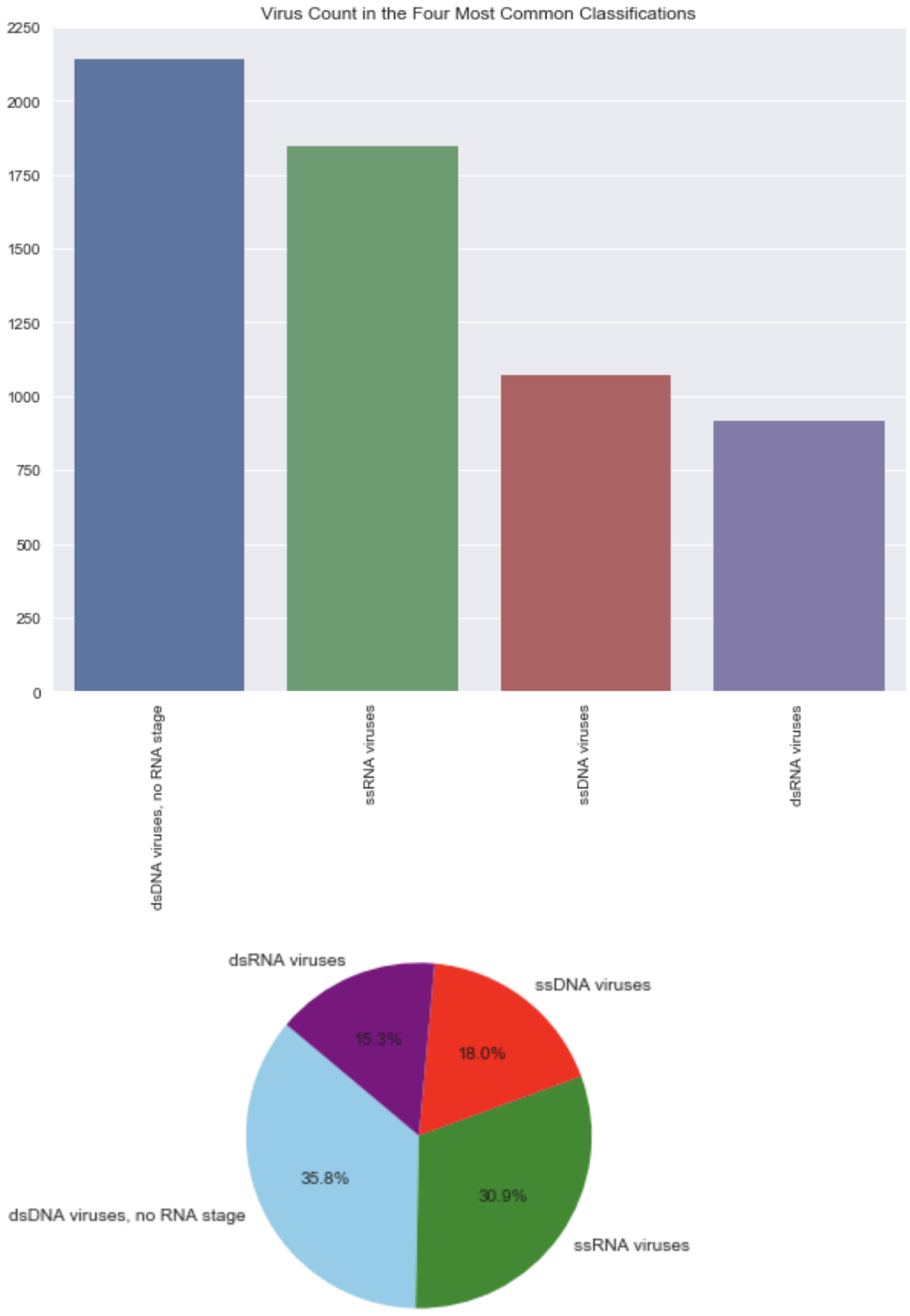

Fig. 8. Count and percentage of the four main classifications of virus sequences in the NCBI viral genomes database, shown as a bar plot (above) and pie chart (below).

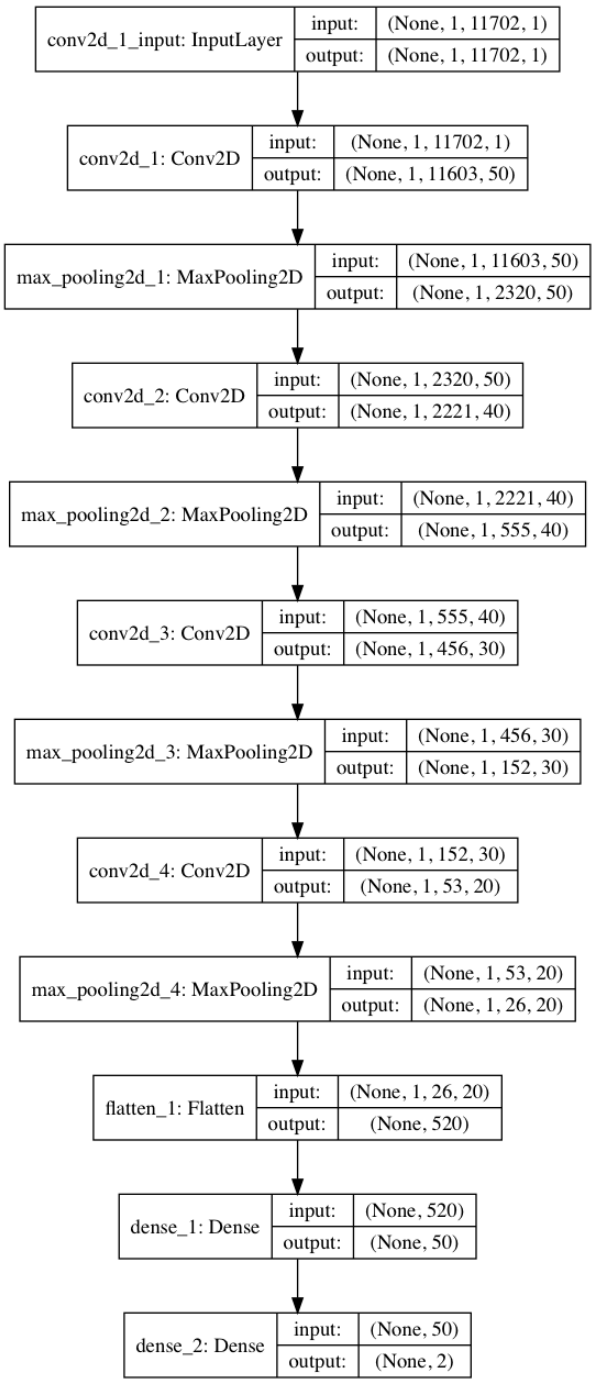

Fig 9. Visual plot of the convoluted neural network built to classify viruses from the Geminiviridae family out of the complete genomes in the single stranded RNA viruses from the NCBI viral genome database.