Review

The Viral Chase: Outsmarting Evolution with Data Trees and AI Predictions

Robert Friedman *

Posted Date: 5 June 2025

doi: 10.20944/preprints202506.0456.v1

Keywords: virus evolution; phylogenetic tree; genetic recombination; reassortment; transformer

 doi:10.20944/preprints202506.0456.v1

Review

The Viral Chase: Outsmarting Evolution with Data Trees and AI Predictions

Robert Friedman †

Department of Biological Sciences, University of South Carolina, Columbia, SC 29208, USA; bob.friedman.2@gmail.com

† Retired.

Abstract: In the ongoing battle against viral pathogens, staying one step ahead requires a blend of deep biological understanding, powerful computational tools, and intelligent foresight. This paper charts a course through these domains, showcasing a synthesized strategy for modern viral management. We start by examining the core processes of viral evolution and the biotechnological breakthroughs that unlocked the secrets of their genomes. We then navigate the world of "big data" in virology with the UShER project, demonstrating how Mutation Annotated Trees allow for efficient, large-scale tracking of viral spread and change. Our journey concludes by peering into the future with advanced AI, illustrating how Transformer models are being trained to predict viral evolutionary trends. This fusion of biological knowledge, data science, and artificial intelligence offers a more complete and dynamic arsenal for confronting the ever-evolving viral world.

Keywords: virus evolution; phylogenetic tree; genetic recombination; reassortment; Transformer

1. Introduction

Viruses are masters of change, constantly evolving in ways that keep scientists on their toes and pose ongoing threats to global health. To keep pace, we need increasingly smart tools to quickly spot, understand, and even predict their next moves.

This paper brings together exciting progress from three key areas that, when combined, give us a much stronger playbook for tackling viral outbreaks and preparing for what the future might hold.

First, we'll explore the fast-paced world of viral evolution itself – how viruses change and adapt. We'll also look at the groundbreaking biotechnologies that have completely transformed how we can study, tweak, and even rebuild viral genetic material (genomes).

With that biological foundation, we'll then introduce you to the UShER project. Think of UShER as a super-efficient computational system built to keep a close watch on viral genomes on a massive scale. It helps scientists manage and make sense of huge family trees of viruses, which is crucial for tracking how they spread and change. We'll also touch on how researchers can practically use these advanced tools.

Finally, we'll venture into the exciting new frontier of using advanced artificial intelligence, specifically powerful models called Transformer architectures. We'll see how AI is being trained to forecast where viruses might be heading evolutionarily and to uncover the intricate ways their genes interact.

By connecting these fields—understanding viral biology, using smart computational tools for surveillance, and harnessing AI for prediction—we gain a more complete and proactive way to face the ever-shifting landscape of viruses.

2. The Fast-Changing World of Viruses and the Tools We Use to Study Them

Even though they're microscopic, viruses are incredibly dynamic creatures, always changing through a few key tricks. They can undergo tiny tweaks in their genetic instruction manual (their genome) – these are called **mutations**. Sometimes, different viruses can swap pieces of their genetic

 doi:10.20944/preprints202506.0456.v1

material, a process known as **recombination**. And for viruses whose genomes are in separate pieces, like a deck of cards, they can shuffle these entire segments if two different strains infect the same cell – this is called **reassortment** [1].

These powerful abilities mean viruses can quickly adapt to new homes (like new animal hosts or humans), cleverly dodge our immune defenses, and even become resistant to the medicines we use to fight them [1,2]. Recombination and reassortment are especially dramatic, sometimes causing big, sudden shifts in a virus's makeup. These shifts can change how sick a virus makes us, what animals it can infect, or whether our bodies even recognize it anymore [1,2]. The flu is a classic example: many major flu pandemics have been traced back to reassortment, where different flu viruses mixed and matched their genes, creating new, dangerous strains that posed serious threats to public health [3].

To keep up with these shape-shifting enemies, scientists have developed some truly revolutionary tools. These biotechnologies have completely changed how we can study, handle, and even rebuild viruses.

One of the earliest game-changers was the creation of **cDNA libraries** [4]. Imagine you want to study a virus whose genetic material is made of RNA, which can be a bit unstable and tricky to work with directly. A cDNA library is essentially a collection of more stable DNA copies made from the virus's RNA messages (called messenger RNA, or mRNA). Here's a simplified idea of how it works:

1. Scientists first isolate the mRNA from the virus. Often, they grab onto a unique "tail" (called a poly-A tail) that most mRNA molecules have.
2. A special enzyme, **reverse transcriptase** (which famously does the reverse of what usually happens in our cells), then uses the mRNA as a template to build a matching strand of DNA. This creates a hybrid molecule, half RNA and half DNA [4–7].
3. The original RNA strand is then removed, and another enzyme builds a second DNA strand, resulting in a stable, double-stranded DNA (ds-cDNA) copy of the original RNA message.
4. This ds-cDNA can then be inserted into tiny biological carriers (like plasmids or bacteriophages, which are viruses that infect bacteria) that can make many copies of it inside host cells like *Escherichia coli*. All these copied DNA pieces together form the cDNA library [7].

This trick of turning fragile RNA into sturdy, easy-to-handle DNA is incredibly important, especially for studying RNA viruses (like influenza or HIV). It allows scientists to get the complete genetic sequence of a virus or to produce viral proteins needed for developing vaccines or understanding how the virus is built [8].

Sometimes, if a virus (or a cell) is making a lot of one particular mRNA, that message can overwhelm the library. To get a more balanced picture of *all* the genes present, especially the rare ones, scientists can create "normalized" cDNA libraries. These libraries are adjusted to reduce the over-representation of common messages, giving a fairer snapshot of everything the virus is expressing [9].

Building on the cDNA technology we just discussed, **plasmid technology** soon became another essential item in the scientist's toolkit [10]. What are plasmids? Imagine tiny, circular pieces of DNA, much smaller than a cell's main chromosome (typically 3,000 to 6,000 DNA "letters" or base pairs long). They live inside bacteria like *E. coli* but can copy themselves independently [11].

Scientists have cleverly engineered these plasmids for easy use. A basic workhorse plasmid usually includes:

- An "ori" (origin of replication), which is like the "start" signal for making copies.
- An antibiotic-resistance gene. This is a neat trick: if you grow bacteria in the presence of an antibiotic, only the bacteria that have successfully taken up your plasmid (and its resistance gene) will survive, making them easy to find.

 doi:10.20944/preprints202506.0456.v1

- A "multiple cloning site" (MCS), which is basically a stretch of DNA with several unique spots where scientists can easily insert a foreign piece of DNA – like the viral cDNA we talked about [10,11].

For plasmids designed to make proteins, this MCS is cleverly placed right after a "promoter." A promoter acts like an "on" switch, telling the cell's machinery to read the inserted gene and make the protein it codes for [10]. This system allows scientists to make lots of copies of viral genes or to produce specific viral proteins, which are invaluable for research or developing new vaccines [12].

3. Bringing Viruses to Life: Building and Rebuilding Genomes

These fundamental tools – like cDNA and plasmids – weren't just small steps; they blew the doors open for incredible breakthroughs in a field called synthetic biology.

Picture this: In 2002, a team led by Eckard Wimmer did something truly astounding [13]. They chemically built the entire genome of the poliovirus from scratch – without using any natural virus or even living cells as a starting point! They did this by carefully stringing together custom-made short pieces of DNA (called oligonucleotides, each about 69 DNA letters long) to create a full-length DNA version of the polio genome (about 7,500 letters). Then, using an enzyme, they copied this synthetic DNA into viral RNA. The most magical part? When they put this lab-made RNA into a special soup (a cell-free extract from human cells), it "booted to life," producing real, infectious poliovirus. This was a landmark moment. It proved that a living, replicating entity could be created purely from its genetic blueprint, showing that viruses are, in essence, complex molecular machines that can be assembled from their chemical parts.

In another monumental feat, a team led by Jeffery Taubenberger didn't build from scratch but instead pieced together a ghost from the past: the infamous 1918 "Spanish" flu virus [14]. This was like a painstaking historical reconstruction. They carefully extracted tiny, broken fragments of viral RNA from preserved lung tissue samples taken from victims of that devastating pandemic nearly a century earlier. Using sophisticated molecular techniques and a process called "reverse genetics" (essentially working backward from the RNA fragments to reconstruct the DNA, and then reassembling the viral genome), they managed to sequence and rebuild the entire genetic code of the 1918 H1N1 influenza virus by 2005.

This wasn't just an academic exercise; others were able to bring the live 1918 virus back to life in secure labs [15]. This gave scientists priceless insights into why it was so deadly and where it came from. Crucially, this research confirmed something vital: later influenza A pandemics were indeed caused by descendants of the 1918 virus that had picked up new genes by swapping bits with other flu viruses (that reassortment process we mentioned earlier). This hammered home just how important this natural gene-swapping is for creating new viruses with the potential to cause global pandemics.

Now, let's look at some even more cutting-edge tools. You've likely heard of **CRISPR-Cas systems** – they've truly revolutionized our ability to edit genes with incredible precision [16]. Think of CRISPR as a natural defense system found in bacteria and other simple organisms called archaea. It's their way of fighting off invading viruses. How does it work? When a virus attacks, these microbes can grab small snippets of the virus's DNA (called "spacers") and weave them into their own genetic code [17,18]. These stored spacers then get turned into guide molecules (CRISPR RNAs or crRNAs). If the same virus tries to invade again, these guides, along with special Cas proteins (like tiny molecular scissors), find the matching viral DNA and chop it up, neutralizing the threat [19].

The most famous version scientists have adapted for gene editing is **CRISPR-Cas9**. It uses the Cas9 enzyme (the "scissors") and a specially designed "guide RNA" (gRNA) [20]. This gRNA is like a programmable GPS: one part of it latches onto the Cas9 enzyme, and another part (a sequence of about 20 DNA letters) precisely matches the spot in the DNA that scientists want to change.

4. Super-Precise Editing and Watching Evolution Happen Live

 doi:10.20944/preprints202506.0456.v1

So, the gRNA leads Cas9 to the exact target DNA sequence. For Cas9 to actually cut, there's one more little requirement: a short specific sequence called a PAM must be sitting right next to the target [20]. Once Cas9 binds and cuts, it creates a clean break in both strands of the DNA [20]. At this point, scientists can cleverly use the cell's own DNA repair crews. One option is a quick-and-dirty fix called Non-Homologous End Joining (NHEJ). It often makes small mistakes, like adding or deleting a few DNA letters, which can be useful if you want to shut down a gene. Alternatively, if scientists provide a "template" piece of DNA with the desired changes, the cell can use a more precise repair system called Homology Directed Repair (HDR) to make exact corrections or insert new genetic information [21].

In the world of virus research, CRISPR-Cas is a game-changer. It allows scientists to edit viral genes with pinpoint accuracy, or even tweak the genes of host cells to make them resistant to viral infection [22]. This technology is buzzing with potential for creating new antiviral drugs, developing gene therapies, and speeding up vaccine design by letting us precisely modify viral genomes.

But what if we want to watch evolution happening right before our eyes? That's where **Experimental Evolution (EE)** comes in [23]. In EE studies, scientists grow huge populations of viruses in the lab, generation after generation, under controlled conditions. This lets them directly observe how viruses adapt, mutate, and swap genes (through recombination or reassortment) over time. Viruses are perfect for this because they reproduce incredibly fast and mutate a lot. A really cool feature of EE is that scientists can freeze samples of the viruses at different time points. Later, they can "revive" these ancestral and evolved viruses to directly compare them and measure how their "fitness" (like their ability to replicate or infect) has changed. EE helps us understand how quickly different genetic changes pile up, and importantly, how events like two different viruses infecting the same cell (co-infection) lead to gene-swapping that can help viruses adapt, jump to new hosts, or become drug-resistant.

Finally, imagine you could test how *every single possible tiny change* in a virus's genetic code affects its ability to do its job – like replicating, latching onto cells, or dodging our immune system. That's essentially what **Deep Mutational Scanning (DMS)**, also called massively parallel mutagenesis, allows us to do [24]. Scientists create vast libraries containing nearly every possible single-letter mutation of a viral gene. They then test all these mutant versions at once (in a "high-throughput" way) and use sophisticated sequencing to see which mutations help the virus thrive under specific conditions [25]. This creates incredibly detailed "fitness maps" that show which parts of a viral protein are absolutely essential and which parts can tolerate changes [24,25]. DMS has been a goldmine for understanding how viruses like influenza and SARS-CoV-2 evolve, for pinpointing critical spots to target with vaccines, and for predicting how viruses might escape treatments [26,27]. While DMS mostly looks at one mutation at a time, the fitness maps it generates are also vital for understanding the impact of more complex genetic shake-ups, like those from recombination. It can even reveal "epistasis," where the effect of one mutation depends on whether other mutations are present or absent – like how a team player's performance can depend on who else is on the field.

Transition: From Understanding Viruses to Tracking Them on a Global Scale

So, we've seen how viruses are constantly changing and explored some amazing biological tools that let us study them up close. But all this research, especially during outbreaks, generates a mountain of genetic data from viruses. To protect public health, we need to be able to sort through, analyze, and make sense of this data – fast. This is where powerful computer tools become absolutely essential, especially for understanding how viruses are evolving and spreading on a massive, even pandemic, scale. And that brings us to the UShER project [28].

5. Keeping Tabs on Viruses Worldwide: The UShER Project and Mutation Trees

Because viruses evolve so quickly, scientists urgently need fast and powerful computer tools to track these genetic changes almost as they happen. This is exactly what the **UShER project** [28] was designed for. (UShER stands for Ultrafast Sample placement on Existing tRees – a bit of a mouthful,

 doi:10.20944/preprints202506.0456.v1

but it describes what it does!). Especially during the SARS-CoV-2 pandemic, with millions of viral genomes being sequenced, UShER became a critical piece of the puzzle. It provides the backbone for rapidly analyzing viral "family trees" (phylogenetic analysis), which is vital for tasks like watching new variants emerge in real-time (genomic surveillance) and figuring out how infections are spreading (contact tracing). UShER's secret sauce is its incredibly smart design. It's built for speed and doesn't get bogged down by the clunky data formats that older systems often used.

5.1. Smart Data Storage: The MAT Format and Protocol Buffers

At the heart of UShER's power is a special way of storing viral evolution data called the **Mutation-Annotated Tree (MAT) format** [29]. Traditional viral family trees might try to store the *entire* genetic sequence for every virus sample. Imagine how huge those files would get! MATs are much cleverer. They are specifically designed to pinpoint and store *only the genetic changes* (mutations) that occur along each branch of the evolutionary tree – from the common ancestor at the root all the way to the newest samples at the tips. This approach dramatically shrinks the amount of data needed, making the files much smaller and easier to handle, which is a lifesaver when you're dealing with information from millions of viruses [28].

To be this compact and speedy, UShER stores MAT data in a special kind of file called a **Protocol Buffer (.pb) file**. This is a highly efficient, binary format developed by Google for organizing and sending data [30]. Choosing Protocol Buffers was a smart move: it makes the MAT data over 300 times smaller than if it were stored in older, more common formats (like a combination of Newick tree files and VCF files, which list variations). Plus, it means the data can be loaded and processed incredibly quickly. Because Protocol Buffers are an open and flexible standard, it also means that MAT data can be easily shared and used by different software tools and programming languages. This encourages teamwork and makes the system adaptable enough to handle the "pandemic-scale phylogenomics" we mentioned – essentially, studying viral family trees on a global, pandemic level.

5.2. How UShER Works: Pre-Processing, Placement, and the Handy matUtils Toolkit

So, how does UShER actually do its job? It mainly works in two big steps: **pre-processing** and **placement**.

1. **Pre-processing:** Think of this as getting the main family tree ready. UShER takes existing viral family trees and their genetic data and cleverly organizes them into that super-compact MAT format we just talked about. It even figures out the likely genetic makeup of the ancestors in the tree. This initial prep work makes everything that comes next much faster and more efficient.
2. **Placement:** This is where the action happens with new virus samples. As new viral genomes are sequenced (say, from new patient samples), UShER rapidly figures out the best spot to add them to the existing, optimized tree. It does this by calculating the fewest genetic changes needed to connect the new sample to the tree. This means the global viral family tree is constantly and quickly updated. Being able to do this over and over again is crucial for watching how a virus is spreading and evolving, almost in real-time – a vital tool for public health officials.

Now, those MAT files are super-efficient, but their underlying binary format isn't something most researchers would want to wrestle with directly. That's where the **matUtils toolkit** comes in – it's a set of user-friendly command-line tools that act as a helpful go-between [29]. This toolkit is indispensable. It lets researchers do all sorts of useful things without needing to be a programming expert:

- Get quick summaries of the tree (like how many branches or samples it contains).
- Pull out smaller sections of the tree for a closer look at specific outbreaks or variants.
- Convert the MAT data into more familiar formats (like Newick for trees or VCF for mutation lists) so it can be used with other scientific software.

 doi:10.20944/preprints202506.0456.v1

- It even has advanced features, like helping to label new viral groups (clades) as they emerge or checking how confident UShER is about where a new sample fits in the tree.

5.3. Seeing the Big Picture: Visualizing Massive Trees with Taxonium

Analyzing all this data is one thing, but being able to *see* it is incredibly powerful. To complete the picture, the UShER system works hand-in-hand with **Taxonium**, a fantastic web-based tool built specifically for looking at incredibly large viral family trees [31,32].

Older visualization tools often choked when trying to display trees with millions of branches (nodes). Taxonium, however, can smoothly show trees with tens of millions of nodes right in your web browser! It's interactive, too. Users can zoom in and out, explore different parts of the tree, click on branches to get more details about specific viruses or mutations, search for particular changes, and even color-code the tree based on different information (like where samples came from or when they were collected).

The ability to easily take the trees generated by UShER and view them in Taxonium creates a smooth path from rapid data crunching to clear, understandable visuals. This powerful combination doesn't just look cool; it genuinely speeds up scientific discovery and helps public health teams make sense of complex genetic data, turning it into actionable insights.

Transition: From Powerful Tools to Practical Steps

We've seen how the UShER project offers incredibly sophisticated ways to manage and analyze huge amounts of viral family tree data. But knowing the tools exist is one thing; actually *using* them is another. So, how do researchers get hands-on with these Mutation-Annotated Tree files to uncover valuable insights? The next section gives a glimpse into the practical side of things.

6. Getting Your Hands Dirty: A Quick Guide to Working with MAT Files

The UShER system isn't just about fancy algorithms; it also provides practical ways for researchers to dive into large-scale genetic data, like the MAT files for SARS-CoV-2. Here's a typical way someone might get started, often using a cloud-based computer setup like Google Colab [33] (which is like having a powerful computer accessible through your web browser):

1. **Set Up Shop:** First, a researcher would set up their digital workspace. This usually means connecting to their cloud storage (like Google MyDrive) to keep files organized. Then, they'd install the necessary bioinformatics software. A handy tool called Conda [34] often helps manage these software installations, including the UShER toolkit (which, as we learned, contains the useful matUtils commands).
2. **Get the Data:** Once the virtual lab is ready, they can download the latest public MAT file. Remember, this file is in that super-efficient Protocol Buffer (.pb) format.
3. **Put matUtils to Work:** With the MAT file downloaded, the matUtils commands become the researcher's best friend. These commands can "unpack" the compressed binary file and pull out all sorts of information. For example:
    - matUtils summary can quickly give basic facts about the tree (like how many viruses are in it).
    - matUtils extract is really flexible. It can be used to grab just a specific part of the big tree for a closer look, or to convert the MAT data into other common formats that different software programs can understand (like Newick files for tree structures or VCF files for lists of mutations).

This straightforward approach allows scientists to take full advantage of the highly efficient MAT data format for their genomic surveillance and analysis work (Appendix A) [35].

 doi:10.20944/preprints202506.0456.v1

Transition: Peering into the Future with a New Kind of AI

So far, we've journeyed through how viruses evolve, how we track their massive datasets, and even how scientists can practically work with this information. But what if we could go a step further? What if we could use the smartest computer tools available to get a glimpse into where viruses might be heading *next*? This is where a truly exciting field of artificial intelligence comes into play, using powerful models called **Transformer architectures** [36]. Building on all the rich genetic information we've gathered (like that stored in MAT files), these AI models are being trained to forecast the evolutionary paths of viruses.

7. Crystal Ball Computing: Using AI (Transformers) to Predict Viral Evolution

We're now at the cutting edge of computational biology, exploring how advanced AI models known as **Transformer architectures** can help us predict future viral versions and their evolutionary journeys [36]. Now, "predicting the future" with viruses isn't like predicting tomorrow's weather with certainty. Evolution is a game of chance and probabilities. So, the goal isn't to pinpoint one exact "future virus" but rather to figure out the *likely* paths evolution might take, or the *probabilities* of certain changes happening.

What kind of "future" are we talking about? It could be:

- Predicting which new viral variant might become dominant.
- Forecasting which mutations are likely to pop up on specific branches of the viral family tree.
- Identifying changes that could help a virus escape our immune system or become resistant to treatments (this is called antigenic drift or immune escape).
- Even helping scientists design hypothetical new virus sequences with desired features – perhaps for making safer vaccines (e.g., a version with low ability to cause disease but still triggers immunity).

7.1. How Transformers Get Smart About Viral Genes

Transformer models, a special kind of AI, can be taught to understand viral evolution data in several powerful ways:

- **Learning to "Translate" Old Viruses into New Ones (Seq2Seq Models [37]):** Imagine you have the genetic sequence of an ancestor virus and the sequence of one of its direct descendants. This approach treats the problem like translating one language into another. The ancestor sequence is the "input sentence," and the descendant (or "future") sequence is the "translated output." The Transformer's "encoder" part reads the entire ancestor sequence, building a deep understanding of every genetic letter and its context. Its special "self-attention" ability lets it potentially consider the whole genome at once when looking at any single spot. Then, its "decoder" part uses this understanding to build the descendant sequence, one genetic letter at a time. As it predicts each new letter, it can look back at the ancestor sequence *and* the part of the descendant sequence it has already built. This helps the AI learn the "rules" of how viruses change – which spots are likely to mutate, and how different mutations might influence each other (remember epistasis, where one mutation's effect depends on others?).
- **Creating Brand New, Believable Virus Sequences (Generative Models):** Some Transformers are designed to be creative. They can learn the underlying "grammar" or patterns of real viral sequences. Once trained, these **generative models** can dream up completely new viral sequences that, while novel, still look and behave like plausible, real-world viruses. Scientists can use these models to explore the vast universe of potential future variants or even to help design artificial sequences for things like new vaccine candidates.

 doi:10.20944/preprints202506.0456.v1

- **Predicting a Virus's Success (Fitness or Escape Potential):** Transformers can also be trained like judges. You feed them a viral sequence (or just a key part, like the spike protein of SARS-CoV-2), and they output a score. This score could predict how "fit" the virus is (how well it can survive and spread), its potential to outgrow other variants, or its likelihood of dodging our immune defenses. This directly tackles the big question: which new viral versions are most likely to succeed in a population? To learn this, the AI is often trained on real-world data, like results from those Deep Mutational Scanning experiments we talked about, or how common different variants are in actual outbreaks.
- **Uncovering Hidden Teamwork Between Mutations (Modeling Epistasis):** Remember how Transformers can potentially pay "attention" to the whole sequence at once? This makes them exceptionally good at spotting and understanding complex, long-distance relationships between different mutations across a virus's entire genome. This is crucial for modeling **epistasis**, where the impact of one mutation is tied to whether other specific mutations are also present or absent – like a complex team play in sports.

7.2. Teaching AI About Viral Family Trees and Handling Super-Long Genetic Codes

When we use these AI Transformer models, it's not just about looking at individual virus sequences in isolation. Viruses evolve as part of a big, branching family tree (a phylogenetic tree), and it's really important for the AI to understand this historical context. Here are a few ways scientists help Transformers "see" the tree:

1. **Learning from Ancestors and Descendants (Implicit Integration):** This is the most common trick. Scientists use the family tree to pick out pairs of viruses where one is a direct ancestor of the other. Each of these "parent-child" sequence pairs becomes a lesson for the AI. The AI sees thousands, or even millions, of these real-life evolutionary steps and gradually learns the "rules" of how viruses change from one generation to the next directly from the tree's structure.
2. **Giving the AI Extra Clues from the Tree (Feature Engineering):** Think of this like giving the AI some extra notes about each virus's place in the family. Scientists can calculate various numbers from the tree for each virus: how long its branch is (which can represent time or the number of mutations since its parent), how big its particular family group (clade) is, what its reconstructed ancestor might have looked like, or how closely related it is to important reference viruses. These numerical clues can then be fed into the Transformer along with the genetic sequence itself, giving the AI more context.
3. **Teaming Up with Other AI: Transformers + Graph Neural Networks (Hybrid Models):** This is a more advanced strategy. It involves combining Transformers with another type of AI called Graph Neural Networks (GNNs) [38]. GNNs are superstars at learning from network-like structures – and a family tree is a perfect example! The GNN can first create a smart summary (an "embedding") for each virus based on its position in the tree. This tree-based summary is then combined with the virus's genetic sequence information and fed into the Transformer. It's a cutting-edge approach that really tries to get the best of both worlds.

7.3. Dealing with DNA Overload: Making Transformers Work for Long Viral Genomes

Now, there's a practical hurdle. Viral genomes can be quite long. The SARS-CoV-2 virus, for instance, has a genetic code of about 30,000 "letters" (base pairs) [39]. The part of a Transformer that makes it so smart – its "self-attention" mechanism – gets computationally very expensive very quickly as sequences get longer. (Technically, the work it has to do often grows with the square of the

 doi:10.20944/preprints202506.0456.v1

sequence length, or $O(L^2)$). This can mean needing way too much computer memory and processing power. Luckily, scientists have clever workarounds:

- **Chopping it Up (Chunking / Sliding Window):** One straightforward idea is to divide the long genome into smaller, more manageable, often overlapping pieces (say, 512 or 1024 letters at a time). Standard Transformers can then work on these smaller chunks. It's simpler to set up and can use less memory for each chunk. The downside? The AI might miss the "big picture" connections that span across the entire genome, and the boundaries between chunks can sometimes cause minor issues.
- **Smarter Transformers for Long Reads (Specialized Long-Sequence Transformers):** Researchers have also designed new types of Transformers specifically for handling long sequences [40,41]. You might hear names like **Longformer, Reformer, Performer,** or **BigBird** [42]. These models use clever tricks to make the attention mechanism more efficient (reducing the computational load to something like $O(L \log L)$ or even $O(L)$). This means they can look at much longer sequences and still pay attention across the entire genome, which is really important for catching those tricky long-distance genetic interactions (epistasis). If the computer power is available, these specialized models are generally preferred over simple chunking, even though they can still be demanding on resources and might need more careful fine-tuning [40].
- **Zooming in on Key Genes (Focus on Specific Genes/Proteins):** Often, instead of tackling the whole genome, researchers will focus the AI on just one or a few key viral genes or the proteins they code for. For example, with SARS-CoV-2, a lot of attention is on the Spike protein (which is about 3,800 letters long) [43]. These shorter segments are much easier for standard Transformers to handle and often contain the mutations that are most important for how the virus spreads and makes us sick.

7.4. Teaching AI the ABCs (or ACGTs) of Viral Genetic Code

Before a Transformer AI can work its magic on viral genetic sequences, we need a way to translate the letters of the genetic code (A, C, G, and T) into a language computers understand: numbers. This translation process is called **tokenization** [44].

You might know that AI models for human languages (like English) use complex "tokenizers" to break down words and sentences. But for viral genomes, things are much simpler. The "alphabet" of DNA or RNA is very small – just A, C, G, and T, plus maybe a symbol for an unknown letter ('N') or for "padding" (we'll get to that in a moment). So, instead of a fancy tokenizer, scientists can just set up a straightforward, direct mapping:

- 'A' might become 0
- 'C' might become 1
- 'G' might become 2
- 'T' might become 3
- An 'N' (for an unknown base or a gap in the sequence) or a special padding symbol might become 4.

It's crucial that this system is consistent. For example, DNA sequences sometimes use lowercase letters (like 'a', 'c', 'g', 't'), perhaps to mark certain regions. The tokenizer needs to be smart enough to treat 'a' the same as 'A' before converting it to its number. This makes sure all valid genetic letters are correctly understood, and any unexpected characters are handled gracefully (often by assigning them the "unknown" code). Two other standard steps are also vital when preparing viral sequences for Transformers:

 doi:10.20944/preprints202506.0456.v1

1. **Padding:** Not all viral sequences (or chunks of sequences) will be the exact same length. To feed them to the AI efficiently, shorter sequences are "padded" out with a special padding token (like our 'N' or code 4) until they all reach a standard maximum length.
2. **Attention Masks:** Because we've added these padding tokens, we need to tell the AI which parts of the sequence are real genetic data and which parts are just padding. An "attention mask" does this – it's like giving the AI a note saying, "Pay attention to these tokens, but ignore these other ones."

These preprocessing steps ensure the data is in the perfect format for the Transformer to learn from (Appendix B).

Funding: This research received no external funding.

Conflicts of Interest: The author declares no conflict of interest.

Acknowledgments: The conceptual development and drafting of this essay benefited significantly from discussions and iterative refinement with an AI language model, Gemini 2.5 Pro, (Google, 5/6/2025).

Appendix A. Processing of SARS-CoV-2 Mutation Data

```
# 1. Mount Google MyDrive in Google Colab
from google.colab import drive
import os

# Mount Google MyDrive for use in Google Colab
# In Colab: activate all allowable permissions for access to MyDrive
# to bypass any authentication error
drive.mount('/content/drive')
os.chdir('/content/drive/MyDrive')

# 2. Installation of Conda in Google Colab
!pip install -q condacolab
import condacolab; condacolab.install()
# Initialize the shell and restart the kernel
# Colab expectedly restarts with a log/report on `crash reported`
!conda init bash
# Verify installation
!conda --version

# 3. Installation of the UShER toolkit via Conda
# See documentation for other setup options:
# https://usher-wiki.readthedocs.io/en/latest/Installation.html
# Create a new environment for UShER
!conda create -n usher-env # python=3.10, if installed, to support BTE library
# Activate the new environment
!conda activate usher-env

# Set up channels
```

 doi:10.20944/preprints202506.0456.v1


```
!conda config --add channels defaults
!conda config --add channels bioconda
!conda config --add channels conda-forge
# Install package
!conda install -q usher

# 4. Download the latest UShER Mutation-Annotated Tree (MAT) data (.pb file; compressed)
!wget http://hgdownload.soe.ucsc.edu/goldenPath/wuhCor1/UShER_SARS-CoV-2/public-latest.all.masked.pb.gz

# Uncompress the MAT data file (-f parameter will force a file overwrite)
!gunzip -f public-latest.all.masked.pb.gz

# Export summary data associated with MAT file (e.g., --clades, --node-stats,
# --mutations, --samples, --get-all)
!matUtils summary --input-mat public-latest.all.masked.pb --clades clades.tsv
# !matUtils summary --input-mat public-latest.all.masked.pb --samples samples.txt

# 5. Obtain mutation data for each node of the subtree

# If any issues arise, verify that public-latest.all.masked.pb is in the current working directory
# Replace "YOUR_CLADE_OF_INTEREST" with the actual clade name, e.g., "20H (Beta)"
# May replace "mutations_for_clade.txt" with another output filename

# Tested with SARS-CoV-2 clade `20H (Beta)` (10179 samples):
# If scaling up to larger clades, note the full SARS-CoV-2 dataset is ~800x as large

!matUtils extract \
    --input-mat public-latest.all.masked.pb \
    --clade "YOUR_CLADE_OF_INTEREST" \
    --all-paths mutations_for_clade.txt

# Explanation of the command:
# `--input-mat public-latest.all.masked.pb`: Specifies the input MAT file.
# `--clade "YOUR_CLADE_OF_INTEREST"`: Focuses the extraction on the members of the named
#   clade. This name must exactly match a clade name present in the MAT file's metadata.
#   May specify multiple clade names as a comma-delimited list. Add double quotes to
#   names with spaces.
# `--all-paths mutations_for_clade.txt`: This crucial option tells `matUtils` to output the mutations
#   along each path from the clade's common ancestor to every sample and internal node within
#   that clade. The output is saved to ` mutations_for_clade.txt`. The list is created by a depth-first
#   traversal order.

# Output Format:
```

 doi:10.20944/preprints202506.0456.v1


```
# The output file (`mutations_for_clade.txt`) will typically list each node (internal nodes often
# labeled like `node_X:` or sample (e.g., `Country/SampleID/Date|Accession|Date:`) followed by
# the mutations inferred to have occurred on the branch immediately leading to it. For example:
#   node_1: G15910T
#   Sample/ID/Date|Accession|Date: C1191T,C11674T
#   node_2: T13090C

# This detailed mutation information is invaluable for understanding the specific evolutionary
# changes within a lineage and can serve as input for further analyses, including preparing data for
# training predictive models like Transformers.

"""
# (Optional) Convert VCF formatted file to Fasta formatted sequence data
# The vcf2fasta binary fails to run in Colab or WSL of Windows 11. Untested in other environments.

# Installation of VCF library
!conda install -q vcflib
!echo "Current working directory: $PWD"

# Download reference sequence for reconstruction of Fasta sequences from VCF file
!wget https://hgdownload.soe.ucsc.edu/goldenPath/wuhCor1/chromosomes/NC_045512v2.fa.gz
!gunzip -f NC_045512v2.fa.gz

# The VCF file contains lists of variants in the nucleotide sequence of the genotypes and depends
# on a Fasta formatted reference sequence to reconstruct full sequences (specified by --reference).
!vcfindex my_clade.vcf > my_clade_idx.vcf
!vcf2fasta --reference NC_045512v2.fa my_clade_idx.vcf
"""
```

Appendix B. Nucleotide Tokenization Code for Transformers (Proof of Concept)

```
# import pytorch library
import torch

# Define a simple vocabulary mapping
nuc_to_id = {'A': 0, 'C': 1, 'G': 2, 'T': 3, 'N': 4} # 'N' for unknown/gap, map to padding or UNK
id_to_nuc = {0: 'A', 1: 'C', 2: 'G', 3: 'T', 4: 'N'}

# Define special tokens and their IDs
PAD_TOKEN_ID = 4 # Using 'N' as padding/unknown for simplicity
MAX_SEQ_LEN = 512 # Your chosen chunk size for Transformer input

def encode_sequence(seq_str):
# Converts a nucleotide sequence string to a list of integer IDs, converting all nucleotides
# to uppercase and mapping unknown characters to PAD_TOKEN_ID
```

 doi:10.20944/preprints202506.0456.v1


```
    return [nuc_to_id.get(nuc.upper(), PAD_TOKEN_ID) for nuc in seq_str]

def prepare_chunk_for_transformer(chunk_str):
    # Encodes, pads, and creates an attention mask for a single sequence chunk,
    # Preparing it for input into a Transformer model
    encoded_ids = encode_sequence(chunk_str)

    # Pad the sequence to MAX_SEQ_LEN
    padding_length = MAX_SEQ_LEN - len(encoded_ids)
    input_ids = encoded_ids + [PAD_TOKEN_ID] * padding_length

    # Create attention mask (1 for real tokens, 0 for padding)
    attention_mask = [1] * len(encoded_ids) + [0] * padding_length

    # Convert to PyTorch tensors
    input_ids_tensor = torch.tensor(input_ids, dtype=torch.long)
    attention_mask_tensor = torch.tensor(attention_mask, dtype=torch.long)
    return input_ids_tensor, attention_mask_tensor

# Example usage (for one ancestor-descendant pair)
# The full ancestral_chunk string is abbreviated below
ancestor_chunk = "GTACGTACGTACGTACGTACGTAC...gtacgtacgtacgtacgtacgtacgtacgtac"
```

References

1. Lowen, A. C. (2017) Constraints, Drivers, and Implications of Influenza A Virus Reassortment. *Annual Review of Virology*, 4, 105-21. https://doi.org/10.1146/annurev-virology-101416-041726
2. Domingo, E., Martin, V., Perales, C., Grande-Pérez, A., García-Arriaza, J., & Arias, A. (2006) Viruses as quasispecies: biological implications. *Current Topics in Microbiology and Immunology*, 299, 51-82. https://doi.org/10.1007/3-540-26397-7_3
3. Hay, A.J., Gregory, V., Douglas, A.R., & Lin, Y.P. (2001) The evolution of human influenza viruses. *Philosophical Transactions of the Royal Society B: Biological Sciences*. 356, 1861-70. https://doi.org/10.1098/rstb.2001.0999
4. Rougeon, F., Kourilsky, P., & Mach, B. (1975) Insertion of a rabbit β-globin gene sequence into an E. coli plasmid. *Nucleic Acids Research*, 2, 2365-78. https://doi.org/10.1093/nar/2.12.2365
5. Temin, H. M., & Mizutani, S. (1970) RNA-dependent DNA polymerase in virions of Rous sarcoma virus. *Nature*, 226, 1211-13. https://doi.org/10.1038/2261211a0
6. Baltimore, D. (1970) RNA-dependent DNA polymerase in virions of RNA tumour viruses. *Nature*, 226, 1209-11. https://doi.org/10.1038/2261209a0
7. Sambrook, J., & Russell, D.W. (2001) Molecular Cloning: A Laboratory Manual (3rd ed.). Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, USA.
8. Kosuri, S., & Church, G. (2014) Large-scale *de novo* DNA synthesis: technologies and applications. *Nature Methods*, 11, 499–507. https://doi.org/10.1038/nmeth.2918

 doi:10.20944/preprints202506.0456.v1

9. Soares, M. B., Bonaldo, M. F., Jelene, P., Su, L., Lawton, L., & Efstratiadis, A. (1994) Construction and characterization of a normalized cDNA library. *Proceedings of the National Academy of Sciences USA*, 91, 9228-32. https://doi.org/10.1073/pnas.91.20.9228
10. Cohen, S. N., Chang, A. C. Y., Boyer, H. W., & Helling, R. B. (1973) Construction of Biologically Functional Bacterial Plasmids *In Vitro*. *Proceedings of the National Academy of Sciences USA*, 70, 3240-44. https://doi.org/10.1073/pnas.70.11.3240
11. Casali, N., & Preston, A. (Eds.) (2008) *E. coli* Plasmid Vectors: Methods and Applications (Vol. 235). Humana Press, Totowa, NJ, USA.
12. Geisbert, T. W., & Feldmann, H. (2011) Recombinant Vesicular Stomatitis Virus–Based Vaccines Against Ebola and Marburg Virus Infections. *The Journal of Infectious Diseases*, 204(Supplement 3), S1075-S1081. https://doi.org/10.1093/infdis/jir349
13. Cello, J., Paul, A. V., & Wimmer, E. (2002) Chemical Synthesis of Poliovirus cDNA: Generation of Infectious Virus in the Absence of Natural Template. *Science*, 297, 1016-18. https://doi.org/10.1126/science.1072266
14. Taubenberger, J. K., Reid, A. H., Lourens, R. M., Wang, R., Jin, G., & Fanning, D. G. (2005) Characterization of the 1918 influenza virus polymerase genes. *Nature*, 437, 889-93. https://doi.org/10.1038/nature04230
15. Tumpey, T. M., Basler, C. F., Aguilar, P. V., Zeng, H., Solórzano, A., Swayne, D. E., ... & García-Sastre, A. (2005) Characterization of the Reconstructed 1918 Spanish Influenza Pandemic Virus. *Science*, 310, 77-80. https://doi.org/10.1126/science.1119392
16. Doudna, J. A., & Charpentier, E. (2014) The new frontier of genome engineering with CRISPR-Cas9. *Science*, 346, 1258096. https://doi.org/10.1126/science.1258096
17. Mojica, F. J. M., Diez-Villasenor, C., Garcia-Martinez, J., & Soria, E. (2005) Intervening Sequences of Regularly Spaced Prokaryotic Repeats Derive from Foreign Genetic Elements. *Journal of Molecular Evolution*, 60, 174-82. https://doi.org/10.1007/s00239-004-0046-3
18. Barrangou, R., Fremaux, C., Deveau, H., Richards, M., Boyaval, P., Moineau, S., ... & Horvath, P. (2007) CRISPR Provides Acquired Resistance Against Viruses in Prokaryotes. *Science*, 315, 1709-12. https://doi.org/10.1126/science.1138140
19. Jinek, M., Chylinski, K., Fonfara, I., Hauer, M., Doudna, J. A., & Charpentier, E. (2012) A Programmable Dual-RNA–Guided DNA Endonuclease in Adaptive Bacterial Immunity. *Science*, 337, 816-21. https://doi.org/10.1126/science.1225829
20. Cong, L., Ran, F. A., Cox, D., Lin, S., Barretto, R., Habib, N., ... & Zhang, F. (2013) Multiplex Genome Engineering Using CRISPR/Cas Systems. *Science*, 339, 819-23. https://doi.org/10.1126/science.1231143
21. Adli, M. (2018) The CRISPR tool kit for genome editing and beyond. *Nature Communications*, 9, 1911. https://doi.org/10.1038/s41467-018-04252-2
22. Jiang, F., & Doudna, J. A. (2017) CRISPR-Cas9 Structures and Mechanisms. *Annual Review of Biophysics*, 46, 505-29. https://doi.org/10.1146/annurev-biophys-062215-010822
23. Elena, S. F., & Sanjuán, R. (2007) Virus Evolution: Insights from an Experimental Approach. *Annual Review of Ecology, Evolution, and Systematics*, 38, 27-52. https://doi.org/10.1146/annurev.ecolsys.38.091206.095637
24. Fowler, D. M., & Fields, S. (2014) Deep mutational scanning: a new style of protein science. *Nature Methods*, 11, 801-7. https://doi.org/10.1038/nmeth.3027
25. Meini, M.R., Tomatis, P. E., Weinreich. D. M., & Vila, A. J. (2015) Quantitative Description of a Protein Fitness Landscape Based on Molecular Features. *Molecular Biology and Evolution*, 32, 1774-87. https://doi.org/10.1093/molbev/msv059
26. Burton T. D., & Eyre, N. S. (2021) Applications of Deep Mutational Scanning in Virology. *Viruses*, 13, 1020. https://doi.org/10.3390/v13061020

 doi:10.20944/preprints202506.0456.v1

27. Starr, T. N., Greaney, A. J., Hilton, S. K., Ellis, D., Crawford, K. H., Dingens, A. S., ... & Bloom, J. D. (2020) Deep Mutational Scanning of SARS-CoV-2 Receptor Binding Domain Reveals Constraints on Folding and ACE2 Binding. *Cell*, 182, 1295-1310.e20. https://doi.org/10.1016/j.cell.2020.08.012
28. Turakhia, Y., Thornlow, B., Hinrichs, A. S., De Maio, N., Gozashti, L., Lanfear, R., ... & Corbett-Detig, R. (2021) Ultrafast Sample placement on Existing tRees (UShER) enables real-time phylogenetics for the SARS-CoV-2 pandemic. *Nature Genetics*, 53, 809-16. https://doi.org/10.1038/s41588-021-00862-7
29. Ultrafast Sample Placement on Existing Trees. Available online: https://github.com/yatisht/usher (accessed on 4 June 2025).
30. Protocol Buffers Documentation. Available online: https://protobuf.dev (accessed on 4 June 2025).
31. Sanderson, T. (2022) Taxonium, a web-based tool for exploring large phylogenetic trees. *Elife*, 11. https://doi.org/10.7554/eLife.82392
32. Taxonium documentation. Available online: https://docs.taxonium.org (accessed June 4, 2025).
33. Bisong, E. (2019) Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA, USA. https://doi.org/10.1007/978-1-4842-4470-8_7
34. Conda: A system-level, binary package and environment manager running on all major operating systems and platforms. Available online: https://github.com/conda/conda (accessed June 4, 2025).
35. A Python Suite for Evolutionary and Comparative Genomics. Available online: https://github.com/bob-friedman/EvolCat-Python (accessed on 4 June 2025).
36. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017) Attention is All you Need. *Advances in Neural Information Processing*, 30. https://arxiv.org/abs/1706.03762v7
37. Yin, X., & Wan, X. (2022, May) How do seq2seq models perform on end-to-end data-to-text generation? In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 7701-10). https://doi.org/10.18653/v1/2022.acl-long.531
38. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2020) A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32, 4-24. https://arxiv.org/abs/1901.00596
39. Naqvi, A. A. T., Fatima, K., Mohammad, T., Fatima, U., Singh, I. K., Singh, A., ... & Hassan, M. I. (2020) Insights into SARS-CoV-2 genome, structure, evolution, pathogenesis and therapies: Structural genomics approach. *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, 1866, 165878. https://doi.org/10.1016/j.bbadis.2020.165878
40. Huang, Y., Xu, J., Lai, J., Jiang, Z., Chen, T., Li, Z., ... & Zhao, P. (2023) Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey. *arXiv*, arXiv:2311.12351. https://arxiv.org/abs/2311.12351v2
41. Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., ... & Ahmed, A. (2020) Big Bird: Transformers for Longer Sequences. *Advances in Neural Information Processing Systems*, 33, 17283-97. https://arxiv.org/abs/2007.14062v2
42. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2019) HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv*, arXiv:1910.03771. https://arxiv.org/abs/1910.03771
43. Zhang, J., Xiao, T., Cai, Y., & Chen, B. (2021) Structure of SARS-CoV-2 spike protein. *Current Opinion in Virology*, 50, 173-82. https://doi.org/10.1016/j.coviro.2021.08.010
44. Friedman, R. (2023) Tokenization in the Theory of Knowledge. Encyclopedia, 3, 380-86. https://doi.org/10.3390/encyclopedia3010024

 doi:10.20944/preprints202506.0456.v1